Honeypots for Windows

ROGER A. GRIMES

Apress®

Honeypots for Windows

ISBN (pbk): 1-59059-335-9

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jim Sumser
Technical Reviewers: Alexzander Nepomnjashiy, Jacco Tunnissen
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Chris Mills, Dominic Shakeshaft, Jim Sumser
Assistant Publisher: Grace Wong
Project Manager: Sofia Marchant
Copy Manager: Nicole LeClerc
Copy Editor: Marilyn Smith
Production Manager: Kari Brooks-Copony
Production Editor: Kelly Winquist
Compositors: Kinetic Publishing Services, LLC; Dina Quan
Proofreader: Katie Stence
Indexer: Carol Burbo
Artist: Kinetic Publishing Services, LLC; Dina Quan
Cover Designer: Kurt Krames
Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013, and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders@springer-ny.com, or visit http://www.springer-ny.com. Outside the United States: fax +49 6221 345229, e-mail orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section. You will need to answer questions pertaining to this book in order to successfully download the code.

To those who fight the good fight with constant vigilance.

Contents at a Glance

About the Author xv
About the Technical Reviewers xvii
Acknowledgments xix
Introduction xxi

PART 1 ■■■ Honeypots in General

Chapter 1 An Introduction to Honeypots 3
Chapter 2 A Honeypot Deployment Plan 35

PART 2 ■■■ Windows Honeypots

Chapter 3 Windows Honeypot Modeling 63
Chapter 4 Windows Honeypot Deployment 89
Chapter 5 Honeyd Installation 121
Chapter 6 Honeyd Configuration 151
Chapter 7 Honeyd Service Scripts 167
Chapter 8 Other Windows-Based Honeypots 189

PART 3 ■■■ Honeypot Operations

Chapter 9 Network Traffic Analysis 223
Chapter 10 Honeypot Monitoring 269
Chapter 11 Honeypot Data Analysis 301
Chapter 12 Malware Code Analysis 337

INDEX 363

Contents

About the Author xv
About the Technical Reviewers xvii
Acknowledgments xix
Introduction xxi

PART 1 Honeypots in General

CHAPTER 1 An Introduction to Honeypots 3

What Is a Honeypot? 3
What Is a Honeynet? 5
Why Use a Honeypot? 5
Low False-Positives 5
Early Detection 7
New Threat Detection 7
Know Your Enemy 8
Defense in Depth 8
Hacking Prevention 8
Internet Simulation Environment 10
Basic Honeypot Components 11
Honeypot Types 13
Honeypot Layers 13
Honeypot Interaction Levels 14
Real Operating System Honeypot 15
Virtual Honeypots 16
Summary of Honeypot Types 20
History of Honeypots 20
GenI Honeypots 21
The GenII Model 24
Future Generations 26

Attack Models 26
Manual Attacks 26
Automated Attack Programs 30
Blended Attacks 31
Summary of Attack Models 32
Risks of Using Honeypots 32
Summary 34

CHAPTER 2 A Honeypot Deployment Plan 35

Honeypot Deployment Steps 35
Honeypot Design Tenets 36
Attracting Hackers 37
Defining Goals 37
Production or Research? 37
Real or Virtual? 39
Hardening a Virtual Honeypot Host 40
Honeypot System Network Devices 41
Hub 41
Bridge 46
Switch 46
Router 47
Firewall 51
Honeywall 51
Honeypot Network Devices Summary 52
Honeypot System Placement 54
External Placement 55
Internal Placement 56
DMZ Placement 57
Honeypot Placement Summary 58
Summary 59

PART 2 Windows Honeypots

CHAPTER 3 Windows Honeypot Modeling 63

What You Need to Know 63
Common Ports and Services 65

Computer Roles 68
Generic Windows Server 68
IIS Server 69
Windows 2000 Domain Controller 69
Windows Workstation 70
SQL Server 70
Exchange Server 71
Services in More Detail 72
RPC 72
NetBIOS 73
RDP 78
Simple TCP/IP Services 78
FTP 79
Telnet Server 80
IIS 80
Exchange Server 83
Common Ports by Platform 83
Common Windows Applications 86
Putting It All Together 87
Summary 88

CHAPTER 4 Windows Honeypot Deployment 89

Decisions to Make 89
Do You Really Need a High-Interaction Honeypot? 90
Real Operating System or Virtual Machine? 90
Which Microsoft Operating System to Choose? 90
Client or Server? 93
Patched or Unpatched? 93
What Support Tools Are Available? 93
Which Services and Applications to Install? 94
SAM or Active Directory? 94
Hacker's Choice? 94
What Hardware Is Required? 95
Installation Guidance 96
Installation Steps 97
Honeypot Installation Tips 99

Hardening Microsoft Windows 100
Physically Securing the Honeypot 100
Installing Necessary Patches 101
Rejecting Defaults 103
Hardening the TCP/IP Stack 104
Removing or Securing Network Shares 104
Filtering Network Traffic 105
Restricting Unauthorized Software Execution 106
Protecting User Accounts 117
Securing Authentication Protocols 118
Automating Security 119
Summary 120

CHAPTER 5 Honeyd Installation 121

What Is Honeyd? 121
Why Use Honeyd? 122
Honeyd Features 123
IP Stack Emulation 123
TCP/IP Port Emulation 131
Honeyd Logging 134
Honeyd Installation 136
Deciding Logistics 137
Hardening the Host 139
Installing WinPcap 140
Installing Cygwin 142
Installing Honeyd 145
Downloading Scripts 146
Installing Snort 146
Installing Ethereal 147
Reviewing the Honeyd Directory Structure 148
Summary 149

CHAPTER 6 Honeyd Configuration 151

Using Honeyd Command-Line Options 151
Creating a Honeyd Runtime Batch File 152
Setting Up Honeyd Configuration Files 154
Configuring Honeyd Templates 154
Assembling Templates in a Honeyd Configuration File 161

Testing Your Honeyd Configuration 165
Summary 166

CHAPTER 7 Honeyd Service Scripts 167

Honeyd Script Basics 167
Common Script Languages 168
Script Input/Output Routines 170
Honeyd Variables 171
Honeyd Configuration File Syntax 171
Default Honeyd Scripts 172
SSH Test Script 172
Cisco Telnet Session Script 173
IIS Web Emulation 176
Downloadable Scripts 178
Custom Scripts 180
A Worm Catcher Script 180
An Offensive Response Script 181
Microsoft FTP Server 183
Summary 188

CHAPTER 8 Other Windows-Based Honeypots 189

Back Officer Friendly 189
LaBrea 190
Installing and Running LaBrea 191
Using LaBrea 191
SPECTER 192
Setting Up SPECTER 193
Logging and Alerting with SPECTER 194
KFSensor 196
Installing and Running KFSensor 197
Emulating Services with KFSensor 198
Logging and Alerting with KFSensor 208
Configuring KFSensor Listeners and Anti-DoS Settings 210
PatriotBox 212
Emulating Services with PatriotBox 212
Creating Custom PatriotBox Port Listeners 214
Logging and Alerting with PatriotBox 214

Jackpot SMTP Tarpit 214
Installing Jackpot 216
Configuring Jackpot 216
Running Jackpot 218
More Honeypots 219
Summary 219

PART 3 Honeypot Operations

CHAPTER 9 Network Traffic Analysis 223

Why Use a Sniffer and an IDS? 223
Sniffer Benefits 223
IDS Benefits 225
How a Sniffer and IDS Complement Each Other 226
Where to Place the Sniffer and IDS 226
Network Protocol Basics 227
The OSI Model 227
TCP/IP Suite Basics 230
Windows Protocols 237
Network Protocol Capturing Basics 239
Ethereal 240
Viewing Packet Information 241
Using Ethereal Features 244
Using Tcpdump or WinDump with Ethereal 249
Using Built-in Ethereal Command-Line Tools 249
Snort 250
Understanding How Snort Works 250
Installing Snort 252
Configuring Snort 252
Using Snort Click-and-Point 268
Summary 268

CHAPTER 10 Honeypot Monitoring 269

Taking Baselines 269
Host Baselines 272
Network Baselines 275

Monitoring 276
In-Band vs. Out-of-Band Monitoring 276
Monitoring Programs 277
Protection for Monitoring Communications 284
Logging 284
Time Synchronization 285
Logging of Security Events 285
Centralized Data Collection 287
Log File Formats 290
Data Filtering 291
Data Correlation 293
A Honeynet Security Console 294
Useful Information Extraction 294
Log Protection 295
Alerting 295
Alert Considerations 295
Alerting Programs 296
Summary 300

CHAPTER 11 Honeypot Data Analysis 301

Why Analyze? 301
Honeypot Analysis Investigations 302
Automated vs. Manual 302
Initial Compromise 303
After the Initial Compromise 303
A Structured Forensic Analysis Approach 304
Taking the Honeypot Offline 305
Recovering RAM Data 305
Making Copies of the Hard Drive 306
Analyzing Network Traffic 309
Analyzing the File System 311
Analyzing Malicious Code 317
Analyzing the Operating System 318
Analyzing Logs 319
Drawing Conclusions 324
Modifying and Redeploying the Honeypot System 324
Forensic Analysis in Action 325
A KFSensor Honeypot 325
The WhiteDoe Real Honeypot 332

Forensic Tool Web Sites 335
Summary 336

CHAPTER 12 Malware Code Analysis 337

An Overview of Code Disassembly 337
Assembly Language 339
Programming Interfaces 340
Assembly Language Instructions on Computer Platforms 345
Assembler and Disassembler Programs 349
Assemblers 350
Disassemblers 353
Text Editors. 357
Malicious Programming Techniques 358
Stealth Mechanisms 358
Encryption 358
Packing 358
Debugger Tricks 359
Disassembly Environment. 360
Disassembly Practice 360
Summary 361

INDEX 363

About the Author

ROGER A. GRIMES is a 17-year computer security industry veteran, full-time teacher, author, and consultant. He is the author of 4 books and more than 150 magazine articles on computer security, specializing in Microsoft Windows security and malware defenses. He is a contributing editor for *Windows IT Pro* and *InfoWorld* magazines. His certifications include CPA, CISSP, CEH, CHFI, TICSA, MCT, MCSE: Security (NT/2000/2003/MVP), Security+, A+, and others. Roger is a frequent presenter at national conferences, including MCP TechMentor, Windows Connections, and SANS, where he is always among the highest rated presenters. Roger has created several courses on advanced Windows security for Microsoft, *Windows IT Pro* magazine, and SANS. His clients have included every branch of the armed forces, Microsoft, VeriSign, Fortune 500 companies, cities, and large public school systems and universities.

About the Technical Reviewers

ALEXZANDER NEPOMNJASHIY is a Microsoft SQL Server database designer for NeoSystems NorthWest, a security services, consulting, and training company. He has more than 11 years of experience in the IT field. His work involves extending and improving clients' corporate ERP systems to manage retail sales data, predict market changes, and calculate trends for future market situations.

JACCO TUNNISSEN has been working in the ISP and security fields since the mid-1990s, mainly focusing on FreeBSD and OpenBSD implementations. Currently, he is "educating the masses" using his web sites, where you can find out all about intrusion detection, honeypots (`http://www.honeypots.net`), incident handling, wireless security, computer forensics, DNS, and BGP routing. In his spare time, he enjoys good food and biking in Rotterdam. Jacco likes working as a technical reviewer for several authors.

Acknowledgments

I wish to thank Apress and my editor Jim Sumser, Sofia Marchant, Marilyn Smith, and StudioB's Neil J. Salkind for seeing the vision for a book like this and putting up with my moving deadlines.

I also want to thank Lance Spitzner, Michael Davis, and Niels Provos, for evangelizing honeypot technology, and answering my many questions. Thanks to Alexzander Nepomnjashiy and Jacco Tunnissen for the excellent technical editing.

Much of this book could not have been written without the previous contributions of The Honeynet Project (`http://project.honeynet.org`), Honeypot: Tracking Hackers (`http://www.tracking-hackers.com`), SANS (`http://www.sans.org`), and the Honeypot mailing list (`http://www.securityfocus.com`).

On a personal note, I would especially like to thank my wife, Tricia, who took care of my every need while I was writing and neglecting her. I could not ask for a better friend and partner.

Introduction

Welcome to the world of honeypots! By reading this book, you are joining a friendly community of like-minded individuals who see honeypots as an important step in preventing and learning about malicious hacking activity.

If you were to ask a honeypot administrator why he uses a honeypot, the first response you would likely get is one or more legitimate business reasons explaining why honeypots are the best tool for the job. We've been trained to respond that way, so we can defend all our time spent managing and learning about honeypots to our bosses and loved ones. But underlying all the official logic is the real reason why most of us get interested in honeypots: the thrill of watching the bad guys expose themselves and their techniques, in an environment explicitly built to exploit them. It's videotaping the thief. It's taking the hacker's favorite tool of social engineering and using it against him. It's hacking the hacker.

For once, the good guys are in control and winning. With honeypots, we can track the hackers back to their lairs, identify them, and learn and defend against their techniques before they really get a chance to use them. Honeypots can stop hacker attacks, Internet worms, spam, and other acts of maliciousness. Honeypots remove the implicit veil of privacy most hackers think they have when they exploit a computer. If honeypots become as big and mature as most security experts expect them to be, they could put an end to the random, no consequence, hacking that so proliferates our Internet experience today. Honeypots are one of the few offensive plays in the computer security world.

Honeypots no longer need to prove their value. They have been responsible for discovering many new threats (called *zero-day exploits*) before they were widely known and used, including a Samba buffer overflow. Honeypots have been essential in capturing encrypted hacker traffic and instrumental in learning that hackers are using version 6 of the Internet Protocol (IPv6) to tunnel their communications under the very noses of network administrators. But perhaps, the greatest demonstration of a honeypot's value was the recent profiling of an extensive credit card fraud operation (`http://www.honeynet.org/papers/profiles/cc-fraud.pdf`).

Researchers followed the online activities and communications of individuals involved with credit card fraud, also known as *carders*. The honeynet was able to capture an incredibly automated Internet Relay Chat (IRC) network that allowed credit card numbers and identifying information to be stolen by simply typing a few keystrokes. Members of the ring could type in `!cc` to receive one random stolen credit card number. Other commands returned credit card account limits and provided web site links to vulnerable online merchant sites. The honeynet tracked key individuals, mostly located in South Asia and Pacific Rim countries, as they bought credit card numbers from legitimate businesses, hacked credit card security systems, and taught newcomers the ins and outs of carding. The credit card ring, by plying its trade online and out in the open, made their illegal activities appear more as a subculture than a crime. This attracted new participants. Honeypots allowed evidence to be recorded that after-the-fact affidavits, search warrants, and bugged phone conversations couldn't provide. The honeypot has solidified its place as a legitimate computer security defense tool.

Why Another Book on Honeypots?

There are already some excellent books, papers, and web sites on honeypots, so why did I feel compelled to write a book on the subject? In one sentence, it's because the world of computer security, and honeypots in particular, is largely Unix-based. Most of the literature about firewalls, intrusion detection systems (IDSs), and honeypots was written by Unix gurus. Most of the tools are Unix-based and work only on Unix platforms. Even when the tools are ported to Windows, they may talk about Windows and give a few Windows examples, but most of the text and examples are for Unix-based users. It can be very frustrating when you are not a Unix person but still want to learn about computer security and use all the cool tools.

The majority of the world's PCs run one of Microsoft's Windows operation systems. This book was written to fill the large gap for Windows administrators trying to learn about honeypots outside the Unix subculture. I don't give examples of software and exploits that do not occur in the typical Windows environment. When the book discusses mail servers, it will be referring to Microsoft Exchange, not Sendmail. When it refers to web servers, it will be talking about Microsoft Internet Information Services (IIS), not Apache. Yes, I know both of those programs have Windows-based counterparts, but they aren't the norm in a Windows network. This doesn't mean that the knowledge and lessons learned in this book cannot be applied to non-Microsoft environments. The opposite is true. You can take anything covered in this book and easily apply it to Unix, Linux, Macintosh, or any other computer environment. But for once, this is a honeypot resource that targets the Windows administrator. It means the following:

- Honeypot planning and setup will target Windows systems.
- Security tools will be Windows versions.
- Hacking examples will be Windows examples.
- When TCP/IP is discussed, it will be as it applies in Windows.
- When TCP/IP ports are discussed, they will be ports common to Windows.

Honeypots for Windows is a book for Windows-based users and administrators.

Who Is This Book For?

This book is for administrators and users with an intermediate understanding of the Windows operating system and computer security. Readers should have experience with the Windows operating system, the Internet, and Windows-based networking; be able to install and troubleshoot network-related software; and have general understanding of the OSI model. It helps if you're familiar with basic computer security concepts, such as computer worms, buffer overflows, and password cracking. An understanding of Windows security mechanisms will make the book more enjoyable.

A strong understanding of TCP/IP network protocol basics is essential for most honeypot administrators. Although this book will cover the fundamentals needed to understand the material presented, readers should understand the following terms prior to beginning this journey: TCP, UDP, ICMP, stateful, stateless, flags, TCP/IP handshake, packet header, and packet payload.

But even if you're not familiar with the details of all these topics, you should still be able to understand every concept discussed in this book. So, don't panic if you can't name all the TCP header flags off the top of your head, or if don't know the exact meaning of stateful inspection. This book will be of value to people newly interested in computer security and honeypots, as well as to experienced security experts.

Readers without a firm foundation in these fundamentals should consider a quick refresher with a TCP/IP protocol reference. There are several good books on the TCP/IP protocol, and here are some online references:

- Webopedia's TCP/IP page: `http://www.webopedia.com/TERM/T/TCP_IP.html`
- An excellent TCP/IP Reference by Cisco: `http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ip.pdf`
- About.com's Computer Networking Guide to TCP/IP: `http://compnetworking.about.com/cs/basictcpip`
- Wikipedia Internet Protocol Suite reference: `http://www.wikipedia.org/wiki/Internet_protocol_suite`
- Internet Engineering Task Force (IETF) references: `http://www.ietf.org`

Note On the IETF web site, at a minimum, read the following Request For Comments (RFCs): 791-IP, 792-ICMP, 768-UDP, and 793-TCP. RFCs are very wordy and long, and a bit like reading IRS tax code, but taking the time to read them will allow you to understand the TCP/IP protocol suite in detail.

What's In This Book?

The book has twelve chapters organized into three main parts.

By the time you get through reading this book, you should have an excellent understanding of honeypots in a Windows environment.

Note Many of the tools covered in this book are Windows ports of open-source Unix tools, like Honeyd, WinPcap, and Snort. All of these tools have been tested on Windows 98 and later Microsoft platforms, but most have been optimized for Windows 2000 and above. Menu options and screenshots were done on Windows 2000 and XP Professional computers, but most commands and screens are identical, no matter which Microsoft operating system you use. Every effort has been made to verify that all commands and utilities work across all current versions of Windows. Exceptions are noted when known.

Part One: Honeypots in General

Part One covers honeypot theory and topics common to all honeypots, along with the particular configuration requirements of a Windows-based environment.

Chapter 1 explains general honeypot theory and reasons to use honeypots. It discusses the main honeypot types, along with advantages and disadvantages of each choice. The chapter also covers hacking basics, such as attack model types and fingerprinting. Understanding the different hacking threats is essential to setting up and using a honeypot.

Chapter 2 describes the general setup and deployment of a honeypot, as well as how to attract hackers to it. Topics include how to decide where to place a honeypot and why. It covers the physical deployment issues involved in placing a honeypot, including hardening the host and configuring your network to route hacking traffic to your honeypot. It includes details on the problems introduced on switched networks and how to correctly configure your routing tables.

Part Two: Windows Honeypots

Part Two provides a detailed lesson in configuring and using Windows-based honeypots. Using an emulated honeypot in a Windows environment takes special consideration to make it appear as a Windows-based host. This means it should have the normal Windows ports open, run the normal Windows services, and respond in a predictable way. Chapter 3 defines normal behaviors, ports, and services on a Windows host, and tells you how to emulate them on a honeypot.

Chapter 4 describes using a real Windows operating system as a honeypot. It reveals what is the best Windows version to attract malicious hackers and presents hardening tips you can use to minimize compromise damage.

Chapters 5 through 7 focus on Honeyd, the most popular honeypot software in use today. Chapter 5 covers how to download and install Honeyd. Honeyd is a fantastic free tool, but like many other open-source programs, not particularly easy to configure. Chapter 6 begins deciphering the Honeyd configuration and provides several sample configuration files that you can adapt for your own needs. Chapter 7 explains how to use service scripts, which allow Honeyd to mimic basic applications, such as FTP, telnet, and IIS. Service scripts are very important in making a honeypot look like a real system.

Honeyd is the most popular and versatile honeypot software in use today, but it isn't the easiest to use. In Chapter 8, we explore six other Windows-based honeypots with front-end graphical user interfaces that make for a more pleasant user experience. Each of these honeypots excels at different goals. The honeypots are Back Officer Friendly, LaBrea, SPECTER, KFSensor, PatriotBox, and Jackpot.

Part Three: Honeypot Operations

Part Three discusses a range of topics related to getting the most out of your honeypot.

Using a network traffic analyzer and understanding how to recognize and decode malicious network traffic is essential to honeypot operations. Chapter 9 discusses how to install and use various tools for analyzing network traffic. It begins with network protocol basics, reviewing the OSI model and TCP/IP suite, and then focuses on using Snort and Ethereal.

Chapter 10 covers the very important issues of monitoring, logging, alerting, and reporting. It discusses how to set up an alert system, how and what to log, and what reports you need to generate.

Honeypots can quickly gather copious amounts of information—sometimes an overwhelming amount. The ultimate success of your honeypot is determined by how well you interpret the attack evidence. Chapter 11 discusses techniques to use in the forensics analysis of your honeypot data.

Chapter 12 discusses analyzing malicious code by disassembling it. For new programmers, this involves learning assembly language, learning how to disassemble executables, and learning about malicious coding in general. Becoming a disassembler is not for the faint of heart, but with a moderate amount of effort and practice, it can reveal malware functions that cannot be found any other way.

The sample files presented in this book, as well as other related files, are available from the Downloads area of the Apress web site (`http://www.apress.com`). You can direct any technical questions or concerns to me at roger@banneretcs.com.

PART ONE

Honeypots in General

Honeypots for Windows **focuses on creating realistic honeypots mimicking production Windows environments. Part One covers honeypot terminology and technology in general, as it applies to any environment.**

Chapter 1 introduces honeypots. Chapter 2 covers where honeypots should be placed in order to meet your goals.

CHAPTER 1

An Introduction to Honeypots

It's an exciting time to be involved with honeypots. If you've ever wanted to participate in something grand just as it's taking off big, honeypots are it.

This chapter gets you started by defining just what honeypots and honeynets are, and why you would want to use them. It also provides an overview of honeypot components and types, as well as a history of their evolution. Finally, we'll talk about the risks and trade-offs involved in operating honeypots.

What Is a Honeypot?

"A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource." This definition was developed by Lance Spitzner (founder of The Honeynet Project) and the unofficial leader of the honeypot community. This definition includes two overall guiding principles of honeypots:

- The phrase *information system resource* is broadly defined intentionally, so that the honeypot can be any type of computer resource. It can be a workstation, file server, mail server, printer, router, any network device, or even an entire network. While honeypots most often mimic servers and workstations, I've seen many router honeypots, and even a honeypot mimicking a Hewlett-Packard JetDirect print server card.
- A honeypot is intentionally put in harm's way to be compromised and has no legitimate production value beyond the honeypot goals. If your web server is frequently exploited and you analyze the information, that doesn't make it a honeypot; that just makes it a poorly configured web server.

Note The Honeynet Project (`http://www.honeynet.org`) is a nonprofit organization founded in October 1999 dedicated to information security and honeypot research. Its goal is to learn the tools, tactics, and motives of the blackhat community and share these lessons learned. All of its work is open source and shared with the security community.

A honeypot can be whatever you want it to be. If you're interested in honeypots as a new user, you must first decide on your goals, which will determine what type of honeypot you will deploy. Ask yourself what you are interested in protecting and what you are interested in learning.

THE HONEYPOT AS A TOOL

Every honeypot administrator has a favorite honeypot story, so it seems appropriate to start this book with one of mine. The client had called me because the company's network was under constant attack. When they connected their corporate network to the Internet, network utilization immediately maxed out. When the Internet was disconnected, network utilization dropped to its normal 2% to 5%. It was visually stunning to see all the network device activity lights staying lit constantly, all at once. It reminded me of a network broadcast storm, but that wasn't the problem. Whatever the malicious hackers were doing, it required a lot of bandwidth.

The client's technicians had run antivirus scanners, had an actively functioning (though misconfigured) firewall, and had looked at all their servers and workstations for signs of foul play. They couldn't find any. They even changed all the passwords in a futile attempt to lock out the hackers. This did prevent the attack for about 15 minutes, but then the hackers were back in. The technicians had placed a network sniffer on their network to capture and analyze packets for malicious content. They had been successful, but so successful that they had captured millions of packets—far too many to manually investigate how the hackers were getting in and discover what they were doing.

We disconnected the network from the Internet, and I set up a honeypot using Honeyd (an open-source honeypot solution). We configured it to accept connections to any UDP or TCP port number. I hooked up Snort (an intrusion detection system, or IDS) to capture all inbound packet traffic to the honeypot. We changed all of the user passwords again, turned on Honeyd, enabled the Internet connection, and then waited. We had connection attempts in seconds. Analyzing the data in real time, the honeypot captured many attempts to connect to a Microsoft SQL Server using the administrator account, called SA, and a blank password. A blank SA password is a common vulnerability in poorly configured SQL Server systems.

I asked the technicians if they had any SQL Server servers with blank SA passwords. They said they did not. We went to the computer room to investigate. They showed me their two production SQL Server computers. Neither had blank SA passwords. Then a technician remembered that one of the dusty boxes sitting in the corner of the room ran SQL Server, too. It was a development box installed by an outside programming company hired to upgrade one of the client's primary applications. The programming job became a debacle, the vendor never delivered the promised application, and the box had remained unused for almost a year, or so they had thought. It did indeed have a blank SA password, and in quick order, we found out it had been exploited.

Using a packet sniffer, I then found heavy traffic between the development SQL Server box and an older Novell NetWare 3.11 server that ran Lotus cc:Mail (an e-mail application) for one user on the network. The NetWare server was being used by the hackers as an Instant Relay Chat (IRC) server and as a storage site for pirated software, pornography, and games. The SQL Server box was simply the hackers' initial backdoor entry to gain access to my client's network. The hackers then connected to the NetWare server using the never-changed SUPERVISOR account and began serving up files. The file storage area on the NetWare file server and traffic between the two compromised boxes were encrypted. We could see that a lot of disk space was used up (using the Novell `syscon` and `ndir` commands), but not what it contained.

We pulled the compromised computers off the network, and the hacking traffic disappeared, except for the hundreds of new attempts knocking against the now properly configured firewall. We knew the hacking was automated, because the time between when we replugged the network back into the Internet and the SQL Server attacks beginning again was seconds. We set up another SQL Server computer as a honeypot, but this time with all the appropriate monitoring tools. We were able to capture the hackers' communications (mostly in a foreign language), learned the encryption method (SSH), and learned that the attacks were coming from different IP addresses in Korea.

Because we could not be certain of the extent of the damage, we spent the rest of the week rebuilding network servers, cleaning workstations, and tightening default security. A company policy was written to ensure that future vendors had to follow a certain set of recommended security steps, including changing all default passwords and not leaving blank ones. The client now runs a honeypot and IDS full time. They continue to get daily visits from the Korean hackers, even though they have complained through the appropriate administrative channels.

There are many ways this particular type of malicious hacking event could have been discovered. If the firewall were appropriately configured, the attack would not have been successful in the first place. The client could have used a network sniffer, auditing log files, or computer security utilities to track down the origination of the high-traffic levels. But in this case, a honeypot used in conjunction with an IDS proved an effective combination. That's because any traffic going to the honeypot is malicious in nature, so it's quick and easy to separate the malicious activity from the legitimate traffic.

What Is a Honeynet?

A collection of honeypots under the control of one person or organization is called a *honeynet*. Figure 1-1 shows an example of a honeynet with four servers. In the example, the honeypots are running different operating systems: Windows 2000, Windows NT 4.0, Windows Server 2003, and Internet Information Server (IIS). These machines are on a demilitarized zone (DMZ), which is an internal routed segment accessible from the Internet, but separated directly from the LAN.

A honeynet can have a single *theme*, say mimicking a military network or series of database servers, or be distributed widely enough that it makes sense to have multiple themes, such as a military site running Windows Server 2003 connected to an educational site running Windows 2000 Server. A single-theme honeynet can still run different operating systems and features. A theme might mimic an e-commerce site, with a web server, back-end database server, customer support File Transfer Protocol (FTP) site, and Simple Mail Transfer Protocol (SMTP) mail server.

Why Use a Honeypot?

If you've done home repair or carpentry in your life, you know how important picking the right tool can be. A screwdriver that is just slightly too big or small can make a simple job unbearable. The incorrect screwdriver can strip screw heads, bend the screwdriver, or worse yet, dig into flesh and draw blood. When you have the right screwdriver, the screw seems to almost turn itself. Using the right tool makes any job easier. Honeypots are often the best tool for a security defense. Here, we'll look at some common reasons for using a honeypot.

Low False-Positives

The number one reason for using a honeypot is the low number of false-positives and false-negatives it has. A *false-positive* is when a security tool indicates that nonmalicious activity is malicious. A *false-negative* is when a security tool does not identify malicious activity as being malicious.

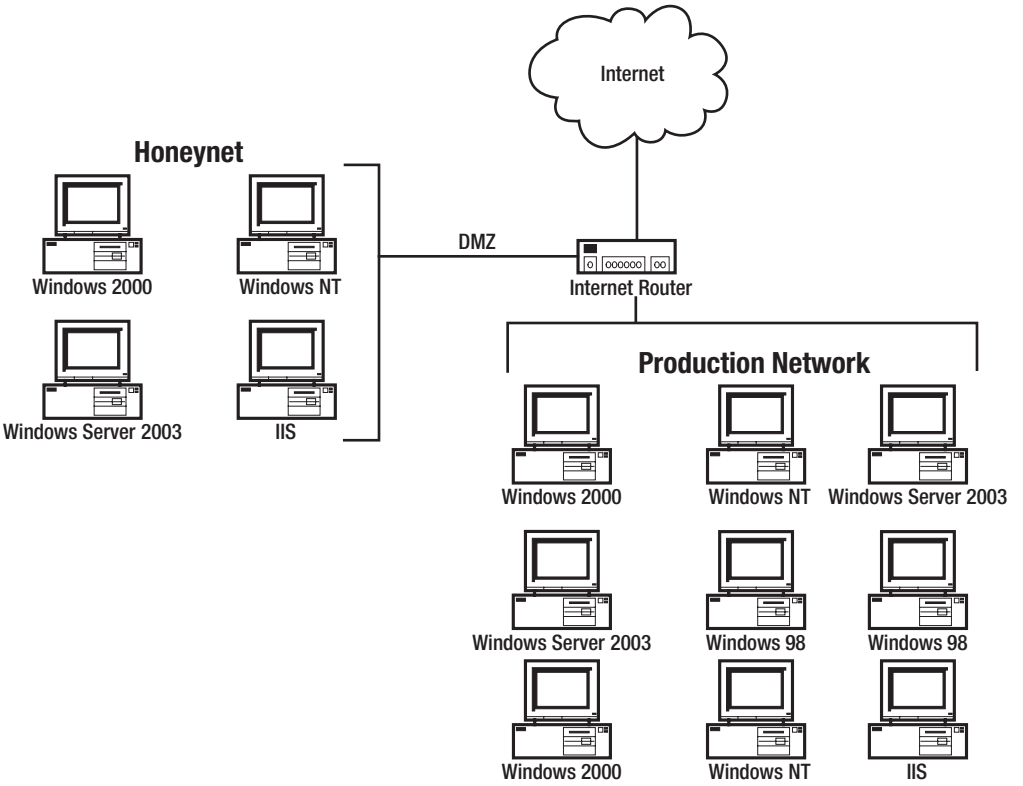

Figure 1-1. *A honeynet example*

Security logs with many false-positives are considered to contain a lot of *noise*. False-positives are very common in intrusion detection systems (IDSs) and firewalls, as are false-negatives, to a lesser extent. Much effort is spent trying to decrease noise coming from firewalls and IDSs. Often, the noise is so high that administrators give up reading and analyzing their logs, decreasing the value of the security device.

In comparison, honeypots have no legitimate production value and should never be accessed by anyone but the honeypot administrator. Any honeypot traffic, outside the expected administrative traffic, is probably malicious. Any traffic leaving the honeypot is malicious.

■ **Note** If a honeypot is used internally, it's not uncommon for the honeypot to detect and report nonmalicious broadcast traffic. Broadcast traffic can be in the form of Address Resolution Protocol (ARP) packets and Windows NetBIOS broadcasts. Honeypot network segments should be designed to filter normal broadcasts away from the honeypot.

The low noise ratio in the captured data is considered to have high value. What a honeypot captures should always be investigated. If you want to get involved with computer and network security only when a real compromise attempt is taking place, then you'll love honeypots.

Early Detection

The low occurrence of false-positives and false-negatives naturally leads to the rapid detection of legitimate threats. Some administrators use a *honeytoken* to ensure early detection. A honeytoken is any object without legitimate production value placed only as an early warning mechanism. Honeytokens can be placed on honeypots or on regular production servers. For example, a honeytoken can be an inactive hoax user account called Administrator, without any permissions. (It is common for the actual Administrator account on Windows computers to be renamed to some other nondescript login name as a way to impede hackers and automated hacking tools.) If anyone tries to log in to the computer or network using the hoax Administrator user account, an alert is generated.

■ **Note** The term *hacker* can be used to describe any computer user who explores computers beyond the normal boundaries of the end-user interface. Throughout this book, the term *hacker* is meant to imply a person with malicious intent, although it's widely understood that not all hackers have malicious intent.

Regardless of how the honeypot detects an active exploit, it can alert you immediately to the attempted compromise. You can respond quickly, close the security hole, and minimize the damage. The same principal applies on a larger scale. For example, during the early morning hours of January 25, 2003, honeypots were among the first security solutions to detect the SQL Slammer worm. Slammer attacked vulnerable Microsoft SQL Server 2000 software with a buffer overflow on UDP port 1434 and infected more than 200,000 computers in its first ten minutes of release. It brought down a major banking ATM network and caused denial of service (DoS) attacks across large portions of the Internet. Early detection allowed an internationally coordinated effort to block port 1434 traffic headed across major backbones, stopping the worm from spreading even farther. By the time most of us awoke, the biggest part of the threat was over. The worm was being quickly eradicated, detection signatures were available, and exploited networks were in cleanup mode. This was due in large part to the early detection work by honeypots and IDSs.

New Threat Detection

Because every connection on a honeypot is a legitimate threat, previously unknown attacks are found just as quickly as known attack vectors. For example, at least two major zero-day exploits were first discovered and documented by honeypots. New hacking methods, while not necessarily zero-day, are discovered routinely by honeypots. There are even research tools, like Honeycomb (`http://www.cl.cam.ac.uk/~cpk25/honeycomb`), that allow brand-new threats discovered by a honeypot to automatically generate an IDS signature. Unlike a virus scanner or signature-detecting IDS, honeypots are excellent at detecting new threats.

Know Your Enemy

Know Your Enemy is the name of a honeypot book by Lance Spitzner and is one of the many mantras of The Honeynet Project. There is no better tool for learning what hackers are up to than a honeypot. You can learn what hackers are doing in general, or you can discover specifically what particular hackers want to do with your information resources. If you put up a honeypot with the goal of learning in general what hackers are up to, it is considered a *research* honeypot. If your goal is in learning about or preventing specific attacks against your organization, it is called a *production* honeypot.

I frequently do work for a large, international, nonprofit religious organization. This organization receives hundreds of attempted attacks a day. Frequently compromised web sites and daily DoS attacks were the norm for the many years. The network administrators have devised a system, using an IDS, that automatically redirects suspected hacking activity to a quarantined area full of different types of honeypots. Now the attackers can attack and malign all day long, without causing a problem to the legitimate targets. The client rarely suffers an attack that is successful against a production asset, and the network administrators use the collected information to better protect their corporate network.

Honeypots can capture everything associated with the hacker, including all network packets, uploaded malware, chat communications, and typed commands. This allows the administrator to learn what the hackers are doing and how they are doing it. As a case in point, it was recently discovered that hackers are setting up Internet Protocol version 6 (IPv6) stacks on machines they have exploited. The hackers then tunnel IPv6 traffic inside the IPv4 traffic, creating a simple but effective virtual private network (VPN). Many IDSs and firewalls, not being designed for IPv6, can't decode the tunneled traffic and are not able to peer inside the malicious packets. A compromised honeypot in an AT&T Mexican honeynet (`http://www.honeynet.org/scans/scan28`) captured hackers using IPv6 to tunnel malicious IRC traffic. The discovery of this led to an increased awareness of the importance of firewalls and IDSs in decoding IPv6 traffic. Honeypots are instrumental in knowing what the enemy is up to.

Defense in Depth

The defense-in-depth security paradigm states that the more defensive tools that are protecting a network, the more successful the overall defense will be. A common use for a honeypot is to place it inside the network perimeter (honeypot placement will be discussed in detail in Chapter 2). If something sneaks past the firewall and IDS and ends up inside the network, there is a chance the honeypot will pick it up. A layered defense will be more likely to catch something that another solution missed. Many of today's computer viruses and worms spread by attempting to infect weakly password-protected NetBIOS shares. Scans made to ports 137 through 139 (the NetBIOS ports) on your honeypot could indicate that a virus or worm has made it inside the perimeter.

Hacking Prevention

Honeypots aren't normally promoted for their ability to prevent malicious activity. Most honeypots, by their very nature, are passive recording devices. Unlike a firewall or IDS, most honeypots are only marginally able to prevent further hacking. But this is not always the case. First, if hackers are spending time attacking a honeypot, you are distracting them from attacking a legitimate production target. This is preventing hacking. Second, it is important to design

your honeypot so that it cannot be used to attack other computers. It is very common for hackers to use a compromised system to attack other systems. It allows the hacker to hide behind another "innocent" computer. If the hacker's attack is traced, the trail stops at another previously compromised box, never leading to the hacker's origination point. A properly designed honeypot will prevent the hacker from successfully attacking other machines.

Tarpits

Some honeypots, like LaBrea and Jackpot, are examples of *tarpits*. Tarpits (also known as black-holes) are *sticky* honeypots built explicitly to slow down or prevent malicious activity. Both LaBrea and Jackpot are open-source honeypots capable of running on Windows computers.

LaBrea (`http://labrea.sourceforge.net`) was developed in response to the Code Red worm. When activated, it listens on the local network to ARP packets and learns all the legitimate IP addresses. When an incoming packet requests an invalid IP address (which should rarely happen for legitimate reasons), LaBrea responds and pretends to be a computer at the corresponding probed IP address. It then works to keep malicious connections hung up in an open, persistent state, maximizing the use of timeout periods transmission retries, actually slowing down automated hacking worms and tools.

Jackpot (`http://jackpot.uk.net`) is a Java-based antispam relay server. When executed, Jackpot accepts connections on port 25. Since Jackpot is installed only as a honeypot decoy, no legitimate mail traffic should ever try to contact it. But spammers will attempt contact when looking for open relays. Jackpot answers the spammer as a valid SMTP server and pretends to have an open relay. When the spammer sends the spam, Jackpot logs the originating IP address and doesn't pass along the spam. Jackpot can be used to slow down and frustrate spammers, but it has also been used to track down the spammers and report their illegal activities.

Wireless honeypots are also being used to detect war drivers, who attempt to detect and exploit weakly protected wireless access points (WAPs). The war driver's Media Access Control (MAC) address can then be recorded and the unauthorized use prevented.

The LaBrea and Jackpot tarpits will be covered in more detail in Chapter 8.

Redirectors

As discussed in the "The GenII Model" section later in this chapter, honeywall gateways can be used to redirect malicious activity away from production assets, or like an IDS, interact with other network defense devices. If the defense system is set up correctly, when an attack is detected, it can be redirected to a honeypot clone of the attacked production system. Clusters of honeypots used in this way make up a *honeypot farm*. In the future, a global network might be able to redirect attacks occurring anywhere in their network to the honeypot farm. If done appropriately, the honeypot clone will appear to have the same IP and MAC address as the original target. To this end, honeywalls already exist, and hardware redirectors are now being developed to make the switch happen at layer 2 of the International Organization for Standardization's (ISO's) Open System Interconnection (OSI) model, a much harder method for the hacker to detect.

Attacks detected by honeypots can also initiate other proactive defenses. For example, if a honeypot detects malicious activity, it can update firewall rules to make sure the hackers never get access to any production assets. On the downside, any type of automated defense tool has the potential to react too quickly to false-positives and generate a self-created DoS attack.

How well the redirector determines what is and isn't malicious activity is important. Imagine if you run an e-commerce site with a honeypot that is an exact clone of your production computer. If legitimate users are redirected to the honeypot system by mistake, they could be making purchase transactions that do not get posted to the company's legitimate site.

■ **Note** Although not available in a Windows version yet, the Linux-based Bait and Switch Honeypot (`http://violating.us/projects/baitnswitch`) is among the most popular redirectors.

Honeypot As a Forensics Tool

Any honeypot has the ability to collect evidence for use against the hacker. With firewall logs, the best you can do is collect meager summary messages as proof of the unauthorized activity. I've often had hackers claim to their Internet service providers (ISPs) and authorities (to whom I had reported them) that they were innocent and that they did not know what they were doing. Without hard evidence, most of the time, they get let off with a warning. With a honeypot, you can replay the entire attack. Interested parties can see the entire sequence of events and decide for themselves what the hacker's intent was.

Better Defenses

Any honeypot, by its very nature, will help its administrator improve defenses. For example, if a honeypot is receiving a lot of SQL Slammer worm probes, more than likely, the administrator will make sure the network's production boxes are patched against Slammer.

When new attacks become popular, honeypot users are the first to warn the Internet community at large. Honeypots help computer security by improving defenses.

Second-generation honeypot products are being developed that will integrate with IDSs, firewalls, antivirus scanners, and other computer-defense mechanisms and report to a common management console. When an attack is noticed, the protected assets can be scanned to see if they have a related vulnerability to the attack. This is known as *relevancy*. If the attack could be successful against protected computers, and hence, relevant, the security administrator should be alerted. If not, the event can just be logged. This process could even begin the downloading of necessary patches or automatically reconfigure production computers to make them invulnerable to the attack.

Although honeypots are not normally thought of as preventing hacking, clearly they are active in the fight.

Internet Simulation Environment

Niels Provos, creator of the Honeyd honeypot (described in the "Emulated Honeypots" section later in this chapter), uses his software to simulate networks of machines during classes that he teaches. He has successfully allowed one physical host to respond for 65,536 different IP addresses, each with different personalities and different port services. Honeyd can create entire virtual networks, with routed segments and latency.

How you decide to use a honeypot is up to you. If you want to create a production honeypot to protect your environment, make it mimic your real environment. If you want to learn about hacking no matter what its form, create a research honeypot that simulates a myriad of different environments. As the definition presented earlier in the chapter says, a honeypot is whatever you want it to be.

Basic Honeypot Components

Besides the honeypot itself, you need the following components to operate a honeypot:

- **Network device hardware:** The network devices will consist of firewalls, routers, and switches. Figure 1-2 shows an example of a honeypot deployment. We will cover honeypot deployment in Chapter 2.

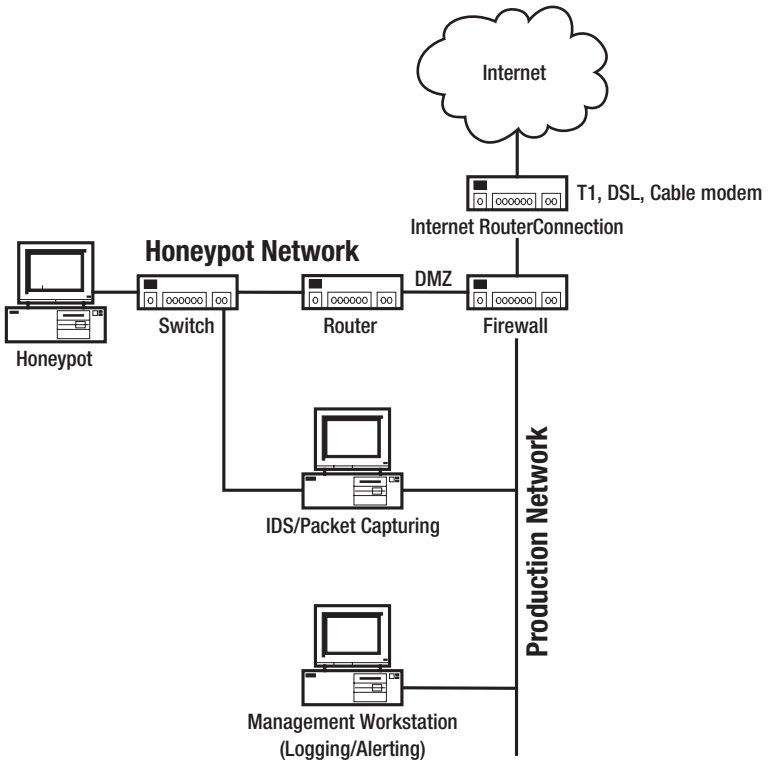

Figure 1-2. *A sample honeypot deployment*