Red Hat Linux 8.0

The Official Red Hat Linux Security Guide

Red Hat Linux 8.0: The Official Red Hat Linux Security Guide

Copyright © 2002 by Red Hat, Inc.

Red Hat, Inc.

1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

rhl-sg(EN)-8.0-Print-RHI (2002-08-30T11:29-0400)

Copyright © 2002 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at http://www.opencontent.org/openpub/).
Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.
Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.
Red Hat, Red Hat Network, the Red Hat "Shadow Man" logo, RPM, Maximum RPM, the RPM logo, Linux Library, PowerTools, Linux Undercover, RHmember, RHmember More, Rough Cuts, Rawhide and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.
Linux is a registered trademark of Linus Torvalds.
Motif and UNIX are registered trademarks of The Open Group.
Intel and Pentium are a registered trademarks of Intel Corporation. Itanium and Celeron are trademarks of Intel Corporation.
AMD, AMD Athlon, AMD Duron, and AMD K6 are trademarks of Advanced Micro Devices, Inc.
Netscape is a registered trademark of Netscape Communications Corporation in the United States and other countries.
Windows is a registered trademark of Microsoft Corporation.
SSH and Secure Shell are trademarks of SSH Communications Security, Inc.
FireWire is a trademark of Apple Computer Corporation.
All other trademarks and copyrights referred to are the property of their respective owners.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

Introduction v

- 1. Document Conventions v
- 2. More to Come viii
  - 2.1. Send in Your Feedback viii

I. A General Introduction to Security ix

- 1. Security Overview 11
  - 1.1. What is Computer Security? 11
  - 1.2. Security Controls 15
  - 1.3. Conclusion 16
- 2. Attackers and Risks 17
  - 2.1. Hackers and Crackers 17
  - 2.2. Threats To Network Security 17
  - 2.3. Threats To Server Security 19
  - 2.4. Threats To Workstation and Home PC Security 21

II. Configuring Red Hat Linux for Security 23

- 3. Security Updates 25
  - 3.1. Using Red Hat Network 25
  - 3.2. Using the Errata Website 25
- 4. Workstation Security 27
  - 4.1. Evaluating Workstation Security 27
  - 4.2. BIOS and Boot Loader Security 27
  - 4.3. Password Security 30
  - 4.4. Administrative Controls 35
  - 4.5. Available Network Services 40
  - 4.6. Personal Firewalls 42
  - 4.7. Security Enhanced Communication Tools 43
- 5. Server Security 45
  - 5.1. Securing Services With TCP Wrappers and `xinetd` 45
  - 5.2. Securing Portmap 47
  - 5.3. Securing NIS 48
  - 5.4. Securing NFS 50
  - 5.5. Securing Apache HTTP Server 51
  - 5.6. Securing FTP 52
  - 5.7. Securing Sendmail 55
  - 5.8. Verifying Which Ports Are Listening 56
- 6. Virtual Private Networks 59
  - 6.1. VPNs and Red Hat Linux 59
  - 6.2. Crypto IP Encapsulation (CIPE) 59
- 7. Firewalls 67
  - 7.1. Netfilter and `iptables` 68
  - 7.2. `ip6tables` 72
  - 7.3. Additional Resources 73
- 8. Hardware and Network Protection 75
  - 8.1. Secure Network Topologies 75
  - 8.2. Hardware Security 78

III. Assessing Your Security 79

- 9. Vulnerability Assessment 81
  - 9.1. Thinking Like the Enemy 81
  - 9.2. Defining Assessment and Testing 81
  - 9.3. Evaluating the Tools 83

IV. Intrusions and Incident Response **87**
  10. Intrusion Detection........ 89
    10.1. Defining Intrusion Detection Systems 89
    10.2. Host-based IDS 89
    10.3. Network-based IDS........ 91
  11. Incident Response 95
    11.1. Defining Incident Response 95
    11.2. Creating an Incident Response Plan 95
    11.3. Implementing the incident response Plan 96
    11.4. Investigating the Incident........ 97
    11.5. Restoring and Recovering Resources 99
    11.6. Reporting the Incident........ 99
V. Appendixes **101**
  A. Common Exploits and Attacks 103
Index **107**
Colophon **111**

Introduction

Welcome to the *Official Red Hat Linux Security Guide*!

The *Official Red Hat Linux Security Guide* is designed to assist users of Red Hat Linux in learning the process and practice of securing workstations and servers against local and remote intrusion, exploitation, and malicious activity. The *Official Red Hat Linux Security Guide* details the planning and the tools involved in creating a secured computing environment for the data center, workplace, and home. With the proper knowledge, vigilance, and tools, systems running Red Hat Linux can be both fully functional and secured from most common intrusion and exploit methods.

This guide discusses several security-related topics in great detail, including:

- Firewalls
- Encryption
- Securing Critical Services
- Virtual Private Networks
- Intrusion Detection

We would like to thank **Thomas Rude** for his generous contributions to this manual. He wrote the *Vulnerability Assessments* and *Incident Response* chapters. Rock on, "farmerdude."

This manual assumes that you have an advanced knowledge of Red Hat Linux. If you are a new user or have basic to intermediate knowledge of Red Hat Linux and would like more information about how to use Red Hat Linux, please refer to the following guides, which discuss the fundamental aspects of Red Hat Linux in greater detail than the *Official Red Hat Linux Security Guide*:

- *Official Red Hat Linux Installation Guide* for information regarding installation
- *Official Red Hat Linux Getting Started Guide* to learn about how to use Red Hat Linux and its many applications
- *Official Red Hat Linux Customization Guide* for more detailed information about configuring Red Hat Linux to suit your particular needs as a user. This guide includes some services that are discussed (from a security standpoint) in the *Official Red Hat Linux Security Guide*.
- *Official Red Hat Linux Reference Guide* provides detailed information suited for more experienced users to refer to when needed, as opposed to step-by-step instructions.

HTML and PDF versions of all Official Red Hat Linux manuals are available online at http://www.redhat.com/docs/.

Note

> Although this manual reflects the most current information possible, you should read the *Red Hat Linux Release Notes* for information that may not have been available prior to our documentation being finalized. They can be found on the Red Hat Linux CD #1 and online at:
>
> `http://www.redhat.com/docs/manuals/linux`

1. Document Conventions

When you read this manual, you will see that certain words are represented in different fonts, typefaces, sizes, and weights. This highlighting is systematic; different words are represented in the same style to indicate their inclusion in a specific category. The types of words that are represented this way include the following:

`command`

Linux commands (and other operating system commands, when used) are represented this way. This style should indicate to you that you can type the word or phrase on the command line and press [Enter] to invoke a command. Sometimes a command contains words that would be displayed in a different style on their own (such as filenames). In these cases, they are considered to be part of the command, so the entire phrase will be displayed as a command. For example:

Use the `cat testfile` command to view the contents of a file, named `testfile`, in the current working directory.

`filename`

Filenames, directory names, paths, and RPM package names are represented this way. This style should indicate that a particular file or directory exists by that name on your Red Hat Linux system. Examples:

The `.bashrc` file in your home directory contains bash shell definitions and aliases for your own use.

The `/etc/fstab` file contains information about different system devices and filesystems.

Install the `webalizer` RPM if you want to use a Web server log file analysis program.

application

This style should indicate to you that the program named is an end-user application (as opposed to system software). For example:

Use **Mozilla** to browse the Web.

[key]

A key on the keyboard is shown in this style. For example:

To use [Tab] completion, type in a character and then press the [Tab] key. Your terminal will display the list of files in the directory that start with that letter.

[key]-[combination]

A combination of keystrokes is represented in this way. For example:

The [Ctrl]-[Alt]-[Backspace] key combination will exit your graphical session and return you to the graphical login screen or the console.

text found on a GUI interface

A title, word, or phrase found on a GUI interface screen or window will be shown in this style. When you see text shown in this style, it is being used to identify a particular GUI screen or an element on a GUI screen (such as text associated with a checkbox or field). Example:

Select the **Require Password** checkbox if you would like your screensaver to require a password before stopping.

top level of a menu on a GUI screen or window

When you see a word in this style, it indicates that the word is the top level of a pulldown menu. If you click on the word on the GUI screen, the rest of the menu should appear. For example:

Under **File** on a GNOME terminal, you will see the **New Tab** option that allows you to open multiple shell prompts in the same window.

If you need to type in a sequence of commands from a GUI menu, they will be shown like the following example:

Go to **Main Menu Button** (on the Panel) => **Programming** => **Emacs** to start the **Emacs** text editor.

button on a GUI screen or window

This style indicates that the text will be found on a clickable button on a GUI screen. For example:

Click on the **Back** button to return to the webpage you last viewed.

`computer output`

When you see text in this style, it indicates text displayed by the computer on the command line. You will see responses to commands you typed in, error messages, and interactive prompts for your input during scripts or programs shown this way. For example:

Use the `ls` command to display the contents of a directory:

```
$ ls
Desktop                 about.html       logs          paulwesterberg.png
Mail                    backupfiles      mail          reports
```

The output returned in response to the command (in this case, the contents of the directory) is shown in this style.

`prompt`

A prompt, which is a computer's way of signifying that it is ready for you to input something, will be shown in this style. Examples:

`$`

`#`

`[stephen@maturin stephen]$`

`leopard login:`

`user input`

Text that the user has to type, either on the command line, or into a text box on a GUI screen, is displayed in this style. In the following example, **`text`** is displayed in this style:

To boot your system into the text based installation program, you will need to type in the **`text`** command at the `boot:` prompt.

Additionally, we use several different strategies to draw your attention to certain pieces of information. In order of how critical the information is to your system, these items will be marked as note, tip, important, caution, or a warning. For example:

Note

Remember that Linux is case sensitive. In other words, a rose is not a ROSE is not a rOsE.

Tip

The directory `/usr/share/doc` contains additional documentation for packages installed on your system.

Important

If you modify the DHCP configuration file, the changes will not take effect until you restart the DHCP daemon.

Caution

Do not perform routine tasks as root — use a regular user account unless you need to use the root account for system administration tasks.

Warning

If you choose not to partition manually, a server installation will remove all existing partitions on all installed hard drives. Do not choose this installation class unless you are sure you have no data you need to save.

2. More to Come

The *Official Red Hat Linux Security Guide* is part of Red Hat's growing commitment to provide useful and timely support to Red Hat Linux users. As new tools and security methodologies are released, this guide will be expanded to include them.

2.1. Send in Your Feedback

If you spot a typo in the *Official Red Hat Linux Security Guide*, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla (http://www.redhat.com/bugzilla) against the component `rhl-sg`.

Be sure to mention the manual's identifier:

```
rhl-sg(EN)-8.0-Print-RHI (2002-08-30T11:29-0400)
```

If you mention this manual's identifier, we will know exactly which version of the guide you have.

If you have a suggestion for improving the documentation, try to be as specific as possible. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

A General Introduction to Security

Chapter 1.

Security Overview

Because of the increased reliance on powerful, networked computers to help run businesses and keep track of our personal information, industries have been formed around the practice of network and computer security. Enterprises have solicited the knowledge and skills of security experts to properly audit systems and tailor solutions to fit the operating requirements of the organization. Because most organizations are dynamic in nature, with workers accessing company IT resources locally and remotely, the need for secure computing environments has become more pronounced.

Unfortunately, most organizations (as well as individual users) regard security as an afterthought, a process that is overlooked in favor of increased power, productivity, and budgetary concerns. Proper security implementation is often enacted "postmortem" — after an unauthorized intrusion has already occurred. Security experts agree that the right measures taken prior to connecting a site to an untrusted network such as the Internet is an effective means of thwarting most attempts at intrusion.

1.1. What is Computer Security?

Computer security is a general term that covers a wide area of computing and information processing. Industries that depend on computer systems and networks to conduct daily business transactions and access crucial information regard their data as an important part of their overall assets. Several terms and metrics have entered our daily business lives, such as total cost of ownership (TCO) and quality of service (QoS). In those metrics, industries calculate aspects such as data integrity and high-availability as part of their planning and process management costs. In some industries, such as electronic commerce, the availability and trustworthiness of data can be the difference between success and failure.

1.1.1. How did Computer Security Come about?

Many readers may recall the movie "Wargames", starring Matthew Broderick in his portrayal of a high school student that breaks into the United States Department of Defense (DOD) supercomputer and inadvertently causes a nuclear war threat. In this movie, Broderick uses his modem to dial into the DOD computer (called WOPR) and plays games with the artificially intelligent software controlling all of the nuclear missile silos. The movie was released during the "cold war" between the former Russian Republic and the United States, and was considered a success in its theatrical release in 1983. The popularity of the movie inspired many individuals and groups to begin implementing some of the methods that the young protagonist used to crack restricted systems, including what is known as *war dialing* — a method of searching phone numbers for analog modem connections in an defined area code and phone prefix combination.

More than 10 years later, after a four-year, multi-jurisdictional pursuit involving the Federal Bureau of Investigation (FBI) and the aid of computer professionals across the country, infamous computer cracker Kevin Mitnick was arrested and charged with 25 counts of computer and access device fraud that resulted in an estimated US$80 Million in losses of intellectual property and source code from Nokia, NEC, Sun Microsystems, Novell, Fujitsu, and Motorola.At the time, the FBI considered it the largest single computer-related criminal offense in U.S. history. He was convicted and sentenced to a combined 68 months in prison for his crimes, of which he served 60 months before his parole on January 21, 2000. He has been further barred from using computers or doing any computer-related consulting until 2003. Investigators say that Mitnick was an expert in *social engineering* — using human beings to gain access to passwords and systems using falsified credentials.

Information security has evolved over the years due to the increasing reliance on public networks to disclose personal, financial, and other restricted information. There are numerous instances such as the Mitnick and the Vladamir Levin case that prompted organizations across all industries to rethink the way they handle information transmission and disclosure. The popularity of the Internet was one of the most important developments that prompted an intensified effort in data security.

An ever-growing number of people are using their personal computers to gain access to the resources that the Internet has to offer. From research and information retrieval to electronic mail and commerce transaction, the Internet has been regarded as one of the most important developments of the 20th century.

The Internet and its earlier protocols, however, were developed as a *trust-based* system. That is, the Internet Protocol was not designed to be secure in itself. There are no approved security standards built into the TCP/IP communications stack, leaving it open to potentially malicious users and processes across the network. Modern developments have made Internet communication more secure, but there are still several incidents that gain national attention and alert us to the fact that nothing is completely safe.

1.1.2. Computer Security Timeline

Several key events contributed to the birth and rise of computer security. The following lists some of the most important events that brought attention to computer and information security and its importance today.

1.1.2.1. The 1960s

- Students at the Massachusetts Institute of Technology (MIT) form the Tech Model Railroad Club (TMRC), which coin the term "hacker" in the context it is known today and begin exploring and programming the school's PDP-1 mainframe computer system.
- The DoD creates the Advanced Research Projects Agency Network (ARPANet), which gains popularity in research and academic circles as a conduit for the electronic exchange of data and information. This paves the way for the creation of the carrier network known today as the Internet.
- Ken Thompson develops the UNIX operating system, widely hailed as the most "hacker-friendly" OS because of its accessible developer tools and compilers and its supportive user community. Around the same time, Dennis Ritchie develops the C programming language, arguably the most popular hacking language in computer history.

1.1.2.2. The 1970s

- Bolt, Beranek, and Newman, a computing research and development contractor for government and industry, develops the telnet protocol, a public extension of the ARPANet. This opens doors to public use of data networks once restricted to government contractors and academic researchers. Telnet, though, is also arguably the most insecure protocol for public networks, according to several security researchers.
- Steve Jobs and Steve Wozniak found Apple Computer and begin marketing the Personal Computer (PC). The PC is the springboard for several malicious users to learn the craft of cracking systems remotely using common PC communication hardware such as analog modems and war dialers.
- Jim Ellis and Tom Truscott create USENET, a bulletin-board style system for electronic communication between disparate users. USENET quickly becomes one the most popular forums for the exchange of ideas in computing, networking, and, of course, cracking.

1.1.2.3. The 1980s

- IBM develops and markets PCs based on the Intel 8086 microprocessor, a relatively inexpensive architecture that brought computing from the office to the home. This serves to commodify the PC as a common and accessible *household* tool that was fairly powerful and easy to use, aiding in the proliferation of such hardware in the homes and offices of malicious users.
- The Transmission Control Protocol, developed by Vint Cerf, is split into two separate parts. The Internet Protocol is born of this split, and the combined TCP/IP protocol becomes the standard for all Internet communication today.
- Based on developments in the area of *phreaking*, or exploring and hacking the telephone system,the magazine *2600: The Hacker Quarterly* is created and begins discussion on topics such as hacking computers and computer networks to a broad audience.
- The 414 gang (named after the area code where they lived and hacked from) are raided by authorities after a nine-day cracking spree where they break into systems from such top-secret locations as the Los Alamos National Laboratory, a nuclear weapons research facility.
- The Legion of Doom and the Chaos Computer Club are two pioneering hacker groups that begin exploiting vulnerabilities in computers and electronic data networks.
- The Computer Fraud and Abuse Act of 1986 was voted into law by congress based on the exploits of Ian Murphy, also known as Captain Zap, who broke into military computers, stole information from company merchandise order databases, and used restricted government telephone switchboards to make phone calls.
- Based on the Computer Fraud and Abuse Act, the courts were able to convict Robert Morris, a graduate student, for unleashing the Morris Worm to over 6,000 vulnerable computers connected to the Internet. The next most prominent case ruled under this act was Herbert Zinn, a high-school dropout who cracked and misused systems belonging to AT&T and the DoD.
- Based on concerns that the Morris Worm ordeal could be replicated, the Computer Emergency Response Team (CERT) is created to alert computer users of network security issues.
- Clifford Stoll writes *The Cuckoo's Egg*, Stoll's account of investigating crackers who exploit his system.

1.1.2.4. The 1990s

- ARPANet is decommissioned. Traffic from that network is transferred to the Internet.
- Linus Torvalds develops the Linux kernel for use with the GNU operating system; the widespread development and adoption of Linux is largely due to the collaboration of users and developers communicating via the Internet. Because of its roots in Unix, Linux is most popular amongst hackers and administrators who found it quite useful for building secure alternatives to legacy servers running proprietary (closed-source) operating systems.
- The graphical Web browser is created and sparks an exponentially higher demand for public Internet access.
- Vladimir Levin and accomplices illegally transfer US$10 Million in funds to several accounts by cracking into the CitiBank central database. Levin is arrested by Interpol and almost all of the money is recovered.
- Possibly the most heralded of all hackers is Kevin Mitnick, who hacked into several corporate systems, stealing everything from personal information of celebrities to over 20,000 credit card numbers and source code for proprietary software. He is caught and convicted of wire fraud charges and serves 5 years in prison.

- Kevin Poulsen and an unknown accomplice rigs radio station phone systems to win cars and cash prizes. He is convicted for computer and wire fraud and is sentenced to 5 years in prison.
- The stories of hacking and phreaking become legend, and several prospective hackers convene at the annual DefCon convention to celebrate hacking and exchange ideas between peers.
- A 19-year-old Israeli student is arrested and convicted for coordinating numerous break-ins to US government systems during the Persian-Gulf conflict. Military officials call it "the most organized and systematic attack" on government systems in US history.
- US Attorney General Janet Reno, in response to escalated security breaches in government systems, establishes the National Infrastructure Protection Center.
- British communications satellites are taken over and ransomed by unknown offenders. The British government eventually seizes control of the satellites.

1.1.3. Security Today

In February of 2000, a Distributed Denial of Service (DDoS) attack was unleashed on several of the most heavily-trafficked sites on the Internet. The attack rendered yahoo.com, cnn.com, amazon.com, fbi.gov, and several other sites completely unreachable to normal users, as it tied up routers for several hours with large-byte ICMP packet transfers, also called a *ping flood*. The attack was brought on by unknown assailants using specially created, widely available programs that scanned vulnerable network servers, installed client applications called *trojans* on the servers, and timed an attack with every infected server flooding the victim sites and rendering them unavailable. Many blame the attack on fundamental flaws in the way routers and the protocols used are structured to accept all incoming data, no matter where or for what purpose the packets are sent.

This brings us to the new millennium, a time where an estimated 400 Million people use or have used the Internet worldwide. At the same time:

- On any given day, there are an estimated 142 major incidences of vulnerability exploits reported to the CERT Coordination Center at Carnegie Mellon University [source: http://www.cert.org]
- In 2001 alone, the number of CERT reported incidences doubled to 52,658 from 21,756 in 2000 [source: http://www.cert.org]
- The search engine Google finds 2,040,000 Web pages containing the term *hackers* [source: http://www.google.com]
- The worldwide economic impact of the three most dangerous Internet Viruses of the last two years was a combined US$13.2 Billion and rising (due to the insidious nature of the still-active Nimda worm) [source: http://www.computereconomics.com]

Computer security has become a quantifiable and justifiable expense for all IT budgets. Organizations that require data integrity and high availability elicit the skills of system administrators, developers, and engineers to ensure 24x7 reliability of their systems, services, and information. To fall victim to malicious users, processes, or coordinated attacks is a direct threat to the success of the organization.

Unfortunately, system and network security can be a difficult proposition, requiring an intricate knowledge of how an organization regards, uses, manipulates, and transmits its information. Understanding the way an organization (and the people that make up the organization) conducts business is paramount to implementing a proper security plan.

1.1.4. Standardizing Security

Enterprises in every industry rely on regulations and rules that are set by standards making bodies such as the American Medical Association (AMA) or the Institute of Electrical and Electronics Engineers

(IEEE). The same ideals hold true for information security. Many security consultants and vendors agree upon the standard security model known as CIA, or *Confidentiality, Integrity, and Availability*. This three-tiered model is a generally accepted component to assessing risks to sensitive information and establishing security policy. The following describes the CIA model in greater detail:

- Confidentiality — Sensitive information must be available only to a set of pre-defined individuals. Unauthorized transmission and usage of information should be restricted. For example, confidentiality of information ensures that a customer's personal or financial information is not obtained by an unauthorized individual for malicious purposes such as identity theft or credit fraud.
- Integrity — Information should not be altered in ways that render it incomplete or incorrect. Unauthorized users should be restricted from the ability to modify or destroy sensitive information.
- Availability — Information should be accessible to authorized users any time that it is needed. Availability is a warranty that information can be obtained with an agreed-upon frequency and timeliness. This is often measured in terms of percentages and agreed to formally in Service Level Agreements (SLAs) used by network service providers and their enterprise clients.

1.2. Security Controls

Computer security is often divided into three distinct master categories, commonly referred to as *controls*:

- Physical
- Technical
- Administrative

These three broad categories define the main objectives of proper security implementation. Within these controls are sub-categories that further detail the controls and how to implement them.

1.2.1. Physical Controls

The physical control is the implementation of security measures in a defined structure used to deter or prevent unauthorized access to sensitive material. Examples of physical controls are:

- Closed-circuit surveillance cameras
- Motion or thermal alarm systems
- Security guards
- Picture IDs
- Locked and dead-bolted steel doors

1.2.2. Technical Controls

The technical control uses technology as a basis for controlling the access and usage of sensitive data throughout a physical structure and over a network. Technical controls are far-reaching in scope and encompass such technologies as:

- Encryption
- Smart cards
- Network authentication

- Access control lists (ACLs)
- File integrity auditing software

1.2.3. Administrative Controls

Administrative controls define the human factors of security. It involves all levels of personnel within an organization and determines which users have access to what resources and information by such means as:

- Training and awareness
- Disaster preparedness and recovery plans
- Personnel recruitment and separation strategies
- Personnel registration and accounting

1.3. Conclusion

Now that you have learned a bit about the origins, reasons and aspects of security, you can determine the appropriate course of action with regards to Red Hat Linux. It is important to know what factors and conditions make up security in order to plan and implement a proper strategy. With this information in mind, the process can be formalized and the path becomes clearer as you delve deeper into the specifics of the security process.

Chapter 2.
Attackers and Risks

In order to plan and implement a good security strategy, you should first be aware of some of the issues which determined, motivated attackers exploit to compromise systems. Before detailing these issues, we will define the terminology used when identifying an attacker.

2.1. Hackers and Crackers

The modern meaning of the term *hacker* has origins dating back to the 1960s and the Massachusetts Institute of Technology (MIT) Tech Model Railroad Club, which designed train sets of large scale and intricate detail. Hacker was a name used for club members who discovered a clever trick or workaround for a problem.

The term hacker has since come to describe everything from computer buffs to gifted programmers. A common trait among most hackers is a willingness to explore in detail how computer systems and networks function with little or no outside motivation. Open source software developers often consider themselves and their colleagues hackers and use the word as a term of respect.

Hackers typically follow a form of the *hacker ethic* which dictates that quest for information and expertise is essential and that sharing this knowledge is the hackers duty to the community. During this quest for knowledge, some hackers enjoy the academic challenges of circumventing security controls on computer systems. For this reason, the press often uses the term hacker to describe those who illicitly access systems and networks with unscrupulous, malicious, or criminal intent. The more accurate term for this type of computer hacker is *cracker* — a term created by hackers in the mid-1980s to differentiate the two communities.

2.1.1. Shades of Grey

There are levels of distinction to describe individuals who find and exploit vulnerabilities in systems and networks. They are described by the shade of hat that they "wear" when performing their security investigations and this shade is indicative of their intent.

The *white hat hacker* is one who tests networks and systems to examine their performance and determine how vulnerable they are to intrusion. Usually, white hat hackers crack their own systems or the systems of a client who has specifically employed them for the purposes of security auditing. Academic researchers and professional security consultants are two examples of white hat hackers.

A *black hat hacker* is synonymous with a cracker. In general, crackers are less focused on programming and the academic side of breaking into systems. They often rely on available cracking programs and exploit well known vulnerabilities in systems to uncover sensitive information for personal gain or to inflict damage on the target system or network.

The *grey hat hacker*, on the other hand, has the skills and intent of a white hat hacker in most situations but uses his knowledge for less than noble purposes on occasion. A grey hat hacker can be thought of as a white hat hacker who wears a black hat at times to accomplish his own agenda.

Grey hat hackers typically subscribe to another form of the hacker ethic, which says it is acceptable to break into systems as long as the hacker does not commit theft or breach confidentiality. Some would argue, however that the act of breaking into a system is in itself unethical.

Regardless of the intent of the intruder, it is important to know the weaknesses a cracker will likely attempt to exploit. The remainder of the chapter will focus on these issues.

2.2. Threats To Network Security

By breaking down a network into its basic segments, we can determine the risks and define what is necessary to prevent unauthorized access.

2.2.1. Insecure Architectures

A misconfigured network is a primary entry point for unauthorized users. Leaving a trust-based, open local network vulnerable to the highly-insecure Internet is much like leaving a door ajar in a crime-ridden neighborhood — nothing may happen for an arbitrary amount of time, but *eventually* someone will exploit the opportunity.

2.2.1.1. Broadcast Networks

System administrators often fail to realize the importance of networking hardware in their security schemas. Simple hardware such as hubs and routers rely on the broadcast or non-switched principle; that is, whenever a node transmits data across the network to a recipient node, the hub or router sends a broadcast of the data packets until the recipient node receives and processes the data. This method is the most vulnerable to address resolution protocol (*arp*) or media access control (*MAC*) address spoofing by both outside intruders and unauthorized users on local nodes. For advice on choosing the right networking hardware and topology, refer to Chapter 8.

2.2.1.2. Centralized Servers

Another potential networking pitfall is the use of centralized computing. A common cost-cutting measure for many businesses is to consolidate all services to a single powerful machine. This can be convenient because it is easier to manage and costs considerably less than multiple-server configurations. However, a centralized server introduces a single point of failure on the network. So if the central server is compromised, it may render the network completely useless or worse, prone to data manipulation or theft. In these situations a central server becomes an an open door, allowing access to the entire network. Refer to Chapter 8 for more information about network segmentation and how they help you avoid an incident.

2.2.1.3. No Firewall

The least likely, but still common, mistake among administrators and home users is the assumption that their network is inherently secure and, thus, they forgo the implementation of a firewall or network packet filtering service. The installation of a dedicated firewall, whether standalone or as part of a server that will act as a gateway, is crucial to segmenting internal and external network traffic. Leaving the internal network exposed to the Internet, especially if the connection to the Internet is constant, is an open invitation to any Internet user that happens to find the network's external IP address. A cracker can potentially act as a node on your internal network or take over machines on the network to act as a proxy. Firewalls can help prevent this by using packet filtering, port forwarding, or *Network Address Translation* (*NAT*). They can also act as a proxy between the internal network and the Internet, further buffering the private network from the Internet. Refusing to implement a firewall or, perhaps more importantly, setting up a firewall incorrectly, leaves a network completely vulnerable. Refer to Chapter 7 for more information on configuring a firewall for your network.

2.2.2. Network Encryption

Password-protected applications and services are sound means of protecting a network. However, these passwords should never be passed over public networks unencrypted. This is because crackers use readably available tools to *sniff* network traffic for data such as passwords to gain access to private networks and services. Unfortunately, many applications and services (such as telnet and FTP) transmit passwords in *plain text* (also known as *clear text*) which makes them vulnerable to these network sniffing applications.

Encryption is a general method of scrambling data, such as passwords, in order to protect it in the event of interception. Depending on the encryption method used, it could conceivably take a cracker several thousand years to decrypt the data using conventional methods. Most encryption methods are done between the client application and the server, making the process transparent to all users. However, encryption is something that most people do not understand. Administrators feel that it is a nuisance to integrate into their network services, even though, in most cases, encrypting network traffic can be a relatively simple procedure. The advantages of using encryption vastly outweigh its liabilities.

2.2.3. Wireless Local Area Networks (WLANS)

The popularity of mobile technology has prompted engineers to develop new ways of connecting to and communicating with others. Cellular and radio-frequency (RF) technology has ushered a new age of wireless communication that boasts competitive speeds and functionality compared to wired or cabled communication solutions.

The recent IEEE 802.11b wireless protocol (*wi-fi*) has become an industry standard for users that need a more mobile networking solution. The 802.11b standard uses 2.4 GHz Direct Sequence Spread Spectrum (DSSS) frequency for communication. It also uses 40-bit Wired Equivalent Privacy (WEP) encryption of all data traffic. It seems to be the ideal solution for users who move frequently or do not have access to traditional RJ-45 or RJ-11 cabling lines.

There have been recent reports, however, that dispute the relative security of 802.11b and other WLAN technologies. One major drawback of wireless networking is that most wireless network interface cards (NICs) must be operated in *promiscuous mode* — that is, data packets must continually be broadcast in order for the wireless NIC to transmit and receive the packets that are intended for it. Moreover, the WEP encryption built into 802.11b NICs and Access Points is, by many estimates, a weak form of encryption that can be cracked using standard desktop or laptop PCs. Many WLAN administrators do not even enable the WEP encryption, making the ability to intercept data even easier. For general information on wireless security, refer to Chapter 8.

2.3. Threats To Server Security

Server security is as important as network security because servers can hold most or all of the organization's vital information. If a server is compromised, all of its contents may become available for the cracker to steal or manipulate at will. There are many ways that a server can be cracked. The following sections detail some of the main issues.

2.3.1. Unused Services and Open Ports

By default, most operating systems install several pieces of commonly used software. Red Hat Linux, for example, can install up to 1200 application and library packages in a single installation. While most server administrators will not opt to install every single package in the distribution, they will install a base installation of packages, including several server applications.

A common occurrence among system administrators is to install an operating system without knowing what is actually being installed. This can be troublesome, as most operating systems will not only install the applications, but also setup a base configuration and turn services on. This can cause unwanted services, such as telnet, DHCP, or DNS to be running on a server or workstation without the administrator realizing it, leading to unwanted traffic to the server or even a path into the system for crackers. See Chapter 5 for information on closing ports and disabling unused services.

2.3.2. Unpatched Services

Most server applications that are included in a default Red Hat Linux installation are solid, thoroughly tested pieces of software. Many of the server applications have been in use in production environments for many years, and their code has been thoroughly refined and many of the bugs have been found and fixed.

However, there is no such thing as perfect software, and there is always room for further refinement. Moreover, newer software is often not as rigorously tested as one might expect, due to its recent arrival to production environments or because it may not be as popular as other server software. Developers and system administrators often find exploitable bugs in server applications and publish the information on bug tracking and security-related websites such as the Bugtraq mailing list or the Computer Emergency Response Team website. CERT and Bugtraq normally alert interested parties of the vulnerabilities. However, even then, it is up to system administrators to patch and fix these bugs whenever they are made public, as crackers also have access to these vulnerability tracking services and will use such information to crack unpatched systems wherever they can. Good system administration requires vigilance, constant tracking of bugs, and proper system maintenance to ensure a secure computing environment.

2.3.3. Inattentive Administration

Similar to server applications which languish unpatched by developers are administrators who fail to patch their systems or are too ignorant to do so. According to the *System Administration Network and Security Institute* (*SANS*), the primary cause of computers security vulnerability is to "assign untrained people to maintain security and provide neither the training nor the time to make it possible to do the job."[1] This applies as much to inexperienced administrators as it does to overconfident or amotivated administrators.

Some administrators fail to patch their servers and workstations, while others fail to watch log messages from their system kernel or from network traffic. Another common error is to leave the default passwords or keys in services that have such authentication methods built into them. For example, some databases leave default administration passwords under the assumption that the system administrator will change this immediately upon configuration. Even an inexperienced cracker can use the widely-known default password to gain administrative privileges to the database. These are just a few-examples of inattentive administration that can eventually lead to a compromised system.

2.3.4. Inherently Insecure Services

Even the most vigilant organization that does their job well and keeps up with their daily responsibilities can fall victim to vulnerabilities if the services they choose for their network are inherently insecure. There are certain services that were developed under the assumption that they will be used over trusted networks; however, this assumption falls short as soon as the service becomes available over the Internet.

Some examples of inherently insecure services include servers that require passwords or passphrases for authentication (in itself, a secure feature), but fail to encrypt the passwords as they are sent over the wire to the authenticating service. Telnet and FTP are two such services. A packet sniffing device set between a remote user and the telnet server can easily be set to steal passwords (especially if the telnet user happens to switch to an administrative user during a telnet session).

The services noted above can also more easily fall prey to what the security industry terms the *man-in-the-middle* attack. In this type of attack, a cracker redirects network traffic by tricking a cracked name server on the network to point to his machine instead of the intended server. Once someone opens a remote session to that server, the attacker's machine acts as an invisible conduit, sitting quietly between the remote service and the unsuspecting user capturing information. This way a cracker can gather administrative passwords and raw data without either the server's or the user's knowledge.

---

[1] Source: http://www.sans.org/newlook/resources/errors.html

Another example of insecure services are network file systems and information services such as NFS or NIS which are developed explicitly for LAN usage but are, unfortunately, extended to include WANs (for remote users). NFS does not, by default, have any authentication or security mechanisms configured that will prevent a cracker from simply mounting the NFS share and accessing anything contained therein. NIS, as well, has vital information that must be known by every computer on a network, including passwords and file permissions, within a plain text ACSII or DBM (ASCII-derived) database. A cracker can take this database and find the passwords of each and every user on a network, including the administrator.

2.4. Threats To Workstation and Home PC Security

Workstations and home PCs may not be as prone to attack as networks or servers, but they may contain sensitive information, such as credit card information, that would be damaging if stolen. They may also be used by attackers as a "slave" machine in coordinated attacks, without the user's knowledge. Knowing the vulnerabilities of your workstation can save you the headache of having to reinstall your operating system — or having your administrator do it for you.

2.4.1. Bad Passwords

Bad passwords are not invalid when running Red Hat Linux. However, a bad password is one of the easiest ways for an attacker to gain access to a system. For more on how to avoid common pitfalls when creating a password, see Section 4.3.

2.4.2. Vulnerable Client Applications

Although an administrator may have a fully secure and patched server, that does not mean that remote users are secure when accessing it. For instance, if the server offers Telnet or FTP services over a public network, an attacker can capture the plain text usernames and passwords as they pass over the network, and then use the account information to access the remote user's workstation.

Even when using secure protocols, such as SHH, a remote user may be vulnerable to certain attacks if they do not keep their client applications updated. For instance, v.1 SSH clients are vulnerable an X-forwarding attack from malicious SSH servers. Once connected to the server, the attacker can quietly capture any keystrokes and mouse clicks made by the client over the network. This problem was fixed in the v.2 SSH protocol, but it is up to the user to keep track of what applications have such vulnerabilities and update them as necessary.

Chapter 4 will discuss in more detail what steps administrators and home users should take to limit the vulnerability of computer workstations.

Configuring Red Hat Linux for Security

Chapter 3.

Security Updates

As security exploits in software are discovered, the software must be fixed to close the possible security risk. If the package is part of an official Red Hat Linux distribution that is currently supported, Red Hat, Inc. is committed to releasing official updated packages that fix security holes as soon as possible. If the announcement of the security exploit is accompanied with a patch (or source code that fixes the problem), the patch is applied to the Red Hat Linux package, tested by the quality assurance team, and released as an official errata update. If the announcement does not include a patch, a Red Hat Linux developer will work with the maintainer of the package to fix the problem. After the problem is fixed, it is tested and released as an official errata update.

If you are using a package for which a security errata report is released, it is highly recommended that you update to the security errata packages as soon as they are released to minimize the time your system is exploitable.

Not only do you want to update to the latest packages that fix any security exploits, but you also want to make sure the latest packages do not contain further exploits such as a trojan horse. A cracker can easily rebuild a version of a package (with the same version number as the one that is supposed to fix the problem) but with a different security exploit in the package and release it on the Internet. If this happens, using security measures such as verifying files against the original RPM will not detect the exploit. Thus, it is very important that you only download RPMs from official sources, such as from Red Hat, Inc., and check the signature of the package to make sure it was built by the official source.

Red Hat offers two ways to retrieve official security updates:

1. Download from Red Hat Network
2. Downloaded from the official Red Hat Linux Errata website

3.1. Using Red Hat Network

Red Hat Network allows you to automate most of the update process. It determines which RPM packages are necessary for your system, downloads them from a secure repository, verifies the RPM signature to make sure they have not been tampered with, and updates them. The package install can occur immediately or can be scheduled during a certain time period.

Red Hat Network requires you to provide a System Profile for each machine that you want updated. The System Profile contains hardware and software information about the system. This information is kept confidential and not give to anyone else. It is only used to determine which errata updates are applicable to each system. Without it, Red Hat Network can not determine whether your system needs updates. When a security errata (or any type of errata) is released, Red Hat Network will send you an email with a description of the errata as well as which of your systems are affected. To apply the update, you can use the **Red Hat Update Agent** or schedule the package to be updated through the website http://rhn.redhat.com.

To learn more about the benefits of Red Hat Network, refer to the *Red Hat Network Reference Guide* available at http://www.redhat.com/docs/manuals/RHNetwork/ or visit http://rhn.redhat.com.

3.2. Using the Errata Website

When security errata reports are released, they are published on the official Red Hat Linux Errata website available at http://www.redhat.com/apps/support/errata/. From this page, select the product and version for your system, and then select **security** at the top of the page to display only Red Hat

Linux Security Advisories. If the synopsis of one of the advisories describes a package used on your system, click on the synopsis for more details.

The details page describes the security exploit and any special instructions that must be performed in addition to updating the package to fix the security hole.

To download the updated package(s), click on the package name(s) and save to the hard drive. It is highly recommended that you create a new directory such as `/tmp/updates` and save all the downloaded packages to it.

All official Red Hat Linux packages are signed with the Red Hat, Inc. GPG key. The RPM utility in Red Hat Linux 8.0 automatically tries to verify the GPG signature of an RPM before installing it. If you do not have the Red Hat, Inc. GPG key installed, install it from a secure, static location such as an official Red Hat Linux distribution CD-ROM.

Assuming the CD-ROM is mounted in `/mnt/cdrom`, use the following command to import it into the keyring:

```
rpm --import /mnt/cdrom/RPM-GPG-KEY
```

To display a list of all keys installed for RPM verification, execute the command:

```
rpm -qa gpg-pubkey*
```

For the Red Hat, Inc. key, the output will include:

```
gpg-pubkey-db42a60e-37ea5438
```

To display details about a specific key, use the `rpm -qi` followed by the output from the previous command:

```
rpm -qi gpg-pubkey-db42a60e-37ea5438
```

It is extremely important that you verify the signature of the RPM files before installing them. This step ensures that they have not been altered (such as a trojan horse being inserted into the packages) from the official Red Hat, Inc. release of the packages. To verify all the downloaded packages at once:

```
rpm -K /tmp/updates/*.rpm
```

For each package, if the GPG key verifies successfully, it should return `gpg OK` in the output.

After verifying the GPG key and downloading all the packages associated with the errata report, install them as root at a shell prompt. For example:

```
rpm -Uvh /tmp/updates/*.rpm
```

If the errata reports contained any special instructions, remember to execute them accordingly. If the security errata packages contained a kernel package, be sure to reboot the machine to enable the new kernel.

Chapter 4.

Workstation Security

Securing a Linux environment begins with the workstation. Whether you are locking down your own personal machine or securing an enterprise system, sound security policy begins with the individual computer. After all, a computer network is only as secure as the weakest node.

4.1. Evaluating Workstation Security

When evaluating the security of a Red Hat Linux workstation, consider the following:

- *BIOS and Boot Loader Security* — Can an unauthorized user physically access the machine and boot into single user or rescue mode without a password?
- *Password Security* — How secure are the user account passwords on the machine?
- *Administrative Controls* — Who has an account on the system and how much administrative control do they have?
- *Available Network Services* — What services are listening for requests from the network and should they be running at all?
- *Personal Firewalls* — What type of firewall, if any, is necessary?
- *Security Enhanced Communication Tools* — What tools should be used to communicate between workstations and what should be avoided?

4.2. BIOS and Boot Loader Security

Password protection for the BIOS and the boot loader can prevent unauthorized users who have physical access to your systems from booting from removable media or attaining root through single user mode. But the security measures one should take to protect against such attacks depends both on the sensitivity of the information the workstation holds and the location of the machine.

For instance, if a machine is used in a trade show and contains no sensitive information, than it may not be critical to prevent such attacks. However, if an employee's laptop with private, non-password protected SSH keys for the corporate network is left unattended at that same trade show, it can lead to a major security breech with ramifications for the entire company.

On the other hand, if the workstation is located in a place where only authorized or trusted people have access, then securing the BIOS or the boot loader may not be necessary at all.

4.2.1. BIOS Passwords

The following are the two primary reasons for password protecting the BIOS of a computer[1]:

1. *Prevent Changes To BIOS Settings* -- If an intruder has access to the BIOS, they can set it to boot off of a diskette or CD-ROM. This makes it possible for them to enter rescue mode or single user mode, which in turn allows them to seed nefarious programs on the system or copy sensitive data.

---

[1] Since system BIOSes differ between manufacturers, some may not support password protection of either type, while others may support one type and not the other.

2. *Prevent Booting the System* -- Some BIOSes allow you to password protect the boot process itself. When activated, an attacker would be forced to enter a password for the BIOS to launch the boot loader.

Because the methods for setting a BIOS password vary between computer manufacturers, you should consult the manual for your computer.

If you forget the BIOS password, it can often be reset either with jumpers on the motherboard or by disconnecting the CMOS battery. However, you should check the manual for your computer or motherboard before attempting this procedure.

4.2.2. Boot Loader Passwords

The following are the primary reasons for password protecting a Linux boot loader:

1. *Prevent Access To Single User Mode* — If an attacker can boot into single user mode, he becomes the root user.
2. *Prevent Access To the GRUB Console* — If the machine uses GRUB as its boot loader, an attacker can use the edit the command's interface to change its configuration or to gather information using the `cat` command.
3. *Prevent Access To Non-Secure Operating Systems* — If it is a dual boot system, an attacker can select at boot time an operating system, such as DOS, which ignores access controls and file permissions.

There are two boot loaders that ship with Red Hat Linux, GRUB and LILO. The next two sections will describe how to password protect these applications.

4.2.2.1. Password Protecting GRUB

You can configure GRUB to address the first two issues listed in Section 4.2.2 by adding a password directive to its configuration file. To do this, first decide on a password, then open a shell prompt, log in as root, and type:

```
/sbin/grub-md5-crypt
```

When prompted, type the GRUB password and press [Enter]. This will return an MD5 hash of the password.

Next, edit the GRUB configuration file: `/boot/grub/grub.conf`. Open the file and below the timeout line in the main section of the document, add the following line:

```
password --md5 password-hash
```

Replace *`password-hash`* with the value returned by `/sbin/grub-md5-crypt`[2].

The next time you boot the system, the GRUB menu will not let you access the editor or command interface without first pressing [p] followed by the GRUB password.

Unfortunately, this solution does not prevent an attacker from booting into a non-secure operating system in a dual boot environment. For this you need to edit a different part of the `/boot/grub/grub.conf` file.

Look for the `title` line of the non-secure operating system and add a line that says **`lock`** directly beneath it.

---

[2] GRUB also accepts plain text passwords, but it is recommended you use the md5 version because `/boot/grub/grub.conf` is world-readable by default.

For a DOS system, the stanza should begin something like the following:

```
title DOS
lock
```

Warning

> You must have a `password` line in the main section of the `/boot/grub/grub.conf` file for this to be work properly. Otherwise an attacker will be able to access the editor interface and remove the lock line.

If you wish to have a different password for a particular kernel or operating system, add a **`lock`** line to the stanza followed by a password line.

Each stanza you protect with a unique password should begin with lines similar to the following example:

```
title DOS
lock
password --md5 password-hash
```

Finally, remember that the `/boot/grub/grub.conf` file is world-readable by default. It is a good idea to change this, as is has no affect on the functionality of GRUB, by typing the following command as root:

```
chmod 600  /boot/grub/grub.conf
```

4.2.2.2. Password Protecting LILO

LILO is a much simpler boot loader than GRUB and does not offer a command interface, so you need not worry about an attacker gaining interactive access to the system before the kernel is loaded. However, there is still a danger in booting in single-user mode or booting to an insecure operating system.

You can configure LILO to ask for a password before booting every operating system or kernel on the system by adding a password directive in the globally. To do this, open a terminal, log in as root, and edit `/etc/lilo.conf`. Before the first `image` stanza, add a password directive similar to this:

```
password=password
```

In the above directive, replace the word *`password`* with your password.

Important

> Anytime you edit `/etc/lilo.conf`, you must run the `/sbin/lilo -v -v` command for the changes to take affect. If you have configured a password and anyone other than root can read the file, LILO will install, but will alert you that the permissions on the configuration file are wrong.

If you do not want a global password, you can apply the password directive to a stanza listed in `/etc/lilo.conf` for any kernel or operating system to which you wish to restrict access. To do this, add the password directive immediately below the `image` line. When finished, the stanza will begin similar to the following:

```
image=/boot/vmlinuz-version
        password=password
```

If you want to allow booting a kernel or operating system without password verification, but do not want to allow users to add arguments without a password, you can add the `restricted` directive on the line below the password line within the stanza. Such a stanza will begin similar to this:

```
image=/boot/vmlinuz-version
        password=password
        restricted
```

If you use the `restricted` directive, you must have a password line in the stanza.

Warning

The `/etc/lilo.conf` file is world-readable. If you are password protecting LILO, it essential that you only allow root to read and edit the file since all passwords are in plain text. To do this, type the following command as root:

```
chmod 600  /boot/grub/grub.conf
```

4.3. Password Security

Passwords are the primary way Red Hat Linux verifies that the user logging into the system is who he claims to be. This is why password security is enormously important for protection of the user, the workstation, and the network.

For security purposes, the Red Hat Linux installation program defaults to using the *Message-Digest Algorithm* (*MD5*) and shadow passwords. It is highly recommended that you do not alter these settings.

If you deselect MD5 passwords during installation, the older *Data Encryption Standard* (*DES*) format is used. This format limits passwords to eight alphanumeric character passwords (disallowing punctuation and other special characters) and provides a modest 56-bit level of encryption.

If you *deselect* shadow passwords, all user passwords will be stored as a one-way hash in the world-readable file `/etc/passwd`. This opens up your system to offline password cracking attacks. If an intruder can gain access to the machine as a regular user, he can view the `/etc/passwd` file and run any number of password cracking programs against it on his own machine. If there is an insecure password in the file, it is only a matter of time before the password cracker discovers it.

Shadow passwords eliminate this type of attack by storing the password hashes in the file `/etc/shadow` which is readable only by the root user.

This forces a potential attacker to attempt password cracking remotely by logging into a network service on the machine, such as SSH or FTP. This sort of brute-force attack is much slower and leaves an obvious trail as hundreds of failed login attempts will appear in the log files. Of course, if the cracker starts an attack in the middle of the night and you have weak passwords, he may have gained access before dawn.

Beyond matters of format and storage is the issue of content. The single most important thing a user can do to protect his account is create a strong password, which make it less susceptible to a password cracking attack.

4.3.1. Creating Strong Passwords

When creating a password, it is a good idea to follow these guidelines:

Do Not Do the Following:

- *Do Not Use Only Words or Numbers* — You should never use solely numbers or words in a password.

  Some examples include the following:

  - 8675309
  - juan
  - hackme

- *Do Not Use Recognizable Words* — Words such as proper names, dictionary words, or even terms from television shows or novels should be avoided, even if they are bookended with numbers.

  - john1
  - DS-9
  - mentat123

- *Do Not Use Words in Foreign Languages* — Password cracking programs often check against word lists that encompass dictionaries of many languages. Relying on foreign languages for secure passwords is of little use.

  Some examples include the following:

  - cheguevara
  - bienvenido1
  - 1dumbKopf

- *Do Not Use Hacker Terminology* — If you think you are elite because you use hacker terminology — also called 1337 (LEET) speak — in your password, think again. Many word lists include LEET speak.

  Some examples include the following:

  - H4X0R
  - 1337

- *Do Not Use Personal Information* — Steer clear of personal information. If the attacker knows who you are, they will have an easier time figuring out your password if it includes information such as:

  - Your name
  - The names of pets
  - The names of family members
  - Any birth dates
  - Your phone number or zip code

- *Do Not Invert Recognizable Words* — Good password checkers always reverse common words, so inverting a bad password does not make it any more secure.

  Some examples include the following:

  - R0X4H
  - nauj
  - 9-DS

- *Do Not Write Down Your Password* — Never store your password on paper. It is much safer to memorize it.
- *Do Not Use the Same Password For All Machines* — It is important that you make separate passwords for each machine. This way if one system is compromised, all of your machines will not be immediately at risk.

Do the Following:

- *Make the Password At Least Eight Characters Long* — The longer the password is, the better. If you are using MD5 passwords, it should be 15 characters long or longer. With DES passwords, use the maximum length — eight characters.
- *Mix Upper and Lower Case Letters* — Red Hat Linux is case sensitive, so by mixing cases, you will enhance the strength of the password.
- *Mix Letters and Numbers* — Adding numbers to passwords, especially when added to the middle (not just at the beginning or the end), can enhance password strength.
- *Include Non-Alphanumeric Characters* — Special characters such as &, $, and > can greatly improve the strength of a password.
- *Pick a Password You Can Remember* — The best password in the world does you little good if you cannot remember it. So use acronyms or other mnemonic devices to aid in memorizing passwords.

With all these rules, it may seem difficult to create a password meeting all of the criteria for good passwords while avoiding the traits of a bad one. Fortunately, there are some simple steps one can take to generate a memorable, secure password.

4.3.1.1. Secure Password Creation Methodology

There are many methods people use to create secure passwords. One of the more popular methods involves acronyms. For example:

- Think of a memorable phrase, such as:

  "over the hills and far away, to grandmothers house we go."

- Next, turn it into an acronym (including the punctuation).

  `othafa,tgmhwg.`

- Add complexity by substituting numbers and symbols for letters in the acronym. For example, substitute **`7`** for **`t`** and the at symbol (**`@`**) for **`a`**:

  `o7h@f@,7gmhwg.`

- Add more complexity by capitalizing at least one letter, such as **`H`**.

  `o7H@f@,7gmHwg.`

- Finally, do not use the example password above on any of your systems.

While creating secure passwords is imperative, managing them properly is also important, especially for system administrators within larger organizations. The next section will detail good practices for creating and managing user passwords within an organization.

4.3.2. Creating User Passwords Within an Organization

If there are a significant number of users in an organization, the system administrator has two basic options available to force the use of good passwords. They can create passwords for the user, or they can let the user create his own passwords, while verifying the passwords are of acceptable quality.

Creating the passwords for the users ensures that the passwords are good, but it becomes a daunting task as the organization grows. It also increases the risk of users writing their passwords down.

For these reasons, system administrators prefer to have the user create their own passwords, but actively verify that the passwords are good and, in some cases, force users to change their passwords periodically through password aging.

4.3.2.1. Forcing Strong Passwords

To protect the network from intrusion it is a good idea for system administrators to verify that the passwords used within an organization are strong ones. When a user is asked to create or change his password, he can use the command line application `passwd`, which is PAM aware and will therefore check to see if the password is easy to crack or too short in length via the `pam_cracklib.so` *Pluggable Authentication manager* (*PAM*) module. Since PAM is customizable, it is possible to add further password integrity checkers, such as `pam_passwdqc` (available from http://www.openwall.com/passwdqc/) or to write your own module. For a list of available PAM modules, see http://www.kernel.prg/pub/linux/libs/pam/modules.html. For more information about PAM, see the chapter titled *Pluggable Authentication Modules (PAM)* in the *Official Red Hat Linux Reference Guide*.

It should be noted, however, that the check performed on passwords at the time of their creation will not discover bad passwords as effectively as running a password cracking program against the passwords within the organization.

There are many password cracking programs available for Red Hat Linux, although none ship with the operating system. Below is a brief list of some of the more popular password cracking programs:

- ***John The Ripper*** — A fast and flexible password cracking program. It allows the use of multiple word lists and is capable of brute-force password cracking. It is available at http://www.openwall.com/john/.
- ***Crack*** — Perhaps the most well known password cracking software, Crack is also very fast, though not as easy to use as John The Ripper. It can be found at http://www.users.dircon.co.uk/~crypto/index.html
- ***Slurpie*** — Slurpie is similar to John The Ripper and Crack except it is designed to run on multiple computers simultaneously, creating a distributed password cracking attack. It can be found along with a number of other distributed attack security evaluation tools at http://www.ussrback.com/distributed.htm

Warning

Always get authorization in writing before attempting to crack passwords within an organization.

4.3.2.2. Password Aging

Password aging is another technique used by system administrators to defend against bad passwords within an organization. Password aging means that after a set amount of time, usually 90 days, the user will be prompted to come up with a new password. The theory behind this is that if a user is forced to change his password periodically, a cracked password is only useful to an intruder for a limited amount of time. The downside to password aging, however, is that users are more likely to write their passwords down.

Their are two primary ways to enforce password aging under Red Hat Linux. you can use the command `chage` or the graphical application **User Manager**.

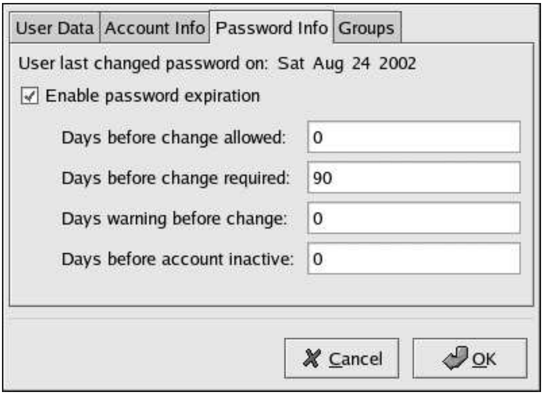
The `-M` option of the `chage` command specifies the maximum number of days the password is valid. So, for instance, if you want a user's password to expire in 90 days, type the following command:

```
chage -M 90 username
```

In the above command, replace *`username`* with the name of the user. If you do not want the password to expire, it is traditional to use a value of **`99999`** after the `-M` option (this equates to a little over 273 years).

If want to use the graphical **User Manager** application to create password aging policies, go to the **Main Menu Button** (on the Panel) => **System Settings** => **Users & Groups** or type the command `redhat-config-users` at a shell prompt (for example, in an XTerm or a GNOME terminal). Click on the **Users** tab, select the user from the user list, and click **Properties** from the button menu (or choose **File** => **Properties** from the pull-down menu).

Then click the **Password Info** tab and enter the number of days before the password expires, as shown in Figure 4-1.



User last changed password on: Sat Aug 24 2002

☑ Enable password expiration

Days before change allowed: 0

Days before change required: 90

Days warning before change: 0

Days before account inactive: 0

Figure 4-1. User Password Info Pane

For more information on using the **User Manager** to do this, see the chapter titled *User and Group Configuration* in the *Official Red Hat Linux Customization Guide*.

4.4. Administrative Controls

When administering a home machine, the user will have to perform some tasks as the root user or by acquiring effective root privileges via a *setuid* program, such as `sudo` or `su`. A setuid program is one that operates with the user ID (*UID*) of the owner of program rather than the user operating the program. Such programs are denoted by a lower case `s` in the owner section of a long format listing.

For a system administrator, however, choices must be made as to how much administrative access each users within the organization should have to their machine. Through a PAM module called `pam_console.so`, some activities normally reserved only for the root user, such as rebooting and mounting removable media are allowed for the first user to log in at the physical console (see the chapter titled *Pluggable Authentication Modules (PAM)* in the *Official Red Hat Linux Reference Guide* for more on the `pam_console.so` module). However, other important system administration tasks such as altering network settings, configuring a new mouse, or mounting network devices are impossible without administrative access. As a result system administrators must decide how much to trust the users on their network.

4.4.1. Allowing Root Access

If the users within an organization are a trusted, computer-savvy group, then allowing them root access may not be a bad thing. Allowing root access users means that minor issues like adding devices or configuring network interfaces can be handled by the individual user, leaving system administrators free to deal with network security and other important issues.

On the other hand, giving root access to individual users can lead to the following issues (to name a few):

- *Machine Misconfiguration* — Users with root access can misconfigure their machines and require assistance or worse, open up security holes without knowing it.
- *Run Insecure Services* — Users with root access may run insecure servers on their machine, such as FTP or telnet, potentially putting usernames and passwords at risk as they pass over the network in the clear.
- *Running Email Attachments As Root* — Although rare, email viruses that effect Linux do exist. The only time they are a threat, however, is when they are run by the root user.

4.4.2. Disallowing Root Access

If an administrator is uncomfortable allowing users to log in as root for these or other reasons, the root password should be kept secret and access to runlevel one or single user mode should be disallowed through boot loader password protection (see Section 4.2.2 for more on this topic).

Table 4-1 shows ways an administrator can further ensure that root logins are disallowed:

Method	Description	Effects	Does Not Effect
Disabling root SSH logins.	Edit the `/etc/ssh/sshd_config` file and set the `PermitRootLogin` parameter to **`no`**.	Prevents root access via the OpenSSH suit of tools. The following programs are prevented from accessing the root account: · `ssh` · `scp` · `sftp`	Since this only effects the OpenSSH suite of tools, no other programs are effected by this setting.

Method	Description	Effects	Does Not Effect
Changing the root shell.	Edit the `/etc/passwd` file and change the shell from `/bin/bash` to `/sbin/nologin`.	Prevents access to the root shell and logs the attempt. The following programs are prevented from accessing the root account: · `login` · `gdm` · `kdm` · `xdm` · `su` · `ssh` · `scp` · `sftp`	Programs that do not require a shell, such as FTP clients, mail clients, and many setuid programs. The following programs are *not* prevented from accessing the root account: · `sudo` · FTP clients · Email clients
Disabling root access via any console device (tty).	An empty `/etc/securetty` file prevents root login on any devices attached to the computer.	Prevents access to the root account via the console or the network. The following programs are prevented from accessing the root account: · `login` · `gdm` · `kdm` · `xdm` · Other network services that open a tty	Programs that do not log in as root, but perform administrative tasks through through setuid or other mechanisms. The following programs are *not* prevented from accessing the root account: · `su` · `sudo` · `ssh` · `scp` · `sftp`
Use PAM to limit root access to services.	Edit the file for the target service in the `/etc/pam.d/` directory. Make sure the `pam_listfile.so` is required for authentication. See Section 4.4.2.4 for more details.	Prevents root access to network services that are PAM aware. The following services are prevented from accessing the root account: · FTP clients · Email clients · `login` · `gdm` · `kdm` · `xdm` · `ssh` · `scp` · `sftp` · Any PAM aware services	Programs and services that are not PAM aware.

Table 4-1. Methods of Disabling the Root Account

4.4.2.1. Disabling the Root Shell

If the administrator does not wish for users to log in directly as root, he can set the root account's shell to `/sbin/nologin` in the `/etc/passwd` file. This will prevent access to the root account through commands that require a shell, such as the `su` and the `ssh` commands.

Important

Programs that do not require access to the shell, such as email clients or the `sudo` command, can still access the root account.

4.4.2.2. Disabling Root Logins

To further enforce this, he can disable root logins at the console by editing the `/etc/securetty` file. This file lists all devices the root user is allowed to log into. If the file does not exist at all, the root user can log in through any communication device on the system, whether it by via the console or a raw network interface. This is dangerous because if configured in this way, a user could telnet into his machine as root, sending his password in plain text over the network. By default, Red Hat Linux's `/etc/securetty` file only allows the root user to log at the console physically attached to the machine. To prevent root from logging in, remove the contents of this file by typing the following command:

```
echo > /etc/securetty
```

Warning

A blank `/etc/securetty` file does *not* prevent the root user from logging in remotely using the OpenSSH suite of tools because the console is not opened until after authentication.

4.4.2.3. Disabling Root SSH Logins

To prevent root logins via the SSH protocol, you will need to edit the SSH daemon's configuration file: `/etc/ssh/sshd_config`. Change the line that says:

```
# PermitRootLogin yes
```

To read as follows:

```
PermitRootLogin no
```

4.4.2.4. Disabling Root Using PAM

PAM allows great flexibility in denying specific accounts via the `/lib/security/pam_listfile.so` module. This allows the administrator to point the module at a list of users that are not allowed to log in. Below is an example of how the module is used for the FTP service in the `/etc/pam.d/ftp` PAM configuration file (the \ character at the end of the first of the first line is *not* necessary if the directive is all on one line):

```
auth    required    /lib/security/pam_listfile.so    item=user \
sense=deny file=/etc/ftpusers onerr=succeed
```

This tells PAM to consult the file `/etc/ftpusers` and deny any user listed access to the service. The administrator is free to change the name of this file and can keep separate lists for each service or use one central list to deny access to multiple services.

If the administrator wants to deny access to multiple services, he can add a similar line to the PAM configuration services, such as `/etc/pam.d/pop` and `/etc/pam.d/imap` for mail clients or `/etc/pam.d/ssh` for SSH clients.

For more information about PAM, see the chapter titled *Pluggable Authentication Modules (PAM)* in the *Official Red Hat Linux Reference Guide*.

4.4.3. Limiting Root Access

Often, rather than completely deny access to the root user, the administrator may wish to allow access only via setuid programs, such as `su` or `sudo`.

4.4.3.1. The `su` Command

When a user types the command `su` she is prompted for the root password and, after authentication, given a root shell prompt.

Once logged in via the `su` command, the user *is* the root user and has absolute administrative access to the system. In addition, once a user has attained root, it is possible in some cases for them to use the `su` command to change to any other user on the system without being prompted for a password.

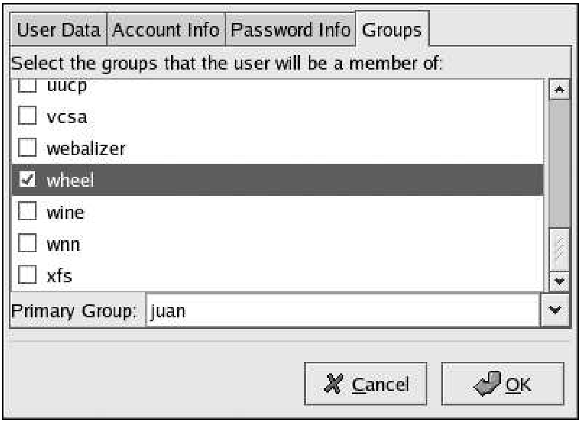
Because this program is so powerful, administrators may wish to limit who has access to the command.

One of the simplest ways to do this is to add users to the special administrative group called *wheel*. To do this, type the following command as root:

```
usermod -G wheel username
```

To use the **User Manager** for this purpose, go to the **Main Menu Button** (on the Panel) => **System Settings** => **Users & Groups** or type the command `redhat-config-users` at a shell prompt. Select the **Users** tab, select the user from the user list, and click **Properties** from the button menu (or choose **File** => **Properties** from the pull-down menu).

Then select the **Groups** tab and click on the wheel group, as shown in Figure 4-2.



Select the groups that the user will be a member of:

☐ uucp
☐ vcsa
☐ webalizer
☑ wheel
☐ wine
☐ wnn
☐ xfs

Primary Group: juan

Figure 4-2. User Groups Pane

Next open the PAM configuration file for `su`, `/etc/pam.d/su`, in a text editor and remove the comment [#] from the following line:

```
auth  required /lib/security/pam_wheel.so use_uid
```

Doing this will permit only members of the administrative group `wheel` to use the program.

Note

> The root user is part of the `wheel` group by default.

4.4.3.2. The `sudo` Command

The `sudo` command offers another approach for giving trusted users administrative access. When a trusted user precedes an administrative command with `sudo`, he is prompted for *his* password. Then, once authenticated and assuming that the command is permitted, the administrative command is executed as if by the root user.

The basic format of the `sudo` command is as follows:

```
sudo command
```

In the above example, *`command`* would be replaces by a command normally reserved for the root user, such as `mount`.

Important

> Users of the `sudo` command should take extra care to log out when they walk away from their machine since sudoers can use the command again without being asked for a password until a five minute period has passed. This setting can be altered via the configuration file, `/etc/sudoers`.

The `sudo` command allows for a high degree of flexibility. For instance, only users listed in the `/etc/sudoers` configuration file are allowed to use the `sudo` command and the command is executed in *their* shell, not root's. This means the root shell can be completely disabled, as shown in Section 4.4.2.1.

The `sudo` command also provides a comprehensive audit trail. Each successful authentication is logged to the file `/var/log/messages` and command that was issued along with the issuer's user name is logged to the file `/var/log/secure`.

Another advantage of the `sudo` command is that an administrator can allow different users access to specific commands based on their needs.

All commands executed via `sudo` are recorded in the `/var/log/secure` file, as well as all attempts to use the `sudo` command.

Administrators wanting to edit the `sudo` configuration file, `/etc/sudoers`, should use the `visudo`.

To give someone full administrative privileges, type `visudo` and add a line similar to the following in the user privilege specification section:

```
juan ALL=(ALL) ALL
```

This example states that the user, `juan`, can use `sudo` from any host and execute any command.

The example below illustrates the granularity possible when configuring `sudo`:

```
%users  localhost=/sbin/shutdown -h now
```

This example states that any user can issue the command `/sbin/shutdown -h now` as long as they issue it from the console.

The man page for `sudoers` has a detailed listing of options for this file.

4.5. Available Network Services

While user access to administrative controls is an important issue for system administrators within an organization, keeping tabs on which network services is of paramount importance to anyone who installs and operates a Linux system.

Many services under Linux behave as network servers. If a network service is running on a machine, then a server application called a *daemon* is listening for connections on one or more network ports. Each of these servers should be treated as potential avenue of attack.

4.5.1. Risks To Services

Network services can pose many risks for Linux systems. Below is a list of some of the primary issues:

- *Buffer Overflow Attacks* — Services which connect to ports number 0 through 1023 must run as an administrative user. If the application has an exploitable buffer overflow, an attacker could gain access to the system as the user running the daemon. Because exploitable buffer overflows exist, crackers will use automated tools to identify systems with vulnerabilities and once they have gained access, they will use automated rootkits to maintain their access to the system.
- *Denial of Service Attacks (DoS)* — By flooding a service with requests, a denial of service attack can bring a system to a screeching halt as it tries to log and answer each request.
- *Script Vulnerability Attacks* — If a server is using scripts to execute server-side actions, as Web servers commonly do, a cracker can mount an attack improperly written scripts. These script vulnerability attacks could lead to a buffer overflow condition or allow the attacker to alter files on the system.

To limit exposure to attacks over the network all services that are unused should be turned off.

4.5.2. Identifying and Configuring Services

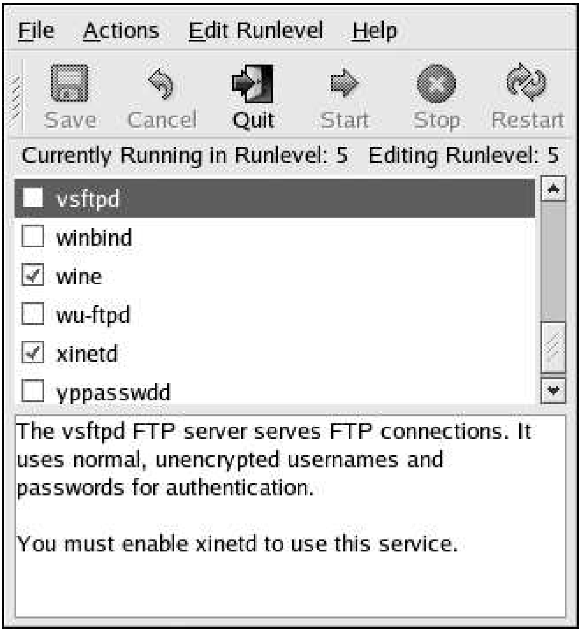
To enhance security, most network services installed with Red Hat Linux are turned off by default. There are, however some notable exceptions:

- `lpd` — A printer server, required by the `lpr` command.
- `portmap` — A necessary component for the NFS, NIS, and other RFC protocols.
- `xinetd` — A super server that controls connections to a host of subordinate servers, such as `wu-ftpd`, `vsftpd`, `telnet`, and `sgi-fam` (which is necessary for the Nautilus file manager).
- `sendmail` — The Sendmail mail transport agent is enabled by default, but only listens for connections on the localhost.
- `sshd` — The OpenSSH server, which is a secure replacement for Telnet.

When determining whether or not to leave these services running, it is best to use common sense and err on the side of caution. For instance, if you do not own a printer, do not leave `lpd` running with the assumption that one day you might buy one. The same is true for `portmap`. If you do not mount NFS volumes or use NIS (the `ypbind` service), then turn portmap should be disabled.

Red Hat Linux ships with three programs designed to switch services on or off. They are **Services Configuration Tool**, **ntsysv**, and `chkconfig`. For information on using these tools, see the chapter titled *Controlling Access to Services* in the *Official Red Hat Linux Customization Guide*.

If you are not sure what purpose a service has, the **Services Configuration Tool** has a description field, illustrated in Figure 4-3, that may be of some use.



Currently Running in Runlevel: 5 Editing Runlevel: 5

☐ vsftpd
☐ winbind
☑ wine
☐ wu-ftpd
☑ xinetd
☐ yppasswdd

The vsftpd FTP server serves FTP connections. It uses normal, unencrypted usernames and passwords for authentication.

You must enable xinetd to use this service.

Figure 4-3. User Groups Pane

But checking to see which network services are configured to start at boot time is not enough. Good system administrators should also check which ports are open and listening. See Section 5.8 for more on this subject.

4.5.3. Insecure Services

Potentially any network service is insecure. This is why turning unused services off is so important. Exploits for services are revealed and patched routinely. But you must remember the importance of keeping the packages associated with any given service updated. See for more on this issue.

Some network protocols are inherently more insecure than others. These include any services which do the following:

- *Pass Usernames and Passwords in Plain Text* — Many older protocols, such as Telnet and FTP, do not encrypt the authentication session and should be avoided whenever possible.
- *Pass Sensitive Information in Plain Text* — Protocols that pass the username and password in plain text also pass everything transfered between the server and client in plain text. These include Telnet, FTP, HTTP (`httpd`), and SMTP (`sendmail`).

  Many network file systems, such as NFS and SMB, also pass information over the network in plain text. It is the user's responsibility when using these protocols to limit what type of data is transmitted.

  Remote memory dump services, like `netdump`, pass the contents of memory over the network. Memory dumps can contain passwords or, even worse, database entries and other sensitive information.

  Other services like `finger` and `rwhod` reveal information about users of the system.

Examples of inherently insecure services includes the following:

- `rlogin`
- `rsh`
- `telnet`
- `vsftpd`
- `wu-ftpd`

All remote login and shell programs (`rlogin`, `rsh`, and `telnet`) should be avoided in favor of SSH. (see Section 4.7 for more information about `sshd`).

FTP is not as inherently dangerous to the security of the system as remote shells, but FTP servers must carefully configured and monitored to avoid problems.

Services which should be carefully implemented and behind a firewall include:

- `finger`
- `identd`
- `netdump`
- `netdump-server`
- `nfs`
- `portmap`
- `rwhod`
- `sendmail`
- `smb` (Samba)
- `yppasswdd`
- `ypserv`
- `ypxfrd`

The next section discusses tools available to set up a simple firewall.

4.6. Personal Firewalls

Once the *necessary* network services are configured, it is important to implement a firewall.

Firewalls prevent network packets from accessing the network interface of the system. If a request is made to a port that is blocked by a firewall, the request will be ignored. If a service is listening on one of these blocked ports, it will not receive the packets and is effectively disabled. For this reason, care should be taken when configuring a firewall to block access to ports not in use, while not blocking access to ports used by configured services.

For most users, the best tools for configuring a simple firewall are the two straight-forward, graphical firewall configuration tools which ship with Red Hat Linux: **Security Level Configuration Tool** and **GNOME Lokkit**.

Both of these tools perform the same task — they create broad `iptables` rules for a general-purpose firewall. The difference between them is in their approach to performing this task. The **Security Level Configuration Tool** is a firewall control panel, while **GNOME Lokkit** presents the user with a series of questions in a wizard-type interface.

For more information about how to use these applications and what options they offer, refer to the chapter called *Basic Firewall Configuration* in the *Official Red Hat Linux Customization Guide*.

For advanced users and server administrators, manually configuring a firewall with `iptables` is likely the best option. Refer to Chapter 7 for more information. For a comprehensive guide to the `iptables` command, consult the chapter titled *Firewalls and* `iptables` in the *Official Red Hat Linux Reference Guide*.

4.7. Security Enhanced Communication Tools

As the size and popularity of the Internet has grown, so has the threat from communication interception. Over the years, tools have been developed to encrypt communications as they are transferred over the network.

Red Hat Linux ships with two basic tools that use high-level, public-key-cryptography-based encryption algorithms to protect information as it travels over the network.

- *OpenSSH* — A free implementation of the SSH protocol for encrypting network communication.
- *Gnu Privacy Guard (GPG)* — A free implementation of the PGP (Pretty Good Privacy) encryption application for encrypting data.

OpenSSH is a safer way to access a remote machine and replaces older, unencrypted services like `telnet` and `rsh`. OpenSSH includes a network service called `sshd` and three command line client applications:

- `ssh` — A secure remote console access client.
- `scp` — A secure remote copy command.
- `sftp` — A secure pseudo-ftp client that allows interactive file transfer sessions.

It is highly recommended that any remote communication with Linux systems occur using the SSH protocol. For more information about OpenSSH, see the chapter titled *OpenSSH* in the *Official Red Hat Linux Customization Guide*. For more information about the SSH Protocol, see the chapter titled *SSH Protocol* in the *Official Red Hat Linux Reference Guide*.

Important

> Although the `sshd` service is inherently secure, the service *must* be kept up-to-date to prevent security threats.

GPG is a great way to keep private data private. It can be used both to email sensitive data over public networks and to protect sensitive data on hard drives.

For more information about using GPG, see the appendix titled *Getting Started with Gnu Privacy Guard* in the *Official Red Hat Linux Customization Guide*.

Chapter 5.

Server Security

When a system is used as a server on a public network, it becomes a target for attacks. For this reason, hardening the system and locking down services is of paramount importance for the system administrator.

Before delving into these specific issues, you should review the following general tips for enhancing server security:

- Keep all services up to date to protect against the latest threats.
- Use secure protocols whenever possible.
- Serve only one type of service per machine whenever possible.
- Monitor all servers carefully for suspicious activity.

5.1. Securing Services With TCP Wrappers and `xinetd`

TCP wrappers provide access control to a variety of services. Most modern network services, such as SSH, Telnet, and FTP, make use of TCP wrappers, a program that is designed to stand guard between an incoming request and the requested service.

The benefits offered by TCP wrappers are enhanced when the `/usr/lib/libwrap.a` library is used in conjunction with `xinetd`, a super service that provides additional access, logging, binding, redirection and resource utilization control.

More information on configuring TCP wrappers and `xinetd` can be found in the chapter titled *TCP Wrappers and `xinetd`* in the *Official Red Hat Linux Reference Guide*.

The following subsections will assume a basic knowledge of each topic and focus on specific security options.

5.1.1. Enhancing Security With TCP Wrappers

TCP wrappers are capable of much more than denying access to services. This section will illustrate how it can be used to send connection banners, warn of attacks from particular hosts, and enhance logging functionality. For a thorough list of TCP wrapper functionality and control language, see the man page for `hosts_options`.

5.1.1.1. TCP Wrappers and Connection Banners

Sending client connections to a service an intimidating banner is a good way to disguise what system the server is running while letting a potential attacker know that system administrator is vigilant. To implement a TCP wrappers banner for a service, use the `banner` option.

This example implements a banner for `wu-ftpd`. To begin you must create a banner file. It can be anywhere on the system, but it must bear same name as the daemon. This example we will name the file `/etc/banners/in.ftpd`.

The contents of the file will look like this:

```
220-Hello, %c
220-All activity on ftp.example.com is logged.
220-Act up and you will be banned.
```

The `%c` token supplies a variety of client information, such as the username and hostname, or the username and IP address to make the connection even more intimidating. The *Official Red Hat Linux Reference Guide* has a list of other tokens available for TCP wrappers.

For this banner to be presented to incoming connections, add the following line to the `/etc/hosts.allow` file:

```
in.ftpd : ALL : banners /etc/banners/
```

5.1.1.2. TCP Wrappers and Attack Warnings

If a particular host or network has been caught attacking the server, TCP wrappers can be used to warn of subsequent attacks from that host or network via the `spawn` directive.

In this example, assume that a cracker from the 206.182.68.0/24 network has been caught attempting to attack the server. By placing the following line in the `/etc/hosts.deny` file, the connection attempt is denied and logged into a special file:

```
 ALL : 206.182.68.0 : spawn /bin/ 'date' %c %d >> /var/log/intruder_alert
```

The `%d` token supplies the name of the service that the attacker was trying to access.

To allow the connection and log it, place the `spawn` directive in the `/etc/hosts.allow` file.

Note

> Since the `spawn` directive executes any shell command, you can create a special script to notify you or execute a chain of commands in the event that a particular client attempts to connect to your server.

5.1.1.3. TCP Wrappers and Enhanced Logging

If certain types of connections are of more concern than others, the log level can be elevated for that service via the `severity` option.

In this example, assume anyone attempting to connect to port 23 (the Telnet port) on our FTP server is a cracker. To denote this, place a `warning` flag in the log files instead of the default flag, `info`, and deny the connection.

To do this, place the following line in `/etc/hosts.deny`:

```
in.telnetd : ALL : severity warning
```

This will use the default `authpriv` logging facility, but elevate the priority from the default value of `info` to `warning`.

5.1.2. Enhancing Security With `xinetd`

The `xinetd` is another useful tool for control access its subordinate services. This section will focus on how `xinetd` can be used to set a trap service and control the amount of resources any given `xinetd` service can use in order to thwart denial of service attacks. For a more thorough list of the options available, see the man pages for `xinetd` and `xinetd.conf`.

5.1.2.1. Setting a Trap

One important feature of `xinetd` is its ability to add hosts to a global `no_access` list. Hosts on this list are denied subsequent connections to services managed by `xinetd` for a specified length of time or until `xinetd` is restarted. This is accomplished using the `SENSOR` attribute. This technique is an easy way to block hosts attempting to port scan the server.

The first step in setting up a `SENSOR` is to choose a service you do not plan on using. For this example, Telnet will be used.

Edit the file `/etc/xinetd.d/telnet` and change the line `flags` line to read:

```
flags           = SENSOR
```

Add the following line within the braces:

```
deny_time       = 30
```

This will deny the host that attempted to connect to the port for 30 minutes. Other acceptable values for the `deny_time` attribute are FOREVER, which keeps the ban in effect until `xinetd` is restarted, and NEVER, which allows the connection and logs it.

Finally, the last line should read:

```
disable         = no
```

While using `SENSOR` is a good way to detect and stop connections from nefarious hosts, it has two drawbacks:

- It will not work against stealth scans.
- An attacker who knows you are running `SENSOR` can mount a denial of service attack against particular hosts by forging their IP addresses and connecting to the forbidden port.

5.1.2.2. Controlling Server Resources

Another important feature of `xinetd` is its ability to control the amount of resources services under its control can utilize.

It does this by way of the following directives:

- `cps = <number_of_connections> <wait_period>` — Dictates the connections allowed to the service per second. This directive accepts only integer values.
- `instances = <number_of_connections>` — Dictates the total number of connections allowed to a service. This directive accepts either an integer value or `UNLIMITED`.
- `per_source = <number_of_connections>` — Dictates the connections allowed to a service by each host. This directive accepts either an integer value or `UNLIMITED`.
- `rlimit_as = <number[K|M]>` — Dictates the amount of memory address space the service can occupy in kilobytes or megabytes. This directive accepts either an integer value or `UNLIMITED`.
- `rlimit_cpu = <number_of_seconds>` — Dictates the amount of time in seconds that a service may occupy the CPU. This directive accepts either an integer value or `UNLIMITED`.

Using these directives can help prevent any one `xinetd` service from overwhelming the system, resulting in a denial of service.

5.2. Securing Portmap

The `portmap` service is a dynamic port assignment daemon for RPC services such as NIS and NFS. It has weak authentication mechanisms and has the ability to assign a wide range of ports for the services it controls. For these reasons, it is difficult to secure.

If you are running RPC services, you should follow some basic rules.

5.2.1. Protect `portmap` With TCP Wrappers

It is important to use TCP wrappers to limit which networks or hosts have access to the `portmap` service since it has no built-in form of authentication.

Further, use *only* IP addresses when limiting access to the service. Avoid these hostnames as they can be more via DNS poisoning and other methods.

5.2.2. Protect `portmap` With `iptables`

To further restrict access to the `portmap` service, it is a good idea to add `iptables` rules to the server, restricting access to specific networks.

Below is an example of an `iptables` command that allows TCP connections to `portmap`, listening on port 111, from the 192.168.0/24 network exclusively. All other packets are dropped.

```
iptables -A INPUT -p tcp -s! 192.168.0.0/24  --dport 111 -j DROP
```

To similarly limit UDP traffic, use the following command.

```
iptables -A INPUT -p udp -s! 192.168.0.0/24  --dport 111 -j DROP
```

5.3. Securing NIS

NIS stands for Network Information Service. It is an RPC service called `ypserv` which is used in conjunction with `portmap` and other related services to distribute maps of usernames, passwords, and other sensitive information to any computer claiming to be within its domain.

An NIS server is comprised of several applications. They include the following:

- `/usr/sbin/rpc.yppasswdd` — Also called the `yppasswdd` service, this daemon allows users to change their NIS passwords.
- `/usr/sbin/rpc.ypxfrd` — Also called the `ypxfrd` service, this daemon is responsible for NIS map transfers over the network.
- `/usr/sbin/yppush` — This application propagates changed NIS databases to multiple NIS servers.
- `/usr/sbin/ypserv` — This is the NIS server daemon.

NIS is rather insecure by todays standards. It has no host authentication mechanisms and passes all of its information in clear text, including password hashes. As a result, extreme care must be taken to set up a network that uses NIS. Further complicating the situation, the default configuration of NIS is inherently insecure.

It is recommended that anyone planning to implement an NIS server first secure the `portmap` service as outlined in Section 5.2, then address following issues.

5.3.1. Carefully Plan the Network

Because NIS passes sensitive information unencrypted over the network, it is important the service be run behind a firewall and on a segmented and secure network. Any time NIS information is passed over an insecure network, it risks being intercepted. Careful network design in these regards can help prevent severe security breaches.

5.3.2. Use a Password-Like NIS Domain Name and Hostname

Any machine within an NIS domain can use commands to extract information from the server without authentication, as long as the user knows the NIS server's DNS hostname and NIS domain name.

For instance, if someone either connects a laptop computer into the network or breaks into the network from outside (and manages to spoof an internal IP address) the following command will reveal the `/etc/passwd` map:

```
ypcat -d <NIS_domain> -h <DNS_hostname> passwd
```

If this attacker is a root user, they can obtain the `/etc/shadow` file by typing the following command:

```
ypcat -d <NIS_domain> -h <DNS_hostname> shadow
```

Note

If Kerberos is used, the `/etc/shadow` file is not stored within an NIS map.

To make access to NIS maps harder for an attacker, create a random string for the DNS hostname, such as `o7hfawtgmhwg.domain.com`. Similarly, create a *different* randomized NIS domain name. This will make it much more difficult for an attacker to access the NIS server.

5.3.3. Edit the `/var/yp/securenets` File

NIS will listen to all networks if the `/var/yp/securenets` file does not exist, as is the case after a default installation, or is blank. One of the first thing you should do is put netmask/network pairs in the file so that `ypserv` will only respond to requests from the proper network.

Warning

Never start an NIS server for the first time without creating the `/var/yp/securenets` file.

Below is a sample entry from a `/var/yp/securenets` file:

```
255.255.255.0     192.168.0.0
```

This technique does not provide protection from an IP spoofing attack, but it does at least place limits on what networks the NIS server will service.

5.3.4. Assign Static Ports and Use

All of the servers related to NIS can be assigned specific ports except for `rpc.yppasswdd` — the daemon that allows users to change their login passwords. Assigning ports to the other two NIS server daemons, `rpc.ypxfrd` and `ypserv`, allows you to create firewall rules to further protect the NIS server daemons from intruders.

To do this, add the following lines to `/etc/sysconfig/network`:

```
YPSERV_ARGS="-p 834"
YPXFRD_ARGS="-p 835"
```

The following `iptables` rules can be issued to enforce which network the server will listen to for these ports:

```
iptables -A INPUT -p ALL -s! 192.168.0.0/24  --dport 834 -j DROP
iptables -A INPUT -p ALL -s! 192.168.0.0/24  --dport 835 -j DROP
```

Tip

Refer to Chapter 7 for more information about implementing firewalls with `iptables` commands.

5.3.5. Use Kerberos Authentication

One of the most glaring flaws inherent when NIS is used for authentication is that whenever a user logs into a machine, a password hash from the `/etc/shadow` map is send over the network. If an intruder gains access to an NIS domain and sniffs network traffic, usernames and password hashes can be quietly collected. With enough time a password cracking program can guess weak passwords, and the attacker has a valid login on the network.

Since Kerberos using secret-key cryptography, no password hashes are ever sent over the network, making the system far more secure. For more about Kerberos, see the chapter titled *Kerberos* in the *Official Red Hat Linux Reference Guide*.

5.4. Securing NFS

The Network File System or NFS is an RPC service used in conjunction with `portmap` and other related services to provide network accessible mount points for client machines. For more information on how NFS works, see the chapter titled *Network File System (NFS)* in the *Official Red Hat Linux Reference Guide*. For more information about configuring NFS, refer to the *Official Red Hat Linux Customization Guide*. The following subsections will assume basic knowledge of NFS.

It is recommended that anyone planning to implement an NFS server first secure the `portmap` service as outlined in Section 5.2, then address following issues.

5.4.1. Carefully Plan the Network

Because NFS passes all information unencrypted over the network, it is important the service be run behind a firewall and on a segmented and secure network. Any time information is passed over NFS an insecure network, it risks being intercepted. Careful network design in these regards can help prevent security breaches.

5.4.2. Beware of Syntax Errors

The NFS server determines which file systems to export and who to export these directories to via the `/etc/exports` file. Be careful not to add extraneous spaces when editing this file.

For instance, the following line in the `/etc/exports` file shares the directory `/tmp/nfs/` to the host `my.example.com` with read and write permissions.

```
/tmp/nfs/     my.example.com(rw)
```

This line in the `/etc/exports` file, on the other hand, shares the same directory to the host `my.example.com` with read-only permissions and shares is to the *world* with read and write permissions due to a single space after the hostname.

```
/tmp/nfs/     my.example.com (rw)
```

It is good practice to check any configured NFS shares by using the following command to verify they are correctly configured:

```
showmount -e <hostname>
```

5.4.3. Do Not Use the `no_root_squash` Option

By default, NFS shares change root-owned files to user `nfsnobody`. This prevents uploading of programs with the setuid bit set.

5.5. Securing Apache HTTP Server

The Apache HTTP Server is one of the most stable and secure services that ships with Red Hat Linux.a There are an overwhelming number of options and techniques available to secure the Apache HTTP Server — too numerous to delve into deeply here.

It is important if you are configuring Apache HTTP Server to read the documentation available for the application. This includes the the chapter titled *Apache HTTP Server* in the *Official Red Hat Linux Reference Guide*, the chapter titled *Apache HTTP Secure Server Configuration* in the *Official Red Hat Linux Customization Guide*, and the Stronghold manuals, available at http://www.redhat.com/docs/manuals/stronghold/.

Below is a list of configuration options administrators should be careful using.

5.5.1. `FollowSymLinks`

This directive is enabled by default, so be careful where you create symbolic links to in the document root of the Web server. For instance, it is a bad idea to provide a symbolic link to `/`.

5.5.2. The `Indexes` Directive

This directive is enabled by default, but may not be desirable. If you do not want users to browse files on the server, it is best to remove this directive.

5.5.3. The `UserDir` Directive

The `UserDir` directive is disabled by default because it can confirm the presence of a user account on the system. If you wish to enable user directory browsing on the server, use the following directives:

```
UserDir enabled
UserDir disabled root
```

These directives activate user directory browsing for all user directories other than `/root`. If you wish to add users to the list of disabled accounts, add a space delimited list of users on the `UserDir disabled` line.

5.5.4. Do Not Remove the `IncludesNoExec` Directive

By default, the server-side includes module cannot execute commands. It is ill advised to change this setting unless you absolutely have to, as it could potentially enable an attacker to execute commands on the system.

5.5.5. Restrict Permissions for Executable Directories

Be certain to only allow write permissions for the root user only for any directory containing scripts or CGIs. This can be accomplished by typing the following commands:

```
chown <directory_name>
chmod 755 <directory_name>
```

Also, always verify that any scripts you are running work as intended *before* putting them into production.

5.6. Securing FTP

The File Transport Protocol (FTP) is an older TCP protocol designed to transfer files over a network. Because all transactions with the server, including user authentication, are unencrypted, it is considered an insecure protocol and should be carefully configured.

Red Hat Linux provides four FTP servers.

- `gssftpd` — A kerberized FTP daemon which does not pass authentication information over the network.
- **Red Hat Content Accelerator** (`tux`) — A kernel-space Web server with FTP capabilities.
- `vsftpd` — A simplified, security oriented implementation of the FTP service.
- `wu-ftpd` — A highly configurable, full-featured FTP daemon.

The following security guidelines are for setting up the `wu-ftpd` and `vsftpd` services.

Important

> If you activate both the `wu-ftpd` and `vsftpd` services, `xinetd` will only activate `vsftpd` because it comes first alphabetically.

5.6.1. FTP Warning Banner

Returning a customized banner to FTP clients when they connect is a good idea, as it helps disguise what system the FTP server is running on. You can send banners to incoming connections either using TCP wrappers as described in Section 5.1.1.1 or as described below.

For `vsftpd`, add the following line to its `xinetd` configuration file, `/etc/xinetd.d/vsftpd`:

```
banner /etc/banners/warning.msg
```

For `wu-ftpd` add the exact same line to its configuration file, `/etc/ftpaccess`.

The contents of the banner file for `vsftpd` should look something like this:

```
220-Hello, all activity on ftp.example.com is logged.
```

Note

The `220-` is not necessary when in the banner file for `wu-ftpd`.

5.6.2. FTP Greeting Banner

After login all users are presented with a greeting banner. By default, this banner includes version information useful to crackers trying to identify weaknesses in a system.

To change the greeting banner for `wu-ftpd`, add the following directive to `/etc/ftpusers`:

```
greeting text <insert_greeting_here>
```

To change the greeting banner for `vsftpd`, add the following directive to `/etc/vsftpd.conf`:

```
ftpd_banner=<insert_greeting_here>
```

5.6.3. Anonymous Access

For both `wu-ftpd` and `vsftpd`, the presence of the `/var/ftp/` directory activates the anonymous account.

The easiest way to create this directory is to install the `anonftp` package. This package sets the directory tree up for the anonymous user and sets up the permissions to read-only for anonymous users.

By default the anonymous user cannot write to any directories.

Caution

If enabling anonymous access to an FTP server, be careful where you store sensitive data.

5.6.3.1. Anonymous Upload

If you want to allow anonymous users to upload, it is recommended you create a write-only directory within `/var/ftp/pub/`.

To do this type:

```
mkdir /var/ftp/pub/upload
```

Next change the permissions so that anonymous users cannot see what is within the directory by typing:

```
chmod 733 /var/ftp/pub/upload
```

A long format listing of the directory should look like this:

```
drwxr--r--    2 root     ftp          4096 Aug 20 18:26 upload
```

Warning

Administrators who allow anonymous users to read and write in directories often find that their server become a repository of stolen software.

5.6.4. User Accounts

Because FTP passes unencrypted usernames and passwords over insecure networks for authentication, it is a good idea to deny system users access to the server from their user accounts.

To disable user accounts in `wu-ftpd`, add the following directive to `/etc/ftpusers`:

```
deny-uid *
```

To disable user accounts in `vsftpd`, add the following directive to `/etc/vsftpd.conf`:

```
local_enable=NO
```

5.6.4.1. Restricting User Accounts

The easiest way to disable a specific group of accounts, such as the root user and those with `sudo` privileges from accessing th FTP server is to use a PAM list file as described in Section 4.4.2.4. The PAM configuration file for `wu-ftpd` is `/etc/pam.d/ftp`. The PAM configuration file for `vsftpd` is `/etc/pam.d/vsftpd`.

It is also possible to perform this test within each service directly.

To disable specific user accounts in `wu-ftpd`, add the username to `/etc/ftpusers`:

To disable specific user accounts in `vsftpd`, add the username to `/etc/vsftpd.ftpusers`:

5.6.5. Use TCP Wrappers To Control Access

You can use TCP wrappers to control access to either FTP daemon as outlined in Section 5.1.1.

5.6.6. Use `xinetd` To Control the Load

You can use `xinetd` to control the amount of resources the FTP server consumes and to limit the effects of denial of service attacks. See Section 5.1.2 for more on how to do this.

5.7. Securing Sendmail

Sendmail is a Mail Transport Agent (MTA) that uses the Simple Mail Transport Protocol (SMTP) to deliver electronic messages between other MTAs and to email clients or delivery agents. Although many MTAs are capable of encrypting traffic between one another, most do not, so sending email over any public networks is considered an inherently insecure form of communication.

For more in formation about how email works and an overview of common configuration settings, see the chapter titled *Email* in the *Official Red Hat Linux Reference Guide*. This section assumes a basic knowledge of how to generate a valid `/etc/mail/sendmail.cf` by editing the `/etc/mail/sendmail.mc` and running the `m4` command as explained in the *Official Red Hat Linux Reference Guide*.

It is recommended that anyone planning to implement a Sendmail server address the following issues.

5.7.1. Limiting Denial of Service Attack

Because of the nature of email, a determined attacker can flood the server with mail fairly easily and cause a denial of service. By setting limits to the following directives to `/etc/mail/sendmail.mc` the effectiveness of such attacks will be limited limited.

- `confCONNECTION_RATE_THROTTLE` — The number of connections the server can receive per second. By default, Sendmail does not limit the number of connections. If a limit is set and reached, further connections are delayed.
- `confMAX_DAEMON_CHILDREN` — The maximum number of child processes that can be spawned by the server. By defaultt, Sendmail does not assign a limit to the number of child processes. If a limit is set and reached, further connections are delayed.
- `confMIN_FREE_BLOCKS` — The minimum number of free blacks which must be available for the server to accept mail. The default is 100 blocks.
- `confMAX_HEADERS_LENGTH` — The maximum acceptable size (in bytes) for a message header.
- `confMAX_MESSAGE_SIZE` — The maximum acceptable size (in bytes) for any one message.

5.7.2. NFS and Sendmail

Never put the mail spool directory, `/var/spool/mail/`, on an NFS shared volume.

Because NFS does not maintain control over user and group IDs, two or more users can have the same UID and therefore recieve and read each other’s mail.

5.7.3. Mail-only Users

To help prevent local user exploits on the Sendmail server, it is best for mail users to only access the Sendmail server using an Email program. Shell accounts on the mail server should not be allowed and all user shells in the `/etc/passwd` file should be set to `/bin/false` (with the possible exception of the root user.

5.8. Verifying Which Ports Are Listening

Once you have configured services on the network, it is important to keep tabs on which ports are actually listening to the systems network interfaces. Any open ports can be evidence of an intrusion.

There are two basic approaches for listing the ports that are listening on the network. The less reliable approach is to query the network stack by typing commands such as `netstat -an` or `lsof -i`. This method is less reliable since the program does not connect to the machine from the network, but rather checks to see what is running. For this reason, these applications are frequent targets for replacement by attackers. In this way, crackers attempt to cover their tracks if they open network ports.

A more reliable way to check which ports are listening on the network by using a port scanner such as `nmap`.

The following command issued from the console determines which ports are listening for TCP connections from the network:

```
nmap -sT -O localhost
```

The output of this command looks like the following:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1596 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
111/tcp    open        sunrpc
515/tcp    open        printer
834/tcp    open        unknown
6000/tcp   open        X11
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.5.25 or Gentoo 1.2 Linux 2.4.19 rc1-
rc7)

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

This output shows the system is running `portmap` due to the presence of the `sunrpc` service. However, there is also a mystery service on port 834. To check if the port is associated with the official list of known services, type:

```
cat /etc/services | grep 834
```

This command returns no output. This indicates that while the port is in the reserved range (meaning 0 through 1023) and requires root access to open, it is not associated with a known service.

Next, you can check for information about the port using `netstat` or `lsof`. To check for port 834 using `netstat`, use the following command:

```
netstat -anp | grep 834
```

The command returns the following output:

```
tcp   0    0 0.0.0.0:834    0.0.0.0:*   LISTEN   653/ypbind
```

The presence of the open port in `netstat` is reassuring because a cracker opening a port surreptitiously on a hacked system would likely not allow it to be revealed through this command. Also, the `[p]` option reveals the process id (PID) of the service which opened the port, in this case the open port belongs to `ypbind` (NIS), which is an RPC service handled in conjunction with the `portmap` service.

The `lsof` command reveals similar information since it is also capable of linking open ports to services:

```
lsof -i | grep 834
```

Below is the relevant portion of the output for this command:

```
ypbind      653       0     7u  IPv4       1319                     TCP *:834 (LISTEN)
ypbind      655       0     7u  IPv4       1319                     TCP *:834 (LISTEN)
ypbind      656       0     7u  IPv4       1319                     TCP *:834 (LISTEN)
ypbind      657       0     7u  IPv4       1319                     TCP *:834 (LISTEN)
```

As you can see, these tools tell can reveal a lot about the status of the services running on a machine. These tools are flexible and can provide a wealth of information about network services and configuration. Consulting the man pages for `lsof`, `netstat`, `nmap`, and `services` is therefore highly recommended.

Chapter 6.

Virtual Private Networks

Organizations with several satellite offices often connect to each other with dedicated lines for efficiency and protection of sensitive data in transit. For example, many businesses use frame relay or Asynchronous Transfer Mode (ATM) lines as an end-to-end networking solution to link one office with others. This can be an expensive proposition, especially for small or medium sized businesses (SMBs) that want to expand without paying the high costs associated with enterprise-level, dedicated digital circuits.

Engineers have developed a cost-effective solution to this problem in the form of *Virtual Private Networks* (VPNs). Following the same functional principles as dedicated circuits, Virtual Private Networks allow for secured digital communication between two parties (or networks), creating a Wide Area Network (WAN) from existing LANs. Where it differs from frame relay or ATM is in its transport medium. VPNs transmit over IP or datagram (UDP) layers, making it a secure conduit through the Internet to an intended destination. Most free software VPN implementations incorporate open standard, open source encryption to further mask data in transit.

Some organizations employ hardware VPN solutions to augment security, while others use the software or protocol-based implementations. There are several vendors with hardware VPN solutions such as Cisco, Nortel, IBM, and Checkpoint. There is a free software-based VPN solution for Linux called *FreeS/Wan* that utilizes a standardized *IPSec* implementation. These VPN solutions act as specialized routers that sit between the IP connection from one office to another. When a packet is transmitted from a client, it sends it through the router or gateway, which then adds header information for routing and authentication called the Authentication Header (AH) and trailer information for CRC file integrity and security called the Encapsulation Security Payload (ESP).

With such a heightened level of security, a cracker must not only intercept a packet, but decrypt the packet as well (which, in the case of most VPNs, usually employ a triple Data Encryption Standard (3DES) 192-bit cypher). Intruders who employ a man-in-the-middle attack between server and client must also have access to the keys exchanged for authenticating sessions. VPNs are secure and effective means to connect multiple remote nodes to act as a unified Intranet.

6.1. VPNs and Red Hat Linux

Red Hat Linux users and administrators have various options in terms of implementing a software solution to secure their WAN. There are, however, two methods of implementing VPN and VPN-equivalent connections that currently ship with Red Hat Linux. One equivalent solution involves using OpenSSH as a tunnel between two remote nodes. This solution is a sound alternative to telnet, rsh, and other remote host communication protocols, but does not completely address the usability needs of all corporate telecommuters. Another solution that is more adherent to the de facto definition of a VPN is Crypto IP Encapsulation (*CIPE*), a method of connecting remote LANs to function as a unified network.

6.2. Crypto IP Encapsulation (CIPE)

CIPE is a VPN implementation developed primarily for Linux. CIPE uses encrypted IP packets that are encapsulated, or "wrapped", in datagram (UDP) packets. Packets are given destination header information and are encrypted using the default CIPE encryption mechanism, then transferred over IP as UDP packets via its own virtual device (cipcb*x*) over a carrier network (such as the Internet) to an intended remote node. The following figure shows a typical CIPE setup connecting two Linux-based networks:

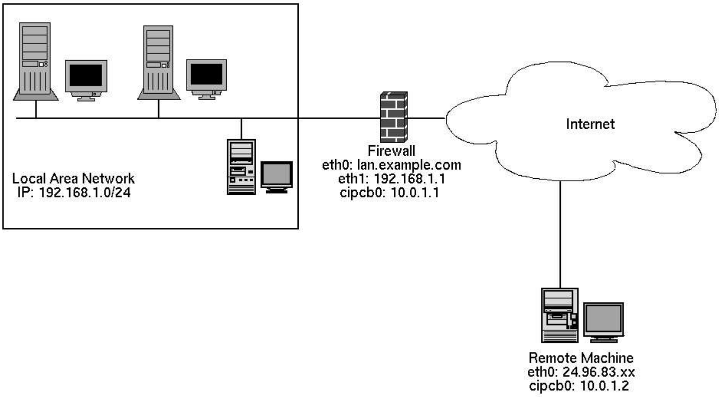


Figure 6-1. A Network and Remote Client Connected by CIPE

The diagram shows a network running CIPE on the firewall, and a remote client machine acting as a CIPE-enabled node. The CIPE connection acts as a tunnel through which all Intranet-bound data is routed between remote nodes. All data is encrypted using dynamically-generated 128-bit keys, and can be further compressed for large file transfers or to tunnel X applications to a remote host. CIPE can be configured for communication between two or more CIPE-enabled Linux machines and also has network drivers for Win32-based operating systems.

6.2.1. Why Use CIPE?

There are several reasons why CIPE would be a smart choice for security and systems administrators:

- Red Hat Linux ships with CIPE, so it is available to all Red Hat Linux edge machines (for example, firewalls or gateways) that you wish to connect to your Intranet. Red Hat Linux also includes CIPE-supported encryption ciphers in its general distribution.
- CIPE supports encryption using either of the standard Blowfish or IDEA encryption algorithms. Depending on encryption export regulations in your country, you may use the default (Blowfish) to encrypt all CIPE traffic on your Intranet.
- Because CIPE is software based, any older machine that is able to run Red Hat Linux can become a CIPE gateway, saving an organization from having to purchase expensive dedicated VPN hardware simply to connect two LANs securely.
- CIPE is actively developed to work in conjunction with IP Tables, IP Chains, and other rules-based firewalls. Simple peer acceptance of incoming CIPE UDP packets is all that is needed to coexist with existing firewall rules.
- Administrators can configure CIPE through text files.

6.2.2. CIPE Installation

The installation of CIPE is equivalent to installing a new network interface under Linux. The CIPE RPM contains configuration files found in `/etc/cipe/`, the CIPE daemon (`/usr/sbin/ciped-cb`), network scripts that load the kernel module and activates/deactivates the CIPE interface (`if*-cipcb`), and sample configuration files found in `/usr/share/doc/cipe-<version>/samples/`. There is also a detailed texinfo page explaining the CIPE protocol.

This guide details a sample configuration involving a workstation client that wishes to connect securely to a remote LAN with a CIPE gateway. The workstation uses a dynamic IP address via cable modem connection, while the CIPE-enabled gateway machine employs the 192.168.1.0/24 range. This is what is known as a "typical" CIPE configuration. The following diagram illustrates the network architecture for this CIPE configuration:

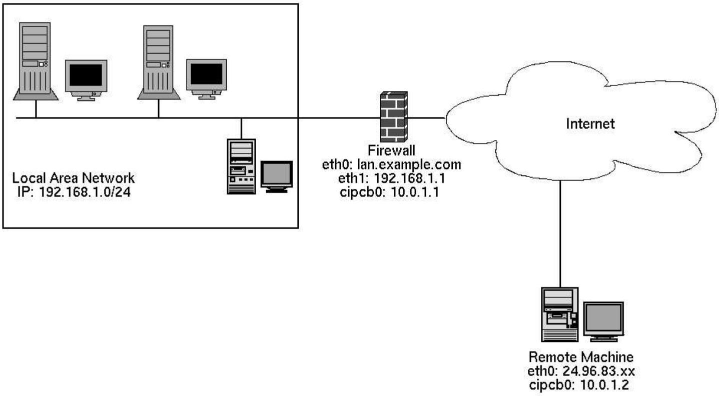


Figure 6-2. Typical CIPE Server and Client Configuration

Installing CIPE between the client and the CIPE server will allow for a secured peer-to-peer connection using the Internet as a medium for transmission of WAN traffic. The client workstation will then transfer a file through the Internet to the CIPE-enabled firewall, where each packet will be timestamped, encrypted, and given the peer address of the receiving CIPE-enabled firewall. The destination firewall then reads the header information, strips it, and sends it through to the remote LAN router to be then routed to its destination node. This process is seamless and completely transparent to end users. The majority of the transaction is done between the CIPE-enabled peers.

6.2.3. CIPE Server Configuration

To setup the CIPE server, simply install the RPM package from the Red Hat Linux disc or via Red Hat Network.

Important

> If you are using an older version of Red Hat Linux and/or have an older version of CIPE, you should upgrade to the latest version.

The next step is to copy the sample configuration files from `/usr/share/doc/cipe-`*`version`*`/samples` (where *`version`* is the version of CIPE installed on your system) to `/etc/cipe/`. Once they are copied, you will need to edit the `/etc/cipe/options.cipcb`*`x`* (*x* is incremental starting from *0*, for those who wish to have more than one CIPE connection on the CIPE server) file to include your LAN subnet addresses and publicly routable firewall IP addresses. The following is the example `options` file included with the Red Hat Linux CIPE RPM which, for this example, is renamed to `options.cibcb0`:

```
# Surprise, this file allows comments (but only on a line by themselves)
# This is probably the minimal set of options that has to be set
# Without a "device" line, the device is picked dynamically

# the peer's IP address
ptpaddr         6.5.4.3

# our CIPE device's IP address
ipaddr          6.7.8.9

# my UDP address. Note: if you set port 0 here, the system will pick
# one and tell it to you via the ip-up script. Same holds for IP 0.0.0.0.
me              bigred.inka.de:6789

# ...and the UDP address we connect to. Of course no wildcards here.
peer            blackforest.inka.de:6543

# The static key. Keep this file secret!
# The key is 128 bits in hexadecimal notation.
key             xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

The `ptpaddr` is the remote LAN's CIPE address. The `ipaddr` is the workstation's CIPE IP address. The `me` address is the client's publicly routable IP address that sends the UDP packets over the Internet, while `peer` is the publicly routable IP address of CIPE server. Note that the client workstation's IP address is 0.0.0.0 because it uses a dynamic connection. The CIPE client will handle the connection to the host CIPE server. The `key` field (represented by x's; your key should be secret) is the shared static key. This key *must* be the same for both peers or connection will not be possible. See Section 6.2.6 for information on how to generate a shared static key for your CIPE machines.

Here is the edited `/etc/cipe/options.cipcb0` that the client workstation will use:

```
ptpaddr        10.0.1.2
ipaddr         10.0.1.1
me             0.0.0.0
peer           LAN.EXAMPLE.COM:6969
key            123456ourlittlesecret7890shhhh
```

Here is the `/etc/cipe/options.cipcb0` file for the CIPE server:

```
ptpaddr        10.0.1.1
ipaddr         10.0.1.2
me             LAN.EXAMPLE.COM:6969
peer           0.0.0.0
key            123456ourlittlesecret7890shhhh
```

6.2.4. Configuring Clients for CIPE

After successfully configuring the CIPE server and testing for functionality, you can now deploy the connection on the client machine.

The CIPE client should be able to connect and disconnect the CIPE connection in an automated way. Therefore, CIPE contains built-in mechanisms to customize settings for individual uses. For example, a remote employee can connect to the CIPE device on the LAN by typing the following:

```
/sbin/ifup cipcb0
```

The device should automatically come up, and any firewall rules and routing information should be executed along with the connection. The remote employee should be able to terminate the connection with the following:

```
/sbin/ifdown cipcb0
```

Configuring clients requires the creation of localized scripts that are run after the device has loaded. The device configuration itself can be configured locally via a user-created file called `/etc/sysconfig/network-scripts/ifcfg-cipcb0`. This file contains pieces of parameters that determine whether the CIPE connection occurs at boot-time and what the name of the CIPE device is, among other things. The following is the `ifcfg-cipcb0` file for a remote client connecting to the LAN A CIPE server:

```
# These first four should be self explanatory. Change as required.
DEVICE=cipcb0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no

# This is the device for which we add a host route to our CIPE peer through.
# You may hard code this, but if left blank, we punt and try to guess from
# the routing table in the /etc/cipe/ip-up.local file.
PEERROUTEDEV=

# We need to use internal DNS when connected via cipe. These may change,
# but for now, they are correct as of 20010604.
DNS=192.168.1.254
```

The CIPE device is named `cipcb0`. The CIPE device will be loaded at boot-time (configured via the `ONBOOT` field) and will not use a boot protocol (for example, DHCP) to receive an IP address for the device. The `PEERROUTEDEV` field determines the CIPE server device name that the client will be connecting to. If no device is specified in this field, one will be determined after the device has been loaded.

If your internal networks are behind a firewall (always a good policy), you need to set rules to allow the CIPE interface on the client machine to send and receive UDP packets. Refer to Chapter 7 for information on configuring a firewall for Red Hat Linux. For our example, IP tables rules are implemented.

Note

Clients should be configured such that all localized parameters are placed in a user-created file called `/etc/cipe/ip-up.local`. The local parameters should be reverted when the CIPE session is shut down using `/etc/cipe/ip-down.local`.

Firewalls should be configured on client machines to accept the CIPE UDP encapsulated packets. Rules may vary widely, but the basic acceptance of UDP packets is required for CIPE connectivity. The following IP tables rules allow UDP packets for the CIPE connection for the remote client connecting to the LAN; the final rule adds IP Masquerading to allow the remote client to communicate to the LAN and the Internet:

```
/sbin/modprobe iptables
/sbin/service iptables stop
/sbin/iptables -P INPUT REJECT
/sbin/iptables -F INPUT
/sbin/iptables -A INPUT -j ACCEPT -p udp -s 10.0.1.1
/sbin/iptables -A INPUT -j ACCEPT -i cipcb0
/sbin/iptables -A INPUT -j ACCEPT -i lo
/sbin/iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth0 -j MASQUERADE
```

You must also add routing rules to the client machine to access the nodes behind the CIPE connection as if they were on the local network. This can be done by running the `route` command. For our example, the client workstation would need to add the following network route:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 10.0.1.2
```

The following shows the final `/etc/cipe/ip-up.local` script for the client workstation:

```
#!/bin/bash -v
if [ -f /etc/sysconfig/network-scripts/ifcfg-$1 ] ; then
        . /etc/sysconfig/network-scripts/ifcfg-$1
else
        cat <<EOT | logger
Cannot find config file ifcfg-$1. Exiting.
EOF
        exit 1
fi

if [ -n ${PEERROUTEDEV} ]; then
        cat <<EOT | logger
Cannot find a default route to send cipe packets through!
Punting and hoping for the best.
EOT
        # Use routing table to determine peer gateway
        export PEERROUTEDEV=‘/sbin/route -n | grep ^0.0.0.0 | head -n 1 \
           | awk '{ print $NF }'‘

fi

####################################################
# Add The routes for the remote local area network #
####################################################

route add -host 10.0.1.2 dev $PEERROUTEDEV
route add -net 192.168.1.0 netmask 255.255.255.0 dev $1

####################################################
# IP TABLES Rules to restrict traffic              #
####################################################

/sbin/modprobe iptables
/sbin/service iptables stop
/sbin/iptables -P INPUT REJECT
/sbin/iptables -F INPUT
/sbin/iptables -A INPUT -j ACCEPT -p udp -s 10.0.1.2
/sbin/iptables -A INPUT -j ACCEPT -i $1
/sbin/iptables -A INPUT -j ACCEPT -i lo
/sbin/iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth0 -j MASQUERADE
```

6.2.5. Customizing CIPE

CIPE can be configured in numerous ways, from passing parameters as command-line arguments when starting `ciped` to generating new shared static keys. This allows a security administrator the flexibility to customize CIPE sessions to ensure security as well as increase productivity. The following chart details some of the command-line parameters when running `ciped`.

Note

The most common parameters should be placed in the `/etc/cipe/options.cipcbx` file for automatic loading at runtime. Be aware that any parameters passed at the command-line as options *will override* respective parameters set in the `/etc/cipe/options.cipcbx` configuration file.

Parameter	Description
`arg`	Passes arguments to the `/etc/cipe/ip-up` initialization script
`cttl`	Sets the Carrier Time To Live value; recommended value is 64
`debug`	Boolean value to enable debugging
`device`	Names the cipe device
`ipaddr`	Publicly-routable IP address of the CIPE machine
`ipdown`	Choose an alternate `ip-down` script than the default `/etc/cipe/ip-down`
`ipup`	Choose an alternate `ip-up` script than the default `/etc/cipe/ip-down`
`key`	Specify a shared static key for CIPE connection
`maxerr`	Number of errors allowable before the CIPE daemon quits
`me`	UDP address of the CIPE machine
`mtu`	Set the device maximum transfer unit
`nokey`	Do not use encryption
`peer`	The peer's CIPE UDP address
`ping`	Set CIPE-specific (non-ICMP) keepalive ping interval
`socks`	IP address and port number of the SOCKS server for proxy connections
`tokey`	Set *dynamic* key lifetime; default is 10 minutes (600 seconds)
`tokxc`	Timeout value for shared key exchange; default is 10 seconds
`tokxts`	Shared key exchange timestamp timeout value; default is 0 (no timestamps)
`toping`	Timeout value for keepalive pings; default is 0

Table 6-1. CIPE Parameters

6.2.6. CIPE Key Management

As previously mentioned, CIPE incorporates a secure combination of static link keys and encrypted traffic to create a secure tunnel over carrier networks such as the Internet. The use of static, *link keys* provides a common point of reference for two CIPE-enabled networks to pass information securely. Therefore, it is imperative that both CIPE-enabled network gateways share the *exact* same key, or CIPE communication will not be possible.

6.2.6.1. Generating CIPE Keys

Generating CIPE keys requires knowledge of what kind of keys are compatible. Random alphanumeric generators do not work. Static keys must be 128-bit, 32-character strings. These can be created by piping an arbitrary file or outputted process through the `md5sum` command. For example:

```
ps -auxw | md5sum
```

Place this key in the `/etc/cipe/options.cipcb0` file for all CIPE servers and clients.

Chapter 7.

Firewalls

Information security is commonly thought of as a process and not a product. However, standard security implementations usually employ some form of dedicated mechanism to control access privileges and restrict network resources to users who are authorized, identifiable, and traceable. Red Hat Linux includes several powerful tools to assist administrators and security engineers with network-level access control issues.

Aside from VPN solutions such as CIPE or IPSec (discussed in Chapter 6), firewalls are one of the core components of network security implementation. Several vendors market firewall solutions catering to all levels of the marketplace: from home users protecting one PC to data center solutions safeguarding vital enterprise information. Firewalls can be standalone hardware solutions, such as firewall appliances by Cisco, Sonicwall, and Nokia. There are also proprietary software firewall solutions developed for home and business markets by vendors such as Checkpoint, McAfee, and Symantec.

Apart from the differences between hardware and software firewalls, there are also differences in the way firewalls function that separate one solution from another. Table 7-1 details three common types of firewalls and how they function:

Method	**Description**	**Advantages**	**Disadvantages**
NAT	*Network Address Translation* (NAT) places internal network IP subnetworks behind one or a small pool of external IP addresses, masquerading all requests to one source rather than several	· Can be configured transparently to machines on a LAN · Protection of many machines and services behind one or more external IP address(es), simplifying administration duties · Restriction of user access to and from the LAN can be configured by opening and closing ports on the NAT firewall/gateway	· Cannot prevent malicious activity once users connect to a service outside of the firewall.
Packet Filter	Packet filtering firewalls read each data packet that passes within and outside of a LAN. It can read and process packets by header information and filters the packet based on sets of programmable rules implemented by the firewall administrator. The Linux kernel has built-in packet filtering functionality through the netfilter kernel subsystem.	· Customizable through the `iptables` front-end utility · Does not require any customization on the client side, as all network activity is filtered at the router level rather than at the application level · Since packets are not transmitted through a proxy, network performance is faster due to direct connection from client to remote host	· Cannot filter packets for content like proxy firewalls · Processes packets at the protocol layer, but cannot filter packets at an application layer · Complex network architectures can make establishing packet filtering rules difficult, especially if coupled with *IP masquerading* or local subnets and DMZ networks

Method	Description	Advantages	Disadvantages
Proxy	Proxy Firewalls filter all requests of a certain protocol or type from LAN clients to a proxy machine, which then makes those requests to the Internet on behalf of the local client. A proxy machine acts as a buffer between malicious remote users and the internal network client machines.	· Gives administrators control over what applications and protocols function outside of the LAN · Some proxy servers can cache data so that clients can access frequently requested data from the local cache rather than having to use the Internet connection to request it, which is convenient for cutting down on unnecessary bandwidth consumption · Proxy services can be logged and monitored closely, allowing tighter control over resource utilization on the network	· Proxies are often application specific (HTTP, telnet, etc.) or protocol restricted (most proxies work with TCP connected services only) · Application services cannot run behind a proxy, so your application servers must use a separate form of network security Proxies can become a network bottleneck, as all requests and transmissions are passed through one source rather than direct client to remote service connections

Table 7-1. Firewall Types

7.1. Netfilter and `iptables`

The Linux kernel features a powerful networking subsystem called *netfilter*. The netfilter subsystem provides stateful or stateless packet filtering as well as NAT and IP masquerading services. Netfilter also has the ability to *mangle* IP header information for advanced routing and connection state management. Netfilter is controlled through the `iptables` executable.

7.1.1. `iptables` Overview

The power and flexibility of netfilter is implemented through the `iptables` interface. This command-line tool is similar in syntax to its predecessor, `ipchains`; however, `iptables` uses the netfilter subsystem to enhance network connection, inspection, and processing; whereas `ipchains` used intricate rule sets for filtering source and destination paths, as well as connection ports for both. `iptables` features advanced logging, pre- and post-routing actions, network address translation, and port forwarding all in one command-line interface.

This section provides an overview of IPTables. For more detailed information about `iptables`, refer to the *Official Red Hat Linux Reference Guide*.

7.1.2. Using `iptables`

The first step in using `iptables` is to start the `iptables` service. This can be done with the command:

```
service iptables start
```

Warning

> The IPChains and IP6Tables services must be turned off to use the IPTables service with the following commands:
>
> ```
> service ipchains off
>
> service ip6tables off
> ```

To make IPTables start by default whenever the system is booted, you must change runlevel status on the service using `chkconfig`.

```
chkconfig --level 345 ip6tables on
```

The syntax of `iptables` is separated into tiers. The main tier is the *chain*. A chain specifies the state at which a packet will be manipulated. For example:

```
iptables -P OUTPUT ACCEPT
```

The `OUTPUT` chain specifies any packets that originate from inside a LAN and travels outside (for example, to a remote website). In the example above, the rule states that all packets coming from the inside to the outside of the local network is allowed to pass through the firewall. This is usually an acceptable rule for administrators because the likelihood of dangerous packets going out into an untrusted carrier network such as the Internet is small compared to malicious packets going into the local network. The three built-in chains of `iptables` (that is, the chains that affect every packet which traverses a network) are INPUT, OUTPUT, and FORWARD. These chains are permanent and cannot be deleted, whereas user-defined chains can be.

Some basic rules established from the outset can aid as a foundation for building more detailed, user-defined rules. For example, you may want to allow all connections originating from the inside by default and then customize unique cases with their own rule sets. Accepting all `OUTPUT` by default is a sufficient foundation to build upon regarding outbound connections. It is also recommended that, by default, all incoming connections be denied by your firewall. The following rule will block all incoming connections:

```
iptables -P INPUT REJECT
```

Additionally, it is recommended that any *forwarded packets* — network traffic that is to be routed from the firewall to its destination node — be denied as well, to restrict internal clients from inadvertent exposure to the Internet (for example, if a LAN user accidentally turns on a service on some arbitrary port, then your network becomes vulnerable because of that machine's service). To do this, use the following rule:

```
iptables -P FORWARD REJECT
```

After setting basic rules, you can now create new rules for your particular network and security requirements. The following sections will outline some common rules you may implement in the course of building your `iptables` firewall.

7.1.2.1. Saving and Restoring IPTables Rules

Firewall rules are only valid for the time the computer is on. If you reboot your system, the rules will be automatically flushed and reset. To save your rules so that they will load later, use the following command:

```
service iptables save
```

The rules will be stored in the file `/etc/sysconfig/iptables` and will be applied whenever the service is started, restarted, or the machine rebooted.

7.1.3. `INPUT` Filtering

Keeping remote attackers out of a LAN is an important aspect of network security, if not the *most* important. The integrity of a LAN should be protected from malicious remote users through the use of stringent firewall rules. In the following example, The LAN (which uses a private class C 192.168.1.0/24 IP range) rejects telnet access from the outside. The rule for this looks like the following:

```
iptables -A INPUT -p tcp --sport telnet -j REJECT
```

The rule rejects all outside tcp connections using the telnet protocol (typically port 23) with a `connection refused` error message. Rules using the `--sport` or `--dport` options can use either port numbers or common service names. So, using both `--sport telnet` and `--sport 23` are acceptable.

Note

> There is a distinction between the `REJECT` and `DROP` target actions. The `REJECT` target denies access and returns a `connection refused` error to users who attempt to `telnet` users. The `DROP`, as the name implies, simply drops the packet without any warning to `telnet` users. Administrators can use their own discretion when using these targets; however, to avoid user confusion and attempts to continue connecting, the `REJECT` target is recommended.

There may be times when certain users require remote access to the LAN from the road or from a field office. Secure services, such as SSH and CIPE, can be used for encrypted remote connection to LAN services. For administrators with PPP-based resources (such as modem banks or bulk ISP accounts), dialup access can used to circumvent firewall barriers securely, as modem connections are typically behind a firewall/gateway because they are direct connections. However, for remote users with broadband connections, special cases can be made. You can set `iptables INPUT` to accept connections from remote SSH and CIPE clients. For example, to allow remote SSH access to the LAN, the following may be used:

```
iptables -A INPUT -p tcp --sport 22 -j ACCEPT
```

CIPE connection requests from the outside can be accepted with the following command:

```
iptables -A INPUT -p udp -i cipcb0  -j ACCEPT
```

Since CIPE uses its own virtual device which transmits datagram (UDP) packets, the rule allows the *`cipcb0`* interface for incoming connections, instead of source or destination ports (though they can be used in place of device options). For information about using CIPE, refer to Chapter 6.

There are other services for which you may need to define `INPUT` rules. Refer to the *Official Red Hat Linux Reference Guide* for comprehensive information on `iptables` and its various options.

7.1.4. `OUTPUT` Filtering

There may be instances when an administrator must block certain users on the internal network from making outbound connections. Perhaps the administrator intends to curtail malicious trojans from contacting their intended hosts or wants to keep an employee from misusing network resources for inappropriate or illicit reasons. In these cases, specialized rules can be established using `OUTPUT` action in `iptables`. The `OUTPUT` action places restrictions on outbound data.

Suppose an administrator notices heavy amounts of network traffic on port 6699 (a commonly used port for peer-to-peer file sharing services). To stop the traffic and conserve bandwidth for legitimate business purposes, the administrator can block LAN users from communicating on this port. After a thorough examination of what other services may be adversely affected by the blockage of port 6699, the administrator can add the following rule to the firewall and effectively block outbound traffic from the source port:

```
iptables -A OUTPUT -p TCP --sport 6699 -j REJECT
```

More elaborate rules can be created that control access to specific subnets, or even specific nodes, within a LAN. You can also restrict certain dubious services such as trojans, worms, and other client/server viruses from contacting their server. For example, there are some trojans that scan networks for services on ports from 31337 to 31340 (called the *elite* ports in cracking lingo). Since there are no legitimate services that communicate via these non-standard ports, blocking it can effectively diminish the chances that potentially infected nodes on your network independently communicate with their remote master servers.

```
iptables -A OUTPUT -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
```

7.1.5. `FORWARD` and NAT Rules

Most organizations are allotted a limited number of publicly routable IP addresses from their ISP. Due to this limited allowance, administrators must find creative ways to share access to Internet services without giving scarce IP addresses to every node on the LAN. Using class C private IP address is the common way to allow all nodes on a LAN to properly access network services internally and externally. Edge routers (such as firewalls) can receive incoming transmissions from the Internet and route the bits to the intended LAN node; at the same time, it can also route outgoing requests from a LAN node to the remote Internet service. This forwarding of network traffic can become dangerous at times, especially with the availability of modern cracking tools that can spoof *internal* IP addresses and make the remote attacker's machine act as a node on your LAN. To prevent this, `iptables` provides routing and forwarding policies that you can implement to prevent aberrant usage of network resources.

The `FORWARD` policy allows an administrator to control where packets can be routed. For example, to allow forwarding for an entire internal IP address range, the following rule can be set:

```
iptables -A FORWARD -i eth1 -j ACCEPT
```

Note

By default, IPv4 policy in Red Hat Linux kernels disables support for IP forwarding, which prevents boxes running Red Hat Linux from functioning as dedicated edge routers. To enable IP forwarding, run the following command or place it in your firewall initialization script:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

`FORWARD` rules can be implemented to restrict certain types of traffic to the LAN only, such as local network file shares through NFS or Samba. The following rules reject outside connections to Samba shares:

```
iptables -A FORWARD -p tcp --sport 137:139 -j DROP
iptables -A FORWARD -p udp --sport 137:139 -j DROP
```

To take the restrictions a step further, you can block all outside connections that attempt to spoof private IP address ranges to infiltrate your LAN. If a LAN uses the 192.168.1.0/24 range, a rule can set the Internet facing network device (for example, eth0) to drop any packets to that device with an address in your LAN IP range. Because it is recommended to reject forwarded packets as a default policy, any other spoofed IP address will be rejected automatically.

```
iptables -A FORWARD -p tcp -s 192.168.1.0/24 -i eth0 -j DROP
iptables -A FORWARD -p udp -s 192.168.1.0/24 -i eth0 -j DROP
```

Rules can also be set to route traffic to certain machines, such as a dedicated HTTP or FTP server, preferably one that is isolated from the internal network on a DMZ. To set a rule for routing all incoming HTTP requests to a dedicated HTTP server at IP address 10.0.4.2 and port 80 (outside of the 192.168.1.0/24 range of the LAN), network address translation (NAT) calls a `PREROUTING` table to forward the packets to the proper destination (the \ denotes a continuation of a one-line command):

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
 -j DNAT --to 10.0.4.2:80
```

With this command, all HTTP connections to port 80 from the outside of the LAN will be routed to the HTTP server on a separate network from the rest of the internal network. This form of network segmentation can prove safer than allowing HTTP connections to a machine on the network.

7.2. `ip6tables`

The introduction of the next-generation Internet Protocol, called IPv6, expands beyond the 32-bit address limit of IPv4 (or IP). IPv6 supports 128-bit addresses and, as such, carrier networks that are IPv6 aware are able to address a larger number of routable addresses than IPv4.

Red Hat Linux supports IPv6 firewall rules using the Netfilter 6 subsystem and the `ip6tables` command. The first step in using `ip6tables` is to start the IP6Tables service. This can be done with the command:

```
service ip6tables start
```

Warning

The IPChains and IPTables services must be turned off to use the IP6Tables service using the following commands:

```
service ipchains stop
```

```
service iptables stop
```

To make IP6Tables start by default whenever the system is booted, you must change runlevel status on the service using `chkconfig`.

```
chkconfig --level 345 ip6tables on
```

The syntax is identical to `iptables` in every aspect except that `ip6tables` supports 128-bit addresses. For example, SSH connections on a IPv6-aware network server can be enabled with the following rule:

```
ip6tables -A INPUT -i eth0 -p tcp -s 3ffe:ffff:100::1/128 --dport 22 -j \
   ACCEPT
```

For more information about IPv6 networking, refer to the IPv6 Information Page at http://www.ipv6.org.

7.3. Additional Resources

There are several aspects to firewalls and the Linux Netfilter subsystem that could not be covered here. For more information, refer to the following resources.

7.3.1. Installed Documentation

- The *Official Red Hat Linux Reference Guide* has a comprehensive chapter on iptables, including definitions for all command options.
- The `iptables` manual page contains a brief summary of the various options, as well.

7.3.2. Useful Websites

- http://www.netfilter.org — The official homepage of the Netfilter/`iptables` project.
- http://www.redhat.com/support/resources/networking/firewall.html — Red Hat Support firewall resource page.
- http://www.tldp.org — The Linux Documentation Project contains several useful guides relating to firewall creation and administration.

7.3.3. Related Documentation

- *Linux Firewalls*, by Robert Ziegler, contains a wealth of information on building firewalls using both 2.2 kernel `ipchains` as well as Netfilter and `iptables`. Additional security topics such as remote access issues and Intrusion Detection Systems are also covered.

Chapter 8.

Hardware and Network Protection

The best practice before deploying a machine into a production environment or connecting your network to the Internet is to determine your organizational needs and how security can fit into the requirements as transparently as possible. Since the main goal of Official Red Hat Linux Security Guide is to explain how to secure Red Hat Linux operating system, a more detailed examination of hardware and physical network security is beyond the scope of this document. However, this chapter is a brief overview of establishing security policies with regard to hardware and physical networks. Important factors to consider are how computing needs and connectivity requirements fit into the overall security strategy. The following explains some of these factors in detail.

- *Connectivity* is the method by which an administrator intends to connect disparate resources on a network. An administrator may use Ethernet (hubbed or switched CAT-5/RJ-45 cabling), token ring, 10-base-2 coaxial cable, or even cable-free (wireless, 802.11*x*) technologies. Depending on which medium an administrator chooses, certain media and network topologies require complementary technologies such as hubs, routers, switches, base stations, and access points. Determining a functional network architecture will allow an easier administrative process if security issues arise.
- *Computing* involves more than just workstations running desktop software. Modern organizations require massive computational power and highly-available services, which can include mainframes, compute/server clusters, powerful workstations, and specialized appliances. With these organizational requirements, however, come increased susceptibility to hardware failure, natural disasters, and tampering or theft of equipment.

From these general considerations, administrators can get a better view of implementation. The design of a computing environment will then be based on both organizational need and security considerations — a true, "ground-up" implementation that places priority on both factors.

8.1. Secure Network Topologies

The foundation of a LAN is the *topology*, or network architecture. A topology is the physical and logical layout of a LAN in terms of resource provided, distance between nodes, and transmission medium. Depending upon the needs of the organization that the network will service, there are several choices available for network implementation. Each topology has its advantages and security issues that network architects should regard when designing their network layout.

8.1.1. Physical Topologies

As defined by the Institute of Electrical and Electronics Engineers (IEEE), there are three common topologies for physical connection of a LAN.

8.1.1.1. Ring Topology

The *Ring* topology connects each node by exactly two connections. This creates a ring structure where each node is accessible to the other either directly by its two physically closest neighboring nodes and indirectly through the physical ring. Token Ring, FDDI, and SONET networks are connected in this fashion (with FDDI utilizing a dual-ring technique); however, there are no common Ethernet connections using this physical topology, so rings are not commonly deployed except in legacy or institutional settings with a large installed base of nodes (for example, a university).

8.1.1.2. Linear Bus Topology

The *linear bus* topology consists of nodes which connect to a terminated main linear cable (the backbone). The linear bus topology requires the least amount of cabling and networking equipment, making it the most cost-effective topology. However, the linear bus depends on the backbone being constantly available, making it a single point-of-failure if it has to be taken off-line or is severed. Linear bus topologies are commonly used in peer-to-peer LANs using co-axial (coax) cabling and 50-93 ohm terminators at both ends of the bus.

8.1.1.3. Star Topology

The *Star* topology incorporates a central point where nodes connect and through which communication is passed. This centerpoint, called a *hub* can be either *broadcasted* or *switched*. This topology does introduce a single point of failure in the centralized networking hardware that will connect the nodes. However, because of this centralization, networking issues that affect segments or the entire LAN itself is easily traceable to this one source.

8.1.2. Transmission Considerations

In a broadcast network, a node will send a packet that traverses through every other node until the recipient accepts the packet. Every node in the network will conceivably receive this packet of data until the recipient processes the packet. In a broadcast network, all packets are sent in this manner.

In a switched network, packets are not broadcasted, but are processed in the switched hub which, in turn, will create a *direct* connection between the sending and recipient nodes using the unicast transmission principles. This eliminates the need to broadcast packets to each node, thus lowering traffic overhead.

The switched network also prevents packets from being intercepted by malicious nodes or users. In a broadcast network, since each node receives the packet en route to its destination, malicious users can set their Ethernet device to *promiscuous* mode and accept all packets regardless of whether or not the data is intended for them. Once in promiscuous mode, a sniffer application can be used to filter, analyze, and reconstruct packets for passwords, personal data, and more. Sophisticated sniffer applications will store such information in a text file and, perhaps, even send the information to an arbitrary source (for example, the malicious user's email address).

A switched network requires a network switch, a specialized piece of hardware which replaces the role of the traditional hub in which all nodes on a LAN are connected. Switches store MAC addresses of all nodes within an internal database, which it uses to perform its direct routing. Several manufacturers, including Cisco Systems, Linksys, and Netgear offer various types of switches with features such as 10/100-Base-T compatibility, gigabit Ethernet support, and support for Carrier Sensing Multiple Access and Collision Detection (CSMA/CD) which is ideal for high-traffic networks because it will queue connections and detect when packets collide in transit.

8.1.3. Wireless Networks

An emerging issue for enterprises today is that of mobility. Remote workers, field technicians, and executives require portable solutions, including laptops, Personal Digital Assistants (PDAs), and wireless access to network resources. The IEEE has established a standards body for the 802.11 wireless specification, which establishes standards for wireless data communication throughout all industries. The current standard in practice today is the 802.11b specification.

The 802.11b specification is actually a group of standards governing wireless communication and access control at the 2.4 GHz communication band. This specification has already been adopted at an industry level, and several vendors market 802.11b (also called *Wi-Fi*) access and compatibility as a value-added feature of their core offerings. Consumers have also embraced the standard for small-office/home-office (SOHO) networks. The popularity has also extended from LANs to MANs (Metropolitan Area Networks), especially in populated areas where a concentration of wireless access points (WAPs) are available. There are also wireless Internet service providers (WISPs) that cater to frequent travelers who require broadband Internet access to conduct business remotely.

The 802.11b specification allows for direct, peer-to-peer connections between nodes with wireless NICs. This loose grouping of nodes, called an *ad hoc* network, is ideal for quick connection sharing between two or more nodes, but introduces scalability issues that are not suitable for long-term wireless connectivity.

A more suitable solution for wireless access in fixed structures is to install one or more WAPs that connect to the traditional network and allowing wireless nodes to connect to through the WAP as if it were on the Ethernet-mediated network. The WAP effectively acts as a bridge router between the nodes connected to it and the rest of the network.

8.1.3.1. 802.11b Security

Although wireless networking is comparable in speed and certainly more convenient than traditional wired networking mediums, there are some limitations to the specification that warrants thorough consideration. The most important of these limitations is in its security implementation.

In the excitement of successfully deploying an 802.11x network, many administrators fail to exercise even the most basic security precautions. Since all 802.11b networking is done using high-band radio-frequency (RF) signals, the data transmitted is easily accessible to any user with a 802.11b NIC, a wireless network scanning tool such as **NetStumbler** or **Wellenreiter**, and common sniffing tools such as `dsniff` and `snort`. To prevent such aberrant usage of private wireless networks, the 802.11b standard uses the Wired Equivalency Privacy (WEP) protocol, which is an RC4-based 64- to 128-bit encrypted key shared between each node or between the AP and the node. This key will encrypt transmissions and decrypt incoming packets dynamically and transparently. Administrators often fail to employ this shared-key encryption scheme, however; either they forget to do so or choose not to do so because of performance degradation (especially over long distances). Enabling WEP on a wireless network can greatly reduce the possibility of data interception.

Relying on WEP, however, is still not a sound enough means of protection against determined malicious users. There are specialized utilities whose purpose is to crack the RC4 WEP encryption algorithm and exposes the shared key. **AirSnort** and **WEP Crack** are two such specialized applications. To protect against this, administrators should adhere to strict policies regarding usage of wireless methods to access sensitive information. Administrators may choose to augment the security of wireless by restricting connectivity to SSH or VPN connections, which introduces an additional encryption layer above the WEP encryption. Using this policy, a malicious user outside of the network that cracks the WEP encryption has to additionally crack the VPN or SSH encryption which, depending on the encryption method, can employ up to triple-strength 168- or 192-bit DES algorithm encryption (3DES) or proprietary algorithms of even greater strength. Administrators who apply these policies should certainly restrict plain text protocols such as TELNET or FTP, as passwords and data can be exposed using any of the aforementioned attacks.

8.1.4. Network Segmentation and DMZs

For administrators who wish to run externally accessible services such as HTTP, email, FTP, and DNS, it is recommended that these publicly available services be physically and/or logically segmented from the internal network. Firewalls and hardening of hosts and applications are effective ways to deter casual intruders. However, determined crackers will find ways into the internal network if the services they have cracked reside on the same logical route as the rest of the network. The externally accessible services become what the security regards as a *demilitarized zone* (DMZ), a logical network segment where inbound traffic from the Internet would only be able to access those services in the DMZ. This is effective in that, even though a malicious user exploits a machine on the DMZ, the rest of the Internal network lies behind a firewall on a separated segment.

Most enterprises have a limited pool of publicly routable IP addresses from which they can host external services, so administrators utilize elaborate firewall rules to accept, forward, reject, and deny packet transmissions. Firewall policies implemented with `iptables` or dedicated hardware firewalls allow for complex routing and forwarding rules, which administrators can use to segment inbound traffic to specific services at specified addresses and ports, as well as allow only the LAN to access internal services, which can prevent IP spoofing exploits. For more information about implementing `iptables`, refer to Chapter 7.

8.2. Hardware Security

According to a study released in 2000 by the FBI and the Computer Security Institute (CSI), over seventy percent of all attacks on sensitive data and resources reported by organizations occurred from within the organization itself. Implementing an internal security policy appears to be just as important as an external strategy. The following sections explain some of the common steps administrators and users can take to safeguard their systems from internal malpractice.

Employee workstations, for the most part, are not as likely to be targets for remote attack, especially those behind a properly configured firewall. However, there are some safeguards that can be implemented to avert an internal or physical attack on individual workstation resources.

Modern workstation and home PCs have BIOSes that control system resources on the hardware level. Workstation users can also set administrative passwords within the BIOS to prevent malicious users from accessing the system. BIOS passwords prevent malicious users from booting the system at all, deterring the user from quickly accessing or stealing information stored on the hard drive.

However, if the malicious user steals the PC (the most common case of theft frequent travelers who carry laptops and other mobile devices) and takes it to a location where they can disassemble the PC, the BIOS password does not prevent the attacker from removing the hard drive, installing it in another PC without BIOS restriction, and mount the hard drive to read any contents within. In these cases, it is recommended that workstations have locks to restrict access to internal hardware. Hardware such as lockable steel cables can be attached to PC and laptop chassis to prevent theft, as well as key locks on the chassis itself to prevent internal access. Such hardware is widely available from manufacturers such as Kensington and Targus.

Server hardware, especially production servers, are typically mounted on racks in server rooms. Server cabinets usually have lockable doors; and individual server chassis also are available with lockable front bezels for increased security from errant (or intentional) shutdown.

Enterprises can also use co-location providers to house their servers, as co-location providers offer higher bandwidth, 24x7 technical support, and expertise in system and server security. This can be an effective means of outsourcing security and connectivity needs for HTTP transactions or streaming media services. However, co-location can be cost-prohibitive, especially for small to medium-sized businesses. Co-location facilities are known for being heavily guarded by trained security staff and tightly monitored at all times.

Assessing Your Security

Chapter 9.

Vulnerability Assessment

Given the time, resources, and motivation, a cracker can break into nearly any system. At the end of the day, all the security procedures and technologies currently available cannot guarantee that your systems are safe from intrusion. Routers can help to secure your gateways to the Internet. Firewalls help secure the edge of the network. Virtual Private Networks can safely pass your data in an encrypted stream. Intrusion detection systems have the potential to warn you of malicious activity. However, the success of each of these technologies is dependent upon a number of variables, including:

- The expertise of the staff responsible for configuring, monitoring, and maintaining the technologies
- The ability to patch and update services and kernels quickly and efficiently
- The ability of those responsible to keep constant vigilance over the network.

Given the dynamic state of data systems and technologies, securing your corporate resources can be quite complex. Because of this complexity, it may be difficult to find expert resources for all of your systems. While it is possible to have personnel knowledgeable in many areas of information security at a high level, it is difficult to retain staff who are experts in more than a few subject areas. This is mainly because each subject area of Information Security requires constant attention and focus. Information security does not stand still.

9.1. Thinking Like the Enemy

Suppose you administer an enterprise network. Such networks are commonly comprised of operating systems, applications, firewalls, intrusion detection systems, and more. Now imagine trying to keep current on every one of these. Given the complexity of today's software and networking environments, exploits and bugs are a certainty. Keeping current with patches and updates for an entire network can prove to be a daunting task in a complex organization with heterogeneous systems.

Combine the expertise requirements with the task of keeping current, and it is inevitable that adverse incidents occur, systems are breached, data is corrupted, and service is interrupted.

To augment security technologies and aid in protecting systems, networks, and data, think like a cracker and gauge the security of systems by checking for weaknesses. Preventative vulnerability assessments against your own systems and network resources can reveal potential issues that can be addressed before a cracker finds it.

A vulnerability assessment is similar to an internal inquiry of your network and system security; the results of which indicate the confidentiality, integrity, and availability (as explained in Section 1.1.4). A vulnerability assessment will typically start with an information gathering phase during which important data regarding the target will be gathered. This phase will lead to the actual checking phase, whereby the target is essentially checked for all known vulnerabilities. The checking phase culminates in the reporting phase, where the findings are classified into categories of high, medium, and low risk; and methods for improving the security (decreasing the level of vulnerability) of the target are discussed.

If you were to perform a vulnerability assessment of your home, you would likely check each door to your home to see if they are shut and locked correctly. You would also check every window, making sure that they shut completely and latch correctly. This same concept applies to systems, networks, and electronic data. The process of checking for weaknesses is the same. Only the targets are different. Malicious users are the thieves and vandals of your data. Focus on their tools, mentality, and motivations, and you will begin to think like them.

9.2. Defining Assessment and Testing

A clear distinction between the types of vulnerability assessments is necessary. Vulnerability assessments may be broken down into one of two types: *Outside looking in* and *inside looking around*.

When performing an outside looking in vulnerability assessment you are attempting to compromise your systems from the outside. Being external to your company provides you with the cracker's viewpoint. You see what a cracker sees — legal IP addresses, systems on your DMZ, external interfaces of your firewall, and more.

When you perform an inside looking around vulnerability assessment you are somewhat at an advantage since you are internal and your status is elevated to trusted. This is the viewpoint you and your co-workers have once logged on to your systems. You see print servers, file servers, databases, and other resources.

There are striking distinctions between these two types of vulnerability assessments. Being internal to your company gives you elevated privileges — more so than any outsider. Still today in most organizations, security is configured in such a manner as to keep intruders out. Very little is done to secure the internals of the organization (such as departmental firewalls). Typically, there are many more resources when inside looking around as most systems are internal to a company. Once you set yourself outside the company, you immediately are given untrusted status. The systems and resources available to you externally are typically much more limited.

Consider the difference between vulnerability assessments and *penetration tests*. Think of a vulnerability assessment as the first step to a penetration test. The information gleaned from the assessment will be used in the testing. Whereas, the assessment is checking for holes and potential vulnerabilities, the penetration testing actually attempts to exploit the findings.

Assessing network infrastructure is a dynamic process. Security, both information and physical, is dynamic. Performing an assessment shows an overview, which can turn up false positives and false negatives.

Security administrators are only as good as the tools they use and the knowledge they retain. Take any of the assessment tools currently available, run them against your system, and it is almost a guarantee that there will be at least some false positives. Whether by program fault or user error, the result is the same. The tool may find vulnerabilities which in reality do not exist (false positive); or, even worse, the tool may not find vulnerabilities that actually do exist (false negative).

Now that the difference between vulnerability assessment and penetration test are defined, it is often good practice to take the findings of the assessment and review them carefully before conducting a penetration test.

Warning

> Attempting to exploit vulnerabilities can be dangerous to the productivity and efficiency of your systems and network.

The following list examines some of the benefits to performing vulnerability assessments.

- Proactive focus on information security
- Finding potential exploits before crackers find them
- Typically results in systems being kept up to date and patched
- Promotes growth and aids in developing staff expertise
- Financial loss and negative publicity abated

9.2.1. Establishing a Methodology

To aid in the selection of tools for vulnerability assessment, it is helpful to establish a vulnerability assessment methodology. Unfortunately, there is no predefined or industry approved methodology at this time; however, common sense and best practices can act as a sufficient guide.

What is the target? Are we looking at one server, or are we looking at our entire network and everything within the network? Are we external or internal to the company? The answers to these questions are important as they will help you determine not only which tools to select but also the manner in which the they will be used.

To learn more about establishing methodologies, refer to the following websites:

- http://www.ideahamster.org/osstmm-description.htm — *The Open Source Security Testing Methodology Manual* (OSSTMM)
- http://www.owasp.org — *The Open Web Application Security Project*

9.3. Evaluating the Tools

A typical assessment can start by using some form of information gathering tool. If assessing the entire network, map the network layout first to find the hosts that are running. Once located, we can then focus on examining them. Focusing on these hosts will require another set of tools. Knowing which tools to use may be the most crucial step in finding vulnerabilities.

Just as in any aspect of everyday life, there are many different tools that perform the same job. This concept applies to performing vulnerability assessments as well. There are tools specific to operating systems, applications, and even networks (based on protocols used). Some tools are free (in terms of cost) while others are not. Some tools are intuitive and easy to use, while others are cryptic and poorly documented.

Deciding which tools are the right tools for you may be a daunting task. In the end, experience counts. If possible, set up a test lab and try out as many tools as you can, noting the strengths and weaknesses of each. Review the README file or man page for the tool. In addition, look to the Internet for more information, such as articles, step-by-step guides, or even mailing lists specific to a tool.

The tools discussed below are just a small sampling of the available tools.

9.3.1. Scanning Hosts with Nmap

Nmap is a popular tool for mapping networks is included in Red Hat Linux. **Nmap** has been available for many years and is probably the most often used tool when gathering information. An excellent man page is included that covers the details, options, and examples of using **Nmap**. Use it on your network to find host systems and open ports on those systems.

Nmap is a competent first step in vulnerability assessment. You can map out all the hosts within your network, and even pass an option that will allow it to attempt to identify the operating system running on those hosts. **Nmap** is a good foundation for establishing a policy of using secure services and stopping unused services.

9.3.1.1. Using Nmap

Nmap can be run from a shell prompt or using a graphical version. At a shell prompt, type the `nmap` command followed by the hostname or IP address of the machine you want to scan.

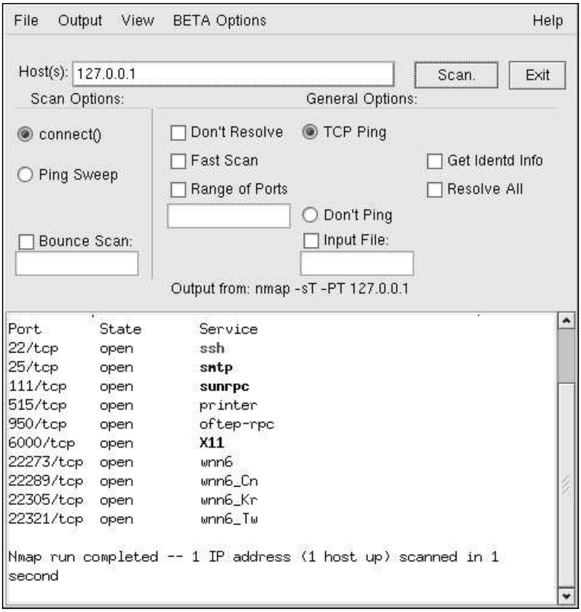
```
nmap foo.example.com
```

The results of the scan (which could take up to a few minutes, depending on where the host is located) should look similar to the following:

```
nmap 127.0.0.1
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1591 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
111/tcp    open        sunrpc
515/tcp    open        printer
950/tcp    open        oftep-rpc
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

If you were to use the graphical version (which can be run by typing `/usr/bin/nmapfe` at a shell prompt), the results will look similar to the following:



Host(s): 127.0.0.1

Scan Options:
● connect()
○ Ping Sweep
☐ Bounce Scan:

General Options:
☐ Don't Resolve
● TCP Ping
☐ Fast Scan
☐ Get Identd Info
☐ Range of Ports
☐ Resolve All
○ Don't Ping
☐ Input File:

Output from: nmap -sT -PT 127.0.0.1

```
Port        State       Service
22/tcp      open        ssh
25/tcp      open        smtp
111/tcp     open        sunrpc
515/tcp     open        printer
950/tcp     open        oftep-rpc
6000/tcp    open        X11
22273/tcp   open        wnn6
22289/tcp   open        wnn6_Cn
22305/tcp   open        wnn6_Kr
22321/tcp   open        wnn6_Tw

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

Figure 9-1. Scanning with Nmap

Nmap tests the most common network communication ports for listening or waiting services. This knowledge can be helpful to an administrator who wants to, for example, close down unnecessary services.

For more information about using **Nmap**, refer to the official homepage at http://www.insecure.org.

9.3.2. Nessus

Nessus is a full-service security scanner. The plug-in architecture of **Nessus** allows users to customize it for their systems and networks. As with any scanner, **Nessus** is only as good as the signature database it relies upon. Fortunately, **Nessus** is updated on a daily basis. It features full reporting, host scanning, and real-time vulnerability searches. Remember that there could be false positives and false negatives, even in a tool as powerful and as frequently updated as **Nessus**.

Note

> **Nessus** is not included with Red Hat Linux and is not supported. It has been included in this document as a reference to users who may be interested in using this popular application.

For more information about **Nessus**, refer to the official website at http://www.nessus.org.

9.3.3. Whisker

Whisker is an excellent CGI scanner. **Whisker** has the capability to not only check for CGI vulnerabilities but do so in an evasive manner, so as to elude intrusion detection systems. It comes with excellent documentation which should be carefully reviewed prior to running the program. When you have found your Web servers serving up CGI scripts, **Whisker** can be an excellent resource for checking the security of these servers.

Note

> **Whisker** is not included with Red Hat Linux and is not supported. It has been included in this document as a reference to users who may be interested in using this popular application.

More information about Whisker can be found at http://www.wiretrip.net.

9.3.4. VLAD the Scanner

VLAD is a scanner developed by the RAZOR team at Bindview, Inc. that may be used to check for vulnerabilities. It checks for the SANS Top Ten list of common security issues (SNMP issues, file sharing issues, etc.). While not as full-featured as **Nessus**, **VLAD** is worth investigating.

Note

> **VLAD** is not included with Red Hat Linux and is not supported. It has been included in this document as a reference to users who may be interested in using this popular application.

More information about **VLAD** can be found on the *Tools* page on the RAZOR team website at http://razor.bindview.com/index.shtml.

9.3.5. Anticipating Your Future Needs

Depending upon your target and resources, there are any number of tools available. There are tools for wireless networks, Novell networks, UNIX systems, Linux systems, and more. Another essential part of performing assessments may include reviewing physical security as well as *war dialing* — dialing numbers and extensions enterprise-wide for modem access to your network. New concepts, such as *war walking* — scanning the perimeter of your enterprise's physical structures for wireless network access — are some emerging concepts that you can investigate and, if needed, incorporate in your assessments. Imagination and exposure are the only limits of planning and conducting vulnerability assessments.

Intrusions and Incident Response

Chapter 10.

Intrusion Detection

Valuable property needs to be protected from the prospect of theft and destruction. Modern homes are equipped with alarm systems that can deter burglars, notify authorities when a break-in has occurred, and even warn owners when their home is on fire. Such measures are necessary to assure the integrity of homes and the safety of homeowners.

The same assurance of integrity and safety should also be applied to computer systems and data. The Internet has facilitated the flow of information, from the personal to the financial. At the same time, it has fostered just as many dangers. Malicious users and crackers seek vulnerable targets such as unpatched systems, systems infected with trojans, and networks running insecure services. Alarms are needed to notify administrators and security team members that an breach has taken place so that they can respond in real-time to the threat. *Intrusion detection systems* have been designed as such a warning system.

10.1. Defining Intrusion Detection Systems

An intrusion detection system (IDS) is an active process or device that analyzes system and network activity for unauthorized entry and/or malicious activity. The way that the IDS detects anomalies can vary widely; however, the aims are the same — catch perpetrators in the act before they do real damage to your resources.

IDSes protect a system from attack, misuse, and compromise. It can also monitor network activity, audit network and and system configuration for vulnerabilities, analyze data integrity, and more. Depending on the detection methods you choose to deploy, there are several direct and incidental benefits to using an IDS.

10.1.1. IDS Types

Understanding what an IDS is and the functions it provides is key in determining what type would be appropriate to include in your computer security policy. This section will discuss the concepts behind IDSes, the functionalities of each type of IDS, and the emergence of hybrid IDSes that employ several detection techniques and tools in one package.

Some IDSes are *knowledge-based*, which preemptively alert security administrators before an intrusion occurs using a database of common attacks. Alternatively, there are *behavioral* IDSes that track all resource usage for anomalies, which is usually a positive sign of malicious activity. Some IDSes are standalone services that work in the background and passively listen for activity, logging any suspicious packets from the outside. Others mix standard system tools, modified configurations, and verbose logging with administrator intuition and experience to create a powerful intrusion detection kit. Evaluating the many intrusion detection techniques can assist in finding one that is right for your organization.

10.2. Host-based IDS

A *host-based* IDS analyzes several areas to determine *misuse* (malicious or abusive activity inside the network) or intrusion (breaches from the outside). Host-based IDSes consult several types of log files (kernel, system, server, network, firewall, and more), and compare the logs against an internal database of common signatures for known attacks. Unix and Linux host-based IDSes make heavy use of `syslog` and its ability to separate logged events by their severity and functionality (for example, printer error messages versus kernel warnings). The IDS will filter logs (which, in the case of some event logs such as network and kernel, can be quite verbose), analyze them, re-tag the anomalous packets with its own system of warning and severity rating, and collect them in its own specialized log for administrator analysis.

Host-based IDSes can also verify data integrity of important files and executables. The IDS will check a database of sensitive files (and any files that you may want to add) and creates a *checksum* of each file with a message-file digest utility such as `md5sum` (128-bit algorithm) or or `sha1sum` (160-bit algorithm). The IDS then stores the sums in a plain text file, and periodically compares the file checksums against the values in the text file. If any of the files checksums do not match, then the IDS will alert the administrator by email or pager. This is the process used by **Tripwire**, which is discussed in Section 10.2.1.

10.2.1. Tripwire

Tripwire is the most popular host-based IDS for Linux. Tripwire, Inc., the developers of **Tripwire**, recently opened the software source code for the Linux version and licensed it under the terms of the GNU General Public License. Red Hat Linux includes **Tripwire**, and is available in RPM package format for easy installation and upgrade.

Detailed information on the installation and configuration of **Tripwire** can be found in the chapter titled "Installing and Configuring Tripwire" in the *Official Red Hat Linux Customization Guide*. Refer to that chapter for more information.

10.2.2. RPM as an IDS

The RPM Package Manager (RPM) is another program that can be used as a host-based IDS. RPM contains various options for querying packages and their contents. These verification options can be invaluable to an administrator who suspects that critical system files and executables have been modified.

The following list details some options for RPM that you can use to verify file integrity on your Red Hat Linux system. Refer to the *Official Red Hat Linux Customization Guide* for complete information about using RPM.

Important

> Some of the commands in the list that follows requires that you import the official Red Hat GPG public key into your RPM keyring. This key verifies that packages installed on your system contain an official Red Hat package signature, which ensures that your packages originated from Red Hat. The key can be imported with the following command (substituting <*version*> with the version of RPM installed on your system):
>
> ```
> rpm --import /usr/share/doc/rpm-<version>/RPM-GPG-KEY
> ```

`rpm -V` *`package_name`*

> This option will verify the files in the installed package called *`package_name`*. If it shows no output and exits, this means that all of the files have not been modified in anyway since the last time the RPM database was updated. If there is an error, such as
>
> ```
> S.5....T c /bin/ps
> ```
>
> then the file has been modified in some way and you need to assess whether to keep the file (such is the case with modified configuration files in `/etc`) or delete the file and reinstall the package that contains it.

`rpm -Va`

This command verifies *all* installed packages and finds any failure in its verification tests (much like the `-V` option, but more verbose in its output since it is verifying every installed package).

`rpm -Vf /bin/ls`

This command verifies individual files in an installed package. This can be useful if you wish to perform a quick verification of a suspect file.

`rpm -K` *`application-1.0.i386.rpm`*

This command is useful for checking the md5 checksum and the GPG signature of an RPM package file. This is useful for checking whether a package you want to install is signed by Red Hat or any organization for which you have the GPG public key imported into your GPG keyring. A package that has not been properly signed will emit an error message similar to the following:

```
application-1.0.i386.rpm (SHA1) DSA sha1 md5 (GPG) NOT OK
    (MISSING KEYS: GPG#897da07a)
```

Exercise caution when installing packages that are unsigned as they are unofficial and could contain malicious code

RPM can be a powerful tool, as evinced by its many verification tools for installed packages and RPM package files. It is strongly recommended that you backup the contents of your RPM database directory (`/var/lib/rpm/`) to read-only media such as CD-ROM after you install Red Hat Linux so that you can safely verify files and packages against the read-only database, rather than against the database on the system, as malicious users may corrupt the database and skew your results.

10.2.3. Other Host-based IDSes

The following list discusses some of the other popular host-based intrusion detection systems available. Refer to the websites of the respective utilities for more information about installing and configuring them in your environment.

Note

These applications are not included with Red Hat Linux and are not supported. They have been included in this document as a reference to users who may be interested in evaluating such applications.

- SWATCH http://www.oit.ucsb.edu/~eta/swatch/ — The Simple WATCHer (SWATCH) uses log files generated by `syslog` to alert administrators of anomalies based on user configuration files. SWATCH was designed to log any event that the user wants to add into the configuration file; however, it has been adopted widely as a host-based IDS.
- LIDS http://www.lids.org — The Linux Intrusion Detection System (LIDS) is a kernel patch and administration tool that can also control file modification with access control lists (ACLs) and protect processes and files, even from the root user.

10.3. Network-based IDS

Network-based intrusion detection systems operate differently from host-based IDSes. The design philosophy of a network-based IDS is to scan network packets at the router or host-level, auditing packet information and logging any suspicious packets into a special log file with extended information. Based on these suspicious packets, a network-based IDS can scan its own database of known network attack signatures and assign a severity level for each packet. If severity levels are high enough, a warning email or pager call is placed to security team members so they can further investigate the nature of the anomaly.

Network-based IDSes have become popular as the Internet grows in size and traffic. IDSes that can scan the voluminous amounts of network traffic and successfully tag suspect traffic are well-received within the security industry. Due to the inherent insecurity of the TCP/IP protocols, it has become imperative to develop scanners, sniffers, and other network auditing and detection tools to prevent security breaches due to such malicious network activity as:

- IP Spoofing
- Denial-of-service attacks
- arp cache poisoning
- DNS name corruption
- Man-in-the-middle attacks

Most network-based IDSes require that the host system network device be set to *promiscuous* mode, which allows the device to capture *every* packet on the network. Promiscuous mode can be set through the `ifconfig` command, like the following:

```
ifconfig eth0 promisc
```

Running ifconfig with no options reveals that `eth0` is now in promiscuous mode.

```
eth0      Link encap:Ethernet  HWaddr 00:00:D0:0D:00:01
          inet addr:192.168.1.50  Bcast:192.168.1.255  Mask:255.255.252.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:6222015 errors:0 dropped:0 overruns:138 frame:0
          TX packets:5370458 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:2505498554 (2389.4 Mb)  TX bytes:1521375170 (1450.8 Mb)
          Interrupt:9 Base address:0xec80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:21621 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21621 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1070918 (1.0 Mb)  TX bytes:1070918 (1.0 Mb)
```

Using a tool such as `tcpdump` (included with Red Hat Linux), we can see the large amounts of traffic flowing throughout a network:

```
# tcpdump
tcpdump: listening on eth0
02:05:53.702142 pinky.exampledomain.com.ha-cluster > \
 heavenly.exampledomain.com.860:  udp 92 (DF)
02:05:53.702294 heavenly.exampledomain.com.860 > \
 pinky.exampledomain.com.ha-cluster:  udp 32 (DF)
02:05:53.702360 pinky.exampledomain.com.55828 > dns1.exampledomain.com.domain: \
 PTR? 254.35.168.192.in-addr.arpa. (45) (DF)
```

```
02:05:53.702706 ns1.rdu.redhat.com.domain > pinky.exampledomain.com.55828: \
 6077 NXDomain* 0/1/0 (103) (DF)
02:05:53.886395 shadowman.exampledomain.com.netbios-ns > \
 172.16.59.255.netbios-ns: NBT UDP PACKET(137): QUERY; BROADCAST
02:05:54.103355 802.1d config c000.00:05:74:8c:a1:2b.8043 root \
 0001.00:d0:01:23:a5:2b pathcost 3004 age 1 max 20 hello 2 fdelay 15
02:05:54.636436 konsole.exampledomain.com.netbios-ns > 172.16.59.255.netbios-ns:\
 NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
02:05:56.323715 pinky.exampledomain.com.1013 > heavenly.exampledomain.com.860:\
 udp 56 (DF)
02:05:56.323882 heavenly.exampledomain.com.860 > pinky.exampledomain.com.1013:\
 udp 28 (DF)
```

Notice that packets that were not intended for our machine (`pinky.exampledomain.com`) are still being scanned and logged by `tcpdump`.

10.3.1. `snort`

While `tcpdump` is a useful auditing tool, it is not considered a true IDS because it does not analyze packets for anomalies; it only *dumps* them to the output screen or to a log file. A proper IDS will analyze the packets and then tag and log suspicious activity.

Snort is an IDS designed to be comprehensive and accurate in successfully logging malicious network activity and notifying administrators when potential breaches occur. Snort uses the standard `libcap` library, and `tcpdump` as a packet logging backend.

The most prized feature of Snort is not in its functionality, but in its flexible attack signature subsystem. Snort has a constantly updated database of attacks that can be added to and updated via the Internet. Users can create signatures based on new network attacks and submit them to the Snort signature mailing lists (located at http://www.snort.org/lists.html), so that all Snort users will benefit. This community ethic of sharing has grown Snort into one of the most up-to-date and robust network-based IDSes available.

Note

Snort is not included with Red Hat Linux and is not supported. It has been included in this document as a reference to users who may be interested in evaluating it.

For more information about using Snort, refer to the official website at http://www.snort.org.

Chapter 11.

Incident Response

In the event that the security of a system has been compromised, an *incident response* is necessary. It is the responsibility of the security team to respond to the problem quickly and effectively.

11.1. Defining Incident Response

Incident response is simply an expedited response to an issue or occurrence. Pertaining to Information Security, an example would be a hacker who has penetrated a firewall and is currently sniffing internal network traffic. The incident is the breach of security. The response depends upon how the security team reacts, what they do to minimize damages, and when they restore resources, all the while attempting to guarantee data integrity.

Think of your organization and how almost every aspect of it relies upon technology and the computer systems. If there is a compromise, think of the potentially devastating results. Besides the obvious system downtime and theft of data, there could be data corruption, identity theft (from online personnel records), and embarrassing publicity or even financially devastating publicity as customers and business partners learn and react to news of such a compromise.

Research on past security breaches (both internal and external) shows that companies can potentially be run out of business as a result of a breach. At minimum, a breach can result in resources being unavailable and data stolen or corrupted. But one cannot overlook issues that are difficult to calculate financially, such as bad publicity. An organization must calculate the cost of a breach and how will it detrimentally affects an organization, both in the short and long term.

11.2. Creating an Incident Response Plan

It is very important that an *incident response plan* is formulated, supported throughout the organization, put into action, and regularly tested. A good incident response plan that is thoroughly tested and acted upon quickly may minimize the effects of a breach. Furthermore, it may even reduce the negative publicity and focus attention on quick reaction time.

From a security team perspective, it does not matter whether a breach occurs (as such occurrences are an eventual part of doing business using an untrusted carrier network such as the Internet), but rather, *when* a breach will occur. Do not think of a system as weak and vulnerable; realize that given enough time and resources someone, somewhere, some day, will breach even the most security-hardened system or network.

The positive aspect of realizing the inevitability of a system breach is that it allows the security team to develop a course of action that minimizes any potential damage. Combining a course of action with expertise allows the team to respond to adverse conditions in a formal and responsive manner.

The incident response plan can be separated into four sections:

- Immediate Response
- Investigation
- Restoring
- Reporting

Incident response must be decisive and executed quickly. There is little room for error in most cases, and by staging practice emergencies and measuring response times it is possible to develop a methodology that fosters speed and accuracy. Reacting quickly may minimize the impact of resource unavailability and the potential damage caused by system compromise.

An incident response plan has a number of requirements, including;

- Appropriate personnel (in-house experts)
- Financial support
- Executive support
- A feasible plan of action
- Physical resources such as hard drives, systems, and backup systems

11.2.1. The Computer Emergency Response Team (CERT)

The term *appropriate personnel* refers to people who will comprise a *Computer Emergency Response Team* (CERT). Finding the core competencies for a CERT can be a challenge. The concept of appropriate personnel goes beyond technical expertise and includes logistics such as location, availability, and desire to put the organization ahead of ones personal life when an emergency occurs. An emergency is never a planned event; it can happen at any moment, and all CERT members must be willing to accept the responsibility that is required of them to respond to an emergency at any hour.

It may not always be feasible, but there should be personnel redundancy within a CERT. If depth in core areas is not applicable to an organization, then cross-training should be implemented wherever possible. Note that if only one person owns the key to data safety and integrity then the entire enterprise becomes helpless in that person's absence.

Typical CERT members include system and network administrators as well as members from the information security department. System administrators will provide the knowledge and expertise of the systems, including data backups, backup hardware available for use, and more. Network administrators provide their knowledge of network protocols, in addition to being able to re-route traffic dynamically. Information Security personnel are useful in tracking and tracing security issues as well as performing post-mortem analysis of media.

11.2.2. Legal Issues

Another important aspect of incident response are legal issues. Security plans should be developed with members of legal staff or some form of legal counsel. Just as every company should have their own corporate security policy, every company has its own way of handling incidents from a legal perspective. Local, state, and federal regulatory issues are beyond the scope of this document, but are mentioned because the methodology for performing a post-mortem analysis, at least in part, will be dictated by (or in conjunction with) legal counsel.

11.3. Implementing the incident response Plan

Once a plan of action is created, it must be agreed upon and actively implemented. Any aspect of the plan that is questioned during active implementation will most likely result in poor response time and downtime in the event of breach. This is where practice exercises become invaluable. Unless something is brought to attention before the plan is actively set in production, implementation should be expedited.

If a breach is detected while the CERT is present for quick action, potential response can vary. The team can decide to pull the network connections, disconnect the affected system or systems, patch the exploit, and then reconnect quickly without further potential complication. The team can also watch the perpetrator and track his actions. The team could even redirect the perpetrator to a *honeypot* —

a system or segment of a network containing intentionally false data — in order to track incursion safely and without disruption to production resources.

Responding to an incident should also be accompanied by information gathering wherever possible. Running processes, network connections, files, directories, and more should be actively audited in real-time. Having a snapshot of production resources for comparison can be helpful in tracking rogue services or processes. CERT members and in-house experts will be great resources in tracking anomalies. These team members should know their systems and should be able to spot an anomaly quicker than someone unfamiliar with the infrastructure.

11.4. Investigating the Incident

Investigating a computer breach is like investigating a crime scene. Investigators collect evidence, note any strange clues, and take inventory on loss and damage. Analysis of computer compromise can either be live (as the attack is happening) or *post-mortem* (after the attack).

Although it is unwise to trust any system log files on an exploited system, there are other forensic utilities to aid us in our analysis. The purpose and features of these tools vary, but they commonly create bit-image copies of media, correlate events and processes, show low level filesystem information, and recover deleted files whenever possible.

11.4.1. Collecting an Evidential Image

Creating a bit-image copy of media is a feasible first step. If performing data forensic work, it is a requirement. It is recommended to make two copies, one for analysis and investigation, and a second to be stored along with the original for evidence in any legal proceedings.

You can use the `dd` command that is part of the `fileutils` package in Red Hat Linux. Suppose there is a single hard drive from a system you want to image. Attach that drive as a slave to your system, and then use `dd` to create the image file, such as the following:

```
dd if=/dev/hdd bs=1k conv=noerror of=/home/evidence/image1
```

This command creates a single file named `image1` using a 1k block size for speed. The `conv=noerror` option forces `dd` to continue reading and dumping data even if bad sectors are encountered on the suspect drive. It is now possible to study the resulting image file, or even attempt to recover deleted files.

11.4.2. Gathering Post-Breach Information

The topic of digital forensics and analysis itself is quite broad, yet the tools are mostly architecture specific and cannot be applied generically. However, incident response, analysis, and recovery are important topics. With proper knowledge and experience, Red Hat Linux can be an excellent platform for performing these types of analysis, as it includes several utilities for performing post-breach response and restoration.

Table 11-1 details some commands for file auditing and management. It also lists some examples that you can use to properly identify files and file attributes, such as permissions and access dates, so that you can collect further evidence or items for analysis. These tools, when combined with intrusion detection systems, firewalls, hardened services, and other security measures, can help in reducing the potential damage when an attack occurs.

Note

For detailed information about each tool, refer to their respective manual pages.

Command	**Function**	**Example**
`dd`	Creates a bit-image copy (or *disk dump*) of files and partitions. Combined with a check of the md5sums of each image, administrators can compare a pre-breach image of a partition or file with a breached system to see if the sums match.	`dd if=/bin/ls of=ls.dd |md5sum ls.dd >ls-sum.txt`
`grep`	Find useful string (text) information on and inside files and directories such as permissions, script changes, file attributes, and more. Used mostly as a piped command of another command such as `ls`, `ps`, or `ifconfig`	`ps auxw |grep /bin`
`strings`	Prints the strings of printable characters in a file. It is most useful for auditing executables for anomalies such as `mail` commands to unknown addresses or logging to a non-standard log files.	`strings /bin/ps |grep 'mail'`
`file`	Determines the characteristics of files based on format, encoding, libraries that it links (if any), and file type (binary, text, and more). Useful for determining whether an executable such as `/bin/ls` has been modified using static libraries, a sure sign that that a modification has occurred.	`file /bin/ls`
`find`	Search directories for particular files. `find` is a useful tool for searching the directory structure by keyword, date and time of access, permissions, and more. This can be useful for administrators that perform general system audits of particular directories or files.	`find -atime +12 -name *log* -perm u+rw`
`stat`	Displays various information about a file, including time last accessed, permissions, UID and GID bit settings, and more. Useful for checking when a breached system executable was last used and/or when it was modified.	`stat /bin/netstat`

Command	Function	Example
`md5sum`	Calculates the 128-bit checksum using the md5 hash algorithm. You can use the command to create a text file that lists all crucial executables that could be modified or replaced in a security compromise. Redirect the sums to a file to create a simple database of checksums, and then copy the file onto a read-only medium such as CD-ROM.	`md5sum /usr/bin/gdm >>md5sum.txt`

Table 11-1. File Auditing Tools

11.5. Restoring and Recovering Resources

During active incident response, there should be considerations toward working on recovery. The actual breach will dictate the course of recovery. This is when having backups on offline systems will prove invaluable. For recovery, the response team should be planning to bring back online any downed systems or applications, such as authentication servers, database servers, and any other production resources.

Having production backup hardware ready for use is highly recommended, such as extra hard drives, hot-spare servers, and the like. Ready-made systems should have all production software loaded and ready for immediate use. Perhaps only the most recent and pertinent data would need to be imported. This ready-made system should be kept disconnected from the rest of the potentially affected network. If a compromise occurs and the backup system is a part of the network, then it defeats the purpose of having a backup system.

System recovery can be a tedious process. In many instances there are one of two options. Administrators can perform a clean reinstallation of the operating system followed by restoration of all applications and data. Alternatively, administrators can patch the system of the offending vulnerability and bring the affected system(s) back into production.

11.5.1. Reinstalling the System

Performing a clean reinstallation insures that the affected system will be clean from any trojans, backdoors, or malicious processes and that any data (if restored from a trusted back up) is cleared of any malicious modification. The drawback to total system recovery is the time involved in rebuilding systems from scratch. However, if there is a *hot* backup system available for use where the only action to take is to dump the most recent data, then system downtime is greatly reduced.

11.5.2. Patching the System

The alternate course to recovery is to patch the affected system(s). This method of recovery is more dangerous to perform and should be undertaken with great caution. The danger with patching a system instead of reinstalling is whether or not you have sufficiently *cleansed* the system of trojans, holes, and corrupted data. If using a modular kernel, then patching a breached system can be even more difficult. Most *rootkits* (programs or packages that a cracker leaves to gain root access to your system), trojan system commands, and shell environments are designed to hide malicious activities from auditors. If this approach is taken, only trusted binaries should be used (for example, from a mounted CD-ROM).

11.6. Reporting the Incident

The last part of the incident response plan is reporting the incident. The security team should take notes as the response is happening to properly report the issue to organizations such as local and federal authorities or multi-vendor software vulnerability portals such as the Common Vulnerabilities and Exposures site (CVE) at http://cve.mitre.org. Depending on the type of legal counsel your enterprise employs, a post-mortem analysis may be required. Even if it is not a functional requirement to a post-compromise analysis, a post-mortem can prove invaluable in helping to learn how a cracker thinks and how your systems are structured so that future compromises can be prevented.

Appendixes

Appendix A.

Common Exploits and Attacks

Table A-1 details some of the most common exploits and entry points used by intruders to access organizational network resources. Key to these common exploits are the explanations of how they are performed and how administrators can properly safeguard their network against such attacks.

Exploit	**Description**	**Notes**
Null or Default Passwords	Leaving administrative passwords blank or using a default password provided by the application package. This is most common in hardware such as routers and BIOSes, though some services that run on Linux can contain default administrator passwords (though Red Hat Linux does not ship with them)	Commonly associated with networking hardware such as routers, firewalls, VPNs and network attached storage (NAS) appliances; Common in many legacy operating systems, especially OSes that bundle services such as UNIX and Windows; Administrators sometimes create privileged users in a rush and leave the password null, a perfect entrypoint for malicious users who discover the user
Default Shared Keys	Secure services sometimes package default security keys for development or evaluation testing purposes. If these keys are left unchanged and placed in a production environment on the Internet, *any* user with the same default keys have access to that shared-key resource, and any sensitive information contained in it	Most common in wireless APs and preconfigured secure server appliances CIPE (refer to Chapter 6) contains an sample static key that must be changed before moving to a production environment
IP Spoofing	A remote machine acts as a node on your local network, finds vulnerabilities with your servers, and installs a backdoor program or trojan to gain control over your network resources.	Spoofing is quite difficult as it involves the attacker predicting TCP/IP SYN-ACK numbers to coordinate a connection to target systems, but several tools are available to assist crackers in performing such a vulnerability Depends on target system running services (such as `rsh`, `telnet`, FTP and others) that use *source-based* authentication techniques, which are not usually recommended compared to PKI or other forms of encryption authentication as used in `ssh` or SSL/TLS.

Exploit	Description	Notes
Eavesdropping	Collecting data that passes between two active nodes on a network by eavesdropping the connection between the two nodes.	This type of attack works mostly with plain text transmission protocols such as telnet, FTP, and HTTP transfers. Remote attacker must have access to a compromised system on a LAN in order to perform such an attack; usually the cracker has used an active attack (such as IP spoofing or Man-in-the-middle) to compromise a system on the LAN Preventative measures include services with cryptographic key exchange, one-time passwords, or encrypted authentication to prevent password snooping; strong encryption during transmission also advised
Service Vulnerabilities	An attacker finds a flaw or loophole in a service run over the Internet; through this vulnerability, the attacker compromises the entire system and and any data that it may hold and could possibly compromise other systems on the network.	HTTP-based services such as CGI are vulnerable to remote command executions and even shell access. Even if the HTTP service runs as a non-privileged user such as "nobody", information such as configuration files and network maps can be read, or the attacker can start a denial of service attack which drains system resources or renders it unavailable to other users. Services sometimes can have vulnerabilities that go unnoticed during development and testing; these vulnerabilities (such as *buffer overflow*, where attackers gain access by filling addressable memory with a quantity over what is acceptable by the service, crashing the service and giving the attacker an interactive command prompt from which they may execute arbitrary commands. Administrators should make sure that services do not run as the root user; stay vigilant of patches and errata updates for their applications from vendors or security organizations such as CERT and CVE.

Exploit	Description	Notes
Application Vulnerabilities	Attackers find faults in desktop and workstation applications such as e-mail clients and execute arbitrary code, implant trojans for future compromise, or crash systems. Further exploitation can occur if the compromised workstation has administrative privileges on the rest of the network.	Workstations and desktops are more prone to exploitation because workers do not have the expertise or experience to prevent or detect a compromise as servers run by an administrator; it is imperative to inform individuals of the risks they are taking when they install unauthorized software or open unsolicited mail Safeguards can be implemented such that email client software does not automatically open or execute attachments. Additionally, the automatic updating of workstation software via Red Hat Network or other system management service can alleviate the burdens of multi-seat security deployments.
Denial of Service (DoS) Attacks	Attacker or group of attackers coordinate an attack on network or server resources by sending unauthorized packets to the target machine (either server, router, or workstation). This forces the resource to become unavailable to legitimate users.	The most reported DoS case occurred in 2000 when several highly-trafficked sites were rendered unavailable by a coordinated ping flood attack using several compromised systems with high bandwidth connections acting as redirected broadcasters Source packets are usually forged (as well as rebroadcasted), making investigation to the true source of the attack difficult. Advances in ingress filtering (IETF rfc2267), and Network IDS technology assist administrators in tracking down and preventing distributed DoS attacks.

Table A-1. Common Exploits

Index

Symbols

- 802.11x, 76
  - and security, 76

A

- Apache HTTP Server
  - cgi security, 52
  - directives, 51
  - introducing, 51
- attackers and risks, 17

B

- basic input output system
  - (See BIOS)
- BIOS
  - security, 27
    - passwords, 27
- black hat hacker
  - (See crackers)
- boot loaders
  - GRUB
    - password protecting, 28
  - LILO
    - password protecting, 29
  - security, 28

C

- CIPE, 59
  - customizing, 64
  - installation, 60
- co-location services, 78
- collecting evidence, 97
  - file auditing tools, 97
    - dd, 98
    - file, 98
    - find, 98
    - grep, 98
    - md5sum, 98
    - stat, 98
    - strings, 98
- common exploits and attacks, 103
  - table, 103
- communication tools
  - secure, 43
    - GPG, 43
    - OpenSSH, 43
- computer emergency response team, 96
- controls, 15
  - administrative, 16
  - physical, 15
  - technical, 15
- conventions
  - document, vi
- cracker
  - black hat hacker, 17
- crackers
  - definition, 17

D

- dd, 97, 98
- Denial of Service (DoS)
  - distributed, 14
- DMZ
  - (See networks)

F

- file, 98
- file auditing
  - tools, 98
- find, 98
- firewall types, 67
  - network address translation (NAT), 67
  - packet filter, 67
  - proxy, 67
- firewalls, 67
  - additional resources, 73
  - personal, 43
  - types, 67
- FTP
  - anonymous access, 53
  - anonymous upload, 54
  - greeting banner, 53
  - introducing, 52
  - TCP wrappers and, 55
  - user accounts, 54
  - vsftpd, 52
  - warning banner, 53
  - wu-ftpd, 52

G

- grep, 98
- grey hat hacker
  - (See hackers)

H

hacker ethic, 17
hackers
  black hat
    (See cracker)
  definition, 17
  grey hat, 17
  white hat, 17
hardware, 75
  and security, 78
  laptops, 78
  servers, 78
  workstations, 78

I

IDS
  (See intrusion detection systems)
incident response, 95
  and legal issues, 96
  collecting evidence, 97
  computer emergency response team (CERT), 96
  creating a plan, 95
  defining, 95
  gathering post-breach information, 97
  implementation, 96
  investigation, 97
  post-mortem, 97
  reporting the incident, 100
  restoring and recovering resources, 99
incident response plan, 95
insecure services, 41
  rsh, 42
  telnet, 42
  vsftpd, 42
  wu-ftpd, 42
introduction, v
  other Red Hat Linux manuals, v
  topics, v
intrusion detection systems, 89
  and log files, 89
  defining, 89
  host-based, 89
  network-based, 92
    Snort, 93
  RPM Package Manager (RPM), 90
  Tripwire, 90
  types, 89
ip6tables, 72
iptables, 68
  additional resources, 73
  using, 68

K

Kerberos
  NIS, 50

L

legal issues, 96
lpd, 40
lsof, 56

M

md5sum, 98

N

Nessus, 85
Netfilter, 68
  additional resources, 73
Netfilter 6, 72
netstat, 56
network services, 40
  identifying and configuring, 40
  risks, 40
    buffer overflow, 40
    denial-of-service, 40
    script vulnerability, 40
network topologies, 75
  linear bus, 75
  ring, 75
  star, 75
networks, 75
  and security, 75
  de-militarized zones (DMZs), 77
  hubs, 76
  segmentation, 77
  switches, 76
  wireless, 76
NFS, 50
  and Sendmail, 55
  network design, 50
  syntax errors, 51
NIS
  introducing, 48
  iptables, 50
  Kerberos, 50
  NIS domain name, 49
  planning network, 49
  securenets, 49
  static ports, 50
nmap, 56, 83
  command line version, 83
  graphical version, 84

O

OpenSSH, 43
  scp, 43
  sftp, 43
  ssh, 43
overview, 11

P

password aging, 34
password security, 30
  aging, 34
  and PAM, 33
  auditing tools, 33
    Crack, 33
    John the Ripper, 33
    Slurpie, 33
  enforcement, 33
  in an organization, 33
  methodology, 32
  strong passwords, 31
passwords
  within an organization, 33
pluggable authentication modules (PAM)
  strong password enforcement, 33
portmap, 40
  and iptables, 48
  and TCP wrappers, 48
ports
  monitoring, 56
post-mortem, 97

R

RAZOR, 85
reporting the incident, 100
restoring and recovering resources, 99
  patching the system, 99
  reinstalling the system, 99
risks
  encryption, 18
  insecure services, 20
  networks, 18
    architectures, 18
  open ports, 19
  patches and errata, 20
  servers, 19
    inattentive administration, 20
  wireless LAN (WLAN), 19
  workstations and PCs, 21, 21
    applications, 21
root, 35
  allowing access, 35
  disallowing access, 35
  limiting access, 38
    and su, 38
    and sudo, 39
    with User Manager, 38
  methods of disabling, 35
    changing the root shell, 37
    disabling SSH logins, 37
    with PAM, 37
root user
  (See root)
RPM
  and intrusion detection, 90
  check GPG signature, 26
  importing GPG key, 26

S

security considerations
  hardware, 75
  network transmission, 76
  physical networks, 75
  wireless, 76
security overview, 11
  conclusion, 16
  controls
    (See controls)
  defining computer security, 11
  Denial of Service (DoS), 14
  evolution of computer security, 11
  viruses, 14
sendmail, 40
  and NFS, 55
  introducing, 55
  limiting DoS, 55
server security
  Apache HTTP Server, 51
    cgi security, 52
    directives, 51
  FTP, 52
    anonymous access, 53
    anonymous upload, 54
    greeting banner, 53
    TCP wrappers and, 55
    user accounts, 54
    vsftpd, 52
    warning banner, 53
    wu-ftpd, 52
  NFS, 50
    network design, 50
    syntax errors, 51
  NIS, 48
    iptables, 50
    Kerberos, 50
    NIS domain name, 49
    planning network, 49

    - securenets, 49
    - static ports, 50
  - overview of, 45
  - portmap, 48
  - ports
    - monitoring, 56
  - Sendmail, 55
    - and NFS, 55
    - limiting DoS, 55
  - TCP wrappers, 45
    - attack wrarnings, 46
    - banners, 45
    - logging, 46
  - xinetd, 46
    - managing resources with, 47
    - preventing DoS with, 47
    - SENSOR trap, 47
- services, 56
- Services Configuration Tool, 41
- Snort, 93
- sshd, 40
- stat, 98
- strings, 98
- su
  - and root, 38
- sudo
  - and root, 39

T

- TCP wrappers
  - and FTP, 55
  - and portmap, 48
  - attack wrarnings, 46
  - banners, 45
  - logging, 46
- Tripwire, 90

U

- updates
  - official security errata, 25
  - via Red Hat Errata website, 25
  - via Red Hat Network, 25

V

- Virtual Private Networks, 59
  - (See CIPE)
- viruses
  - trojans, 14
- VLAD the Scanner, 85
- VPN, 59
- vulnerabilities
  - assessing with Nessus, 85
  - assessing with Nmap, 83
  - assessing with VLAD the Scanner, 85
  - assessing with Whisker, 85
  - assessment, 81
    - defining, 82
    - establishing a methodology, 83
    - testing, 82

W

- Whisker, 85
- white hat hacker
  - (See hackers)
- Wi-Fi networks
  - (See 802.11x)
- wireless security, 76
  - 802.11x, 76
- workstation security, 27
  - BIOS, 27
  - boot loaders
    - passwords, 28
  - evaluating, 27
    - administrative control, 27
    - BIOS, 27
    - boot loaders, 27
    - communications, 27
    - passwords, 27
    - personal firewalls, 27

X

- xinetd, 40
  - managing resources with, 47
  - preventing DoS with, 47
  - SENSOR trap, 47

Colophon

The Official Red Hat Linux manuals are written in DocBook SGML v4.1 format. The HTML and PDF formats are produced using custom DSSSL stylesheets and custom jade wrapper scripts.

Marianne Pecci <`goddess@ipass.net`> created the admonition graphics (note, tip, important, caution, and warning). They may be redistributed with written permission from Marianne Pecci and Red Hat, Inc..

The Red Hat Linux Product Documentation Team consists of the following people:

Sandra A. Moore — Primary Writer/Maintainer of the *Official Red Hat Linux x86 Installation Guide*; Contributing Writer to the *Official Red Hat Linux Getting Started Guide*

Tammy Fox — Primary Writer/Maintainer of the *Official Red Hat Linux Customization Guide*; Contributing Writer to the *Official Red Hat Linux Getting Started Guide*; Writer/Maintainer of custom DocBook stylesheets and scripts

Edward C. Bailey — Contributing Writer to the *Official Red Hat Linux x86 Installation Guide*

Johnray Fuller — Primary Writer/Maintainer of the *Official Red Hat Linux Reference Guide*; Co-writer/Co-maintainer of the *Official Red Hat Linux Security Guide*

John Ha — Primary Writer/Maintainer to the *Official Red Hat Linux Getting Started Guide*; Co-writer/Co-maintainer of the *Official Red Hat Linux Security Guide*