howlett_fm.fm Page i Tuesday, June 29, 2004 2:10 PM

Open Source Security Tools

Open Source Security Tools

Practical Applications for Security

Tony Howlett

PRENTICE HALL PTR

Prentice Hall
Professional Technical Reference
Upper Saddle River, NJ 07458
www.phptr.com

Visit Prentice Hall on the Web: www.phptr.com

Library of Congress Cataloging-in-Publication Data

Howlett, Tony.
Open source security tools : practical applications for security / Tony Howlett
p. cm.
Includes index.
ISBN 0-321-19443-8 (pbk. : alk. paper)
1. Computer security. 2. Computer networks—Security measures. 3. Open source software. I. Title.

QA76.9.A25H6985 2004
005.8—dc22
2004009479

Printed in the United States of America

First Printing, July 2004

ISBN 0-321-19443-8

Pearson Education Ltd.
Pearson Education Australia Pty., Limited
Pearson Education South Asia Pte. Ltd.
Pearson Education Asia Ltd.
Pearson Education Canada, Ltd.
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan
Pearson Malaysia S.D.N. B.H.D.

Contents

HowlettTOC.fm Page vii Wednesday, June 23, 2004 10:48 PM

Preface

Open source software is such an integral part of the Internet that is it safe to say that the Internet wouldn't exist as we know it today without it. The Internet never would have grown as fast and as dynamically as it did without open source programs such as BIND, which controls the domain name system; Sendmail, which powers most e-mail servers; INN, which runs many news servers; Major Domo, which runs many of the thousands of mailing lists on the Internet; and of course the popular Apache Web server. One thing for sure is that the Internet is a lot cheaper due to open source software. For that, you can thank the Free Software Foundation, BSD UNIX, Linux and Linus Torvalds, and the thousands of nameless programmers who put their hard work and sweat into the programs that run today's Internet.

While open source programs cover just about every aspect of computer software—from complete operating systems and games to word processors and databases—this book primarily deals with tools used in computer security. In the security field, there are programs that address every possible angle of IT security. There are open source firewalls, intrusion detection systems, vulnerability scanners, forensic tools, and cutting-edge programs for areas such as wireless communications. There are usually multiple choices in each category of mature, stable programs that compare favorably with commercial products. I have tried to choose the best of breed in each major area of information security (in my opinion, of course!). I present them in a detailed manner, showing you not just how to install and run them but also how to use them in your everyday work to have a more secure network. Using the open source software described in this book, you can secure your enterprise from both internal and external security threats with a minimal cost and maximum benefit for both the company and you personally.

I believe combining the concepts of information security with open source software offers one of the most powerful tools for securing your company's infrastructure, and by extension the entire Internet. It is common knowledge that large-scale virus infections and worms are able to spread because many systems are improperly secured. I believe that by educating the rank-and-file system managers and giving them the tools to get the job done, we can make the Internet more secure, one network at a time.

Audience

The audience for this book is intended to be the average network or system administrator whose job duties are not specifically security and who has at least several years of experience. This is not to say that security gurus won't get anything out of this book; there might be areas or tools discussed that are new to you. And likewise, someone just getting into IT will learn quite a bit by installing and using these tools. The concepts discussed and techniques used assume a minimal level of computer and network proficiency.

There is also a broad group of readers that is often overlooked by the many open source books. These are the Windows system administrators. The info-security elite often has a certain disdain for Windows-only administrators, and little has been written on quality open source software for Windows. However, the fact remains that Windows servers make up the lion's share of the Internet infrastructure, and ignoring this is doing a disservice to them and the security community at large. While overall the book is still tilted towards Linux/UNIX because most open source programs are still Linux/UNIX-only, I have tried to put Windows-based security tools in every chapter. I've also included helpful hints and full explanations for those who have never run a UNIX machine.

Contents

This book covers most of the major areas of information security and the open source tools you can use to help secure them. The chapters are designed around the major disciplines of information security and key concepts are covered in each chapter. The tools included on the book's CD-ROM allow for a lab-like environment that everyone can participate in. All you need is a PC and this book's CD-ROM to start using the tools described herein.

This book also contains some quick tutorials on basic network terminology and concepts. I have found that while many technicians are well-schooled in their particular platforms or applications, they often lack an understanding of the network protocols and how they work together to get your information from point A to point B. Understanding these concepts are vital to securing your network and implementing these tools properly. So while this book may seem slanted towards the network side of security, most of the threats are coming from there these days, so this is the best place to start.

Coverage of each security tool is prefaced by a summary of the tool, contact information, and various resources for support and more information. While I give a fairly detailed look at the tools covered, whole books can and have been written on many of the programs discussed. These resources give you options for further research.

Helpful and sometimes humorous tips and tricks and tangents are used to accent or emphasize an area of particular importance. These are introduced by Flamey the Tech, our helpful yet sometimes acerbic mascot who is there to help and inform the newbies as well as keeping the more technical readers interested in sections where we actually make some minor modifications to the program code. He resembles the denizens you may encounter in the open source world. In exploring the open source world, you will meet many diverse, brilliant, and sometimes bizarre personalities (you have to be a least a little bent to spend as much unpaid time on these programs as some of us do). Knowing the proper etiquette and protocol will get you a lot farther and with fewer flames. On a more serious note, many of the tools in this book can be destructive or malicious if used in the wrong ways. You can unintentionally break the law if you use these tools in an uninformed or careless manner (for example, accidentally scanning IP addresses that aren't yours with safe mode off). Flamey will always pipe up to warn you when this is a possibility.

Open Source Security Tool Index

Immediately following this Preface is a listing of all the tools and the pages where they are covered. This way you can skip all the background and go straight to installing the tools if you want.

Chapter 1: Information Security and Open Source Software

This chapter offers an introduction to the world of information security and open source software. The current state of computer security is discussed along with a brief history of the open source movement.

Chapter 2: Operating System Tools

This chapter covers the importance of setting up your security tool system as securely as possible. A tool for hardening Linux systems is discussed as well as considerations for hardening Windows systems. Several operating system-level tools are reviewed too. These basic tools are like a security administrator's screwdriver and will be used again and again throughout the course of this book and your job.

Chapter 3: Firewalls

The basics of TCP/IP communications and how firewalls work are covered here before jumping into installing and setting up your own open source firewall.

Chapter 4: Port Scanners

This chapter delves deeper into the TCP/IP stack, especially the application layer and ports. It describes the installation and uses for a port scanner, which builds up to the next chapter.

Chapter 5: Vulnerability Scanners

This chapter details a tool that uses some of the earlier technology such as port scanning, but takes it a step further and actually tests the security of the open ports found. This security Swiss army knife will scan your whole network and give you a detailed report on any security holes that it finds.

Chapter 6: Network Sniffers

This chapter primarily deals with the lower levels of the OSI model and how to capture raw data off the wire. Many of the later tools use this basic technology, and it shows how sniffers can be used to diagnose all kinds of network issues in addition to tracking down security problems.

Chapter 7: Intrusion Detection Systems

A tool that uses the sniffer technology introduced in the previous chapter is used here to build a network intrusion detection system. Installation, maintenance, and optimal use are also discussed.

Chapter 8: Analysis and Management Tools

This chapter examines how to keep track of security data and log it efficiently for later review. It also looks at tools that help you analyze the security data and put it in a more usable format.

Chapter 9: Encryption Tools

Sending sensitive data over the Internet is a big concern these days, yet it is becoming more and more of a requirement. These tools will help you encrypt your communications and files with strong encryption as well as create IPsec VPNs.

Chapter 10: Wireless Tools

Wireless networks are becoming quite popular and the tools in this chapter will help you make sure that any wireless networks your company uses are secure and that there aren’t wireless LANs you don’t know about.

Chapter 11: Forensic Tools

The tools discussed in this chapter will help you investigate past break-ins and how to properly collect digital evidence.

Chapter 12: More On Open Source Software

Finally, this chapter will give you resources for finding out more about open source software. Various key Web sites, mailing lists, and other Internet-based resources are identified. Also, I give a number of ways to become more involved in the open source movement if you so desire.

Appendix A: Common Open Source Licenses

Contains the two main open source licenses, the GPL and BSD software licenses.

Appendix B: Basic Linux/UNIX Commands

Contains basic navigation and file manipulation commands for those new to UNIX and Linux.

Appendix C: Well-Known TCP/IP Port Numbers

Contains a listing of all the known port numbers as per IANA. Note that this section is not intended to be comprehensive and is subject to constant update. Please check the IANA Web site for the most current information.

Appendix D: General Permission and Waiver Form

Contains a template for getting permission to scan a third-party network (one that is not your own). This is intended to be used as an example only and is not intended as a legal document.

Appendix E: Nessus Plug-ins

Contains a partial listing of plug-ins for the Nessus Vulnerability Scanner discussed in Chapter 5. This listing will not be the most current since the plug-ins are updated daily. The Nessus Web site should be consulted for plug-ins added after January 12, 2004.

CD-ROM Contents and Organization

The CD-ROM that accompanies this book has most of the open source security tools on it for easy access and installation. The disk is organized into directories labeled by tool. If there are separate files for Windows and Linux, they will be in their own directories. The directory “Misc” has various drivers and other documentation such as RFCs that will be of general use through your reading.

Using the Tools

Whenever possible, the tools in this book are provided in RedHat Package Manager (RPM) format. Of course, you don't have to be running RedHat Linux to use RPM. The RedHat folks originally designed it, but now it comes with most Linux versions. The RedHat Package Manager automates the installation process of a program and makes sure you have all the supporting programs and so forth. It is similar to a Windows installation process where you are guided through the process graphically and prompted where necessary. Using the RPM is almost always preferable to doing a manual installation. When you need to set custom install parameters or if a RPM file is not available for your distribution, I describe how to install the program manually. If the RPM file is provided, simply download the file or copy it from the CD-ROM that comes with this book and click on it. Your version of RPM will take care of the rest.

If you use any of the other variations of UNIX (BSD, Solaris, HP/UX, and so on), they will probably work with the tools in this book, but the installation instructions may be different. You can run most of the tools in this book on alternative versions of UNIX or Linux. Staying within the Linux family will certainly make compatibility more likely with the actual tools on the CD-ROM. If you have to download a different version of the program, some of the features discussed may not be supported. But if you are a Solaris aficionado or believe that BSD is the only way to go, feel free to use it as your security workstation. Just be aware that the instructions in this book were designed for a specific implementation and you may have to do some additional homework to get it to work. The platforms supported are listed at the beginning of each tool description.

Reference Installation

Most of the tools in this book were tested and reviewed on the following platforms:

- Mandrake Linux 9.1 on a HPVectra series PC and a Compaq Presario laptop.
- Windows XP Pro and Windows 2000 Pro on a Compaq Prosignia series desktop and Compaq Armada laptop.

Input or Variables

In code and command examples, italics are used to designate user input. The words in italics should be replaced with the variables or values specific to your installation. Operating system-level commands appear like this:

```
ssh -l login hostname
```

Due to page size limits, code lines that wrap are indented with a small indent.

I hope you enjoy and learn from this book. There are many, many more tools that I couldn't include due to space limitations, and I apologize in advance if I didn't include your favorite tool. I had room to cover only *my* favorites and tried to pick the best of breed in each category. I'm sure some will differ with my choices; feel free to e-mail me at tony@howlett.org, and perhaps those will make it into a future edition.

Acknowledgments

This book wouldn't be possible without the tireless efforts of programmers all around the world, making great open source software. I'd name a few but would certainly leave too many out. Thanks for your great software! I'd like to thank my business partner, Glenn Kramer, for assisting with proofing this book (as well as minding the business while I was busy trying to make deadlines) and my Nessus Command Center (NCC) project mates, Brian Credeur, Lorell Hathcock, and Matt Sisk. Finally, my love and gratitude goes to my lovely wife, Cynthia, and daughters, Carina and Alanna, who sacrificed countless hours without husband and daddy to make this book happen.

Open Source Security Tools Index

Tool Name	On CD?	Linux/ UNIX?	Windows?	Page Number
Swatch	Yes	Yes	No	236
Tcpdump	Yes	Yes	No	167
Traceroute	No	Yes	Yes	32
Tripwire	Yes	Yes	No	226
Turtle Firewall	Yes	Yes	No	71
Whois	No	Yes	Yes	35
Windump	Yes	No	Yes	181

Howlett_CH01.fm Page 1 Wednesday, June 23, 2004 2:58 PM

CHAPTER 1

Information Security and Open Source Software

When Tom Powers took a new job as system administrator at a mid-sized energy company, he knew his computer security skills had been a critical factor for being hired. The company had been hacked several times in the last year and their home page had been replaced with obscene images. Management wanted him to make their company information more secure from digital attacks in addition to running the computer network day to day.

After only his first day on the job, he knew he was in for a challenge. The company lacked even the most basic security protections. Their Internet connection, protected only by a simple ISP router, was wide open to the world. Their public servers were ill-maintained and looked like they hadn't been touched since they were installed. And his budget for improving this situation was practically nothing.

Yet within four months Tom had stabilized the network, stopped any further attacks, locked down the public access points, and cleaned up the internal network, as well as adding services that weren't there before. How could he do all this with such limited resources? He knew the basic principles and concepts of information security and found the right software tools to get the job done. He developed a plan and methodically carried out the following steps using security tools to improve company security.

Securing the Perimeter

First, Tom had to establish some basic defenses to protect his network from the outside so he could direct his time to securing the servers and the inside of the network. He built a firewall for their Internet connections using a program called Turtle Firewall (covered in Chapter 3). Using this software and an old server that wasn't being used for anything else, he configured this machine to allow connections only from the inside of the network outwards; all incoming connections not requested from the inside were blocked. He made some exceptions for the public servers operated by his new employer that needed access from the outside. He was even able to set up a Virtual Private Network (VPN) through the firewall so that his users could connect securely from the outside (see Chapter 3). Now he was able to repel most of the basic attacks coming from the Internet and focus on closing up the other holes in the network.

Plugging the Holes

Tom knew that he needed to assess his network for security holes and figure out where the intruders were getting in. Even though the firewall was now protecting the internal workstations from random incursions, the public servers, such as Web and mail, were still vulnerable to attack. His firewall was also now a target, so he needed a way to ensure it was secure from all attacks. He installed a program called Bastille Linux on his firewall server to make sure it was configured securely (Chapter 2). He then ran a program called Nmap from both outside and inside his network (Chapter 4). This reported what application ports were "visible" from the outside on all his public IP addresses. The internal scan let him know if there were any unusual or unnecessary services running on his internal machines.

Next, he used a program called Nessus to scan the network from the outside and inside again (Chapter 5). This program went much deeper than Nmap, actually checking the open ports for a large number of possible security issues and letting him know if machines were improperly configured on his internal network. The Nessus program created reports showing him where there were security holes on the Web and mail servers and gave him detailed instructions on how to fix them. He used these reports to resolve the issues and then ran the Nessus program again to make sure he had eliminated the problems.

Establishing an Early Warning System

Even though he had sealed up all the holes he knew about, Tom still wanted to know if there was unusual activity happening on his LAN or against his public IP addresses. He used a network sniffer called Ethereal to establish a baseline for different types of activity on his network (Chapter 6). He also set up a Network Intrusion Detection System (NIDS) on a server, using a software package called Snort (Chapter 7). This program watched his network 24/7, looking for suspicious activity that Tom could define specifically, telling him if new attacks were happening, and if people on the inside were doing something they shouldn't be.

Building a Management System for Security Data

Tom was initially overwhelmed with all the data from these systems. However, he set up a database and used several programs to manage the output from his security programs. One called Analysis Console for Intrusion Database (ACID) helped him sort and interpret his NIDS data (Chapter 8). A program called Nessus Command Center (NCC) imported all his Nessus security scan data into a database and ran reports on it (Chapter 8). Tom also had a program called Swatch keeping an eye on his log files for any anomalous activity (Chapter 8). These programs allowed him to view the reports from a Web page, which consolidated all his security monitoring jobs into a half-hour a day task. For a guy like Tom, who was wearing many hats (technical support, programmer, and of course security administrator), this was a crucial time saver.

Implementing a Secure Wireless Solution

Another of Tom's assignments was to set up a wireless network for his company. Tom knew wireless network technology to be rife with security issues, so he used two programs, NetStumbler and WEPCrack, to test the security of his wireless network, and deployed a wireless network that was as secure as it could be (Chapter 10).

Securing Important Files and Communications

One of the things that worried his company's management was the use of e-mail to transfer potentially sensitive documents. As Tom knew, sending information via regular e-mail was akin to sending it on a postcard. Any one of the intermediaries handling a message could potentially read it. He replaced this way of doing business with a system using PGP software, which allowed users to send encrypted files whenever sending confidential or sensitive information and to secure important internal files from unauthorized prying eyes (Chapter 9).

Investigating Break-ins

Finally, with his network as secure as it could be, he checked each server for any remains of past break-ins, both to make sure nothing had been left behind and to see if he could determine who had done the dirty work. Using system-level utilities such as wtmp and lsof, and a program called The Coroner's Toolkit, Tom was able to identify the probable culprits responsible for the past break-ins (Chapter 11). While his evidence wasn't hard enough to turn in to authorities for criminal prosecution, he blocked the offending IP addresses at his new firewall so they couldn't come back to haunt him. He also used this information to file an abuse complaint with their Internet provider.

Tom had accomplished an impressive turnabout in his first few months on the job. And the most amazing thing of all was that he had been able to do it with almost no budget. How did he do this? His training in the information security field helped him develop his plan of attack and carry it out. He was able to leverage this knowledge to install low-cost but effective security solutions by using open source software to build all his systems. Using these packages, Tom was able to turn a poorly secured network into one that could rival the security of much larger networks. And he did this with no staff and a minimal amount of money.

You too can use open source software to secure your company or organization. This book will introduce you to dozens of software packages that will help you accomplish this as well as educate you on the proper policies and procedures to help keep your information secure. As I emphasize many times in this book, software tools are a great help, but they are only half the equation. A well-rounded information security program is also comprised of polices and procedures to maximize the benefits of the software. So, before you start installing software, let's first discuss the basics of information security and the background of open source software.

The Practice of Information Security

The discipline of information security (often shortened to **info-security**) has many different elements, but they all boil down to the main goal of keeping your information safe. They can be distilled into three areas that are the foundation for all information security work: confidentiality, integrity, and availability. The acronym **C.I.A.** is often used to refer to them (no relation to the government agency). This triad represents the goals of information security efforts (see Figure 1.1). Each one requires different tools and methods and protects a different area or type of information.

Confidentiality

The confidentiality segment of info-security keeps your data from being viewed by unauthorized individuals. This can be information that is confidential to your company, such as engineering plans, program code, secret recipes, financial information, or marketing plans. It can be customer information or top-secret government data. Confidentiality also refers to the need to keep information from prying eyes within your own company or organization. Obviously, you don't want all employees to be able to read the CEO's e-mail or view the payroll files.

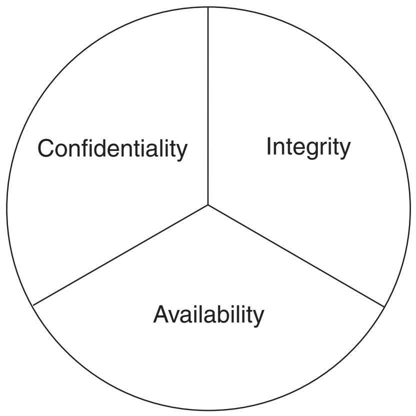


Figure 1.1 Principles of Information Security

There are multiple ways to protect your private data from getting out. The first way is to deny access to it in the first place. But sometimes that is not possible, as in the case of information going over the Internet. In that case, you have to use other tools, such as encryption, to hide and obscure your data during its journey.

Integrity

The integrity factor helps to ensure that information can't be changed or altered by unauthorized individuals. It also means that people who are authorized don't make changes without the proper approval or consent. This can be a subtle distinction. If a bank teller is secretly debiting someone's account and crediting another, that is an integrity problem. They are authorized to make account changes but they didn't have approval to make those ones. Also, data integrity means your data is properly synchronized across all your systems.

Availability

Having your information secure doesn't do you much good if you can't get to it. With denial of service attacks becoming more common, a major part of your info-security goals is not only keeping the bad guys from accessing your information, but making sure the right people can access it. Many computer criminals are just as satisfied to destroy your data or take your Web site offline. The availability element also includes preparing for disasters and being able to recover cleanly when they do occur.

In this example, Tom knew he had to apply each of these principles to completely secure his company's network. He found the software tools that would tackle each area. He was going to need all the help he could get. From the news and trade articles he had read, he knew the chilling statistics.

The State of Computer Crime

Computer crime has become an epidemic that affects every computer user from Fortune 500 CEO to the home user. According to the FBI's annual study on computer crime, conducted in connection with the Computer Security Institute (CSI), over 90 percent of U.S. companies have fallen victim to some form of computer crime. Eighty percent of those surveyed had experienced some financial loss associated with those attacks. Losses of $445 million were attributed to computer crime in 2001, up from $337 million in 2000. And it is certain that many more attacks go unreported. Many companies do not want to publicize that their computer systems were broken into or compromised and therefore avoid going to the authorities because they fear bad publicity could hurt their stock prices or business, especially firms in industries like banking that rely on the public trust.

As the FBI's National Infrastructure Protection Center (NIPC) predicted, computer attacks in 2002 were more frequent and more complex, often exploiting multiple avenues of attack like the Code Red worm did in 2001. They had expected hackers to concentrate on routers, firewalls, and other noncomputer devices as these are less visible and offer fuller access to a corporate LAN if exploited. They had also predicted that the time between the release of a known exploit and tools to take advantage of it would shrink, giving companies less time to respond to a potential threat. Sure enough, the average time from announcement of a security vulnerability and publishing exploit code has dropped from months to weeks. For example, the Blaster worm debuted a mere six weeks after the Microsoft Remote Procedure Call (RPC) vulnerabilities were discovered in early 2003.

The Computer Emergency Response Team (CERT), which is run jointly by Carnegie Mellon University and the federal government, tracks emerging threats and tries to warn companies of newly discovered exploits and security holes. They found that reports of computer security incidents more than doubled in 2001 over the previous year, from 21,756 to 52,658. They have been recording over 100 percent increase in attacks each year since 1998. In 2003, the number of incidents rose 70 percent even though the overall number of new vulnerabilities, defined as weaknesses in hardware or software that allow unauthorized entry or use, dropped (see Figure 1.2). This is due to the emergence of worms that spread quickly across the Internet affecting many systems with a single virus.

This exponential growth in both the number of attacks and the methods for making those attacks is a troubling trend as businesses connect their enterprises to the Internet in record numbers. Unfortunately, many businesses have chosen to stick their heads in the sand and ignore the information security problem. A common excuse for not properly securing their computer network is "Why would a hacker come after my company? We don't have anything they want." In years past, they would have been right. Old-school hackers generally only went after large institutions with data that was valuable to them or someone else.

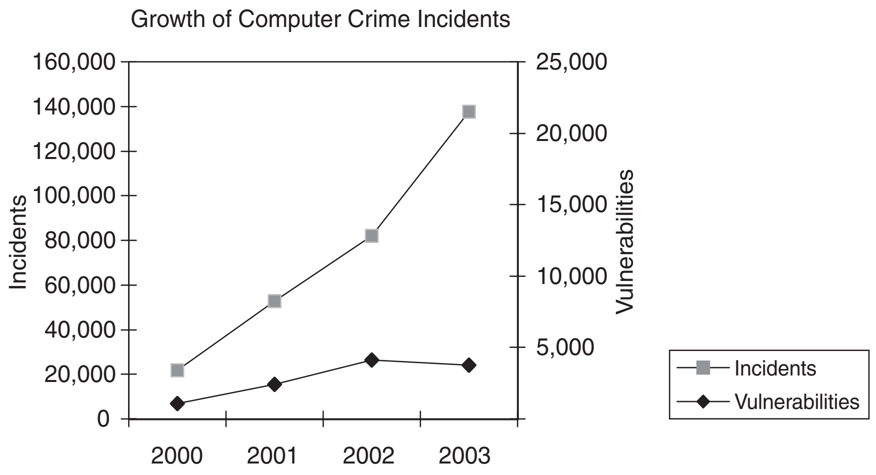


Figure 1.2 CERT Incident and Vulnerability Graph

However, a sea change in the computer security equation has made everyone a target, even small business users. In fact, small- and medium-sized companies now comprise over 50 percent of the attacks reported by the FBI. This change has been caused by several factors, which are described in the following sections.

The Advent of the Internet

When only a few networks were connected to the Internet, companies primarily had to worry about the risk of someone gaining access to a computer console or a virus being introduced by a floppy disk. Protecting against this kind of physical threat is something businesses have been doing for years. Locks on doors, alarm systems, and even armed guards can protect the computers and systems from physical access. Anti-virus software and passwords served as the only necessary technical security precaution for firms in the pre–World Wide Web age.

With the Internet, hackers can attack from thousands of miles away and steal critical company assets, bypassing any and all physical barriers. They can then sink back into the anonymity that the Internet provides. They can come from foreign countries with no extradition treaties with the United States. They leave few clues as to who they are or even what they did. When you are connected to the Internet, you are literally no more than a few keystrokes away from every hacker, cracker, and ne'er-do-well on the network. Password protection and anti-virus software is not enough to keep intruders out of your virtual office.

Ubiquitous, Inexpensive Broadband

Not too long ago, dedicated Internet connections were the sole domain of large companies, educational institutions, and the government. Now, you can get DSL or cable modem access for your business or home use for less than $100 per month. Companies are getting online by the thousands, and this is a good thing overall for business. However, having a dedicated connection exposes them to more risk than their previous dial-up or private line connections. First of all, broadband is quite different from just dialing up via a modem from a network standpoint. Usually when you dial up, you are connected only while you are using it. With always-on broadband, hackers can work away, trying to get in, taking as much time as they need. They especially like working during the late night hours, when system administrators who might notice something awry have gone home.

Having access to a site with dedicated broadband access is very attractive to hackers. They can use that bandwidth and leverage it to attack other sites. If a hacker's goal is to take down a hugely popular site like Yahoo or Amazon by sheer brute force, they need a lot of bandwidth. Most of these sites have bandwidth that is measured in gigabits, not megabits. In order to flood those sites, they need a huge bandwidth pipe, which the average hacker can't afford. However, if they break into other machines on the Internet with broadband connections, they can use these machines to attack their real target. If they can "own" enough sites, they suddenly have a very big gun to wield. This is known as a **distributed denial of service** (DDOS) attack. It has the added benefit of throwing the authorities off their trail because all of the attacks are coming from unsuspecting victims, rather than the attackers themselves. These victim machines are known as **zombies**, and hackers have special software they can load to make these computers or servers “awake” on special commands that only they can issue. These programs are often very hard to find and eradicate because the host computer shows no ill effects while the zombie software is dormant. The one thing that the hacker hordes want is your bandwidth; they could generally care less who you are.

Another reason hackers want to break into machines is to store their tools and other ill-gotten loot. These exploited machines are called **storage lockers** by the hackers, who often traffic in illicit files. The files might be pornography, pirated software or movies, or other hacker tools. Rather than store these on their own machines, where they might be found and used against them in court, they prefer to hide them on unsuspecting victim’s servers. A broadband connection is nice because they have lots of bandwidth for uploading and downloading files. A small company is even better because it is likely they don’t have a large IT staff monitoring their Internet connection and probably don’t have very sophisticated security measures in place. They can give the hacked server IP address out to their buddies and use them for informal swap meets. Again, these kinds of intrusions are hard to find because the computer acts normally, although you might notice a slowdown in performance or download speeds while it is being used for these unauthorized activities.

Attack of the Script Kiddies

Another thing that has changed the targets for computer crime is simply a rise in the number of participants, especially at the low end of expertise. These hacker novices are called **Script Kiddies** because they often use point-and-click hacking tools or “scripts” found on the Web rather than their own knowledge. Hackers used to be part of an elite community of highly skilled (albeit morally challenged) individuals who were proficient in writing code and understood computers at their most fundamental level. They even had an informal Hacker Ethics code, which, although eschewing the idea of privacy, stated that no harm should be done to computers invaded. The hacker experience was primarily about learning and exploring. However, that community soon splintered and was watered down by newcomers. Now one can find hundreds of Web sites that can teach you how to hack in a matter of minutes. Many so-called hackers are teenagers with little knowledge of coding. Rather than seeking knowledge, they are intent on joyriding hacked computers, bragging rights, and outright vandalism. And with the influx of new bodies to the hacking community, like any thief or criminal, they look for the easiest “mark.” These inexperienced criminals attack the systems of smaller companies, those with fewer defenses and less-experienced administrators who are not as likely to notice their neophyte mistakes. Most of them wouldn’t dare taking on the Pentagon or the CIA’s computers, which have impressive digital defenses and significant prosecutorial powers. Few small companies can afford to investigate, much less prosecute, a computer intrusion even if they do notice it. And since most Script Kiddies’ main goal is not learning but mischief, they often cause more damage than an experienced computer criminal would.

Worms, Auto-rooters, and Other Malware

Finally, a major reason that the fundamental computer security scene has changed is that much hacking nowadays is automated and random. Script kiddies can use tools that scan IP addresses at random to look for weak or exploitable machines. They will often let these programs run all night, harvesting potential victims for them. There are packages, called **auto-rooters**, that gain "root" or admin privileges on a machine. These tools not only do the reconnaissance for them, but also actually carry out the act of breaking into the machine and placing their Trojan horse or other malicious software (**malware**) in place. The result is that with a single click of a mouse, someone with no more computer experience than a six-year old can "own" dozens of machines in a single evening.

With the advent of Internet worms like Nimda in 2001, even the human element has been taken out of the picture. These autonomous cousins to the computer virus roam the Internet, looking for computers with a certain set of security holes. When they find one, they insert themselves into that computer, perform whatever function they were programmed to do, and then set that machine up to search for more victims. These automated hacking machines have infected far more networks than have human troublemakers. They also spread incredibly fast. It is estimated that the Code Red worm spread to over 300,000 servers within a few days of its release.

Info-Security Business Risks

So it's clear that the playing field has changed. Before, few small companies really had to worry about their data security; now firms of all sizes are forced to spend time and money to worry about it—or risk the consequences. What are these risks? Few companies stop to think about all the possible risks that they are exposed to from an information security standpoint. You should understand all these risks, recognize which ones apply to your organization, and know what the value or dollar cost of each one is. This will help you make a business case for better computer security and justify the expenditures you need.

Data Loss

While computer viruses have kept this threat current since the 1980s, few managers stop to think what it would really cost them to lose part or all of their data. Without proper backups, which many small firms lack, the loss of critical data can be catastrophic. Years of accounting, payroll, or customer data can be wiped out. Orders can be lost. If the data belongs to customers, the company could be liable for its loss. Certain professions, such as legal or accounting, can be subject to regulatory fines or punishment for loss of such data. And this doesn't include the loss of business and productivity while employees restore the data or have to revert to paper records. Even when they have backups, the time and hassle involved to get systems back up and running is considerable. The bottom line is that few businesses can survive long without their computerized records and systems. Does your company have a written Disaster Recovery Plan that covers data and systems? If not, you could be in for a nasty surprise in the event of an unexpected outage.

Denial of Service

Many of today's hackers are more high-tech vandals than computer geniuses. They take joy in knocking down servers or denying service for any reason, and sometimes for no reason at all. Often the denial of service is accidental or incidental to the hacker's real goal. The Code Red and Nimda worms brought many networks to their knees just from trying to respond to all the attempts at infection. With the reliance of today's business on the Internet, this can be like shutting off the electricity. E-mail communication comes to a halt. A company Web site might go down. For a company that does a considerable amount of business over the Internet, this could mean a total stoppage of work.

How many companies know the hourly or daily cost to their business of a loss of Internet access? In certain industries or companies, it is very large due to their reliance on information technology. Few companies these days are without some dependence on Internet access. Depending on how much the business relies on the Internet, a denial of service attack can either be a minor annoyance or a major blow to a company's business. Try calculating the cost for your company based on the number of employees unable to work, the number of orders processed online, and so on.

Embarrassment/Loss of Customers

Being offline can make a company look very bad. Not being able to communicate via e-mail or missing critical messages can be embarrassing at best. If their Web site is offline, customers will immediately begin asking questions. For public companies, it could mean a loss of stock value if the news gets out. Witness the drop in stock prices of Yahoo and Amazon after well-publicized denial of service attacks. Millions or even hundreds of millions of dollars of stockholder value can disappear in an instant. For businesses like financial intuitions or e-commerce companies that depend on people feeling safe about putting their financial information online, a single Web defacement can wipe out years of goodwill. CD Universe, an online CD retailer who had their credit card database stolen, never recovered from that attack. Cloud Nine Communications, an ISP in England, was down for a week due to a concerted and lengthy denial of service attack and eventually had to close its doors. There are now gangs of hackers who go on mass Web site defacement binges, sometimes hitting hundreds of sites per night. The admission to these hacker clubs is racking up a certain number of Web site defacements. Do you want your Web site to become a notch on their scorecard?

Liability

In this litigious age, making a small mistake can result in a lawsuit costing millions. Imagine the results if your entire customer database is stolen and then traded on the Internet. Class action suits have resulted from such events. With the huge rise in identity theft, laws are being passed that require companies to exercise the proper standard of care when dealing with a customer's personal or financial data. One industry that has been particularly affected by legislation is healthcare. The Health Insurance Portability and Accountability Act of 1996 (HIPAA) requires any company dealing with patient information to properly secure that data from unauthorized use. The privacy provisions of the act affecting computer networks went into effect in 2003. There are civil and criminal penalties for violators, so it is no longer just a money issue. Executives and managers could go to jail if found in violation.

Also, hackers are always looking for unsecured computers to launch their distributed denial of service attacks from. If your company's computers are used in such an attack and victims can't find the original perpetrator, they might come after you, charging that you were negligent in securing your network. After all, companies tend to have deeper pockets than most hackers.

Another area to be concerned about is liability for copyright violations. Copying of pirated movies, music, and software over the Internet has reached a fever pitch. Media companies are fed up and are starting to go after violators directly by tracking down the IP addresses of the downloaders and sending lawyers after them. InternetMovies.com, a Hawaii-based Web site, had their ISP service disconnected when their ISP was served with a lawsuit for alleged pirated files found on their network. Pirates who want to distribute their wares are resorting to storing them on third-party computers, often compromised servers on corporate networks. If your company is unknowingly running one of these servers or has such files stored on it, you could be disconnected from the Internet, liable for fines, or sued. Stories like these can often help you persuade reluctant executives to implement stricter personnel policies when it comes to information security, such as banning file sharing software or implementing stronger password requirements.

Disclosure of Corporate Secrets and Data

It is hard to put a dollar value on this risk because it varies from firm to firm. For example, the value of the recipe for Coca-Cola or Colonel Sander's fried chicken could reach into the billions. At a smaller company, detailed plans for a proprietary device or formula may be invaluable. In some cases, much of the value of the company may be locked up in this important data. For example, a biotech company may have their research for their latest gene patents on their corporate network.

Customer lists are always valuable to competitors, especially in very competitive markets. Hewlett-Packard was served with a shareholder lawsuit after sensitive discussions between their executives were released to the public during a contentious merger.

However, even at companies where there are no secret plans or recipes, this risk exists. For instance, think of the damage of releasing the corporate payroll file to the rank-and-file workers. This happens all the time, usually due to snoopy or vindictive employees. The discord and subsequent loss of morale and perhaps employee exodus due to being disgruntled over pay differences can be huge. Often, all this could be avoided if the system administrator had simply secured the system properly.

Tampering with Records

Sometimes an intruder is not intent on stealing or destroying data but rather just making changes to existing records, hopefully without being detected. This can be one of the most difficult kinds of computer crime to detect because the systems keep functioning just as they were before. There is no system crash or performance drain to point to an intrusion. There is no defaced Web site to raise an alarm. Obviously, for banks and government agencies, this can be a very serious problem. But every company has to worry about someone getting into the payroll system and changing pay amounts. Schools and universities have to deal with students trying to change grades. Often it is up to the accounting auditors to find evidence of foul play. However, with the right system security, these problems can be avoided up front.

Loss of Productivity

This is a much more subtle risk and often very hard to avoid. It can range from bandwidth being used by employees to download music or movies, thereby slowing down other workers, to employees surfing objectionable or nonwork Web sites. While these are employee policy issues, the system administrator is often called on to fix them with technology such as content filters and firewalls. And many of these unauthorized programs, such as Napster, Kazaa, and instant messengers, in addition to being productivity drainers, can create security holes in a company's network defenses.

Given all these risks, you would think that companies would be falling over themselves to put the proper protections in place. Yes, the largest companies have implemented significant defenses, but most small- and medium-sized companies have little in the way of network security. At best, a company will install a firewall and anti-virus software and consider that enough to protect them. Unfortunately, it is often not enough.

A whole industry has sprung up to offer solutions to these problems. There are commercial hardware and software solutions such as firewalls, intrusion detection systems, and vulnerability scanners. However, most of these products are priced so high that only larger firms can afford them. A simple firewall costs several thousands of dollars. Commercial intrusion detection systems and vulnerability testing solutions can run into the tens of thousands or more. In addition to the up-front costs, there are often yearly maintenance fees to support the software. And many of the software solutions require high-end computers to run on. They also often require pricey database software such as Oracle for reporting features. Given these costs, proper computer security is often seemingly out of reach for the small- and medium-sized firms. And as you have seen, the risk is just as great for these businesses as the Fortune 500, and perhaps even more so, since their financial resources to withstand such an attack will be much more limited than a large firm.

So what's a harried, overworked, underfunded system administrator to do? Well, there is a solution that can provide companies with quality computer security for little or no cost: *open source software*.

Open Source History

The open source software movement has its roots in the birth of the UNIX platform, which is why many people associate open source with UNIX and Linux systems, even though the concept has spread to just about every other computer operating system available. UNIX was invented by Bell Labs, which was then the research division of AT&T. AT&T subsequently licensed the software to universities. Because AT&T was regulated, it wasn't able to go into business selling UNIX, so it gave the universities the source code to the operating system, which was not normally done with commercial software. This was an afterthought, since AT&T didn't really think there was much commercial value to it at the time.

Universities, being the breeding grounds for creative thought, immediately set about making their own additions and modifications to the original AT&T code. Some made only minor changes. Others, such as the University of California at Berkley, made so many modifications that they created a whole new branch of code. Soon the UNIX camp was split into two: the AT&T, or System V, code base used by many mainframe and minicomputer manufacturers, and the BSD code base, which spawned many of the BSD-based open source UNIX versions we have today. Linux was originally based on MINIX, a PC-based UNIX, which has System V roots.

The early open sourcers also had a philosophical split in the ranks. A programmer named Richard Stallman founded the Free Software Foundation (FSF), which advocated that all software should be open source. He developed a special license to provide for this called the General Public License (GPL). It offers authors some protection of their material from commercial exploitation, but still provides for the free transfer of the source code. Berkley had developed its own open source license earlier, the BSD license, which is less restrictive than the GPL and is used by the many BSD UNIX variants in the open source world.

These two licenses allowed programmers to fearlessly develop for the new UNIX platforms without worry of legal woes or having their work being used by another for commercial gain. This brought about the development of many of the applications that we use today on the Internet, as well as the underlying tools you don't hear as much about, such as the C++ compiler, Gcc, and many programming and scripting languages such as Python, Awk, Sed, Expect, and so on.

However, open source didn't really get its boost until the Internet came to prominence in the early 1990s. Before then, developers had to rely on dial-up networks and Bulletin Board Systems (BBSs) to communicate and transfer files back and forth. Networks such as USENET and DALnet sprung up to facilitate these many specialized forums. However, it was difficult and expensive to use these networks, and they often didn't cross international boundaries because of the high costs of dialing up to the BBSs.

The rise of the Internet changed all that. The combination of low-cost global communications and the ease of accessing information through Web pages caused a renaissance of innovation and development in the open source world. Now programmers could collaborate instantly and put up Web sites detailing their work that anyone in the world could easily find using search engines. Projects working on parallel paths merged their resources and combined forces. Other splinter groups spun off from larger ones, confident that they could now find support for their endeavors.

Linux Enters the Scene

It was from this fertile field that open source's largest success to date grew. Linus Torvalds was a struggling Finnish college student who had a knack for fiddling with his PC. He wanted to run a version of UNIX on it since that is what he used at the university. He bought MINIX, which was a simplified PC version of the UNIX operating system. He was frustrated by the limitations in MINIX, particularly in the area of terminal emulation, since he needed to connect to the school to do his work. So what became the fastest growing operating system in history started out as a project to create a terminal emulation program for his PC.

By the time he finished with his program and posted it to some USENET news groups, people began suggesting add-ons and improvements. At that point, the nucleus of what is today a multinational effort, thousands of people strong, was formed. Within six months he had a bare-bones operating system. It didn't do much, but with dozens of programmers contributing to the body of code, it didn't take long for this "science project" to turn into what we know as the open source operating system called Linux.

Linux is a testament to all that is good about open source. It starts with someone wanting to improve on something that already exists or create something totally new. If it is any good, momentum picks up and pretty soon you have something that would take a commercial company years and millions of dollars to create. Yet it didn't cost a dime (unless you count the thousands of hours invested). Because of this, it can be offered free of charge. This allows it to spread even farther and attract even more developers. And the cycle continues. It is a true meritocracy, where only the good code and good programs survive.

However, this is not to say that there is no commercial motive or opportunity in open source. Linus himself has made quite a bit of money by his efforts, though he would be the first to tell you that was never his intention. Many companies have sprung up around Linux to either support it or to build hardware or software around it. RedHat and Turbo Linux are just a few of the companies that have significant revenues and market values (albeit down from their late 1990s heights). Even companies that were known as proprietary software powerhouses, such as IBM, have embraced Linux as a way to sell more of their hardware and services.

This is not to say that all software should be free or open source, although some of the more radical elements in the open source world would argue otherwise. There is room for proprietary, closed source software and always will be. But open source continues to gain momentum and support. Eventually it may represent a majority of the installed base of software. It offers an alternative to the commercial vendors and forces them to continue to innovate and offer real value for what they charge. After all, if there is an open source program that does for free what your commercial program does, you have to make your support worth the money you charge.

Open Source Advantages

You and your company can use open source both to cut costs and improve your security. The following sections touch on the myriad of reasons why open source security tools might make sense for you and your company.

Cost

It's hard to beat free! Although open source does not necessarily always mean free, most open source software is available at no charge. The most common open source license is the GNU GPL license, which is a free software license. Other open source software might be shareware or even charge up front, like the commercial servers available from RedHat. But either way, open source is usually available for a fraction of the cost of commercial alternatives. This helps greatly in justifying new security projects within your company. When all that is needed is a little of your time and maybe a machine to run the software, it is a lot easier to get approval for a new solution. In fact, depending on your authority level, you may be able to go ahead and implement it without having to make a business case for it. If you want to take it a step further, after successful installation, you can bring the results to your boss and demonstrate that you saved the company thousands of dollars while making the network more secure (and that may improve your job security!).

Extendability

By definition, open source software is modifiable and extendable, assuming you have the programming skills. Many open source programs have scripting languages built in so that you can write small add-on modules for them without having to be a programming guru. Nessus, the open source vulnerability scanner does this with their NASL scripting language (this is demonstrated later in this book, and you'll learn how to write some custom security tests too). Snort, the open source intrusion detection system mentioned earlier, lets you write your own alert definitions. This means that if there is something specific to your company that you need to test for, you can easily write a custom script to look for it. For example, if you have a database file called customer.mdb that is specific to your company and that should only be used by certain departments, you could write a Snort rule that looks for that file traversing the network and alerts you.

And of course if you are a real programming guru, you can get involved in contributing to the core code and gain both valuable experience and recognition within the open source community. This could also be helpful in terms of your job marketability.

Security

There are some people, mostly those involved with commercial software concerns, who advocate that closed source software is inherently more secure since hackers do not have the internal workings of the software easily available to them. This school of thought relies on the security premise of obfuscation—keeping the design of your product secret. However, this logic breaks down when you look at the facts. Windows is the largest proprietary software product in the world, yet the number of security holes announced in the Windows platforms is about the same as those found in Linux and other open source platforms. The truth is that whether the source code is open or closed doesn't make programmers write more secure programs.

Independence

Discovery and remediation of security issues in software can be much faster with open source programs. Commercial companies often have strong monetary motivations for not admitting to security flaws in their products. Multiple security holes found in a product, especially if it is a security product, could hurt sales to new customers. If it is a publicly traded company, the stock price could fall. Additionally, developing security patches and distributing them to customers are expensive endeavors, ones that usually don't generate any revenue. So getting a company to confirm a security issue with its software can be a major effort. This means days or weeks can go by while customer systems are still vulnerable. Frustration with this process has prompted some security researchers to adopt a policy of releasing new security vulnerabilities directly to the public rather than privately to the company.

Once a security hole is known to the public, a company will often go through a complicated development and testing process before releasing a patch to the public, ensuring that there aren't any liability issues and that the patch can be released for all platforms at once. So more time may go by while you have a known security hole that hackers can exploit.

Open source software projects have no such limitations. Security patches are usually available within hours or days, not weeks. And of course you don't have to wait for an official patch; if you understand the code well enough, you can write your own or design a workaround while you wait for one.

The general thinking in the open source community is that the best overall security comes from a critical review by a large body of people who don't have a vested interest in not finding any holes. This is the same measure of quality that cryptographic researchers apply to their work. The open source concept, while not guarantying that you will get more secure software, means you don't have to take a company's word that a product is secure, and then wait for them to come up with a solution for any security holes.

User Support

Commercial software products usually have support lines and a formal channel to go through for help. One of the main reasons many people shy away from open source solutions is that they feel like they have to pay for a product to get decent support. However, the support you often get for your money is not that great. If the software company is small, you might have to wait hours or days for a return call. If the vendor is large, you will probably be shunted into a call queue. When you finally get connected, it will be with an entry-level technical person who can't do much more than enter your problem into a knowledge base to see if anyone has had the problem before and then parrot back a generic solution. Usually you have to get to a level two or three technician before you get someone who truly understands the product and can help you with complicated problems. Not to mention that companies don't like to admit their products have bugs; they will tend to blame it on everything else beside their product (your operating system, your hardware, and so on).

Add to that, many companies are now charging separately for support. The price you pay over several years for support of the software can exceed the initial purchase price of it. These charges create a nice steady stream of revenue for the company even if you never upgrade. Most software companies, if they aren't already doing it, are moving in this direction. Toll-free numbers for software technical support are becoming a thing of the past.

Open source products often have terrific support networks, albeit somewhat non-traditional. Open source support is less organized but often more helpful and more robust. There will rarely be a phone number to call, but there are usually several options to get answers on the software. On a smaller project, it might be as simple as e-mailing the developer directly. The larger packages usually have a mailing list you can post questions to. Many have several different lists depending on your question (user, developer, specific modules, or platforms). Many now have chat rooms or IRC channels where you can ask questions, ask for new features, or just sound off in real time.

The neat thing is that you are usually talking to people who are very familiar with the software, possibly even the actual developers. You can even ask them for new features or comment on recently added ones. You will end up talking to some of the brightest and most experienced people in the industry. I've learned a lot by just following the conversations on the mailing lists.

Most questions I've posed to these lists have been answered in a few hours or less. The answers are usually insightful and informative (and sometimes witty). You will often get several different opinions or solutions to your problem, all of which may be right! Besides getting very detailed answers to your questions, you can talk about the state of the art in that particular area or engage in philosophical debates about future versions, and so forth (if you have a lot of extra time on your hands). And of course, if you are knowledgeable about the software, you are free to chime in with your own answers to questions.

Keep in mind that these folks usually aren't employees of a company producing the software and might sometimes seem a bit harsh or rude. Asking simple questions that are answered fully in the INSTALL pages or in a FAQ might earn you a rebuke. But it will also usually get you the answer or at least a pointer to where you can find it. Sometimes the flame wars on the lists crowd out the real information. However, I'll take impassioned debate over mindless responses any day.

Finally, if you really do feel like you have to pay for support, there are companies that do just that for open source platforms. Numerous Linux companies offer supported versions of that open source operating system. Many of the more popular applications also have companies providing support for them. You can buy a prepackaged Snort IDS box from several companies that will support you and provide regular updates. This way you can have the same vaulted support that commercial products offer but still keep all the benefits of an open source platform.

Product Life Span

With commercial software, you are at the mercy of the corporation that owns the product you select. If it's a large company like Microsoft, then you are probably in good shape. However, even Microsoft has tried to get into market segments and then decided they wanted out and dropped product lines. Smaller companies could go out of business or get bought or merged. In this day and age, it is happening more and more. If the company that buys them has competing products, more than likely they will get rid of one of the lines. If they decide to drop your product, then you are out of luck for future support. With a closed source product, you have no way of asking any questions or making any necessary upgrades to it once the company decides they don't want to play anymore.

Open source projects never die a final death. That's not to say that they don't go dormant. Projects go by the wayside all the time as the participants graduate or move on to a new stage of life. This is more prevalent in the smaller programs and tools. The larger ones (which comprise the majority of programs mentioned in this book) always have someone willing to step up and grab the reins. In fact, there are sometimes power struggles in the hierarchy for control of a project. However, if someone doesn't like the direction it is going, there is nothing to stop him or her from branching off and taking the product where he or she wants it to go. Even in the smaller ones, where there is a single developer who might not be actively developing it anymore, you can simply pick up where they left off. And if you need to fix something or add a feature, the code is wide open to let you do that. With open source software, you are never at the mercy of the whims of the market or a company's financial goals.

Education

If you want to learn about how security software works or polish your programming skills, open source software is a great way to do it. The cost is low, so you don't have to worry about dropping a couple of thousand dollars on training or programs. If you are doing this yourself, all you need is a machine to run it on and an Internet connection to download the software (or the CD-ROM included with this book). If you are doing it for a company, it is the cheapest training course your company will ever approve. Plus, your company has the added benefit that you will be able to use the technology to improve the company's computer security without spending a lot of money. Talk about a win-win situation!

Of course, budding programmers love open source software because they can get right into the guts of the program and see how it works. The best way to learn something is to do it, and open source software offers you the ability to see all the code, which is usually fairly well documented. You can change things, add new features, and extend the base code—all things that are impossible with closed source software. The most you can ever be with a closed source program is an experienced user; with open source, you can be an innovator and creator if you want.

The mailing lists and chat rooms for open source projects are excellent places to ask questions and make friends with people who can really mentor your career. Getting involved with an open source project is probably the quickest way to learn about how software is developed. Which leads into my next point.

Reputation

After you've cut your teeth, gotten flamed a few times, and become a regular contributing member of an open source package, you will notice that you are now the go-to guy for all the newbies. Building a reputation in the open source world looks great on a resume. Being able to say you were integrally involved in the development of an open source product speaks volumes about your dedication and organization skills, not to mention your programming skills. Designing an open source software package makes for a great graduate research project. And of course, once you get good enough, you may end up producing your own open source software and building quite a following. More than a few authors of open source software have gone on to parley their user base into a real company making real money. So whether your efforts in open source are just a hobby, as most are, or become your sole aim in life, it can be very rewarding and a lot of fun.

When Open Source May Not Fit Your Needs

I've said a lot about how great open source software is. You'd think it was going to solve all the world's problems with the way I have gone on about it. However, there are instances when it is just not appropriate. There aren't many of them, but here they are.

Security Software Company

If you work for a company that is designing proprietary, closed source security software, then open source software is not appropriate as a base of code to start from. This is not to say you can't play around with open source software to get ideas and learn the art, but be very careful about including any code from an open source project. It could violate the open source licenses and invalidate your work for your company. If your company can work with the license that's included with the open source software, then you may be okay. Also, some companies are beginning to open source some part of their software. These "hybrid" licenses are becoming more common. If you do decide to do this, you will want to make sure you clearly understand the open source license and have your legal department research it thoroughly.

This doesn't mean that you can't use open source software within your company. If you are a network administrator, you can use an open source firewall, for example. Many closed source software companies do this, as hypocritical as it sounds. You just can't use the code to create a product that won't be open sourced.

100 Percent Outsourced IT

Another case where open source may not fit is if your IT department is not technically capable of handling program installations, compilations, and so on. While most open source software is fairly easy to use, it does require a certain level of expertise. If your IT department consists of the administrative assistant who does it in his or her spare time, or you outsource your entire IT department, then it probably doesn't make sense, unless your contractor has experience in that area.

Restrictive Corporate IT Standards

Finally, you may be faced with corporate standards that either require you to use specific vendors or outright forbid open source. This is becoming less and less common as companies are realizing that locking into a single vendor is silly. Ignored for a long time by the big boys, open source is coming on strong in corporate America. Companies like IBM, once the champion of closed source and proprietary products, are embracing and even promoting open source. The old adage of "no one ever got fired for buying (insert blue-chip vendor of choice)" is no longer valid in most companies. An updated version of the proverb might be "no one ever got fired for saving the company money with a solution that worked." Certainly, however, going out on limb with a new concept can be more risky than the status quo.

Windows and Open Source

It used to be that open source software was primarily developed only for UNIX-based operating systems. Many developers consider Windows and the company behind it as being the antithesis of what open source software stands for. And the company hasn't denied the charge; in fact, Microsoft has commissioned studies that show open source in a bad light, and heavily markets against the Linux operating system, which is starting to encroach on its market share in the server arena. However, no matter what the Microsoft attitude is towards the concept, Windows users have been busy creating programs for it and releasing them as open source. There are ports of most of the major tools in the UNIX and Linux world for Windows. These programs are sometimes not full versions of their UNIX brethren, but there are also open source programs that are released only on the Windows platform, such as the wireless sniffer NetStumbler that is reviewed in Chapter 10.

Many times, technical personnel will be limited in what operating systems they can run on their company's LAN. Even if they have carte blanche, they may just not be able to dedicate the time to loading and learning one of the open source operating systems I recommend in the next chapter. So for each area mentioned in this book, I try to present both a UNIX and a Windows option (they are often the same program). Like it or not, Windows is the dominant operating system on most desktops, and ignoring this would be doing a disservice to a large body of technical professionals who could benefit from open source software.

Open Source Licenses

Many people assume that open source means software free of all restrictions. Indeed, in many cases there is no charge for the software. However, almost all open source software is covered by a license that you must agree to when using the software, just as you do when using a commercial product. Generally this license is much less restrictive than a traditional closed source license; nonetheless, it does put limits on what you can do with the software. Without these limits, no programmer would feel safe releasing the results of his or her hard work into the public domain. When using open source software, make sure you are in accordance with the license. Also be sure that any modifications or changes you make also comply. This is the important part: If your company spends a lot of time customizing an open source program for its own use, you should be aware that you will have some responsibilities under the open source license.

There are two main types of open source licenses: the GNU General Public License and the BSD license. As long as you understand them thoroughly, you should be able to confidently use most open source software without fear of running afoul of any copyright issues. There are some unusual open source licenses coming out for things like artwork created in games and so forth. These “hybrid” licenses are a little murkier to deal with, and you should definitely be careful when using them, because you could be incurring charges or be in violation of their copyright without knowing it.

The goal of both major open source licenses is not so much to protect the existing software, but to control the uses of derivative code from that software. After all, it is usually free and the original developer shouldn’t care if you make a million copies of it and distribute them to your friends. It’s when you start making changes to the software and want to distribute it that you have to be careful. The two major open source licenses and their similarities and differences are described next.

The GNU General Public License

The GNU General Public License (GPL) is probably the more commonly used open source license. It is championed by the Free Software Foundation, which promotes the creation and proliferation of free software using this license. The actual GNU project works on certain specific software projects and puts their stamp of approval on them. These projects are usually core tools and libraries, such as the Gcc compiler and other major works. Anyone can use the GPL license for software as long as you use it verbatim and without changes or additions. Many developers use it because it has been vetted by a team of lawyers and has withstood the test of time. It is so common that if someone says that something is “GPL’d,” generally people understand that to mean that it has been released open source under the GPL license.

The GPL is more complicated than the other major open source license, the BSD license. It has a few more restrictions on the use of the code by the licensee, which makes it more appropriate for companies that are making a commercial product. Generally, if you are licensing something under the GPL, it is understood that it is free software. A vendor, however, may charge for packaging, distribution, and support. This is the area that a lot of companies make money from what is supposedly a free package. Witness the retail packages of various flavors of Linux and commercial versions of the Apache Web servers and Sendmail communication package. However, if you download or load from a CD-ROM something that is covered under the GPL and didn't put a credit card number in somewhere, you can reasonably assume that you don't owe anyone any money for it.

The real beauty of the GPL from a developer's standpoint is that it allows the original author of the program to maintain the copyright and some rights while releasing it for free to the maximum number of people. It also allows for future development, without worry that the original developer could end up competing against a proprietary version of his or her own program.

In its basic form, the GPL allows you to use and distribute the program as much as you want with the following limitations.

- If you distribute the work, you must include the original author's copyright and the GPL in its entirety. This is so that any future users of your distributions fully understand their rights and responsibilities under the GPL.
- You must always make a version of the source code of the program available when you distribute it. You can also distribute binaries, but you must also make the source code easily available. This gets back to the goal of the open source concept. If all that is floating around is the binaries of a free program and you have to track down the original designer to get access to the source, the power of free software is greatly diminished. This ensures that every recipient of the software will have the full benefit of being able to see the source code.
- If you make any changes to the program and release or distribute it, you must also make available the source code of those modifications in the same manner as the original code, that is, freely available and under the GPL. The key phrase here is "and release or distribute it." If you don't release it, then you are not obligated to release the source code. If you are making custom changes to the code for your company, they might be worried about giving out the results of your efforts. As long as you don't release it publicly or intend to sell it, it can remain proprietary.

  However, it usually makes good sense to go ahead and release the new code with the GPL. This not only generates lots of good will with the open source community, but it will also ensure that your changes are compatible with future versions of the program and are fully tested. You can use this logic to convince your company that they can get the experience and free labor of all the other programmers on the project by doing this. It will generally not hurt a company competitively to release this kind of code unless that program is part of the core business of the company, in which case open source software may not make sense anyway. And finally, it won't hurt your reputation and leverage with the other developers on the project and elsewhere in the software community.

Appendix A has the entire text of the GPL. You can get it in different text formats from www.gnu.org/licenses/gpl.html.

The BSD License

The BSD license is the open source license under which the original University of California at Berkley version of UNIX was released. After they won their lawsuit with AT&T over the original license, they released the software into the public domain with the permissive BSD license. The primary difference from the GPL is that the BSD license does not include the requirement of releasing future modifications under the same license. Based on this, several companies went on to release commercial versions of UNIX based on the BSD code base. BSDI is one such company. Some say that this goes against the idea of open source, when a company can take an improved version and charge for it, while others feel that it encourages innovation by giving a commercial incentive. Either way, it spawned a whole family of UNIX versions, including FreeBSD, NetBSD, and OpenBSD, from the free side of the house, and others such as BSDi on the commercial side. Appendix A has the full text of the BSD license. You can also access it at www.opensource.org/licenses/bsd-license.php.

Now that you understand the background of info-security and open source software, we are going to get into the specifics: installing, configuring, and using actual software packages. The following chapters review programs that can help you secure your network and information in a variety of ways. The chapters are loosely organized into different info-security subjects, and most of the most major areas of information security are covered. Also, many tools can have multiple uses. For example, even though Snort is covered in the chapter on intrusion detection systems, it can be used in forensic work too. And certainly if your interest is in a tool for particular area, you can skip right to that section.

CHAPTER 2

Operating System Tools

Most of the tools described in this book are application programs. As such, they require an underlying operating system to run on. If you think of these programs as your information security toolkit, then your operating system is your workbench. If your OS is unstable, your security work will suffer; you will never be able to truly trust the data coming from it. In fact, your OS might introduce even more insecurity into your network than you started with. In computer security jargon, having a secure OS to build on is part of what is known as a **Trusted Computing Base** (TCB). The TCB consists of the entire list of elements that provides security, the operating system, the programs, the network hardware, the physical protections, and even procedures. An important base of that pyramid is the operating system. Without that, you are building your Trusted Computing Base on quicksand.

Chapter Overview

Concepts you will learn:

- Introduction to Trusted Computing Base
- Guidelines for setting up your security tool system
- Operating system hardening
- Basic use of operating system-level tools

Tools you will use:

Bastille Linux, ping, traceroute, whois, dig, finger, ps, OpenSSH, and Sam Spade for Windows

Many attacks on computers are directed at the operating system. Modern operating systems have ballooned to such size that it is extremely difficult for any one person to completely understand what is going on “under the hood.” XP, the most current version of Windows, contains over 50 million lines of code. While it is supposed to be the most secure version of Windows yet (according to Microsoft), new security bugs are found in it almost daily. The more complexity you add to a product, the more likely it is to give unexpected results when given unexpected input. Hackers count on these unexpected results.

It used to be that a computer had a limited number of possible inputs—the application programs that were either designed by or approved by the computer vendor. Now, with the Internet and Java- and Active X-enabled Web browsers, all kinds of traffic and code can come at a computer that the designers never allowed for. The sheer volume of programs combined with the types of traffic coming from the Internet means that operating systems are getting less secure, not more secure, as times goes on, especially when you use them “straight out of the box.”

Add to this vendors’ tendency to try to make computers as ready as possible so users can simply “plug and play.” While some might argue that this is a good thing for the masses of computer illiterates, it is certainly not a good thing from a security standpoint. Most security features are turned off by default, many programs and services are loaded automatically, whether the user will need them or not, and many “extras” are thrown onto the system in an effort to outdo the competition. While Microsoft Windows has been the worst offender in this area, consumer versions of Linux aren’t much better, and even server-level operating systems are guilty of this sin. A standard installation of RedHat Linux still loads far too many services and programs than the average user needs or wants. Windows Small Business Server 2000 loads a Web server by default. And while Windows XP improved on the past policy of “everything wide open,” there are still insecurities in the product when using the default installation.

Making sure your security tool system is secure is important for several reasons. First of all, if a front-line security device such as a firewall is breached, you could lose the protection that the firewall is supposed to provide. If it’s a notification device, for example, an intrusion detection system, then potential intruders could invade the box and shut off your early warning system. Or worse yet, they could alter the data so that records of their activities are not kept. This would give you a false sense of security while allowing the intruders free reign of your network.

There are hacker programs designed to do just this. They alter certain system files so that any data coming out of the machine can be under the control of the hacker. Any computer that has been infected with one of these programs can never be trusted. It is often more cost effective to reformat the drive and start over.

Finally, if unauthorized users commandeer your security box, they could use the very security tools you are using against you and other networks. An Internet-connected machine with these tools loaded could be very valuable to someone intent on mischief.

Ensuring that the base operating system of your security machine is secure is the first thing you should do, before you load any tools or install additional programs. Ideally, you should build your security tool system from scratch, installing a brand new operating system. This way you can be sure that no programs or processes will interfere with your security tools. Also, this guarantees that the base operating system is secure from any previous tampering or malicious programs. If for some reason you have to install your tools on an existing installation of an operating system, make sure you follow the directions later in this chapter for OS hardening and securing your system. Later in this chapter I review Bastille Linux, a tool for doing this on a Linux platform. There are free utilities available from Microsoft for hardening Windows. You can also use the tools described in Chapter 5 to scan an existing system for vulnerabilities.

Your choice of operating system for your security tool system determines how you go about securing it. I recommend an open source operating system such as Linux or BSD, but Windows will work fine as long as you properly secure it first. I used Mandrake Linux to install and run the Linux-based tools recommended in this book, and most Linux distributions and BSD or UNIX operating system can use these tools.

There are many open source operating systems available as mentioned in Chapter 1. Most of them are UNIX-based, although they all have a graphical interface available called X-Windows, and window managers such as KDE and GNOME. These interfaces will be familiar to anyone who has used Microsoft Windows, but there are a few differences.

I do not advocate that one operating system is intrinsically better than the others as far as security goes. It is all in the way you use it and configure it; hence the lengthy section that follows on hardening the OS installation. I used Linux because it is the one I have the most experience with, and I felt that it was compatible with most systems being used. With over 50 million users worldwide and dozens of variants, Linux has the widest variety of programs, and most of the open source security tools I mention in this book are designed specifically for it.

The first tool discussed automates locking down a Linux system. This will ensure you are working with a workstation that is as secure as it can be initially. There are also some basic tips on how to properly secure the Windows operating system for use as a security workstation. Finally, you will use some tools at the operating system level. There are certain system-level functions that you will use regularly in your security applications, and several of these are included in the tools section.

This chapter is not intended to be a definitive guide on securing any of these operating systems, but it gives you an overview of the basics and some tools to use.

Hardening Your Security Tool System

Once you have installed your operating system, you need to **harden** it for use as a security system. This process involves shutting off unneeded services, tightening permissions, and generally minimizing the parts of the machine that are exposed. The details of this vary depending on the intended uses of the machine and by operating system.

Hardening used to be an intensive manual process whereby you walked through each possible setting and modified it. Many books have been written on the subject of hardening each different operating system. However, you don’t have to read a whole other book to do this if you are using the Linux operating system—there are now tools that will do this for you automatically on a Linux system. This both saves time and makes it much less likely that you will miss something.

Bastille Linux: An OS Hardening Program for Linux

Bastille Linux

Author/primary contact:	Jay Beale
Web site:	www.bastille-linux.org
Platforms:	Linux (RedHat, Mandrake, Debian), HP/UX
License:	GPL
Version reviewed:	2.1.1
Important e-mails:	
General inquiries:	jon@lasser.org
Technical inquires:	jay@bastille-Linux.org

Mailing lists:
Bastille Linux announcement:
http://lists.sourceforge.net/mailman/listinfo/bastille-Linux-announce
Bastille Linux development:
http://lists.sourceforge.net/mailman/listinfo/bastille-Linux-discuss

System requirements:
Perl 5.5_003 or greater
Perl TK Module 8.00.23 or greater
Perl Curses Module 1.06 or greater

This first security tool is an operating system hardening tool called Bastille Linux. Contrary to what the name sounds like, it isn't a stand-alone operating system, but rather a set of scripts that goes through and makes certain system settings based on prompts from you. It greatly simplifies the hardening process and makes it as easy as answering some questions. It can also set up a firewall for you (that's covered in the next chapter). Bastille Linux can run on Mandrake, RedHat, Debian, and HP/UX, which is not even Linux. Jay Beale, the developer, is continuing to release support for other Linux distributions.

Installing Bastille Linux

Bastille is written using a toolkit called Curses (finally an appropriate name for a programming language!).

1. You first need to download and install the Perl Curses and TK modules, which Bastille depends on. They can be obtained from this chart on the Bastille site:

   www.bastille-Linux.org/perl-rpm-chart.html.

2. RedHat users: You also need to install a package called Pwlib, which you can obtain from the same chart. Run RPM to install it from the command line with the parameters given in the chart there.

3. Once you've installed the required modules, download the Bastille RPM or get it from the CD-ROM that accompanies this book. Click on it, and Bastille should install automatically.

Now you are ready to run Bastille to harden or lock down your operating system.

Flamey the Tech Tip:

Run Bastille on Nonproduction Systems First!

Always run these tools for the first time on nonproduction or test systems. These programs might turn off services needed for a Web server or mail server to function and cause an outage. Once you've fully tested the effect and verified that it's stable, you can run them in your production environment.

Running Bastille Linux

1. If you didn't select to start X-Windows at boot time when installing your OS, type `startx` at a command prompt and the X-Windows graphical interface will display.
2. Start Bastille in Interactive mode by clicking on the Bastille icon located in /usr/bin/bastille. You can also type `bastille` from a terminal window opened in X.
3. If you don't want to use Bastille in X-Windows or can't for some reason, you can still run Bastille from the command line using the Curses-based user interface. Type

```
bastille c
```

at any command prompt. Both interfaces will give you the same result.

You can also run Bastille in what is called Non-Interactive mode. This runs Bastille automatically, without asking any questions, from a predesignated configuration file. Every time you run Bastille, a configuration file is created. You can then use it to run Bastille on other machines in Non-Interactive mode. This technique is useful for locking down multiple machines quickly. Once you have a configuration file that does the things you want, simply load Bastille on additional machines and copy the configuration file onto those machines (or have them access the file over the network). Then type `bastille non-interactive` *`config-file`* (*`config-file`* is the name and location of the configuration file you want to use).

Most of the time, however, you will run Bastille in Interactive mode. In this mode you answer a series of questions on how you will use the machine. Based on the answers, Bastille shuts down unneeded services or restricts the privileges of users and services. It asks you things like, "Do you intend to use this machine to access Windows machines?" If not, it shuts off the Samba server, which allows your machine to interact with Windows machines. Samba could introduce some potential security holes into your system, so it is a good idea to turn it off if you don't need it. If you do have to run some servers (SSH, for example), it will attempt to set them up with limited privileges or use a **chrooted jail**. This means that if the server has to run with root access, it has a limited ability to affect other parts of the system. This blunts the effects of any successful attacks on that service.

Each question is accompanied by an explanation of why this setting is important, so you can decide if it is appropriate to your installation. There is also a More detail button that has additional information. Bastille takes the novel approach of trying to educate the administrator while it is locking down the system. The more information you have, the better armed you will be in your network security duties.

You can skip a question if you aren't quite sure and come back to it later. Don't worry; it gives you a chance at the end to finalize all the settings. You can also run Bastille later after you have researched the answer and change the setting at that time. Another nice thing that Bastille does is gives you a "to do" list at the end of the hardening session for any items that Bastille doesn't take care of.

Now you have a secure Linux computer from which to run your security tools. If you are new to a UNIX-based operating system, you will want to familiarize yourself with the common commands and navigation. If you have ever used DOS, many of the commands will be familiar although the syntax is somewhat different. One of the most significant differences between Windows and Linux and other UNIX-based operating systems is that the file system is case sensitive. Appendix B contains a cheat sheet of the most commonly used Linux and UNIX commands. Take some time to practice moving around the operating system and make sure you can do simple things like change directories, copy files, and so on.

There are several operating system commands you will be using frequently in your security work. They are not truly separately security programs but rather operating system utilities that can be used to generate security information. They are used so much in later chapters and in security work in general that I want to discuss them in detail here.

ping: A Network Diagnostic Tool

ping

Author:	Mike Muus (deceased)
Web site:	http:/ftp.arl.mil/~mike/ping.html
Platforms:	Most UNIX platforms and Windows
Licenses:	Various

UNIX manual pages:
Type `man ping` at any command prompt.

If you've been around Internet systems for any time at all, you've probably used ping. But there are some unique uses for ping in security applications as well as various considerations for how pings are handled by certain security programs. **Ping** stands for Packet

Internet Groper (which sounds a little politically incorrect) and is a diagnostic tool now built into most TCP/IP stacks. Many people think that ping is like submarine radar: a ping goes out, bounces off a target, and comes back. While this is a good general analogy, it doesn't accurately describe what happens when you ping a machine. Pings use a network protocol called **ICMP** (Internet Control Message Protocol). These messages are used to send information about networks. Ping uses ICMP message types 8 and 0, which are also known as Echo Request and Echo Reply, respectively. When you use the ping command, the machine sends an echo request out to another machine. If the machine on the other end is accessible and running a compliant TCP stack, it will reply with an echo reply. The communications in a ping basically look like this.

System A sends a ping to System B: Echo Request, "Are you there?"

System B receives the Echo Request and sends back an Echo Reply, "Yes, I'm here."

In a typical ping session this is repeated several times to see if the destination machine or the network is dropping packets. It can also be used to determine the **latency**, the time that it takes packets to cross between two points.

You may also get these other types of ICMP messages back when you ping a host. Each has its own meaning and will be explained in later chapters.

- Network unreachable
- Host unreachable

You can tell a lot more about a host with a ping than just if it is alive or not. As you will see, the way a machine responds to a ping often identifies what operating system it is running. You can also use ping to generate a DNS lookup request, which gives the destination's host name (if it has one). This can sometimes tell you if this machine is a server, a router, or perhaps someone on a home dial-up or broadband connection. You can ping an IP address or a fully qualified domain name. Table 2.1 lists additional switches and options for the ping command that you might find useful.

Table 2.1 ping Options

Options	Descriptions
-c *count*	Sends count number of pings out. The default on Linux and UNIX systems is continuous pings. On Windows, the default count is four pings.
-f	Ping flood. Sends as many packets as it can, as fast as it can. This is useful for testing to see if a host is dropping packets, because it will show graphically how many pings it responds to. Be very careful with this command, as it can take down a machine or network quite easily.

(continues)

Table 2.1 ping Options (*continued*)

Options	Descriptions
-n	Don't perform DNS on the IP address. This can speed up a response and rule out DNS issues when diagnosing network issues.
-s *size*	Sends packets of *size* length. This is good for testing how a machine or router handles large packets. Abnormally large packets are often used in denial of service attacks to crash or overwhelm machines.
-p *pattern*	Sends a specific pattern in the ICMP packet payload. This is also good for testing how a machine responds to unusual ICMP stimuli.

traceroute (UNIX) or tracert (Windows): Network Diagnostic Tools

traceroute (UNIX) or tracert (Windows)

Author/primary contact:	Unknown
Web sites:	www.traceroute.org
	www.tracert.com
Platforms:	Most UNIX and all Windows platforms
Licenses:	Various
UNIX manual pages:	

Type `man traceroute` at any UNIX command prompt.

This command is similar to ping, but it provides a lot more information about the remote host. Basically, traceroute pings a host, but when it sends out the first packet, it sets the TTL (Time to Live) setting on the packet to one. This setting controls how many hops a packet will take before dying. So the first packet will only go to the first router or machine beyond yours on the Internet, and then a message acknowledging that the packet has "expired" will return. Then, the next packet is set with a TTL of 2, and so on until it reaches your target. This shows the virtual path (the route) that the packets took. The name of each host along the way is resolved, so you can see how your traffic traverses the Internet. It can be very interesting to see how a packet going from Houston to Dallas might bounce from the East Coast to the West Coast, traveling thousands of miles before reaching its target a fraction of a second later.

This tool comes in handy when you are trying to track down the source or location of a perpetrator you have found in your log files or alerts. You can traceroute to the IP address and learn a number of things about it. The output might tell you if they are a home user or inside a company, who their ISP is (so you can file an abuse complaint), what type of service they have and how fast it is, and where geographically they are (sometimes, depending on the descriptiveness of the points in-between). Listings 2.1 and 2.2 show examples of traceroutes.

Listing 2.1 traceroute Example 1

```
Tracing route to www.example.com
over a maximum of 30 hops:

1   <10 ms  <10 ms  <10 ms 192.168.200.1

2   40 ms  60 ms  160 ms 10.200.40.1

3   30ms  40ms   100ms  gateway.smallisp.net

4   100 ms  120 ms  100 ms iah-core-03.inet.genericisp.net
    [10.1.1.1]

5   70 ms  100 ms  70 ms dal-core-03.inet.genericisp.net
    [10.1.1.2]

6   61 ms  140 ms  70 ms dal-core-02.inet.genericisp.net
    [10.1.1.3]

7   70 ms  71 ms  150 ms dal-brdr-02.inet.genericisp.net
    [10.1.1.4]

8   60 ms  60 ms  91 ms 192.168.1.1

9   70 ms  140 ms  100 ms sprintds1cust123.hou-pop.sprint.com
    [192.168.1.2]

10  101 ms  130 ms  200 ms core-cr7500.example.com
    [216.34.160.36]

11  180 ms  190 ms  70 ms acmefirewall-hou.example.com
    [216.32.132.149]

12  110 ms  110 ms  100 ms www.example.com [64.58.76.229]

Trace complete.
```

In Listing 2.1, the DNS names have been changed to generic names, but you get the general idea. From this simple command, you can tell that the IP address in question belongs to a company called Acme, that it is probably a Web server, it is inside their firewall or on the DMZ, their ISP is Sprint, and they are in Houston. Many network administrators and large ISPs use geographical abbreviations or initials to name their routers, so by looking at the DNS name and following the trail of routers, you can deduce that hou-pop.sprint.com is a Sprint router in Houston.

Listing 2.2 traceroute Example 2

```
Tracing route to resnet169-136.plymouth.edu [158.136.169.136]

over a maximum of 30 hops:



1   <1 ms  <1 ms  <1 ms 192.168.200.1

2   12 ms   7 ms   8 ms 10.200.40.1

3   26 ms  28 ms  11 ms iah-edge-04.inet.qwest.net
    [63.237.97.81]

4   37 ms  15 ms  12 ms iah-core-01.inet.qwest.net
    [205.171.31.21]

5   51 ms  49 ms  47 ms dca-core-03.inet.qwest.net
    [205.171.5.185]

6   52 ms  55 ms  65 ms jfk-core-03.inet.qwest.net
    [205.171.8.217]

7   73 ms  63 ms  58 ms jfk-core-01.inet.qwest.net
    [205.171.230.5]

8   94 ms  67 ms  55 ms bos-core-02.inet.qwest.net
    [205.171.8.17]

9   56 ms  56 ms  60 ms bos-brdr-01.ip.qwest.net
    [205.171.28.34]

10  64 ms  63 ms  61 ms 63.239.32.230

10  67 ms  59 ms  55 ms so-7-0-0-0.core-rtr1.bos.verizon-gni.net
    [130.81.4.181]

11  56 ms  61 ms  62 ms so-0-0-1-0.core-rtr1.man.verizon-gni.net
    [130.81.4.198]

12  58 ms  59 ms  57 ms so-0-0-0-0.core-rtr2.man.verizon-gni.net
    [130.81.4.206]
```

```
13   59 ms   57 ms   64 ms a5-0-0-732.g-rtr1.man.verizon-gni.net
    [130.81.5.126]

15   74 ms   62 ms   61 ms 64.223.133.166

16   68 ms   67 ms   68 ms usnh-atm-inet.plymouth.edu
    [158.136.12.2]

17   80 ms 2968 ms   222 ms xhyd04-3.plymouth.edu [158.136.3.1]

18   75 ms 2337 ms   227 ms xspe04-2.plymouth.edu [158.136.2.2]

19   74 ms   65 ms   72 ms resnet169-136.plymouth.edu
    [158.136.169.136]


Trace complete.
```

From the traceroute example in Listing 2.2 you can tell that the IP in question is probably being used by a student at Plymouth State University in Plymouth, New Hampshire. How can you tell this? First of all, the final domain name is a giveaway. If you follow the traceroute, it goes from bos (Boston) to man (Manchester), then to plymouth.edu. The .edu means that it's a university. This was an educated guess, but you can verify it by going to the plymouth.edu Web site. Also, the resolved host name is resnet169-136. The name suggests it is the network for their student residences.

As you can see, sometimes reading traceroutes is like being a detective, more of an art than a science, but over time you will learn more and get better at recognizing what each abbreviation means.

Traceroute gives lots of information to use to follow up on this IP if it was the source of an intrusion or attack. In the example in Listing 2.1, you could look up the company Web site to find a main number. You can call their ISP and complain. Larger ISPs usually have a main e-mail or contact to use for complaints, and will usually enforce their terms of service with the customer. Or you can use the next command, whois, to find specific technical contacts for the company or organization.

whois: A DNS Query Tool

whois

Author/Primary contact:	N/A
Web site:	N/A
Platforms:	Most UNIX platforms
Licenses:	Various
UNIX manual pages:	Type `man whois` at any UNIX command prompt.

The whois command is useful when trying to track down a contact for someone causing trouble on your network. This command queries the primary domain name servers and returns all the information that Internic (or whoever their name registrar is) has. Internic used to be the quasi-government agency that was responsible for keeping track of all the domain names on the Internet. Internic became a commercial company called Network Solutions, and was then acquired by VeriSign. Now that name registration has been opened up for competition, there are literally dozens of official name registrars. However, you can still usually find out who owns a domain by using the whois command.

This command is useful for attacks coming both from within companies or within ISP networks. Either way, you can track down the person responsible for that network and report your problems to them. They won't always be helpful, but at least you can try. The syntax is:

```
whois domain-name.com
```

The variable *domain-name.com* is the domain name you are looking for information on. Listing 2.3 shows the kinds of information returned that might be returned.

Listing 2.3 whois Results

```
Registrant:
Example Corp (EXAMPLE.DOM)
  123 Elm, Suite 123
  New York, NY 10000
  US
  212-123-4567
  Domain Name: EXAMPLE.COM

  Administrative Contact:
   Jones, Jane (JJ189)   jane.jones@example.com
    123 Elm, Ste 123
    New York, NY 10000
    212-123-4567

  Technical Contact:
   John Smith  (JS189)   john.smith@example.com
    123 Elm, Ste 123
    New York, NY 10000
    212-123-4567


  Record expires on 06-Oct-2006.
  Record created on 05-Oct-2002.
  Database last updated on 30-Apr-2004 21:34:52 EDT.

  Domain servers in listed order:

  NS.EXAMPLE.COM        10.1.1.1
  NS2.EXAMPLE.COM       10.1.1.2
```

As you can see, you can contact the technical person in charge of that domain directly. If that doesn't work, you can always try the administrative person. The whois command usually displays an e-mail address, a mailing address, and sometimes phone numbers. It tells when the domain was created and if they've made recent changes to their whois listing. It also shows the domain name servers responsible for that domain name. Querying these numbers with the dig command (described next) can generate even more information about the remote network's configuration.

Unfortunately, whois is not built into the Windows platforms, but there are plenty of Web-based whois engines, including the one located on Network Solutions Web site at:

www.networksolutions.com/cgi-bin/whois/whois

Flamey the Tech Tip:

Don't Drop Your Corporate Drawers on *whois*!

If you administer domains of your own, you should make sure your whois listing is both up to date and as generic as possible. Putting real e-mail addresses and names in the contact information fields gives information that an outsider can use either for social engineering or password-cracking attacks. Also, people might leave the company, making your record outdated. It is better to use generic e-mail addresses, such as **dnsmaster@example.com** or **admin@example.com**. You can forward these e-mails to the people responsible, and it doesn't give out valuable information on your technical organization structure.

dig: A DNS Query Tool

dig

Author/primary contact:	Andrew Scherpbeir
Web site:	
http://www-search.ucl.ac.uk/htdig-docs/author.html	
Platforms:	Most UNIX Platforms
Licenses:	Various
UNIX manual pages:	Type `man dig` at any UNIX command prompt.

The dig command queries a name server for certain information about a domain. Dig is an updated version of the nslookup command, which is being phased out. You can use it to determine the machine names used on a network, what the IP addresses tied to those machines are, which one is their mail server, and other useful tidbits of information. The general syntax is:

```
dig @server domain type
```

where *server* is the DNS server you want to query, *domain* is the domain you are asking about, and *type* is the kind of information you want on it. You will generally want to query the authoritative DNS for that domain; that is, the one listed in their whois record as being the final authority on that domain. Sometimes the company runs this server; other times its ISP runs the server. Table 2.2 lists the kinds of records you can ask for with the type option.

Listing 2.4 shows an example of results of the dig command. As you can see, their whole domain zone file has been downloaded. This yields valuable information, such as the host name of their mail server, their DNS server, and other important machines on their network. If you run a DNS server, you should be able to configure it to respond only to these kinds of request from authorized machines.

Listing 2.4 Output from `dig @ns.example.com AXFR`

```
; <<>> DiG 9.2.1 <<>> @ns.example.com.com example.com ANY
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54042
;; flags: qr aa rd; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 4

;; QUESTION SECTION:
;example.com   IN  ANY

;; ANSWER SECTION:
example.com.  86400   IN  MX   10 mail.example.com.
example.com.  2560    IN  SOA  ns.example.com
hostmaster.example.com. 1070057380 16384 2048 1048576 2560
example.com.  259200  IN  NS  ns.example.com.
example.com.  259200  IN  NS  ns2.example.com.
example.com.  86400   IN  A   10.1.1.1

;; ADDITIONAL SECTION:
nat1.example.com.  86400  IN  A  10.1.1.2
ns.example.com.    86400  IN  10.1.1.3
ns2.example.com.  86400  IN  A  10.1.1.4
sql.example.com   86400  IN  A  10.1.1.5
www.example.com   86400  IN  A  10.1.1.6


;; Query time: 107 msec
;; SERVER: 64.115.0.245#53(ns.example.com)
;; WHEN: Wed Dec 31 18:39:24 2003
;; MSG SIZE rcvd: 247
```

Table 2.2 dig Record Types

Options	Descriptions
AXFR	Attempts to get the whole file for the domain or “zone” file. Some servers are now configured not to allow zone file transfers, so you may have to ask for specific records.
A	Returns any “A” records. “A” records are individual host names on the network, such as webserver.example.com and firewall1.example.com.
`MX`	Returns the registered mail host name for that domain. This is useful if you want to contact an administrator (try administrator@mailhost.example.com or root@mailhost.example.com).
`CNAME`	Returns any CNAMED hosts, also known as aliases. For example: fido.example.com = www.example.com.
ANY	Returns any information it can generate on the domain. Sometimes this works when AXFR doesn’t.

finger: A User Information Service

finger

Author/primary contact:	Unknown
Web site:	Various including: www.infonet.st-johns.nf.ca/adm/finger.html www.developer.com/net/cplus/article.php/627661
Platforms:	Most UNIX and Windows platforms
Licenses:	Various

UNIX manual pages:
Type `man finger` at any command prompt.

Finger is an old UNIX command that isn’t used much anymore, but it is still running on many machines as a legacy service. It was originally designed when the Internet was a friendlier place and users didn’t mind people halfway across the world knowing their schedule, office numbers, and other information. Most competent system administrators turn this daemon off now because it has been associated with many security holes. However, you’d be surprised how many servers still run it. Many routers come with it (I can’t figure out why, except maybe the vendor implemented a TCP stack that included it), and some UNIX operating systems still enable it by default on installation, and people forget or don’t know how to turn it off.

The finger command lets you query the remote system for information on its users. The syntax is:

```
finger user@hostname.example.com
```

Replace the variables *user* with the username you are trying to find out about and *hostname.example.com* with the fully qualified host name. You can also use an IP address. Listing 2.5 shows the results of a finger query run on the user bsmith on server1.example.com.

Listing 2.5 *finger* Query Results

```
Login name: bsmith In real life: Bob Smith
Directory: /home/bsmith Shell: /bin/bash
Last Login: 7/03/04 0800:02
No unread mail
Project: Writing a book

Plan:  I'll be on vacation in Europe from September 1-15th.
```

As you can see, there quite a bit of information on Bob available through finger, including the last time he logged on, if he has any new e-mail, and any personal information he entered. He was also kind enough to let us know when he will be out of the office. This could be used by hackers to divine information about Bob for use in social engineering. It also can help them to learn his log-on habits and schedule so they could attempt to crack his account when he is out of town.

Another crafty use of finger is to send the command without a user name. This generates a list of all the users currently logged on. Listing 2.6 shows the results of what this query might look like on the fictitious example.com. You can see who is logged on and what their real names are. You can also see if they have been idle (perhaps they forgot to log out) and for how long. Finally, it lists what station they are coming from (whether they are local or remote) and the hostname or IP of where they are logging on from if it is not local. You can see one user is logged on multiple times with one session idle. A malicious viewer of this data might decide to attempt to hijack this idle session.

You could also run full finger queries on any of those users that looked worth pursuing further. Using the command `finger -l @`*hostname.example.com* generates a full finger query on every user logged in at that moment.

Listing 2.6 *finger -l* with No Username

```
[hostname.example.com]


User   Real Name   What  Idle TTY Host  Console Location

bsmith  Bob Smith          2 lab1-30 (cs.example.edu)
```

```
ajohnson Andrew Johnson         2 lab1-10 (dialup.genericisp.com)
bjones  Becky Jones             co lab3-22

atanner Allen H Tanner          0:50 co lab3-9

atanner Allen H Tanner          co lab3-1

atanner Allen H Tanner          4:20 co lab3-8

cgarcia Charles Garcia          3 lab1-10
```

ps: A UNIX Process Query Command

ps

Author/primary contact:	Unknown
Web sites:	Various, including www.nevis.columbia.edu/cgi-bin/man/sh?man=ps
Platforms:	Most UNIX platforms
Licenses:	Various

UNIX manual pages:
Type `man ps` at any UNIX command prompt.

The ps command, short for process, shows you all the processes running on a system. This can be very useful to determine if there is some daemon or process running that shouldn't be. It can also be used to debug many of the tools in this book. Table 2.3 lists some useful ps switches.

Table 2.3 ps Switches

Switches	Descriptions
A	Shows all users' processes.
a	Shows users' processes for all processes with a tty.
u	Shows the name of the process user.
x	Displays processes with controlling ttys.

Listing 2.7 shows the output from a ps command with the `-aux` switch.

Listing 2.7 *ps -aux* Output

```
USER     PID %CPU %MEM  VSZ RSS TTY    STAT START  TIME COMMAND
root      1 0.1 0.7 1288 484 ?     S  18:00  0:04 init [3]
root      2 0.0 0.0    0  0 ?    SW  18:00  0:00 [keventd]
root      3 0.0 0.0    0  0 ?    SW  18:00  0:00 [kapmd]
root      5 0.0 0.0    0  0 ?    SW  18:00  0:00 [kswapd]
root      6 0.0 0.0    0  0 ?    SW  18:00  0:00 [bdflush]
root      7 0.0 0.0    0  0 ?    SW  18:00  0:00[kupdated]
root      8 0.0 0.0    0  0 ?    SW< 18:00  0:00 [mdrecoveryd]
root     12 0.0 0.0    0  0 ?    SW  18:00  0:00 [kjournald]
root     137 0.0 0.0    0  0 ?    SW  18:00  0:00 [khubd]
root     682 0.0 1.0 1412 660 ?    S  18:01  0:00 /sbin/cardmgr
rpc    700 0.0 0.8 1416 532 ?    S  18:01  0:00 portmap
root     720 0.0 1.2 1640 788 ?    S  18:01  0:00 syslogd -m 0
root     757 0.0 1.8 1940 1148 ?    S  18:01  0:00 klogd -2
root     797 0.0 0.8 1336 500 ?    S  18:01  0:00 gpm -t ps/2 -m
xfs    869 0.0 5.8 5048 3608 ?    S  18:01  0:00 xfs -port -1
daemon   884 0.0 0.8 1312 504 ?    S  18:01  0:00 /usr/sbin/atd
root     928 0.0 2.0 2660 1244 ?    S  18:01  0:01 /usr/sbin/SSHd
root     949 0.0 1.5 2068 948 ?    S  18:01  0:00 xinetd -stayalive
root     951 0.0 0.7 1292 496 ?    S  18:01  0:00 /sbin/dhcpcd -h m
root    1078 0.0 1.0 1492 628 ?    S  18:01  0:00 crond
root    1132 0.0 3.4 3808 2152 ?    S  18:01  0:02 nessusd: waiting
root    1134 0.0 1.9 2276 1224 ?    S  18:01  0:00 login -- tony
tony    1394 0.0 2.6 2732 1624 tty1   S  18:29  0:00 -bash
tony    1430 0.0 2.6 2744 1636 tty1   S  18:29  0:00 bash
tony    1805 0.0 1.2 2676 796 tty1   R  18:56  0:00 ps -aux
```

You can see each process running on the system with its process ID. This is important if you want to kill the service or take some other action. The `-u` switch shows the user at the far left. This readout shows various system processes owned by root. It also shows a user running the ps command. If you see some mysterious service running, you should investigate it further. This listing shows what might be a suspicious service: the nessusd daemon, which is the vulnerability scanner you will use in Chapter 5. However, this is your security tool system, so it is all right for it to be running here.

You can also pipe the ps command into a grep command to search for specific services running. For example, the command

```
ps -ax |grep snort
```

will tell you if Snort is running on your system and its associated process ID (PID). So, as you'll find with many of the operating system level tools in this book, the ps command can be useful for all kinds of system administration activities, not just security.

OpenSSH Client: A Secure Terminal Service

OpenSSH Client

Author/primary contact:	Tatu Ylönen
Web site:	www.openSSH.com
Platforms:	Most UNIX platforms, Windows, OS/2
License:	BSD

Other Web sites:
www.uni-karlsruhe.de/~ig25/SSH-faq/
www.SSH.com
http://kimmo.suominen.com/SSH/

SSH is such a useful tool that there is a separate section on it in Chapter 9 as a server-side tool. However, I highly recommend using the client whenever possible for interactive logins in lieu of Telnet or some other nonsecure method. You will be using it so much I want to give some basic details and syntax of the client here. SSH (secure shell) is a remote access tool that allows you to log into a remote system securely. A major Achilles' heel of most networks is the fact that inter-system communications are generally passed over a network in plain text. So you can harden the individual machines all you want, but if you log into them remotely with an insecure terminal program, thieves could still grab your log-on credentials off the network with a sniffer. They can then log on as you without breaking a sweat. One of the most popular remote access tools, Telnet, suffers from this deficiency. SSH fixes this problem by encrypting all the communications from the first keystroke.

SSH is an open source program that is available on almost every platform, and it comes by default with most Linux-based operating systems. There is a commercial version, available at the www.ssh.com Web site, which is also open source. The one I review here is OpenSSH, the free version that comes with most Linux distributions and is on the CD-ROM that comes with this book. While there are a few differences, most of the commands and syntax should work and the two are interoperable.

In order to access a remote system with SSH, you need an SSH client on your end and there must be an SSH server running on the remote side. While SSH isn't as widespread as Telnet, it is catching on. Cisco is finally installing SSH on it routers, although it still leaves the Telnet server enabled by default while SSH is optional.

SSH is released under an open source license that is similar in effect to the BSD license. Make sure you are using version 3.6 or newer; some earlier versions had flaws in their implementation of cryptographic protocols and are susceptible to being cracked. In fact, it is a good idea to make sure you have the latest version available, as the code is constantly being improved and the algorithms are being tweaked.

SSH has a number of really interesting uses other than just logging into a remote system securely. It can be used to tunnel almost any service through an encrypted channel between servers (this application is discussed more in later chapters). Basic SSH syntax to log in remotely is:

```
ssh -l login hostname
```

Replace *login* with your login name on that remote system and *hostname* with the host you are trying to SSH into. You can also use:

```
ssh login@hostname
```

So, to log onto the Web server called web.example.com using my login of tony, I would type

```
ssh tony@web.example.com
```

I can also use `ssh -l tony web.example.com` to log into the server using SSH. If you simply type `ssh web.example.com`, the server will assume the same user name as your system login.

Table 2.4 lists some more SSH options.

Table 2.4 More SSH Options

Options	Descriptions
-c *protocol*	Uses a specific cryptographic protocol. Replace protocol with blowfish, 3des, or des, depending on the cryptographic algorithm you want to use. Note that your version of SSH must support these algorithms.
-p *port#*	Connects to a specific port number rather than the default SSH port of 22.
-P *port#*	Uses a specific port that is not part of the standard list of proprietary ports. This usually means a port number above 1024. This can be useful if you have a firewall that knocks down communications on lower port numbers.
-v	Displays verbose output. This is useful for debugging.
-q	Reports in quiet mode, opposite of verbose.
-C	Uses compression on the encrypted traffic. This can be useful for extremely slow connections like dial-up, but you better have a powerful processor to do the compression or it will slow you down more than it will speed you up.
-1	Forces SSH to use only SSH protocol version 1. This is not recommended for the reasons mentioned in the -C option, but it may be required if the server you are connecting to isn't upgraded to version 2.
-2	Forces SSH to use SSH protocol version 2 only. This may keep you from connecting to some servers.

Considerations for Hardening Windows

While not the subject of this book, it's important if you're using a Windows system to lock the system down as much as possible so you can establish that Trusted Computing Base discussed earlier. Windows is notorious for running all kinds of network-aware services. Some vendors of Windows PCs even load small Web servers on them so their technical support staff can "come in" and help you out interactively if you call in. Needless to say, this is horribly insecure and hacks have been published for many of these little "helpful" tools. Most people are unaware of all these programs running in the background.

One thing you can do if you are running one of the newer versions of Windows (NT, 2000, or XP) is to go to the Services window located under Administrative Tools in the Control Panel menu. This lists all the processes running on your computer (similar to the UNIX ps command). You can scroll down through this list and see all the little programs that Windows helpfully starts up for you. Most of these are services that are required for the basic operation of Windows. However, some of them you don't need and are just taking up processor cycles, slowing down your computer, and possibly creating a security hole. You can shut them down by clicking on the service and selecting Stop. Make sure you also set the start-up type to Manual or Disabled, or they will just start up again when you reboot.

Flamey the Tech Tip:

Be Sure You Know What You're Turning Off!

You need to be very careful when shutting things down like this. If you don't explicitly know what the service is and that you don't need it, then don't shut it off. Many processes depend on others, and shutting them down arbitrarily might cause your system to stop functioning properly.

There are some excellent guides created by the National Security Agency (www.nsa.gov) for secure configuration of Windows operating systems. Guides are currently available for Windows 2000 and NT, and more are being added as they become available. You can download them from http://nsa1.www.conxion.com/index.html.

The Center for Internet Security (www.cisecurity.org) publishes a benchmark and scoring tools for Windows 2000 and NT as well. You can use these tools to help configure your Windows machines securely.

Many books and Internet resources cover this subject in more depth. You can also use some of the tools discussed later in this book, such as the port scanner and vulnerability scanner, to scan and secure Windows systems as well. However you do it, make sure you harden your system before you begin installing tools on it.

While Windows has some of the network diagnostic and query tools that UNIX has, such as ping and traceroute, it does not offer some of the other services, such as whois and dig, right out of the box. There is, however, an add-on security tool, Sam Spade for Windows, that adds this functionality to your Windows system and improves on the existing ones.

Sam Spade for Windows: A Network Query Tool for Windows

Sam Spade for Windows

Author/primary contact:	Steve Atkins
Web site:	www.samspade.org
Platforms:	Windows 95, 98, ME, NT, 2000, XP
Version reviewed:	1.14
License:	GPL
Other resources:	

See the Help file included with the installation.

This wonderful Swiss army knife for Windows machines fixes the dearth of real network tools in the Windows OS. No longer can UNIX system administrators gloat over their Windows counterparts who don't have neat things like dig, whois, and other valuable tools. In fact, Sam Spade for Windows even adds a few that the UNIX guys don't have. It is an invaluable tool for finding out information on networks. Like the fictional detective of the same name, Sam Spade can find out just about anything about a network.

Installing and Using Sam Spade for Windows

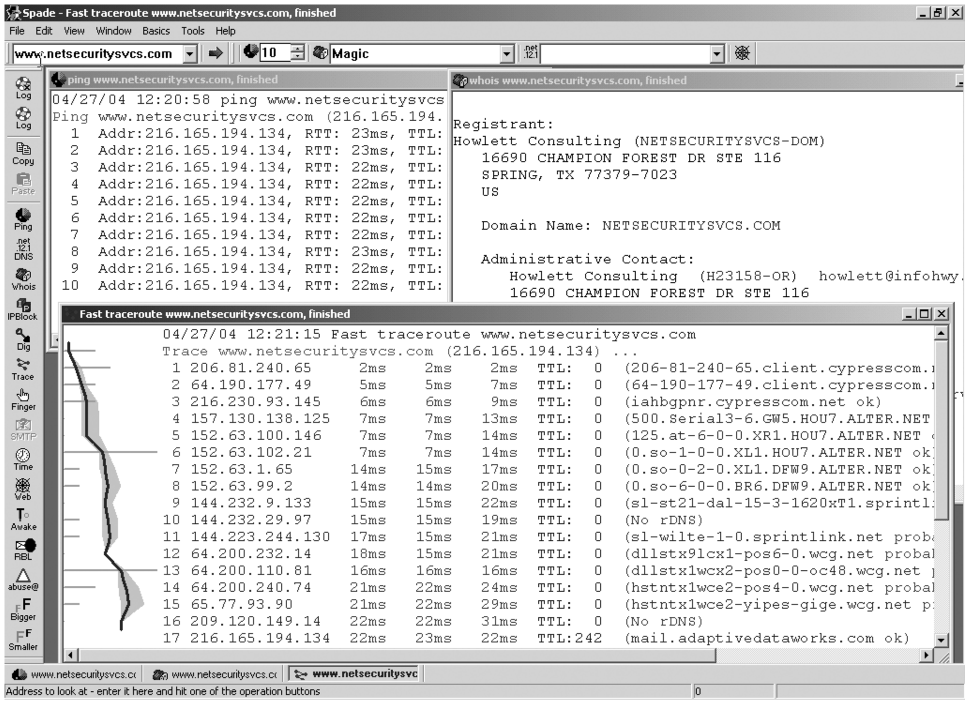
Start by visiting the Samspade.org Web site and downloading the program, or get it from the CD-ROM that comes with this book. Then simply double-click on the file and let the install program take care of everything for you. Once you've installed Sam Spade, fire it up and you will get the main console screen (see Figure 2.1).

Sam Spade has an easy-to-use interface. You enter the IP address or host name you want to run tests on in the upper-left field, and then click the icons below it to run different tests against that target. Each test runs in a window of its own, and all the output is stored in a log file that you can save for later use and documentation. You must set up a default name server under the Options menu so that any tests that rely on DNS will function. You can also enter this number in the menu bar to the far right.

Flamey the Tech Tip:

Be a Responsible Sam Spade

Running Sam Spade on your own network or one you are responsible for is fine. However, be very careful when running these tools against networks outside your control. While most of these tests are benign, some could put a heavy load on a server or set off intrusion monitors. So make sure you have



Spade - Fast traceroute www.netsecuritysvcs.com, finished
File Edit View Window Basics Tools Help
www.netsecuritysvcs.com
10
Magic
Log
Log
Copy
Paste
Ping
DNS
Whois
IPBlock
Dig
Trace
Finger
SMTP
Time
Web
Awake
RBL
abuse@
Bigger
Smaller
ping www.netsecuritysvcs.com, finished
whois www.netsecuritysvcs.com, finished
Fast traceroute www.netsecuritysvcs.com, finished
www.netsecuritysvcs.cc
www.netsecuritysvcs.cc
www.netsecuritysvc
Address to look at - enter it here and hit one of the operation buttons
0

```
04/27/04 12:20:58 ping www.netsecuritysvcs
Ping www.netsecuritysvcs.com (216.165.194.
  1  Addr:216.165.194.134, RTT: 23ms, TTL:
  2  Addr:216.165.194.134, RTT: 23ms, TTL:
  3  Addr:216.165.194.134, RTT: 22ms, TTL:
  4  Addr:216.165.194.134, RTT: 22ms, TTL:
  5  Addr:216.165.194.134, RTT: 22ms, TTL:
  6  Addr:216.165.194.134, RTT: 22ms, TTL:
  7  Addr:216.165.194.134, RTT: 22ms, TTL:
  8  Addr:216.165.194.134, RTT: 23ms, TTL:
  9  Addr:216.165.194.134, RTT: 22ms, TTL:
 10  Addr:216.165.194.134, RTT: 22ms, TTL:
```

```
Registrant:
Howlett Consulting (NETSECURITYSVCS-DOM)
   16690 CHAMPION FOREST DR STE 116
   SPRING, TX 77379-7023
   US

   Domain Name: NETSECURITYSVCS.COM

   Administrative Contact:
      Howlett Consulting  (H23158-OR)  howlett@infohwy.
      16690 CHAMPION FOREST DR STE 116
```

```
04/27/04 12:21:15 Fast traceroute www.netsecuritysvcs.com
Trace www.netsecuritysvcs.com (216.165.194.134) ...
 1 206.81.240.65     2ms    2ms    2ms  TTL:  0  (206-81-240-65.client.cypresscom.
 2 64.190.177.49     5ms    5ms    7ms  TTL:  0  (64-190-177-49.client.cypresscom.
 3 216.230.93.145    6ms    6ms    9ms  TTL:  0  (iahbgpnr.cypresscom.net ok)
 4 157.130.138.125   7ms    7ms   13ms  TTL:  0  (500.Serial3-6.GW5.HOU7.ALTER.NET
 5 152.63.100.146    7ms    7ms   14ms  TTL:  0  (125.at-6-0-0.XR1.HOU7.ALTER.NET
 6 152.63.102.21     7ms    7ms   14ms  TTL:  0  (0.so-1-0-0.XL1.HOU7.ALTER.NET ok
 7 152.63.1.65      14ms   15ms   17ms  TTL:  0  (0.so-0-2-0.XL1.DFW9.ALTER.NET ok
 8 152.63.99.2      14ms   14ms   20ms  TTL:  0  (0.so-6-0-0.BR6.DFW9.ALTER.NET ok
 9 144.232.9.133    15ms   15ms   22ms  TTL:  0  (sl-st21-dal-15-3-1620xT1.sprintl
10 144.232.29.97    15ms   15ms   19ms  TTL:  0  (No rDNS)
11 144.223.244.130  17ms   15ms   21ms  TTL:  0  (sl-wilte-1-0.sprintlink.net prob
12 64.200.232.14    18ms   15ms   21ms  TTL:  0  (dllstx9lcx1-pos6-0.wcg.net proba
13 64.200.110.81    16ms   16ms   16ms  TTL:  0  (dllstx1wcx2-pos0-0-oc48.wcg.net
14 64.200.240.74    21ms   22ms   24ms  TTL:  0  (hstntx1wce2-pos4-0.wcg.net proba
15 65.77.93.90      21ms   22ms   29ms  TTL:  0  (hstntx1wce2-yipes-gige.wcg.net p
16 209.120.149.14   22ms   22ms   31ms  TTL:  0  (No rDNS)
17 216.165.194.134  22ms   23ms   22ms  TTL:242  (mail.adaptivedataworks.com ok)
```

Figure 2.1 Sam Spade Main Screen

permission before running these tools on outside networks. Not only is it in a gray area legally, but it's also just good manners. You wouldn't want some other system administrator running these against your network without your permission, would you?

Table 2.5 lists the main functions of Sam Spade and describes what they do. Table 2.6 lists other useful tests located under the Tools menu.

Table 2.5 Sam Spade Main Functions

Functions	Descriptions
Ping	This is the same as the built-in Windows and UNIX ping, except you can easily configure the number of pings and the output is a little more verbose.
Nslookup	Similar to the UNIX command of the same name.

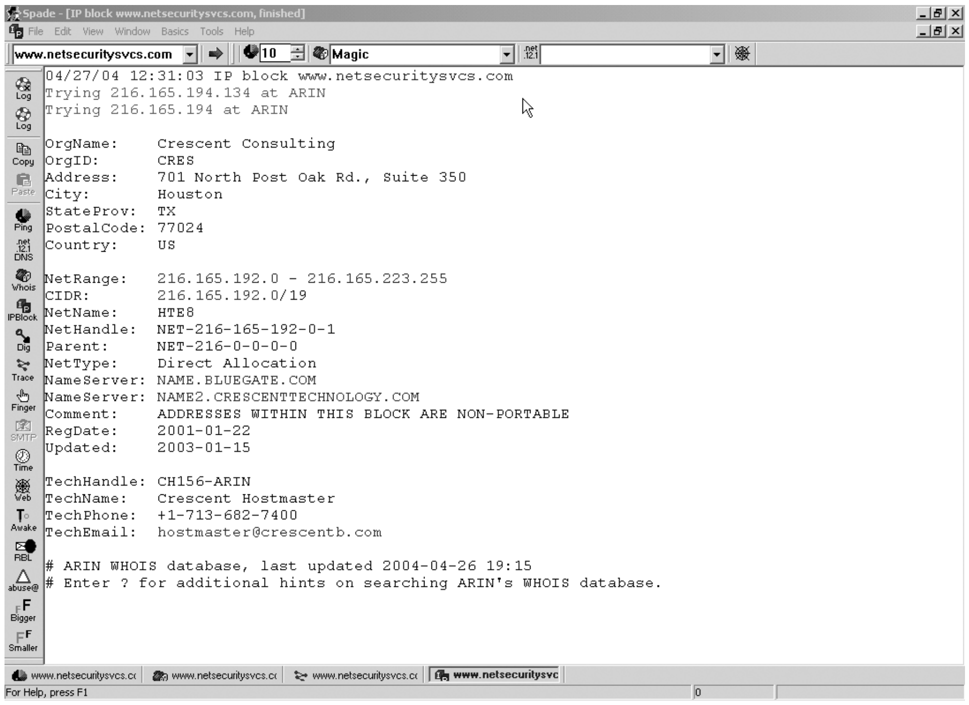
(continues)

Table 2.5 Sam Spade Main Functions (*continued*)

Functions	Descriptions
Whois	Similar to the UNIX command of the same name.
IPBlock	This command checks the ARIN database for an IP address or set of IP addresses and generates some useful information on it. This data includes the organization that owns those IPs, where they were allocated from an ISP, and different contacts, including a contact to report abuse if they registered one. See Figure 2.2 for an example of the output.
Trace	Similar to the traceroute command. However, additional information is generated, such as any reverse DNS entry and a graphical display of the latency between hops.
Finger	Similar to the UNIX finger command.
Time	Checks the time clock on the remote system. This is good for ensuring that your server's time clocks are synchronized.

Table 2.6 Sam Spade Tools Menu Tests

Tests	Descriptions
Blacklist	Checks to see if your mail server is listed in any of the e-mail black hole lists (databases that contain the addresses of known spammers). If your address somehow gets in there (by leaving your server open to mail relays, for example), then some people won't be able to get mail from you.
Abuse	Looks up the official abuse contact for a set of IP addresses so you can register a complaint if you are having a problem with one of their addresses.
Scan Addresses	Performs a basic port scan of a range of addresses. This very simple port scanner identifies open network ports. If you are going to need to scan addresses, I recommend you use one of the fully featured port scanners reviewed in Chapter 4. Also, keep in mind that port scanning can be considered hostile activity by outside networks.
Crawl website	Takes a Web site and "crawls" it, identifying each link and page and any other forms or files it can reach. This is useful for finding all the pages that a Web site references and for looking for files that you weren't aware were there.



```
04/27/04 12:31:03 IP block www.netsecuritysvcs.com
Trying 216.165.194.134 at ARIN
Trying 216.165.194 at ARIN

OrgName:    Crescent Consulting
OrgID:      CRES
Address:    701 North Post Oak Rd., Suite 350
City:       Houston
StateProv:  TX
PostalCode: 77024
Country:    US

NetRange:   216.165.192.0 - 216.165.223.255
CIDR:       216.165.192.0/19
NetName:    HTE8
NetHandle:  NET-216-165-192-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Allocation
NameServer: NAME.BLUEGATE.COM
NameServer: NAME2.CRESCENTTECHNOLOGY.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2001-01-22
Updated:    2003-01-15

TechHandle: CH156-ARIN
TechName:   Crescent Hostmaster
TechPhone:  +1-713-682-7400
TechEmail:  hostmaster@crescentb.com

# ARIN WHOIS database, last updated 2004-04-26 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Figure 2.2 Sam Spade IP Block Output

There are several other tools that are not the subject of this book, such as Check cancels for USENET News and Decode URLs, that you may find useful if you are developing a Web site. Sam Spade can give you UNIX-like capabilities in terms of network discovery. The next tool, PuTTY, gives you the capabilities of SSH, another UNIX-based program for secure remote terminal access on Windows.

PuTTY: An SSH Client for Windows

PuTTY

Author/primary contact:	Sam Tatham
Web site:	www.chiark.greenend.org.uk/~sgtatham/putty
Platforms:	Windows 95, 98, ME, NT, 2000, XP
Version reviewed:	.54b
License:	MIT (similar to BSD license)
Other resources:	

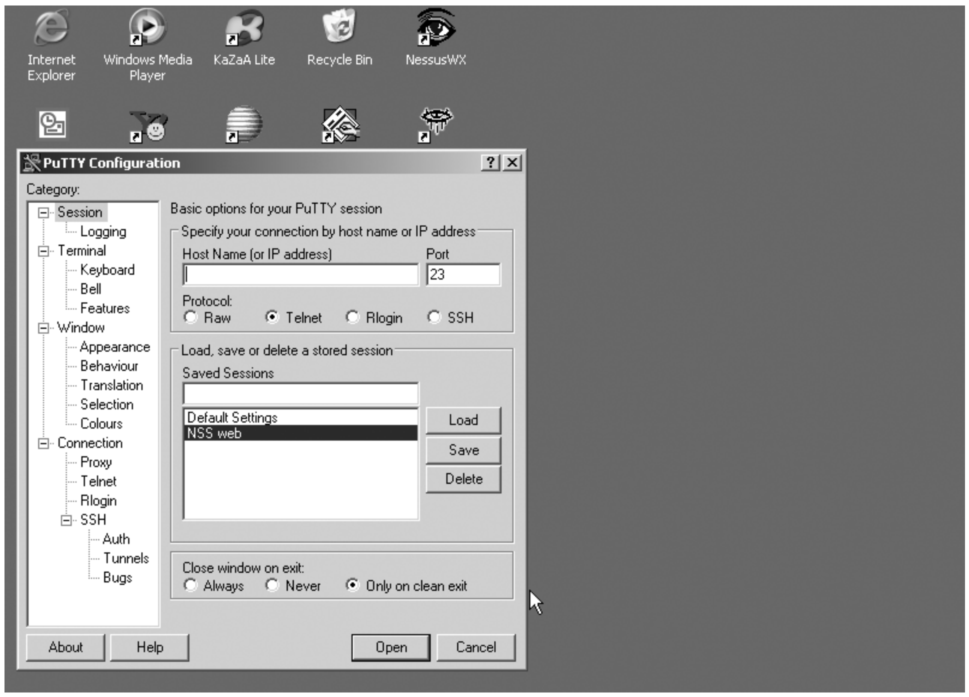
See Help file or Web site.

One of these days Microsoft will get with the program and begin including a built-in SSH client with Windows. In the meantime, PuTTY is an excellent SSH client for Windows, and it also includes an enhanced, encryption-enabled Telnet client. You can use PuTTY to securely communicate with any server running the SSH protocol.

Installing and Running PuTTY

Download the file from the Web site or get it from the CD-ROM that comes with this book and double-click on it to install it. PuTTY has a pretty clean interface and should be able to emulate almost all terminals. You can configure the port number you come in on if the SSH server is using a nonstandard port number. You can also fiddle with all the settings by using the menus on the left.

You can log all your sessions to a text file, which can be quite useful (I used PuTTY to log all of the terminal session listings in this book). You can also mess with the configuration ad infinitum, including which encryption protocols it will accept. It will even warn



Basic options for your PuTTY session

Specify your connection by host name or IP address

Host Name (or IP address):

Port: 23

Protocol: Raw [✓ Telnet] Rlogin SSH

Load, save or delete a stored session

Saved Sessions

Default Settings

NSS web

Close window on exit: Always Never [✓ Only on clean exit]

Figure 2.3 PuTTY Main Screen

you if it is attempting to connect to a SSH server that uses one of the weak versions of SSH that may be vulnerable to cracking.

When connecting to a server for the first time, PuTTY will warn you that it is adding that server's fingerprint and key to your database. This is normal—just make sure the certificate looks appropriate, accept it, and it won't appear in future connections to that server.

CHAPTER 3

Firewalls

So now that you have a fairly secure operating system and know a few basic tricks, let's get into using some more complex security tools. This chapter describes how to configure and run a secure open source firewall. If you already have a firewall, you may still want to read this chapter if you need a refresher or primer on how firewalls function. This will come in handy in later chapters that discuss port scanners and vulnerability scanners.

A **firewall** is a device that acts as the first line of first defense against any incoming attacks or misuses of your network. It can deflect or blunt many kinds of attacks and shield your internal servers and workstations from the Internet. A firewall can also prevent internal LAN machines from being accessed from outside your network. With the growing use of random scanners and automated worms and viruses, keeping your internal machines shielded from the Internet is more important than ever. A properly configured firewall will get you a long way towards being safe from outside attacks. (Protecting yourself from inside attacks is a different thing altogether and is a subject of Chapters 4 through 7.)

Chapter Overview

Concepts you will learn:

- Basic concepts of TCP/IP networking
- How firewalls operate
- The philosophy of firewall configuration
- Business processes for firewalls
- Sample firewall configurations

Tools you will use:

Iptables, Turtle Firewall, and SmoothWall

It's pretty much a given these days that firewalls are an essential part of any secure infrastructure. There are many very viable commercial alternatives available: Cisco, NetScreen, SonicWALL, and Checkpoint are just a few of the vendors making high-end, commercial firewall solutions. These products are built to handle large corporate networks and high traffic volumes

Linksys (now owned by Cisco), D-Link, and NETGEAR are some of the vendors making low-end consumer-grade firewalls. These devices generally don't have much configurability or expandability; they basically act as a packet filter, blocking incoming TCP and UDP connections and as a NAT appliance. They are usually marketed for DSL and cable-type connections and may buckle under heavier loads.

The higher end firewalls will do just about anything you want them to do. However, that comes at a price: most of them start at several thousand dollars and go up from there. And they often require you to learn a new syntax or interface in order to configure them. Some of the newer models, like SonicWALL and NetScreen, are going to a Web-based configuration interface, but that usually comes at the expense of less depth in the configuration options.

The little known and rarely advertised secret of some commercial firewalls is that they have open source software just underneath the hood. What you are really paying for is the fancy case and the technical support line. This may be worth it for companies that need the extra support. However, if you are going to have to learn yet another interface, and if they are using the same technologies that are available to you for free, why not create your own firewall with the open source tools provided in this book and save your firm thousands of dollars? Even if you don't want to throw out your commercial firewall, learning more about firewall basics and what happens behind the scenes will help you keep your firewall more securely configured.

Before we dive into the tools, I want to go over the basics of what a firewall does and how it works with the various network protocols to limit access to your network. Even if you are not planning to use open source software for your firewall, you can still benefit from knowing a little more about what is really going on inside that black box.

Network Architecture Basics

Before you can truly understand network security, you have to first understand network architecture. Although this book is not intended to serve as a network primer, this section is a quick review of network concepts and terms. I will be referring to these terms often and it will help you to have a basic understanding of the TCP/IP protocol. If you are already well-schooled in network topologies, then you can skip over this section and jump straight into the tools.

As you may know, every network design can be divided into seven logical parts, each of which handles a different part of the communication task. This seven-layered design is called the **OSI Reference Model**. It was created by the International Standards Organizations (ISO) to provide a logical model for describing network communications, and it

OSI Layer Number	Layer Name	Sample Protocols
Layer 7	Application	DNS, FTP, HTTP, SMTP, SNMP, Telnet
Layer 6	Presentation	XDR
Layer 5	Session	Named Pipes, RPC
Layer 4	Transport	NetBIOS, TCP, UDP
Layer 3	Network	ARP, IP, IPX, OSPF
Layer 2	Data Link	Arcnet, Ethernet, Token Ring
Layer 1	Physical	Coaxial, Fiber Optic, UTP

Figure 3.1 The OSI Reference Model

helps vendors standardize equipment and software. Figure 3.1 shows the OSI Reference Model and gives examples of each layer.

Physical

This layer is the actual physical media that carries the data. Different types of media use different standards. For example, coaxial cable, unshielded twisted pair (UTP), and fiber optic cable each serve a different purpose: coaxial cable is used in older LAN installations as well as Internet service through cable TV networks, UTP is generally used for in-house cable runs, while fiber optic is generally used for long-haul connections that require a high load capacity.

Data Link

This layer relates to different pieces of network interface hardware on the network. It helps encode the data and put it on the physical media. It also allows devices to identify each other when trying to communicate with another node. An example of a data link layer address is your network card's MAC address. (No, the MAC address doesn't have anything to do with Apple computers; it's the Medium Access Control number that uniquely identifies your computer's card on the network.) On an Ethernet network, MAC addresses are the way your computer can be found. Corporations used many different types of data link standards in the 1970s and 80s, mostly determined by their hardware vendor. IBM used Token Ring for their PC networks and SNA for most of their bigger hardware, DEC used a different standard, and Apple used yet another. Most companies use Ethernet today because it is widespread and cheap.

Network

This layer is the first part that you really see when interacting with TCP/IP networks. The network layer allows for communications across different physical networks by using a secondary identification layer. On TCP/IP networks, this is an IP address. The IP address on your computer helps get your data routed from place to place on the network and over the Internet. This address is a unique number to identify your computer on an IP-based network. In some cases, this number is unique to a computer; no other machine on the Internet can have that address. This is the case with normal publicly routable IP addresses. On internal LANs, machines often use private IP address blocks. These have been reserved for internal use only and will not route across the Internet. These numbers may not be unique from network to network but still must be unique within each LAN. While two computers may have the same private IP address on different internal networks, they will never have the same MAC address, as it is a serial number assigned by the NIC manufacturer. There are some exceptions to this (see the sidebar Follow the MAC), but generally the MAC address will uniquely identify that computer (or at least the network interface card inside that computer).

Flamey the Tech Tip:

Follow the MAC

MAC addresses can help you troubleshoot a number of network problems. Although the MAC address doesn't identify a machine directly by name, all MAC addresses are assigned by the manufacturer and start with a specific number for each vendor. Check out www.macaddresses.com for a comprehensive list. They are also usually printed on the card itself.

By using one of the network sniffers discussed in Chapter 6, you can often track down the source of troublesome network traffic using MAC addresses. Mac addresses are usually logged by things like a Windows DHCP server or firewalls, so you can correlate MAC addresses to a specific IP address or machine name. You can also use them for forensic evidence—amateur hackers often forge IP addresses, but most don't know how to forge their MAC address, and this can uniquely identify their PCs.

Transport

This level handles getting the data packet from point A to point B. This is the layer where the TCP and UDP protocols reside. TCP (Transmission Control Protocol) basically ensures that packets are consistently sent and received on the other end. It allows for bit-level error correction, retransmission of lost segments, and fragmented traffic and packet reordering. UDP (User Datagram Protocol) is a lighter weight scheme used for multimedia traffic and short, low-overhead transmissions like DNS requests. It also does error detection and data multiplexing, but does not provide any facility for data reordering or ensured data arrival. This layer and the network layer are where most firewalls operate.

Session

The session layer is primarily involved with setting up a connection and then closing it down. It also sometimes does authentication to determine which parties are allowed to participate in a session. It is mostly used for specific applications higher up the model.

Presentation

This layer handles certain encoding or decoding required to present the data in a format readable by the receiving party. Some forms of encryption could be considered presentation. The distinction between application and session layers is fine and some people argue that the presentation and application layers are basically the same thing.

Application

This final level is where an application program gets the data. This can be FTP, HTTP, SMTP, or many others. At this level, some program handling the actual data inside the packet takes over. This level gives security professionals fits, because most security exploits happen here.

TCP/IP Networking

The TCP/IP network protocol was once an obscure protocol used mostly by government and educational institutions. In fact, it was invented by the military research agency, DARPA, to provide interruption-free networking. Their goal was to create a network that could withstand multiple link failures in the event of something catastrophic like a nuclear strike. Traditional data communications had always relied on a single direct connection, and if that connection was degraded or tampered with, the communications would cease. TCP/IP offered a way to "packetize" the data and let it find its own way across the network. This created the first fault-tolerant network.

However, most corporations still used the network protocols provided by their hardware manufacturers. IBM shops were usually NetBIOS or SNA; Novell LANs used a protocol called IPX/SPX; and Windows LANs used yet another standard, called NetBEUI, which was derived from the IBM NetBIOS. Although TCP/IP became common in the 1980s, it wasn't until the rise of the Internet in the early 90s that TCP/IP began to become the standard for data communications. This brought about a fall in the prices for IP networking hardware, and made it much easier to interconnect networks as well.

TCP/IP allows communicating nodes to establish a connection and then verify when the data communications start and stop. On a TCP/IP network, data to be transmitted is chopped up into sections, called **packets**, and encapsulated in a series of "envelopes," each one containing specific information for the next network layer. Each packet is stamped with a 32-bit sequence number so that even if they arrive in the wrong order, the transmission can be reassembled. As the packet crosses different parts of the network each layer is opened and interpreted, and then the remaining data is passed along according to those instructions. When the packet of data arrives at its destination, the actual data, or payload, is delivered to the application.

It sounds confusing, but here is an analogy. Think of a letter you mail to a corporation in an overnight envelope. The overnight company uses the outside envelope to route the package to the right building. When it is received, it will be opened up and the outside envelope thrown away. It might be destined for another internal mailbox, so they might put in an interoffice mail envelope and send it on. Finally it arrives at its intended recipient, who takes all the wrappers off and uses the data inside. Table 3.1 shows how some network protocols encapsulate data.

As you can see, the outside of our data "envelope" has the Ethernet address. This identifies the packet on the Ethernet network. Inside that layer is the network information, namely the IP address; and inside that is the transport layer, which sets up a connection and closes it down. Then there is the application layer, which is an HTTP header, telling the Web browser how to format a page. Finally comes the actual payload of packet—the content of a Web page. This illustrates the multi-layered nature of network communications.

There are several phases during a communication between two network nodes using TCP/IP (see Figure 3.2). Without going into detail about Domain Name Servers (DNS)

Table 3.1 Sample TCP/IP Data Packet

Protocol	Contents	OSI Layer
Ethernet	MAC address	Datalink
IP	IP address	Network
TCP	TCP header	Transport
HTTP	HTTP header	Application
Application Data	Web page	Data

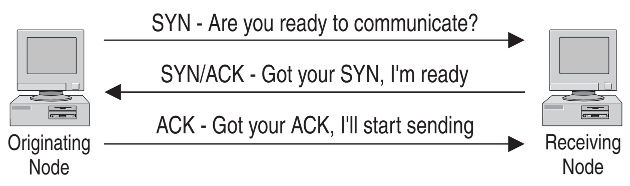

SYN - Are you ready to communicate?
SYN/ACK - Got your SYN, I'm ready
ACK - Got your ACK, I'll start sending
Originating Node
Receiving Node

Figure 3.2 TCP Three-Way Handshake

and assuming we are using IP addresses and not host names, the first thing that happens is that the machine generates an ARP (Address Resolution Protocol) request to find the corresponding Ethernet address to the IP it is trying to communicate with. ARP converts an IP address into a MAC address on an Ethernet network. Now that we can communicate to the machine using IP, there is a three-way communication between the machines using the TCP protocol to establish a session. A machine wishing to send data to another machine sends a SYN packet to synchronize, or initiate, the transmission. The SYN packet is basically saying, “Are you ready to send data?” If the other machine is ready to accept a connection from the first one, it sends a SYN/ACK, which means, “Acknowledged, I got your SYN packet and I’m ready.” Finally, the originating machine sends an ACK packet back, saying in effect, “Great, I’ll start sending data.” This communication is called the **TCP three-way handshake**. If any one of the three doesn’t occur, then the connection is never made. While the machine is sending its data, it tags the data packets with a sequence number and acknowledges any previous sequence numbers used by the host on the other end. When the data is all sent, one side sends a FIN packet to the opposite side of the link. The other side responds with a FIN/ACK, and then the other side sends a FIN, which is responded to with a final FIN/ACK to close out that TCP/IP session.

Because of the way TCP/IP controls the initiation and ending of a session, TCP/IP communications can be said to have **state**, which means that you can tell what part of the dialogue is happening by looking at the packets. This is a very important for firewalls, because the most common way for a firewall to block outside traffic is to disallow SYN packets from the outside to machines inside the network. This way, internal machines can communicate outside the network and initiate connections to the outside, but outside machines can never initiate a session. There are lots of other subtleties in how firewalls operate, but basically that’s how simple firewalls allow for one-way only connections for Web browsing and the like.

There are several built-in firewall applications in Linux: these are known as **Iptables** in kernel versions 2.4x, **Ipchains** in kernel versions 2.2x, and **Ipfwadm** in kernel version 2.0. Most Linux-based firewalls do their magic by manipulating one of these kernel-level utilities.

All three applications operate on a similar concept. Firewalls generally have two or more interfaces, and under Linux this is accomplished by having two or more network cards in the box. One interface typically connects to the internal LAN; this interface is called the **trusted** or **private** interface. Another interface is for the public (WAN) side of your firewall. On most smaller networks, the WAN interface is connected to the Internet. There also might be a third interface, called a **DMZ** (taken from the military term for Demilitarized Zone), which is usually for servers that need to be more exposed to the Internet so that outside users can connect to them. Each packet that tries to pass through the machine is passed through a series of filters. If it matches the filter, then some action is taken on it. This action might be to throw it out, pass it along, or masquerade ("Masq") it with an internal private IP address. The best practice for firewall configuration is always to deny all and then selectively allow traffic that you need (see the sidebar on firewall configuration philosophy).

Firewalls can filter packets at several different levels. They can look at IP addresses and block traffic coming from certain IP addresses or networks, check the TCP header and determine its state, and at higher levels they can look at the application or TCP/UDP port number. Firewalls can be configured to drop whole categories of traffic, such as ICMP. ICMP-type packets like ping are usually rejected by firewalls because these packets are often used in network discovery and denial of service. There is no reason that someone outside your company should be pinging your network. Firewalls will sometimes allow echo replies (ping responses), though, so you can ping from inside the LAN to the outside.

Security Business Processes

At some point, preferably before you start loading software, you should document in writing a business process for your firewall(s). Not only will this be a useful tool for planning your installation and configuration, but it may also help if you have to justify hardware purchases or personnel time to your boss. Documenting your security activities will make you look more professional and emphasize the value you add to the organization, which is never a bad thing. It also makes it easier for anyone who comes after you to pick up the ball.

This plan documents the underlying processes and procedures to make sure that you get a business benefit from the technology. Installing a firewall is all well and good, but without the proper processes in place, it might not actually give the organization the security it promises. The following steps outline a business process for firewall implementation and operation.

1. Develop a network use policy.
   There may already be some guidelines in your employee manual on proper computer use. However, many computer use polices are intentionally vague and don't specify which applications count as misuse. You may have to clarify this with your manager or upper management. Are things like instant messengers allowed? Do you want to follow a stringent Web and e-mail only outbound policy? Remember that it is safer to write a rule for any exceptions rather than allowing all types of activity by default. Getting the answers to these questions (hopefully in writing) is crucial before you start writing rules.

2. Map out services needed outward and inward.
If you don't already have a network map, create one now. What servers need to be contacted from the outside and on which ports? Are there users who need special ports opened up for them? (Hint: technical support staff often need FTP, Telnet, and SSH.) Do you want to set up a DMZ for public servers or forward ports to the LAN from the outside? If you have multiple network segments or lots of public servers, this could take longer than the firewall setup itself. Now is the time to find out about these special requests, not when you turn on the firewall and it takes down an important application.
3. Convert the network use policy and needed services into firewall rules.
This is when you finally get to write the firewall rules. Refer to your list of allowed services out, required services in, and any exceptions, and create your firewall configuration. Be sure to use the "deny all" technique described in the sidebar to drop anything that doesn't fit one of your rules.
4. Implement and test for functionality and security.
Now you can turn on your firewall and sit back and wait for the complaints. Even if your rules conform exactly to policy, there will still be people who didn't realize that using Kazaa to download movies was against company policy. Be ready to stand your ground when users ask for exceptions that aren't justified. Every hole you open up on your firewall is a potential security risk.
Also, once your firewall is operating to your users' satisfaction, make sure that it is blocking what it is supposed to be blocking. By using two tools discussed later in this book together, you can run tests against your firewall: A port scanner on the outside and a network sniffer on the inside will tell you which packets are getting through and which ones aren't. This setup can also be useful for troubleshooting applications that are having problems with the firewall.
5. Review and test your firewall rules on a periodic basis.
Just because your firewall is working great today doesn't mean it will be tomorrow. New threats may evolve that require new rules to be written. Rules that were supposed to be temporary, just for a project, may end up being left in your configuration. You should review your rules periodically and compare them with the current business requirements and security needs. Depending on the size and complexity of your configuration and how often it changes, this may be as infrequently as once a year for firewalls with a small rule set (20 or fewer rules), or once a month for very complex firewalls. Each review should include an actual test using the scanner/sniffer setup mentioned above using the tools in Chapters 4, 5, and 6 to verify that the rules are indeed doing what they are supposed to be.

Designing and using a business process such as this will help ensure you get a lot more out of your firewall implementation, both professionally and technically. You should also develop plans for the other technologies discussed in this book, such as vulnerability scanning and network sniffing.

Flamey the Tech Tip:

"Deny all!" When It Comes to Firewall Rules

There are two ways set up a firewall: You can start with an "allow all" stance and then add the behavior you want blocked, or start with a "deny all" statement and then add what you want to allow (permissible user behavior). The overwhelmingly preferred method is starting with "deny all." By beginning with this statement, you automatically block all traffic unless it is specifically allowed in the configuration. This method is both more secure and easier to maintain securely than the other route.

Most commercial firewalls use this philosophy. The idea behind it is that if you have to define what is bad behavior, you will be continually behind as the Internet changes and evolves. You cannot predict what form the next new attack might take, so you will be vulnerable until it is published and you can add a new line to your firewall configuration. By using the "deny all" approach, you categorically deny anything that isn't known good activity.

The "allow all" type of configuration might make sense in a extremely permissive environment where the overhead of adding lines for allowed items overrides the value of the information on the network, for example, on a nonprofit or purely informational site. But for most sites the "deny all" approach is much safer. However, just because you use this approach doesn't mean your network is totally secure. Attacks can still come in via any holes you've created, such as for the Web and e-mail. Also, keep in mind that even when the "deny all" statement is used, you have to be careful not to negate it with an overly permissive statement higher up in your configuration.

Iptables: A Linux Open Source Firewall

Iptables

Author/primary contact:	Paul "Rusty" Russell
Web site:	www.netfilter.org
Platforms:	Most Linux
License:	GPL
Version reviewed:	1.2.8
Resources:	
Netfilter mailing lists:	
Netfilter-announce	General announcement list for news of new releases and updates. Subscribe at:

https://lists.netfilter.org/mailman/listinfo/netfilter-announce

Netfilter-users	General questions about using Netfilter/Iptables. Post general discussion topics and questions here. Subscribe at:
https://lists.netfilter.org/mailman/listinfo/netfilter-users	
Netfilter-devel	Development and contributor discussions. Subscribe at:
https://lists.netfilter.org/mailman/listinfo/netfilter-devel	

This section describes how to configure a firewall with Iptables, which is the firewall/packet filter utility built into most Linux systems with kernel version 2.4 and later. This utility lets you create a firewall using commands in your operating system. Iptables evolved from earlier attempts at firewalls on Linux. The first system, called Ipfwadm, could be used to create a simple set of rules to forward or deny packets based on certain criteria. Ipchains was introduced in kernel 2.2 to overcome the limitations of Ipfwadm. Ipchains worked pretty well and was modular in architecture. However, with the growing number of people using their firewalls for multiple functions (for example, proxy server and NAT device), Ipchains also became insufficient. Iptables represents an update to these programs and allows for the multiple uses that today's firewalls are expected to perform. (Note that the concepts and terms for Iptables are pretty much the same for Ipchains.)

Iptables is a powerful but complex tool, and is usually recommended for users who are familiar with firewalls and the art of configuring them (see the sidebar on writing shell scripts). If this is your first firewall, I suggest using one of the autoconfiguration tools discussed later in the chapter to create your firewall configuration, at least at first. These tools use Iptables (or its predecessor, Ipchains) to create a firewall by using your input. However, it is good to have a basic understanding of what is going on "under the hood" with Iptables before start configuring with one of the graphical tools.

Installing Iptables

Most Linux systems on kernel 2.4 or higher will have Iptables built right in, so you don't have to install any additional programs. (If your system is earlier than kernel 2.4, it will use Ipchains or Ipfwadm. These are similar systems, but they are not reviewed in this book.) You can issue Iptables statements from the command line or via a script (see the sidebar). To double-check that Iptables is installed, type `iptables -L` and see if you get a response. It should list your current rule set (which is probably empty if you haven't configured a firewall yet).

If your system doesn't have Iptables or if you want to get the latest version of the code, go to www.netfilter.org and download the RPM for your operating system. You can also get it from the CD-ROM that comes with this book.

If you don't have a Webmin RPM on your installation disks, check www.webmin.com to see if there is a version of Webmin available for your operating system. Webmin is required for the Turtle Firewall, and there are specific versions for each distribution and operating system. If there isn't one for your particular operating system, then you can't use Turtle Firewall, but the list of supported systems is quite large. Click on the RPM file in X-Windows and it will install automatically.

Using Iptables

The idea behind Iptables and Ipchains is to create pipes of input and process them according to a rule set (your firewall configuration) and then send them into pipes of output. In Iptables, these pipes are called **tables**; in Ipchains, they are called **chains** (of course!). The basic tables used in Iptables are:

- Input
- Forward
- Prerouting
- Postrouting
- Output

The general format of an Iptables statement is

```
iptables command rule-specification extensions
```

where `command`, `rule-specification`, and `extensions` are one or more of the valid options. Table 3.2 lists the Iptables commands, and Table 3.3 contains the Iptables rule specifications.

Table 3.2 Iptables Commands

Commands	Descriptions
-A *chain*	Appends one or more rules to the end of the statement.
-I *chain rulenum*	Inserts chain at the location *rulenum*. This is useful when you want a rule to supercede those before it.
-D *chain*	Deletes the indicated chain.
-R *chain rulenum*	Replaces the rule at *rulenum* with the provided *chain*.
-L	Lists all the rules in the current chain.
-F	Flushes all the rules in the current chain, basically deleting your firewall configuration. This is good when beginning a configuration to make sure there are no existing rules that will conflict with your new ones.

Commands	Descriptions
-Z *chain*	Zeros out all packet and byte counts in the named chain.
-N *chain*	Creates a new chain with the name of *chain*.
-X *chain*	Deletes the specified chain. If no chain is specified, this deletes all chains.
-P *chain policy*	Sets the policy for the specified chain to *policy*.

Table 3.3 Iptables Rule Specifications

Rule Specifications	Descriptions	
-p *protocol*	Specifies a certain protocol for the rule to match. Valid protocol types are icmp, tcp, udp, or all.	
-s *address/mask!port*	Specifies a certain address or network to match. Use standard slash notation to designate a range of IP addresses. A port number or range of port numbers can also be specified by putting them after an exclamation point.	
-j *target*	This tells what to do with the packet if it matches the specifications. The valid options for target are:	
	DROP	Drops the packet without any further action.
	REJECT	Drops the packet and sends an error packet in return.
	LOG	Logs the packet to a file.
	MARK	Marks the packet for further action.
	TOS	Changes the TOS (Type of Service) bit.
	MIRROR	Inverts the source and destination addresses and sends them back out, essentially "bouncing" them back to the source.

(continues)

Table 3.3 Iptables Rule Specifications (*continued*)

Rule Specifications	**Descriptions**	
	SNAT	Static NAT. This option is used when doing Network Address Translation (NAT). It takes the source address and converts it into another static value, specified with the switch --to-source.
	DNAT	Dynamic NAT. Similar to above but using a dynamic range of IP addresses.
	MASQ	Masquerades the IP using a public IP.
	REDIRECT	Redirects the packet.

There are other commands and options but these are the most common operations. For a full listing of commands, refer to the Iptables man page by typing `man iptables` at any command prompt.

Creating an Iptables Firewall

The best way to learn is to do, so let's walk through a couple of commands to see how they are used in practical application. Here is an example of how to create a firewall using Iptables. You can enter these commands interactively (one at a time) to see the results right away. You can also put them all into a script and run it at boot time to bring your firewall up at boot time (see the sidebar on writing scripts). Remember to type them exactly as shown and that capitalization is important.

Writing Shell Scripts

Often you will need to automate a process or have a single command initiate a number of statements. In the firewall example, you will generally want to have all your firewall commands executed when your system boots. The best way to do this is to write a shell script. A **shell script** is a simple text file that contains a command or list of commands. The shell editor executes the commands when it is invoked by a user typing the name of the script.

1. To create a shell script, first open a text editor such as vi or EMACS and type in your command(s).
2. Make sure you put a line at the very top that looks like this:

```
#! /bin/bash
```

This tells the script which shell to use to execute the command. You must have that shell on your OS, and the commands you put in your script will have to be valid commands for that shell. This example is for the bash shell location on Mandrake Linux. You can use a different shell, for example, Tcsh or Csh. Just put the path to it on this line. Then save your file.

3. Make the file executable so the shell can run it as a program. You do this with the `chmod` command. Type:

```
chmod 700 script_name
```

where you replace *`script_name`* with your file name. This makes the permissions on the file readable, writable, and executable.

To run the script, type the file's name in the command line. (In the bash shell, you need to add a `./` before the file name to run the script from your local directory.) When you press Enter, the commands in your script should run.

You have to be in the same directory as the file or type the path in the command line statement when you run it. Alternatively, you could add the directory for the script to your PATH statement so it will run from anywhere or put the script in one of your PATH directories.

The example in the following procedure assumes that your local LAN subnet is 192.168.0.1 - 192.168.0.254, that the eth1 interface is your local LAN connection, and that the eth0 interface is your Internet orWAN connection.

1. Start by eliminating any existing rules with a Flush command:

```
iptables -F FORWARD
```

This flushes all rules for the FORWARD chain, which is the main "funnel" for any packets wanting to pass through the firewall.

2. Flush the other chains:

```
iptables -F INPUT
iptables -F OUTPUT
```

This flushes any rules to your local machine and your output chain.

3. Put your standard "deny all" statement right up front.

```
iptables -P FORWARD DROP
iptables -A INPUT -i eth0 -j DROP
```

4. To accept fragmented packets in Iptables, this must be done explicitly.

```
iptables -A FORWARD -f -j ACCEPT
```

5. There are two types of common attacks that you should block right away. One is what is known as **spoofing**, which is when someone forges the IP packet headers to make it look like an outside packet has in internal address. By doing this, someone

could route onto your LAN even if you have private IP addresses. The other type of attack is done by sending a stream of packets to the broadcast address of the LAN to overwhelm the network. This is called a **smurf** attack (although I'm not sure what this has to do with little blue cartoon characters). You can block these types of attacks with two simple statements.

```
iptables -A FORWARD -s 192.168.0.0/24 -I eth0 -j DROP
iptables -A FORWARD -p icmp -i eth0 -d 192.168.0.0/24 -j
  DENY
```

The first statement drops any packets coming from the Internet interface eth0 with the internal address 192.168.0.0/24. By definition, no packets should be coming from the untrusted interface with an internal, private source address. The second statement drops any packets of protocol ICMP coming from the outside address to the inside.

6. You generally do want to accept incoming traffic based on connections initiated from the inside, for example, someone surfing a Web page. As long as the connection is ongoing and it was initiated internally, then it is probably okay. You can, however, limit the type of traffic allowed in. Let's say that you only want to allow employees Web and e-mail access. You can specify the types of traffic to allow through and only if it is on an already-initiated connection. You can tell if it is an existing connection by seeing that the ACK bit has been set, that is, that the TCP three-way handshake has occurred. The following statements allow HTTP and Web traffic based on this criteria.

```
iptables –A FORWARD –p tcp –i eth0 –d 192.168.0.0/24 --
  dports
www,smtp --tcp-flags SYN,ACK –j ACCEPT

iptables –A FORWARD –p tcp –i eth0 –d 192.168.0.0/24 --
  sports
www,smtp --tcp-flags SYN,ACK –j ACCEPT
```

The -dport statement says to only allow e-mail and Web, and the –tcp flags statement says you only want packets with the ACK field set.

7. To be able to accept incoming connections from the outside only on certain ports, such as e-mail coming into your mail server, use a statement like this:

```
iptables –A FORWARD –m multiport –p tcp –i eth0 –d
  192.168.0.0/24
--dports smtp --syn –j ACCEPT
```

The -m multiport flag tells Iptables that you will be issuing a match statement for ports. The -syn statement tells it to allow SYN packets, which means to initiate TCP connections. And the -dports flag allows only the SMTP mail traffic.

8. You can allow outgoing connections to be initiated by your users, but only on the protocols you want them using. This is where you can prevent your users from using FTP and other nonessential programs. The all-zero IP address is shorthand for saying "any address."

```
iptables –A FORWARD –m multiport –p tcp –i eth0 –d
  0.0.0.0 --dports www,smtp --syn –j ACCEPT
```

9. You need to allow certain incoming UDP packets. UDP is used for DNS, and if you block that your users won't be able to resolve addresses. Because they don't have a state like TCP packets, you can't rely on checking the SYN or ACK flags. You want to allow UDP only on port 53, so you specify domain (a built-in variable for port 52) as the only allowable port. You do that with these statements.

```
iptables –A FORWARD –m multiport –p udp –i eth0 –d
  192.168.0.0/24  --dports domain –j ACCEPT

iptables –A FORWARD –m multiport –p udp –i eth0 –s
  192.168.0.0/24  --sports domain –j ACCEPT

iptables –A FORWARD –m multiport –p udp –i eth1 –d
  0.0.0.0 --dports domain –j ACCEPT

iptables –A FORWARD –m multiport –p udp –i eth1 –s
  0.0.0.0 --sports domain –j ACCEPT
```

10. The first two statements allow the incoming UDP datagrams, and the second two allow the outbound connections. You also want to do this for ICMP packets. These are the network information packets discussed in Chapter 2. You want to allow all types of internal ICMP outwards, but only certain types such as echo-reply inwards. This can be accomplished with the following statements.

```
iptables –A FORWARD –m multiport –p icmp –I eth0 –d
192.168.0.0/24 --dports 0,3,11 –j ACCEPT

iptables –A FORWARD –m multiport –p icmp –I eth1 –d
0.0.0.0
 --dports 8,3,11 –j ACCEPT
```

11. Finally, you want to set up logging so you can look at the logs to see what is being dropped. You will want to view the logs from time to time even if there isn't a problem, just to get an idea of the kinds of traffic being dropped. If you see dropped packets from the same network or address repeatedly, you might be being attacked. There is one statement to log each kind of traffic.

```
iptables –A FORWARD –m tcp –p tcp –j LOG
iptables –A FORWARD –m udp –p udp –j LOG
iptables –A FORWARD –m udp –p icmp –j LOG
```

That's it! This will provide you with firewall protection from the most common attacks from the Internet.

IP Masquerading with Iptables

When the Internet was originally designed, several large blocks of addresses were set aside for use on private networks. These addresses will not be routed by the Internet and can be used without worrying that they will conflict with other networks. The private address ranges are:

10.0.0.0 – 10.255.255.255

192.168.0.0 – 192.68.255.255

172.16.0.0 – 172.31.255.255

By using these addresses on your internal LAN and having one external, routable IP on your firewall, you effectively shield your internal machines from outside access. You can provide this additional layer of protection easily with Iptables using **IP masquerading**. The internal IP header is stripped off at the firewall and replaced with a header showing the firewall as the source IP. The data packet is then sent out to its destination with a source IP address of the public interface of the firewall. When it comes back, the firewall remembers which internal IP it goes to and re-addresses it for internal delivery. This process is also known as **Network Address Translation** (NAT). You can do this in Iptables with the following statements.

```
iptables –t nat –P POSTROUTING DROP
iptables –t nat –A POSTROUTING –o eth0 –j MASQUERADE
```

The MASQUERADE flag can be abbreviated to MASQ. One of the improvements of Iptables over previous systems like Ipchains and Ipfwadm is the way that it handles secondary tasks like NAT.

So now you know how to build a basic firewall. This is just a simple configuration; the possible variations are endless. You can forward certain ports to internal servers so they don't have to have a public IP address.You can put another network card in your firewall box and make it a DMZ interface for servers with public addresses. There are entire books on advanced firewall configuration and many mailing lists. One of the better lists is firewall-wizards. To subscribe to this list, send an e-mail with "subscribe" in the body to:

firewall-wizards-request@honor.icsalabs.com

The firewall-wizards list hosts discussions about all levels of firewall configuration and is vendor agnostic, that is, all firewall brands are discussed, from open source to commercial.

If you want to build a quick firewall without entering all those Iptables statements and remembering the syntax, there is tool that builds the firewall statements using a graphical interface—so it's all done for you in the background.

Turtle Firewall: An Iptables-Based Firewall with a Graphical User Interface

Turtle Firewall

Author/primary contact:	Andrea Frigido
Web site:	www.turtlefirewall.com/
Platforms:	Most Linux-compatibles that support Iptables
License:	GPL 2.0
Contact information:	andrea@friweb.com
System requirements:	Linux operating system with kernel 2.4 or newer Perl with expat library Webmin server

This neat little contraption, called Turtle Firewall, was created by Andrea Frigido. Turtle is basically a set of Perl scripts that do all the dirty work for you to set up an Iptables firewall. This program makes it much easier to see your rules and to make sure you are getting the statements in the right order. It runs as a service, so you don't have to worry about initializing your firewall with a shell script. It uses the Linux Webmin service, which is a little Web server that allows you to make configuration changes to your server via a Web browser. While this might introduce some insecurity into your system by running a Web server on the firewall, it may be worth it for the ease of configuration it brings. Many commercial vendors now use a Web browser interface for configuration. A big benefit of this application is that you can reach the configuration screen from any Windows or UNIX machine.

For support, Andrea offers a commercial support option. For a mere 100 euros (don't ask me to convert that to dollars exactly, but when this book was printed it was about $100.00), you can get 30 days of e-mail support so you can get help setting it up. It also might be worth subscribing if you have a problem with an existing installation that you can't solve on your own.

Installing Turtle Firewall

Installing and setting up Turtle Firewall is very easy because it uses the Webmin administration module, which is available on most Linux platforms.

1. If you did not install the Webmin administration module during your OS installation, you will need to in order to use Turtle Firewall. Locate and run the RPM, which should be on most Linux distributions disks. Click on the RPM file and it will install automatically.
2. Once that is done, you should be able to log into your firewall's configuration screen by putting its IP address in your browser window and pressing Enter.

3. Now you are ready to install Turtle Firewall. Download the packed distribution from www.turtlefirewall.com or get it from the CD-ROM that comes with this book and unzip it.
4. Change to the turtlefirewall directory and type:

```
./setup
```

   This runs an installation script that puts the Perl modules and other things that are needed in the right places.
5. Log into the Webmin server using a Web browser pointed at the IP address or host name the server is using. The Webmin interface will display.
6. Click the Module Index tab, and the Turtle Firewall Main screen displays (see Figure 3.3).

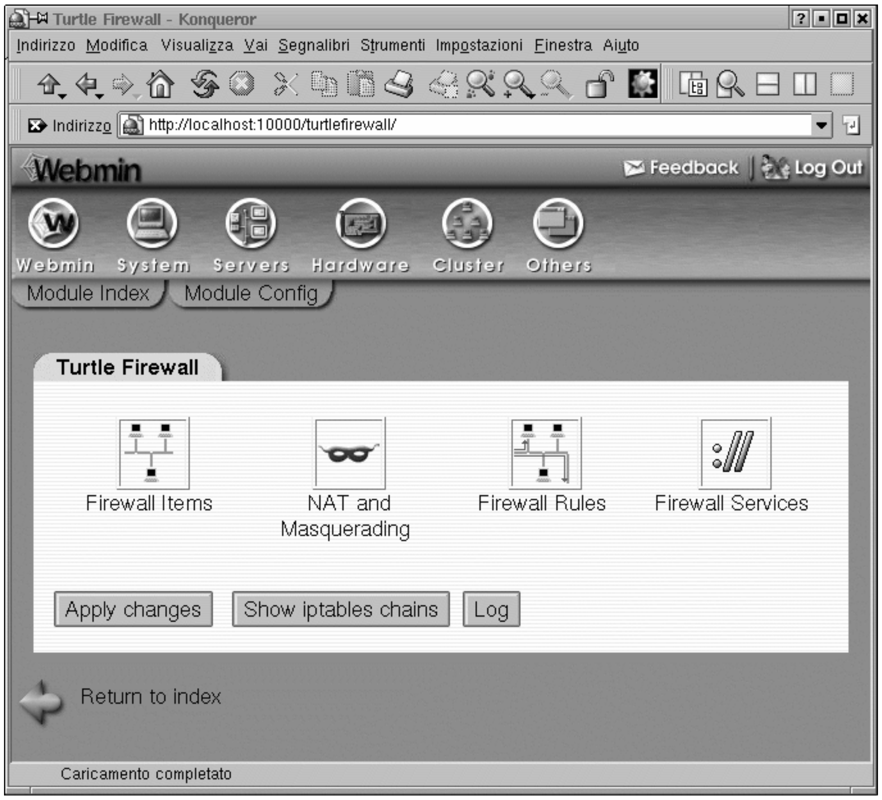
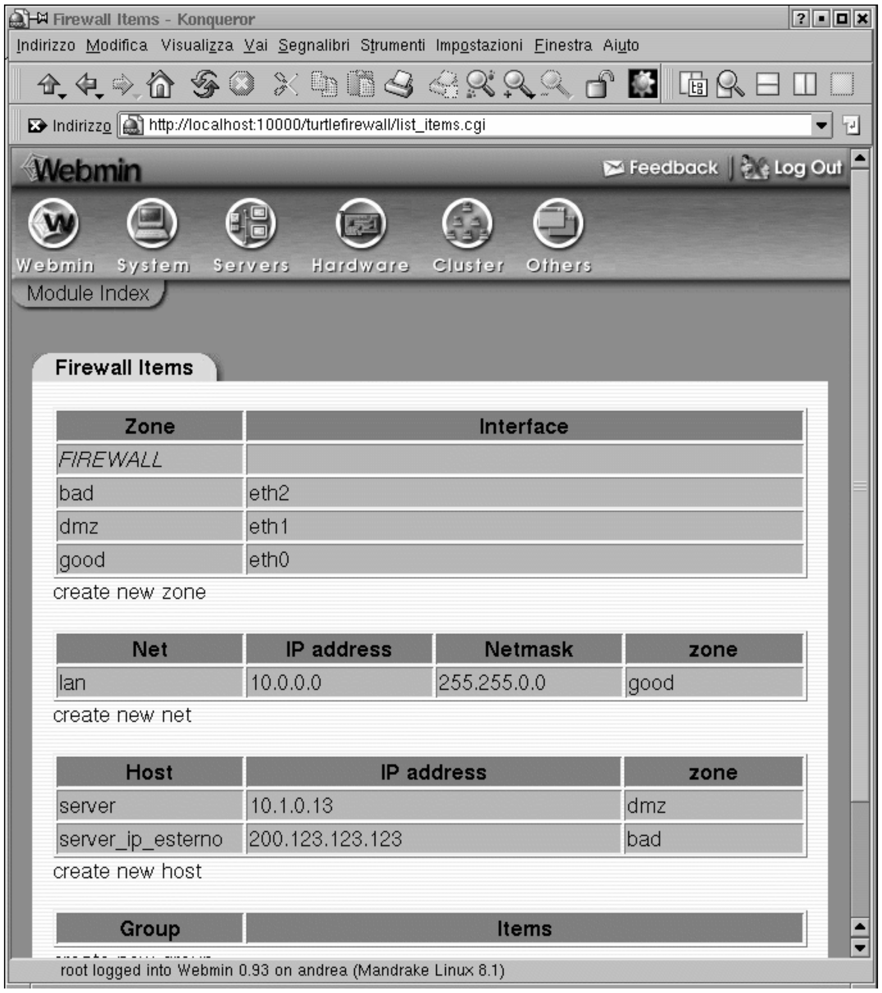

Turtle Firewall - Konqueror
Indirizzo Modifica Visualizza Vai Segnalibri Strumenti Impostazioni Finestra Aiuto
Indirizzo http://localhost:10000/turtlefirewall/
Webmin
Feedback
Log Out
Webmin System Servers Hardware Cluster Others
Module Index
Module Config
Turtle Firewall
Firewall Items
NAT and Masquerading
Firewall Rules
Firewall Services
Apply changes
Show iptables chains
Log
Return to index
Caricamento completato

Figure 3.3 Turtle Firewall Main Screen

7. Click on the Firewall Items icon to begin configuring your firewall.

First you will need to define some basic things about your firewall (see Figure 3.4). Turtle Firewall uses the concept of zones to define trusted and untrusted networks. A **trusted zone** connects to a network with employees or people who should generally be trusted on it, such as your internal network. An **untrusted zone** is a network that could have anything on it, from employees to customers, vendors, or even people with malevolent intentions. Turtle calls them "good" and "bad," but it is basically the same thing as trusted and untrusted.


Firewall Items - Konqueror
Indirizzo Modifica Visualizza Vai Segnalibri Strumenti Impostazioni Finestra Aiuto
Indirizzo http://localhost:10000/turtlefirewall/list_items.cgi
Webmin
Feedback
Log Out
Webmin System Servers Hardware Cluster Others
Module Index

Firewall Items

Zone	Interface
FIREWALL	
bad	eth2
dmz	eth1
good	eth0

create new zone

Net	IP address	Netmask	zone
lan	10.0.0.0	255.255.0.0	good

create new net

Host	IP address	zone
server	10.1.0.13	dmz
server_ip_esterno	200.123.123.123	bad

create new host

Group	Items

root logged into Webmin 0.93 on andrea (Mandrake Linux 8.1)

Figure 3.4 Turtle Firewall Configuration

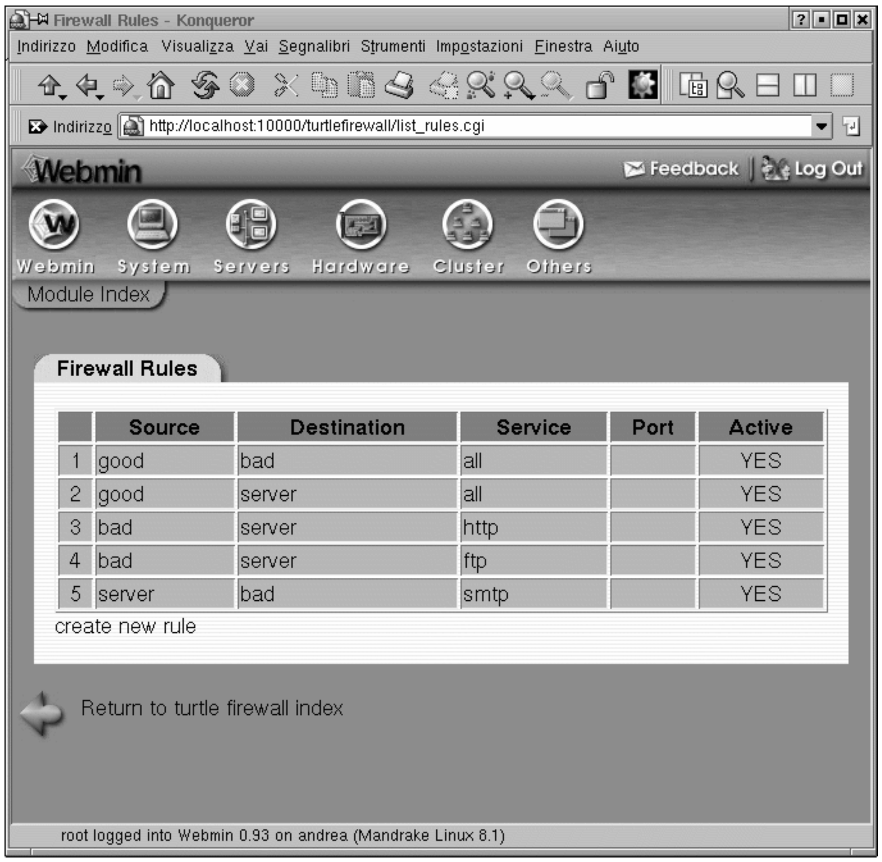
Turtle also has an entry for a DMZ or "Demilitarized Zone" segment. A DMZ segment is used to put servers that need unfettered access to the untrusted zone. Put the interfaces for your good, bad, and DMZ (if any) interfaces here.

8. Next you need to define your internal network IP addresses in the Net box. Put the IP address range with subnet mask for your internal LAN to be protected by the firewall in the box provided (see Figure 3.4).

9. Next, define any internal or DMZ hosts that will need special consideration, such as your mail server or Web server. Do this in the Hosts box (see Figure 3.4).

10. Finally, you can define any special hosts that you want to treat differently, such as administrators, in the Group area. Now your firewall is up and running in basic mode.

There are probably some additional restrictions or permissions you will want to add, for example, the ability for someone from the outside to use SSH to get in.You can do this by writing a rule on the Firewall Rules tab. Click on that tab, and it will graphically walk you through writing a new firewall rule. You will notice the format is similar to Iptables (see Figure 3.5).


Firewall Rules - Konqueror
Indirizzo Modifica Visualizza Vai Segnalibri Strumenti Impostazioni Finestra Aiuto
Indirizzo http://localhost:10000/turtlefirewall/list_rules.cgi
Webmin
Feedback
Log Out
Webmin System Servers Hardware Cluster Others
Module Index

Firewall Rules

	Source	Destination	Service	Port	Active
1	good	bad	all		YES
2	good	server	all		YES
3	bad	server	http		YES
4	bad	server	ftp		YES
5	server	bad	smtp		YES

create new rule

Return to turtle firewall index

root logged into Webmin 0.93 on andrea (Mandrake Linux 8.1)

Figure 3.5 Turtle Firewall Rules
]

If you want to implement the Iptables Masquerade function using private IP addresses for your internal LAN, click on the NAT and Masquerading icon on the main screen. Here you can define what zone will be masqueraded (see Figure 3.6). Generally, it will be your "good" or trusted interface. You can also set up hosts to be "NAT'ed" here. Putting a host to be your virtual IP makes it act as the front for your real host, and the firewall will forward all packets through the virtual host to the real host. This provides an extra level of protection for your internal servers.

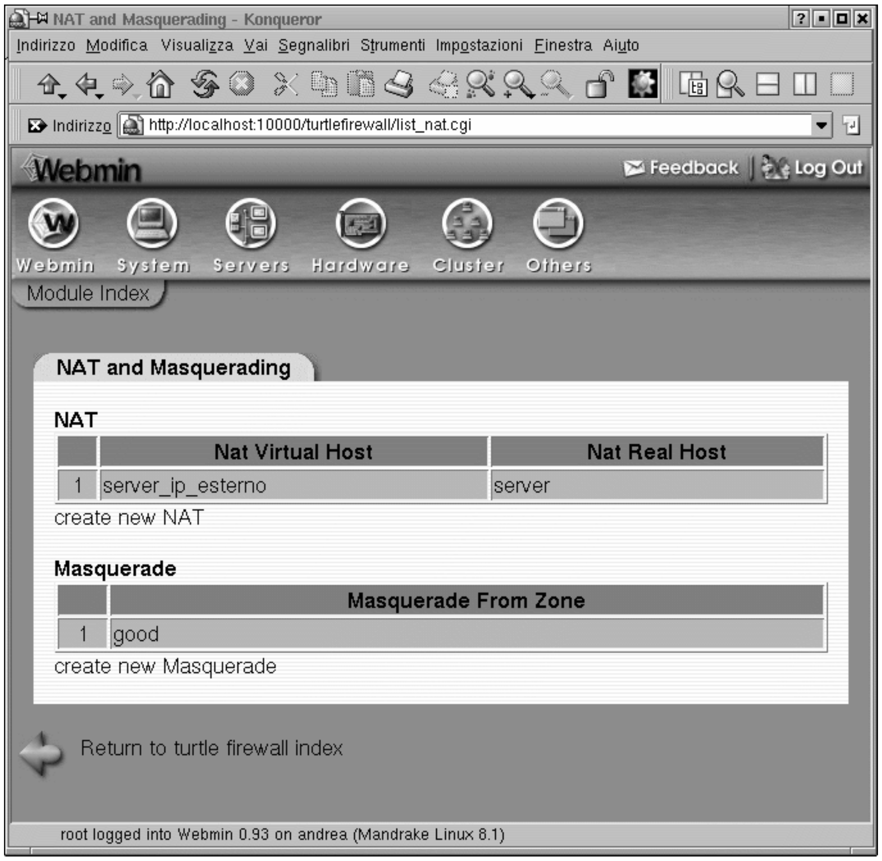
SmoothWall Express: A Complete Multi-Function Firewall

SmoothWall Express

Authors/primary contacts:	Lawrence Manning, Richard Morrell, Jon Fautley, and Tom Ellis (original authors) SmoothWall Limited (current contact)
Web site:	www.smoothwall.org
Platform:	Linux
License:	GPL
Version reviewed:	2.0

Web forums:
http://community.smoothwall.org/forum/
IRC chat channels:
Use IRC server irc.smoothwall.org 6667.
Join the channel #help for SmoothWall questions and general chat.
Mailing lists:
For general/installation support, subscribe at:
http://lists.smoothwallusers.org/mailman/listinfo/gpl

The two programs discussed previously, Iptables and Turtle Firewall, offer an inexpensive way to set up a simple firewall. But if you need a DHCP server, you have to set that up separately. And if you want to be able to SSH into the machine, that is another program to install. SmoothWall is an open source firewall that offers a robust firewall package with all those features and more built in. It is designed by a company that offers both a free GPL version and a commercial version with some additional features and enhanced support. This is another example of how a product can take advantage of the power of open source and also reap commercial gains for a company. The free version is called SmoothWall Express and is currently on version 2.0; the commercial version is called SmoothWall Corporate Server version 3.0.


NAT and Masquerading - Konqueror
Indirizzo Modifica Visualizza Vai Segnalibri Strumenti Impostazioni Finestra Aiuto
Indirizzo http://localhost:10000/turtlefirewall/list_nat.cgi
Webmin
Feedback
Log Out
Webmin System Servers Hardware Cluster Others
Module Index

NAT and Masquerading

NAT

	Nat Virtual Host	Nat Real Host
1	server_ip_esterno	server

create new NAT

Masquerade

	Masquerade From Zone
1	good

create new Masquerade

Return to turtle firewall index

root logged into Webmin 0.93 on andrea (Mandrake Linux 8.1)

Figure 3.6 Turtle Firewall NAT and Masquerading

SmoothWall Express contains several options beyond Iptables that most companies would want in a fully functional firewall. Granted, you can cob most of these together with other programs and Iptables, but SmoothWall offers it all in one program in an easy to install package. Some of these features are:

- VPN support: SmoothWall integrates an IPsec VPN with firewall capabilities. This allows people on the outside to securely access the local area network via an encrypted tunnel. This can be a fixed remote office or a roaming salesperson (nonstatic IP VPN is only supported in the corporate edition).
- DHCP client and server: The client allows the firewall to get a dynamic IP address for its WAN interface. This is common practice on DSL and cable modem ISP service. It also allows the firewall to act as a DHCP server for the internal LAN, handing out IP addresses according to a preset policy. Again, you can add these

things to an Iptables firewall, but then you have two separate programs to install and manage.

- SSH and Web access to firewall: Secure access via command line and a Web browser. The Turtle Firewall gives this capability for Iptables but doesn't allow SSH access. SmoothWall has both built in with no additional software to install.
- Web proxy server: The ability to set up a Web proxy so that all Web sites are accessed through a firewall. This provides some level of Web security, since any exploits would have to run on the firewall and not the local machine. It can also allow for further protection through a content filtering option available from SmoothWall Limited.
- Web caching server: This feature stores the most popular Web pages for local access so that access times are improved and bandwidth usage is lowered.
- Intrusion detection: SmoothWall offers some basic network intrusion detection capabilities.
- Graphs and reports: SmoothWall allows you to run some simple reports on firewall activity and generate graphs based on this data.
- Support for additional connection types: SmoothWall supports many types of interfaces including dial-up, cable, ADSL, ISDN, and Ethernet. Some of these interfaces require additional software and configuration when supported under Ipchains.

One major difference between SmoothWall and the programs mentioned earlier is that SmoothWall needs to run on a dedicated machine. When you install SmoothWall, it wipes everything off the hard disk and installs its own operating system. This is basically a stripped down and hardened version of Linux, but you don't have to know anything about it to run your SmoothWall firewall. This means you won't be able to run any other tools on that machine or use it for anything else (at least not without a lot of hassle and the potential of breaking the SmoothWall software), so it may not be the right fit for everyone. But if you are looking for a cheap and quick way to set up a turnkey firewall with a lot of features, SmoothWall may be right for you.

SmoothWall Hardware Requirements

As mentioned earlier, SmoothWall needs a dedicated machine to run on. The good news is that the requirements for this machine are quite low since it will be running only the firewall software. The minimum specifications required for SmoothWall are a Pentium-class Intel-compatible PC running at 200Mhz or higher with at least 32MB of RAM and 512MB of disk space. A more optimal configuration would be a 500Mhz processor with 64MB of RAM and 2GB of disk space. These specifications should be easy to meet on all but the oldest machines.You will also need a CD-ROM drive and at least one network card (typically two, if the WAN interface is Ethernet).

SmoothWall Express Versus SmoothWall Corporate

If you have a little money to spend and are considering other commercial alternatives, you might look at the SmoothWall Corporate edition. This firewall has all the benefits of the Express version with the following important differences:

- Enhanced IDS support
- Connection fail-over capabilities
- VPN roaming support (dynamic IPs)
- Additional graphs and reports
- Enhanced graphical user interface
- Certificate authentication support for VPN

You can see a complete list of the differences at

http://download.smoothwall.org/archive/docs/promo/CorporateServer_vs_Express_Comparison_20040113.pdf.

Pricing for the commercial version is quite reasonable (check the Web site for the latest prices). The cost is significantly less than what you'd pay to buy a server to run it on. SmoothWall also makes other software products for network monitoring and content filtering. Check out their full product line at www.smoothwall.net.

Installing SmoothWall

Caution: Remember, installing SmoothWall will erase any data on the hard disk and put its own operating system on it. Do not run this installation on a computer on which you have data or programs you need.

1. You must first create a bootable CD-ROM disk. To do this, use CD-writing software, such as Nero or Easy CD Creator, and create a disk from the .iso image file from the SmoothWall directory on the CD-ROM that accompanies this book. The disk it creates will be bootable.
2. Set your PC to boot from the CD-ROM first. Otherwise, it will search the hard drive and load the operating system it finds there. You usually do this in the BIOS settings of a PC accessed at boot-up before the OS loads. Many PCs use the F2 function key to enter this mode.
3. Boot the machine from the CD-ROM. A title screen displays some basic licensing and disclaimer information. Click on OK.

   You have the choice of loading from the CD-ROM or HTTP. Remember, do not enter this mode unless you are ready for all the data on that hard disk to be erased and replaced with the SmoothWall software.

   Choose CD-ROM, and the installation will begin.