Red Hat Enterprise Linux 4 Security Guide

For Red Hat Enterprise Linux 4
Edition 2

Red Hat Inc.

Red Hat Enterprise Linux 4 Security Guide

For Red Hat Enterprise Linux 4
Edition 2

Red Hat Inc.

Abstract

This book assists users and administrators in learning the processes and practices of securing workstations and servers against local and remote intrusion, exploitation and malicious activity. Focused on Red Hat Enterprise Linux but detailing concepts and techniques valid for all Linux systems, this guide details the planning and the tools involved in creating a secured computing environment for the data center, workplace, and home. With proper administrative knowledge, vigilance, and tools, systems running Linux can be both fully functional and secured from most common intrusion and exploit methods.

Table of Contents

Introduction

Welcome to the *Security Guide*!

The *Security Guide* is designed to assist users of Red Hat Enterprise Linux in learning the processes and practices of securing workstations and servers against local and remote intrusion, exploitation, and malicious activity. The *Security Guide* details the planning and the tools involved in creating a secured computing environment for the data center, workplace, and home. With proper administrative knowledge, vigilance, and tools, systems running Red Hat Enterprise 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

The manual is divided into the following parts:

- General Introduction to Security
- Configuring Red Hat Enterprise Linux for Security
- Assessing Your Security
- Intrusions and Incident Response
- Appendix

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

This manual assumes that you have an advanced knowledge of Red Hat Enterprise Linux. If you are a new user or only have basic to intermediate knowledge of Red Hat Enterprise Linux and need more information on using the system, refer to the following guides which discuss the fundamental aspects of Red Hat Enterprise Linux in greater detail than the *Security Guide*:

- The *Installation Guide* provides information regarding installation.
- The *Red Hat Enterprise Linux Introduction to System Adminitration* contains introductory information for new Red Hat Enterprise Linux system administrators.
- The *System Administrators Guide* offers detailed information about configuring Red Hat Enterprise Linux to suit your particular needs as a user. This guide includes some services that are discussed (from a security standpoint) in the *Security Guide*.
- *Reference Guide* provides detailed information suited for more experienced users to refer to when needed, as opposed to step-by-step instructions.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **`Buttons`** tab, select the **`Left-handed mouse`** check box and click **`Close`** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **`gedit`** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **`Search`** field and click **`Next`**. The character you sought will be highlighted in the **`Character Table`**. Double-click this highlighted character to place it in the **`Text to copy`** field and then click the **`Copy`** button. Now switch back to your document and choose **Edit** → **Paste** from the **`gedit`** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                  struct kvm_assigned_pci_dev *assigned_dev)
{
          int r = 0;
          struct kvm_assigned_dev_kernel *match;

          mutex_lock(&kvm->lock);

          match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                        assigned_dev->assigned_dev_id);
          if (!match) {
                  printk(KERN_INFO "%s: device hasn't been assigned before, "
                    "so cannot be deassigned\n", __func__);
                  r = -EINVAL;
                  goto out;
          }

          kvm_deassign_device(kvm, match);

          kvm_free_assigned_device(kvm, match);

out:
          mutex_unlock(&kvm->lock);
          return r;
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. More to Come

The *Security Guide* is part of Red Hat's growing commitment to provide useful and timely support and information to Red Hat Enterprise 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 *Security Guide*, or if you have thought of a way to make this manual better, we would love to hear from you! Submit a report in Bugzilla (**http://bugzilla.redhat.com/bugzilla/**) against the component **rhel-sg**.

Be sure to mention the manual's identifier:

```
rhel-sg
```

By mentioning the identifier, we 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, include the section number and some of the surrounding text so we can find it easily.

Part I. A General Introduction to Security

This part defines information security, its history, and the industry that has developed to address it. It also discusses some of the risks that computer users or administrators face.

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 vocabulary, such as total cost of ownership (TCO) and quality of service (QoS). In these 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 who 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 Soviet Union 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 a 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 to be the largest 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. Mitnick was 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 Vladimir Levin cases (refer to Section 1.1.2, “Computer Security Timeline” for more information) 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 timeline lists some of the more 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) begin exploring and programming the school's PDP-1 mainframe computer system. The group eventually coined the term "hacker" in the context it is known today.
- 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 for the public use of data networks which were 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 of 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 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 from 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 cracking 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 cracker groups that begin exploiting vulnerabilities in computers and electronic data networks.
- The Computer Fraud and Abuse Act of 1986 is 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 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 among 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 crackers 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 arrested and convicted of wire fraud charges and serves 5 years in prison.
- Kevin Poulsen and an unknown accomplice rig 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 cracking and phreaking become legend, and several prospective crackers convene at the annual DefCon convention to celebrate cracking 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 945 Million people use or have used the Internet worldwide (Computer Industry Almanac, 2004). At the same time:

- On any given day, there are approximately 225 major incidences of security breach reported to the CERT Coordination Center at Carnegie Mellon University. [1]
- In 2003, the number of CERT reported incidences jumped to 137,529 from 82,094 in 2002 and from 52,658 in 2001. [2]
- The worldwide economic impact of the three most dangerous Internet Viruses of the last three years was estimated at US$13.2 Billion. [3]

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. Falling 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 of sensitive information and establishing security policy. The following describes the CIA model in further 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

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
- Biometrics (includes fingerprint, voice, face, iris, handwriting, and other automated methods used to recognize individuals)

1.2.2. Technical Controls

Technical controls use 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 about the origins, reasons, and aspects of security, you can determine the

appropriate course of action with regard to Red Hat Enterprise 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.

[1] Source: http://www.cert.org

[2] Source: http://www.cert.org/stats/

[3] Source: http://www.newsfactor.com/perl/story/16407.html

Chapter 2. Attackers and Vulnerabilities

To plan and implement a good security strategy, first be aware of some of the issues which determined, motivated attackers exploit to compromise systems. But before detailing these issues, the terminology used when identifying an attacker must be defined.

2.1. A Quick History of Hackers

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 to be hackers, and use the word as a term of respect.

Typically, hackers follow a form of the *hacker ethic* which dictates that the 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

Within the community of individuals who find and exploit vulnerabilities in systems and networks are several distinct groups. These groups are often 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 may likely attempt to exploit. The remainder of the chapter focuses on these issues.

2.2. Threats to Network Security

Bad practices when configuring the following aspects of a network can increase the risk of attack.

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 exploits the opportunity.

2.2.1.1. Broadcast Networks

System administrators often fail to realize the importance of networking hardware in their security schemes. 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 hosts.

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 as 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. 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 open door which allows access to the entire network.

2.3. Threats to Server Security

Server security is as important as network security because servers often hold a great deal of an 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. The following sections detail some of the main issues.

2.3.1. Unused Services and Open Ports

A full installation of Red Hat Enterprise Linux contains 1000+ application and library packages. However, most server administrators do not opt to install every single package in the distribution, preferring instead to install a base installation of packages, including several server applications.

A common occurrence among system administrators is to install the operating system without paying attention to what programs are actually being installed. This can be problematic because unneeded services may be installed, configured with the default settings, and possibly turned on. This can cause unwanted services, such as Telnet, DHCP, or DNS, to run on a server or workstation without the administrator realizing it, which in turn can cause unwanted traffic to the server, or even, a potential pathway into the system for crackers. Refer To Chapter 5, *Server Security* for information on closing ports and disabling unused services.

2.3.2. Unpatched Services

Most server applications that are included in a default installation are solid, thoroughly tested pieces of software. Having been in use in production environments for many years, 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, because of 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 (http://www.securityfocus.com) or the Computer Emergency Response Team (CERT) website (http://www.cert.org). Although these mechanisms are an effective way of alerting the community to security vulnerabilities, it is up to system administrators to patch their systems promptly. This is particularly true because crackers have access to these same vulnerability tracking services and will use the information to crack unpatched systems whenever they can. Good system administration requires vigilance, constant bug tracking, and proper system maintenance to ensure a more secure computing environment.

Refer to Chapter 3, *Security Updates* for more information about keeping a system up-to-date.

2.3.3. Inattentive Administration

Administrators who fail to patch their systems are one of the greatest threats to server security. According to the *System Administration Network and Security Institute* (*SANS*), the primary cause of computer 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." [4] 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 the system kernel or network traffic. Another common error is when default passwords or keys to services are left unchanged. For example, some databases have default administration passwords because the database developers assume that the system administrator changes these passwords immediately after installation. If a database administrator fails to change this password, even an inexperienced cracker can use a widely-known default password to gain administrative privileges to the database. These are only a few examples of how inattentive administration can lead to compromised servers.

2.3.4. Inherently Insecure Services

Even the most vigilant organization can fall victim to vulnerabilities if the network services they choose are inherently insecure. For instance, there are many services developed under the assumption that they are used over trusted networks; however, this assumption fails as soon as the service becomes available over the Internet — which is itself inherently untrusted.

One category of insecure network services are those that require unencrypted usernames and passwords for authentication. Telnet and FTP are two such services. If packet sniffing software is monitoring traffic between the remote user and such a service usernames and passwords can be easily intercepted.

Inherently, such services 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 the server, the attacker's machine acts as an invisible conduit, sitting quietly between the remote service and the unsuspecting user capturing information. In this way a cracker can gather administrative passwords and raw data without the server or the user realizing it.

Another category of insecure services include 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 to prevent a cracker from 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

who gains access to this database can then access every user account on a network, including the administrator's account.

By default, Red Hat Enterprise Linux is released with all such services turned off. However, since administrators often find themselves forced to use these services, careful configuration is critical. Refer to Chapter 5, *Server Security* for more information about setting up services in a safe manner.

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 since they often contain sensitive data, such as credit card information, they are targeted by system crackers. Workstations can also be co-opted without the user's knowledge and used by attackers as "slave" machines in coordinated attacks. For these reasons, knowing the vulnerabilities of a workstation can save users the headache of reinstalling the operating system, or worse, recovering from data theft.

2.4.1. Bad Passwords

Bad passwords are 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, refer to Section 4.3, “Password Security”.

2.4.2. Vulnerable Client Applications

Although an administrator may have a fully secure and patched server, that does not mean 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 SSH, 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 to 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, *Workstation Security* discusses in more detail what steps administrators and home users should take to limit the vulnerability of computer workstations.

[4] Source: https://www.sans.org/reading_room/whitepapers/hsoffice/addressing_and_implementing_computer_security_for_a_small_branch_office_620

Part II. Configuring Red Hat Enterprise Linux for Security

This part informs and instructs administrators on proper techniques and tools to use when securing Red Hat Enterprise Linux workstations, Red Hat Enterprise Linux servers, and network resources. It also discusses how to make secure connections, lock down ports and services, and implement active filtering to prevent network intrusion.

Chapter 3. Security Updates

As security vulnerabilities are discovered, the affected software must be updated in order to limit any potential security risks. If the software is part of a package within an Red Hat Enterprise Linux distribution that is currently supported, Red Hat, Inc is committed to releasing updated packages that fix the vulnerability as soon as possible. Often, announcements about a given security exploit are accompanied with a patch (or source code that fixes the problem). This patch is then applied to the Red Hat Enterprise Linux package, tested by the Red Hat quality assurance team, and released as an errata update. However, if an announcement does not include a patch, a Red Hat developer works with the maintainer of the software to fix the problem. Once the problem is fixed, the package is tested and released as an errata update.

If an errata update is released for software used on your system, it is highly recommended that you update the effected packages as soon as possible to minimize the amount of time the system is potentially vulnerable.

3.1. Updating Packages

When updating software on a system, it is important to download the update from a trusted source. An attacker can easily rebuild a package with the same version number as the one that is supposed to fix the problem but with a different security exploit and release it on the Internet. If this happens, using security measures such as verifying files against the original RPM does not detect the exploit. Thus, it is very important to only download RPMs from trusted sources, such as from Red Hat, Inc and check the signature of the package to verify its integrity.

Red Hat offers two ways to find information on errata updates:

1. Listed and available for download on Red Hat Network
2. Listed and unlinked on the Red Hat Errata website

Note

Beginning with the Red Hat Enterprise Linux product line, updated packages can be downloaded only from Red Hat Network. Although the Red Hat Errata website contains updated information, it does not contain the actual packages for download.

3.1.1. Using Red Hat Network

Red Hat Network allows the majority of the update process to be automated. It determines which RPM packages are necessary for the 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 a *System Profile* for each machine to be updated. The System Profile contains hardware and software information about the system. This information is kept confidential and is not given to anyone else. It is only used to determine which errata updates are applicable to each system, and, without it, Red Hat Network can not determine whether a given system needs updates. When a security errata (or any type of errata) is released, Red Hat Network sends an email with a description of the errata as well as a list of systems which are affected. To apply the update, use the **Red Hat User Agent** or schedule the package to be updated through the website http://rhn.redhat.com.

Note>

Red Hat Enterprise Linux includes the **up2date**, a convenient panel icon that displays visible alerts when there is an update for a registered Red Hat Enterprise Linux system. Refer to the following URL for more information about the applet: http://rhn.redhat.com/help/basic/applet.html

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.

Important

Before installing any security errata, be sure to read any special instructions contained in the errata report and execute them accordingly. Refer to Section 3.1.5, “Applying the Changes” for general instructions about applying the changes made by an errata update.

3.1.2. Using the Red Hat Errata Website

When security errata reports are released, they are published on the Red Hat Errata website available at http://www.redhat.com/security/. 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 Enterprise 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 link to login to Red Hat Network, click 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.

3.1.3. Verifying Signed Packages

All Red Hat Enterprise Linux packages are signed with the Red Hat, Inc *GPG* key. GPG stands for GNU Privacy Guard, or GnuPG, a free software package used for ensuring the authenticity of distributed files. For example, a private key (secret key) held by Red Hat locks the package while the public key unlocks and verifies the package. If the public key distributed by Red Hat does not match the private key during RPM verification, the package may have been altered and therefore cannot be trusted.

The RPM utility within Red Hat Enterprise Linux automatically tries to verify the GPG signature of an RPM package before installing it. If the Red Hat GPG key is not installed, install it from a secure, static location, such as an Red Hat Enterprise Linux installation CD-ROM.

Assuming the CD-ROM is mounted in **/mnt/cdrom**, use the following command to import it into the *keyring* (a database of trusted keys on the system):

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

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

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

For the Red Hat key, the output includes the following:

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

To display details about a specific key, use the **rpm -qi** command followed by the output from the previous command, as in this example:

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

It is extremely important to verify the signature of the RPM files before installing them to ensure that they have not been altered from the Red Hat, Inc release of the packages. To verify all the downloaded packages at once, issue the following command:

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

For each package, if the GPG key verifies successfully, the command returns **gpg OK**. If it doesn't, make sure you are using the correct Red Hat public key, as well as verifying the source of the content. Packages that do not pass GPG verfications should not be installed, as they may have been altered by a third party.

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

3.1.4. Installing Signed Packages

Installation for most packages can be done safely (except kernel packages) by issuing the following command:

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

For kernel packages use the following command:

```
rpm -ivh /tmp/updates/<kernel-package>
```

Replace ***<kernel-package>*** in the previous example with the name of the kernel RPM.

Once the machine has been safely rebooted using the new kernel, the old kernel may be removed using the following command:

```
rpm -e <old-kernel-package>
```

Replace ***<old-kernel-package>*** in the previous example with the name of the older kernel RPM.

Note

It is not a requirement that the old kernel be removed. The default boot loader, GRUB, allows for multiple kernels to be installed, then chosen from a menu at boot time.

Important

Before installing any security errata, be sure to read any special instructions contained in the errata report and execute them accordingly. Refer to Section 3.1.5, “Applying the Changes” for general instructions about applying the changes made by an errata update.

3.1.5. Applying the Changes

After downloading and installing security errata via Red Hat Network or the Red Hat errata website, it is important to halt usage of the older software and begin using the new software. How this is done depends on the type of software that has been updated. The following list itemizes the general categories of software and provides instructions for using the updated versions after a package upgrade.

Note

In general, rebooting the system is the surest way to ensure that the latest version of a software package is used; however, this option is not always available to the system administrator.

Applications

User-space applications are any programs which can be initiated by a system user. Typically, such applications are used only when a user, script, or automated task utility launches them and they do not persist for long periods of time.

Once such a user-space application is updated, halt any instances of the application on the system and launch the program again to use the updated version.

Kernel

The kernel is the core software component for the Red Hat Enterprise Linux operating system. It manages access to memory, the processor, and peripherals as well as schedules all tasks.

Because of its central role, the kernel cannot be restarted without also stopping the computer. Therefore, an updated version of the kernel cannot be used until the system is rebooted.

Shared Libraries

Shared libraries are units of code, such as **glibc**, which are used by a number of applications and services. Applications utilizing a shared library typically load the shared code when the application is initialized, so any applications using the updated library must be halted and relaunched.

To determine which running applications link against a particular library, use the **lsof** command as in the following example:

```
lsof /usr/lib/libwrap.so*
```

This command returns a list of all the running programs which use TCP wrappers for host access control. Therefore, any program listed must be halted and relaunched if the **tcp_wrappers** package is updated.

SysV Services

SysV services are persistent server programs launched during the boot process. Examples of SysV services include **sshd**, **vsftpd**, and **xinetd**.

Because these programs usually persist in memory as long as the machine is booted, each updated SysV service must be halted and relaunched after the package is upgraded. This can be done using the **Services Configuration Tool** or by logging into a root shell prompt and issuing the **/sbin/service** command as in the following example:

```
/sbin/service <service-name> restart
```

In the previous example, replace ***<service-name>*** with the name of the service, such as **sshd**.

Refer to the chapter titled *Controlling Access to Services* in the *System Administrators Guide* for more information regarding the **Services Configuration Tool**.

xinetd Services

Services controlled by the **xinetd** super service only run when a there is an active connection. Examples of services controlled by **xinetd** include Telnet, IMAP, and POP3.

Because new instances of these services are launched by **xinetd** each time a new request is received, connections that occur after an upgrade are handled by the updated software. However, if there are active connections at the time the **xinetd** controlled service is upgraded, they are serviced by the older version of the software.

To kill off older instances of a particular **xinetd** controlled service, upgrade the package for the service then halt all processes currently running. To determine if the process is running, use the **ps** command and then use the **kill** or **killall** command to halt current instances of the service.

For example, if security errata **imap** packages are released, upgrade the packages, then type the following command as root into a shell prompt:

```
ps -aux | grep imap
```

This command returns all active IMAP sessions. Individual sessions can then be terminated by issuing the following command:

```
kill -9 <PID>
```

In the previous example, replace ***<PID>*** with the process identification number (found in the second column of the **ps** command) for an IMAP session.

To kill all active IMAP sessions, issue the following command:

```
killall imapd
```

Refer to the chapter titled *TCP Wrappers and **xinetd*** in the *Reference Guide* for general information regarding **xinetd**.

Chapter 4. Workstation Security

Securing a Linux environment begins with the workstation. Whether locking down a 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 its weakest node.

4.1. Evaluating Workstation Security

When evaluating the security of a Red Hat Enterprise 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* — Which tools should be used to communicate between workstations and which should be avoided?

4.2. BIOS and Boot Loader Security

Password protection for the BIOS (or BIOS equivalent) and the boot loader can prevent unauthorized users who have physical access to systems from booting using removable media or attaining root privileges 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, unencrypted SSH keys for the corporate network is left unattended at that same trade show, it could lead to a major security breach 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 [5]:

1. *Preventing Changes to BIOS Settings* — If an intruder has access to the BIOS, they can set it to boot from 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 start arbitrary processes on the system or copy sensitive data.
2. *Preventing System Booting* — Some BIOSes allow password protection of the boot process. When activated, an attacker is forced to enter a password before the BIOS launches the boot loader.

Because the methods for setting a BIOS password vary between computer manufacturers, consult the computer's manual for specific instructions.

If you forget the BIOS password, it can either be reset with jumpers on the motherboard or by disconnecting the CMOS battery. For this reason, it is good practice to lock the computer case if

possible. However, consult the manual for the computer or motherboard before attempting to disconnect the CMOS battery.

4.2.1.1. Securing Non-x86 Platforms

Other architectures use different programs to perform low-level tasks roughly equivalent to those of the BIOS on x86 systems. For instance, Intel® Itanium™ computers use the *Extensible Firmware Interface* (*EFI*) shell.

For instructions on password protecting BIOS-like programs on other architectures, refer to the manufacturer's instructions.

4.2.2. Boot Loader Passwords

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

1. *Preventing Access to Single User Mode* — If attackers can boot the system into single user mode, they are logged in automatically as root without being prompted for the root password.
2. *Preventing Access to the GRUB Console* — If the machine uses GRUB as its boot loader, an attacker can use the use the GRUB editor interface to change its configuration or to gather information using the **cat** command.
3. *Preventing 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.

The GRUB boot loader ships with Red Hat Enterprise Linux on the x86 platform. For a detailed look at GRUB, consult the chapter titled *The GRUB Boot Loader* in the *Reference Guide*.

4.2.2.1. Password Protecting GRUB

GRUB can be configured to address the first two issues listed in Section 4.2.2, “Boot Loader Passwords” 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 returns 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** [6].

The next time the system boots, the GRUB menu does not allow access to 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, a different part of the **/boot/grub/grub.conf** file must be edited.

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

For a DOS system, the stanza should begin similar to the following:

```
title DOS
lock
```

Warning

A **password** line must be present in the main section of the **/boot/grub/grub.conf** file for this method to work properly. Otherwise, an attacker can access the GRUB editor interface and remove the lock line.

To create a different password for a particular kernel or operating system, add a **lock** line to the stanza, followed by a password line.

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

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

4.3. Password Security

Passwords are the primary method Red Hat Enterprise Linux uses to verify a user's identity. This is why password security is enormously important for protection of the user, the workstation, and the network.

For security purposes, the installation program configures the system to use *Message-Digest Algorithm* (*MD5*) and shadow passwords. It is highly recommended that you do not alter these settings.

If MD5 passwords are deselected 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 shadow passwords are deselected during installation, all passwords are stored as a one-way hash in the world-readable **/etc/passwd** file, which makes the system vulnerable to offline password cracking attacks. If an intruder can gain access to the machine as a regular user, he can copy the **/etc/passwd** file to his own machine and run any number of password cracking programs against it. 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 are written to system files. Of course, if the cracker starts an attack in the middle of the night on a system with weak passwords, the cracker may have gained access before dawn and edited the log files to cover his tracks.

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

4.3.1. Creating Strong Passwords

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

Do Not Do the Following:

- *Do Not Use Only Words or Numbers* — Never use only numbers or words in a password.

  Some insecure 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.

  Some insecure examples include the following:

  - 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 not secure.

  Some insecure 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 l337 (LEET) speak — in your password, think again. Many word lists include LEET speak.

  Some insecure examples include the following:

  - H4X0R
  - 1337
- *Do Not Use Personal Information* — Steer clear of personal information. If the attacker knows your identity, the task of deducing your password becomes easier. The following is a list of the types of information to avoid when creating a password:

  Some insecure examples include the following:

  - 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 insecure examples include the following:

  - R0X4H
  - nauj
  - 9-DS
- *Do Not Write Down Your Password* — Never store a password on paper. It is much safer to memorize it.
- *Do Not Use the Same Password For All Machines* — It is important to make separate passwords for each machine. This way if one system is compromised, all of your machines

are not immediately at risk.

Do the Following:

- *Make the Password At Least Eight Characters Long* — The longer the password, the better. If using MD5 passwords, it should be 15 characters or longer. With DES passwords, use the maximum length (eight characters).
- *Mix Upper and Lower Case Letters* — Red Hat Enterprise Linux is case sensitive, so mix cases to 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 (this is not possible if using DES passwords).
- *Pick a Password You Can Remember* — The best password in the world does little good if you cannot remember it; 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 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 river and through the woods, to grandmother's house we go."
- Next, turn it into an acronym (including the punctuation).

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

  o7r@77w,7ghwg.
- Add more complexity by capitalizing at least one letter, such as **H**.

  o7r@77w,7gHwg.
- *Finally, do not use the example password above for any systems, ever*.

While creating secure passwords is imperative, managing them properly is also important, especially for system administrators within larger organizations. The following section details 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 within an organization, the system administrators have two basic options available to force the use of good passwords. They can create passwords for the user, or they can let users create their 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, most system administrators prefer to have the users 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 users are asked to create or change passwords, they can use the command line application **passwd**, which is *Pluggable Authentication Manager* (*PAM*) aware and therefore checks to see if the password is easy to crack or too short in length via the **pam_cracklib.so** 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 a new module. For a list of available PAM modules, refer to http://www.kernel.org/pub/linux/libs/pam/modules.html. For more information about PAM, refer to the chapter titled *Pluggable Authentication Modules (PAM)* in the *Reference Guide*.

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

There are many password cracking programs that run under Red Hat Enterprise Linux although none ship with the operating system. Below is a brief list of some of the more popular password cracking programs:

Note

None of these tools are supplied with Red Hat Enterprise Linux and are therefore not supported by Red Hat, Inc in any way.

- ***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 online 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 online at http://www.crypticide.com/users/alecm/.
- ***Slurpie*** — **Slurpie** is similar to **John The Ripper** and **Crack**, but 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 online 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 is prompted to create 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.

There are two primary programs used to specify password aging under Red Hat Enterprise Linux: the

chage command or the graphical **User Manager** (**system-config-users**) application.

The **-M** option of the **chage** command specifies the maximum number of days the password is valid. So, for instance, to set 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. To disable password expiration, it is traditional to use a value of **99999** after the **-M** option (this equates to a little over 273 years).

The graphical **User Manager** application may also be used to create password aging policies. To access this application, go to the **Main Menu** button (on the Panel) => **System Settings** => **Users &Groups** or type the command **system-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, “**Password Info** Pane”.

Figure 4.1. Password Info Pane

For more information about user and group configuration (including instructions on forcing first time passwords), refer to the chapter titled *User and Group Configuration* in the *System Administrators Guide*. For an overview of user and resource management, refer to the chapter titled *Managing User Accounts and Resource Access* in the *Red Hat Enterprise Linux Introduction to System Adminitration*.

4.4. Administrative Controls

When administering a home machine, the user must 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 program's owner 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, as in the following example:

```
-rwsr-xr-x    1 root     root        47324 May  1 08:09 /bin/su
```

For the system administrators of an organization, however, choices must be made as to how much administrative access 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 that logs in at the physical console (see the chapter titled *Pluggable Authentication Modules (PAM)* in the *Reference Guide* for more about 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 not possible without administrative priveleges. As a result, system administrators must decide how much access the users on their network should receive.

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 an issue. Allowing root access by users means that minor activities, like adding devices or configuring network interfaces, can be handled by the individual users, 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:

- *Machine Misconfiguration* — Users with root access can misconfigure their machines and require assistance or worse, open up security holes without knowing it.
- *Running 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 affect 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 (refer to Section 4.2.2, “Boot Loader Passwords” for more information on this topic.)

The following are four different ways that an administrator can further ensure that root logins are disallowed:

Changing the root shell

To prevent users from logging in directly as root, the system administrator can set the root account's shell to **/sbin/nologin** in the **/etc/passwd** file.

Table 4.1. Disabling the Root Shell

Effects	Does Not Affect
Prevents access to the root shell and logs any such attempts. 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)

To further limit access to the root account, administrators 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 via the console or a raw network interface. This is dangerous, because a user can log in to their machine as root via Telnet, which transmits the password in plain text over the network.

By default, Red Hat Enterprise Linux's **/etc/securetty** file only allows the root user to log in

at the console physically attached to the machine. To prevent the root user from logging in, remove the contents of this file by typing the following command at a shell prompt as root:

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

To enable **securetty** support in the KDM, GDM, and XDM login managers, add the following line:

```
auth [user_unknown=ignore success=ok ignore=ignore default=bad]
pam_securetty.so
```

to the files listed below:

- **/etc/pam.d/gdm**
- **/etc/pam.d/gdm-autologin**
- **/etc/pam.d/gdm-fingerprint**
- **/etc/pam.d/gdm-password**
- **/etc/pam.d/gdm-smartcard**
- **/etc/pam.d/kdm**
- **/etc/pam.d/kdm-np**
- **/etc/pam.d/xdm**

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.

Table 4.2. Disabling Root Logins

Effects	Does Not Affect
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 setuid or other mechanisms. The following programs are *not* prevented from accessing the root account: • **su** • **sudo** • **ssh** • **scp** • **sftp**

Disabling root SSH logins

To prevent root logins via the SSH protocol, edit the SSH daemon's configuration file, **/etc/ssh/sshd_config**, and change the line that reads:

```
#PermitRootLogin yes
```

to read as follows:

```
PermitRootLogin no
```

Table 4.3. Disabling Root SSH Logins

Effects	Does Not Affect
Prevents root access via the OpenSSH suite of tools. The following programs are prevented from accessing the root account: • **ssh** • **scp** • **sftp**	Programs that are not part of the OpenSSH suite of tools.

Using PAM to limit root access to services

PAM, through the **/lib/security/pam_listfile.so** module, allows great flexibility in denying specific accounts. The administrator can use this module to reference a list of users who are not allowed to log in. To limit root access to a system service, edit the file for the target service in the **/etc/pam.d/** directory and make sure the **pam_listfile.so** module is required for authentication.

The following is an example of how the module is used for the **vsftpd** FTP server in the **/etc/pam.d/vsftpd** PAM configuration file (the \ character at the end of the first line is *not* necessary if the directive is on a single line):

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

This instructs PAM to consult the **/etc/vsftpd.ftpusers** file and deny access to the service for any listed user. The administrator can 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, a similar line can be added to the PAM configuration files, 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, refer to the chapter titled *Pluggable Authentication Modules (PAM)* in the *Reference Guide*.

Table 4.4. Disabling Root Using PAM

Effects	Does Not Affect
Prevents root access to network services that are PAM aware. The following services are prevented from accessing the root account: • **login** • **gdm** • **kdm** • **xdm** • **ssh** • **scp** • **sftp** • FTP clients • Email clients • Any PAM aware services	Programs and services that are not PAM aware.

4.4.3. Limiting Root Access

Rather than completely deny access to the root user, the administrator may want to allow access only via setuid programs, such as **su** or **sudo**.

4.4.3.1. The su Command

Upon typing the **su** command, the user is prompted for the root password and, after authentication, is 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 become root, it is possible 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 within an organization 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>
```

In the previous command, replace **<*username*>** with the username you want to add to the **wheel** group.

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 **system-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, "**Groups** Pane".

Figure 4.2. 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/$ISA/pam_wheel.so use_uid
```

Doing this permits 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 to giving users administrative access. When trusted users precede an administrative command with **sudo**, they are prompted for *their own* 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 replaced 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 before walking away from their machines since sudoers can use the command again without being asked for a password within a five minute period. 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 *the user's* shell, not a root shell. This means the root shell can be completely disabled, as shown in Section 4.4.1, “Allowing Root Access”.

The **sudo** command also provides a comprehensive audit trail. Each successful authentication is logged to the file **/var/log/messages** and the command 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.

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

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 it is issued 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 are active is of paramount importance to anyone who administers and operates a Linux system.

Many services under Red Hat Enterprise 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:

- *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 on improperly written scripts. These script vulnerability attacks can lead to a buffer overflow condition or allow the attacker to alter files on the system.
- *Buffer Overflow Attacks* — Services which connect to ports numbered 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 use automated tools to identify systems with vulnerabilities, and once they have gained access, they use automated rootkits to maintain their access to the system.

Note

The threat of buffer overflow vulnerabilities is mitigated in Red Hat Enterprise Linux by *ExecShield*, an executable memory segmentation and protection technology supported by x86-compatible uni- and multi-processor kernels. ExecShield reduces the risk of buffer overflow by separating virtual memory into executable and non-executable segments. Any program code that tries to execute outside of the executable segment (such as malicious code injected from a buffer overflow exploit) triggers a segmentation fault and terminates.
Execshield also includes support for *No eXecute* (NX) technology on AMD64 platforms and *eXecute Disable* (XD) technology on Itanium and EM64T systems. These technologies work in conjunction with ExecShield to prevent malicious code from running in the executable portion of virtual memory with a granularity of 4kb of executable code, lowering the risk of attack from stealthy buffer overflow exploits.
For more information about ExecShield and NX or XD technologies, refer to the whitepaper entitled *New Security Enhancements in Red Hat Enterprise Linux v.3, Update 3*, available at the following URL:
http://www.redhat.com/solutions/info/whitepapers/

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 Enterprise Linux are turned off by default. There are, however, some notable exceptions:

- **cupsd** — The default print server for Red Hat Enterprise Linux.
- **lpd** — An alternate print server.
- **xinetd** — A super server that controls connections to a host of subordinate servers, such as **vsftpd** and **telnet**.
- **sendmail** — The Sendmail mail transport agent is enabled by default, but only listens for connections from the localhost.
- **sshd** — The OpenSSH server, which is a secure replacement for Telnet.

When determining whether to leave these services running, it is best to use common sense and err on the side of caution. For example, if a printer is not available, do not leave **cupsd** running. The same is true for **portmap**. If you do not mount NFSv3 volumes or use NIS (the **ypbind** service), then **portmap** should be disabled.

Red Hat Enterprise Linux ships with three programs designed to switch services on or off. They are the **Services Configuration Tool** (**system-config-services**), **ntsysv**, and **chkconfig**. For information on using these tools, refer to the chapter titled *Controlling Access to Services* in the *System Administrators Guide*.

Figure 4.3. Services Configuration Tool

If unsure of the purpose for a particular service, the **Services Configuration Tool** has a description field, illustrated in Figure 4.3, “Services Configuration Tool”, that may be of some use.

But checking which network services are available to start at boot time is not enough. Good system

administrators should also check which ports are open and listening. Refer to Section 5.8, “Verifying Which Ports Are Listening” 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, making it very important to keep packages associated with any network service updated. Refer to Chapter 3, *Security Updates* for more information about this issue.

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

- *Pass Usernames and Passwords Over a Network Unencrypted* — Many older protocols, such as Telnet and FTP, do not encrypt the authentication session and should be avoided whenever possible.
- *Pass Sensitive Data Over a Network Unencrypted* — Many protocols pass data over the network unencrypted. These protocols include Telnet, FTP, HTTP, and SMTP. Many network file systems, such as NFS and SMB, also pass information over the network unencrypted. It is the user's responsibility when using these protocols to limit what type of data is transmitted.

  Also, remote memory dump services, like **netdump**, pass the contents of memory over the network unencrypted. 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**

All remote login and shell programs (**rlogin**, **rsh**, and **telnet**) should be avoided in favor of SSH. (refer to Section 4.7, “Security Enhanced Communication Tools” for more information about **sshd**.)

FTP is not as inherently dangerous to the security of the system as remote shells, but FTP servers must be carefully configured and monitored to avoid problems. Refer to Section 5.6, “Securing FTP” for more information on securing FTP servers.

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

- **finger**
- **authd** (this was called **identd** in previous RHEL releases)
- **netdump**
- **netdump-server**
- **nfs**
- **rwhod**
- **sendmail**
- **smb** (Samba)
- **yppasswdd**
- **ypserv**

- **ypxfrd**

More information on securing network services is available in Chapter 5, *Server Security*.

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 system's network interface. If a request is made to a port that is blocked by a firewall, the request is ignored. If a service is listening on one of these blocked ports, it does 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 tool for configuring a simple firewall is the straight-forward, graphical firewall configuration tool which ships with Red Hat Enterprise Linux: the **Security Level Configuration Tool** (**system-config-securitylevel**). This tool creates broad **iptables** rules for a general-purpose firewall using a control panel interface.

For more information about using this application and the options it offers, refer to the chapter titled *Basic Firewall Configuration* in the *System Administrators Guide*.

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

4.7. Security Enhanced Communication Tools

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

Red Hat Enterprise 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, refer to the chapter titled *OpenSSH* in the *System Administrators Guide*. For more information about the SSH Protocol, refer to the chapter titled *SSH Protocol* in the *Reference Guide*.

Important

Although the **sshd** service is inherently secure, the service *must* be kept up-to-date to prevent security threats. Refer to Chapter 3, *Security Updates* for more information about this issue.

GPG is one way to ensure private email communication. It can be used both to email sensitive data over public networks and to protect sensitive data on hard drives.

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

[6] GRUB also accepts unencrypted passwords, but it is recommended that an md5 hash be used for added security.

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 specific issues, review the following general tips for enhancing server security:

- Keep all services current, to protect against the latest threats.
- Use secure protocols whenever possible.
- Serve only one type of network 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, which stand guard between an incoming request and the requested service.

The benefits offered by TCP wrappers are enhanced when used in conjunction with **`xinetd`**, a super service that provides additional access, logging, binding, redirection, and resource utilization control.

Note>

It is a good idea to use IPTables firewall rules in conjunction with TCP wrappers and **`xinetd`** to create redundancy within service access controls. Refer to Chapter 7, *Firewalls* for more information about implementing firewalls with IPTables commands.

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

The following subsections 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 illustrates 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, refer to the **`hosts_options`** man page.

5.1.1.1. TCP Wrappers and Connection Banners

Sending a client an intimidating banner when they connect to a service 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 **`vsftpd`**. To begin, create a banner file. It can be anywhere on the system, but it must bear same name as the daemon. For this example, the file is called **`/etc/banners/vsftpd`**.

The contents of the file 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 *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:

```
vsftpd : 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 the administrator 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, create a special script to notify the administrator or execute a chain of commands in the event that a particular client attempts to connect to the 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.

For this example, assume anyone attempting to connect to port 23 (the Telnet port) on an FTP server is a cracker. To denote this, place a **emerg** 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 emerg
```

This uses the default **authpriv** logging facility, but elevates the priority from the default value of **info** to **emerg**, which posts log messages directly to the console.

5.1.2. Enhancing Security With xinetd

The **xinetd** super server is another useful tool for controlling access to its subordinate services. This section focuses on how **xinetd** can be used to set a trap service and control the amount of resources

any given **xinetd** service can use to thwart denial of service attacks. For a more thorough list of the options available, refer to 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 is used.

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

```
flags = SENSOR
```

Add the following line within the braces:

```
deny_time = 30
```

This denies 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 does not work against stealth scans.
- An attacker who knows that a **SENSOR** is running 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 which 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.

Note

Securing **portmap** only affects NFSv2 and NFSv3 implementations, since NFSv4 no longer requires it. If you plan to implement a NFSv2 or NFSv3 server, then **portmap** is required, and the following section applies.

If running RPC services, follow these 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 using hostnames, as they can be forged 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 and restrict access to specific networks.

Below are two example IPTables commands that allow TCP connections to the **portmap** service (listening on port 111) from the 192.168.0/24 network and from the localhost (which is necessary for the **sgi_fam** service used by **Nautilus**). All other packets are dropped.

```
 iptables -A INPUT -p tcp -s! 192.168.0.0/24 --dport 111 -j DROP iptables -A
INPUT -p tcp -s 127.0.0.1 --dport 111 -j ACCEPT
```

To similarly limit UDP traffic, use the following command.

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

Note>

Refer to Chapter 7, *Firewalls* for more information about implementing firewalls with IPTables commands.

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 over the network unencrypted, 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, “Securing Portmap”, then address the following issues, such as network planning.

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 reveals 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 makes it much more difficult for an attacker to access the NIS server.

5.3.3. Edit the /var/yp/securenets File

NIS listens to all networks, if the **/var/yp/securenets** file is blank or does not exist (as is the case

after a default installation). One of the first things to do is to put netmask/network pairs in the file so that **ypserv** only responds to requests from the proper network.

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

```
255.255.255.0 192.168.0.0
```

Warning

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

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

5.3.4. Assign Static Ports and Use IPTables Rules

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 for the creation of 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 listens 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
```

Note>

Refer to Chapter 7, *Firewalls* 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 sent 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 an attacker can gain access to a valid account on the network.

Since Kerberos uses secret-key cryptography, no password hashes are ever sent over the network, making the system far more secure. For more about Kerberos, refer to the chapter titled *Kerberos* in the *Reference Guide*.

5.4. Securing NFS

The *Network File System*, or *NFS*, is service that provides network accessible file systems for client machines. For more information on how NFS works, refer to the chapter titled *Network File System (NFS)* in the *Reference Guide*. For more information about configuring NFS, refer to the *System Administrators Guide*. The following subsections assume a basic knowledge of NFS.

Important

The version of NFS included in Red Hat Enterprise Linux, NFSv4, no longer requires the **portmap** service as outlined in Section 5.2, “Securing Portmap”. NFS traffic now utilizes TCP in all versions, rather than UDP, and requires it when using NFSv4. NFSv4 now includes Kerberos user and group authentication, as part of the **RPCSEC_GSS** kernel module. Information on **portmap** is still included, since Red Hat Enterprise Linux supports NFSv2 and NFSv3 which utilize it.

5.4.1. Carefully Plan the Network

Now that NFSv4 has the ability to pass all information encrypted using Kerberos over a network, it is important that the service be configured correctly if it is behind a firewall or on a segmented network. NFSv2 and NFSv3 still pass data insecurely, and concerns should be taken into consideration. Careful network design in all of these regards can help prevent security breaches.

5.4.2. Beware of Syntax Errors

The NFS server determines which file systems to export and which hosts 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 **bob.example.com** with read/write permissions.

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

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

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

It is good practice to check any configured NFS shares by using the **showmount** command to verify what is being shared:

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

5.4.3. Do Not Use the no_root_squash Option

By default, NFS shares change the root user to the **nfsnobody** user, an unprivileged user account. In this way, all root-created files are owned by **nfsnobody**, which prevents uploading of programs with the setuid bit set.

If **no_root_squash** is used, remote root users are able to change any file on the shared file system and leave trojaned applications for other users to inadvertently execute.

5.5. Securing the Apache HTTP Server

The Apache HTTP Server is one of the most stable and secure services that ships with Red Hat Enterprise Linux. 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 when configuring the Apache HTTP Server to read the documentation available for the application. This includes the chapter titled *Apache HTTP Server* in the *Reference Guide*, the chapter titled *Apache HTTP Server Configuration* in the *System Administrators Guide*.

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

5.5.1. `FollowSymLinks`

This directive is enabled by default, be sure to use caution when creating symbolic links to 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. To prevent visitors from browsing files on the server, 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. 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/`**. 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 absolutely necessary, 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 assign write permissions to the root user for any directory containing scripts or CGIs. This can be accomplished by typing the following commands:

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

Also, always verify that any scripts running on the system work as intended *before* putting them into production.

5.6. Securing FTP

The *File Transport Protocol*, or *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 Enterprise Linux provides three FTP servers.

- **gssftpd** — A kerberized **xinetd**-based 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 standalone, security oriented implementation of the FTP service.

The following security guidelines are for setting up the **vsftpd** FTP service.

5.6.1. FTP Greeting Banner

Before submitting a username and password, 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 **vsftpd**, add the following directive to the **/etc/vsftpd/vsftpd.conf** file:

```
ftpd_banner=<insert_greeting_here>
```

Replace **<insert_greeting_here>** in the above directive with the text of the greeting message.

For mutli-line banners, it is best to use a banner file. To simplify management of multiple banners, place all banners in a new directory called **/etc/banners/**. The banner file for FTP connections in this example is **/etc/banners/ftp.msg**. Below is an example of what such a file may look like:

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

Note

It is not necessary to begin each line of the file with **220** as specified in Section 5.1.1.1, “TCP Wrappers and Connection Banners”.

To reference this greeting banner file for **vsftpd**, add the following directive to the **/etc/vsftpd/vsftpd.conf** file:

```
banner_file=/etc/banners/ftp.msg
```

It also is possible to send additional banners to incoming connections using TCP wrappers as described in Section 5.1.1.1, “TCP Wrappers and Connection Banners”.

5.6.2. Anonymous Access

The presence of the **/var/ftp/** directory activates the anonymous account.

The easiest way to create this directory is to install the **vsftpd** package. This package sets a directory tree up for anonymous users and configures the permissions on directories 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 aware of where sensitive data is stored.

5.6.2.1. Anonymous Upload

To allow anonymous users to upload, it is recommended that a write-only directory be created 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 730 /var/ftp/pub/upload
```

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

```
drwx-wx---    2 root     ftp          4096 Feb 13 20:05 upload
```

Warning

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

Additionally, under **vsftpd**, add the following line to the **/etc/vsftpd/vsftpd.conf** file:

```
anon_upload_enable=YES
```

5.6.3. 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 **vsftpd**, add the following directive to **/etc/vsftpd/vsftpd.conf**:

```
local_enable=NO
```

5.6.3.1. 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 an FTP server is to use a PAM list file as described in Section 4.4.1, “Allowing Root Access”. The PAM configuration file for **vsftpd** is **/etc/pam.d/vsftpd**.

It is also possible to disable user accounts within each service directly.

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

5.6.4. Use TCP Wrappers To Control Access

Use TCP wrappers to control access to either FTP daemon as outlined in Section 5.1.1, “Enhancing Security With TCP Wrappers”.

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 information about how email works and an overview of common configuration settings, refer to the chapter titled *Email* in the *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 *Reference Guide*.

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

5.7.1. Limiting a 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 in **/etc/mail/sendmail.mc**, the effectiveness of such attacks are 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 default, 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 blocks 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 NFSv2 and NFSv3 do not maintain control over user and group IDs, two or more users can have the same UID, and receive and read each other's mail. With NFSv4 using Kerberos, this is not the case, since the **SECRPC_GSS** kernel module does not utilize UID-based authentication.

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 **/sbin/nologin** (with the possible exception of the root user).

5.8. Verifying Which Ports Are Listening

After configuring network services, it is important to pay attention to which ports are actually listening on the system's 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 these programs do not connect to the machine from the network, but rather check to see what is running on the system. For this reason, these applications are frequent targets for replacement by attackers. In this way, crackers attempt to cover their tracks if they open unauthorized network ports.

A more reliable way to check which ports are listening on the network is to use 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 3.55 ( http://www.insecure.org/nmap/ ) at 2004-09-24 13:49 EDT
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1653 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
111/tcp   open  rpcbind
113/tcp   open  auth
631/tcp   open  ipp
834/tcp   open  unknown
2601/tcp  open  zebra
32774/tcp open  sometimes-rpc11
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.5.25 - 2.6.3 or Gentoo 1.2 Linux 2.4.19 rc1-rc7)
Uptime 12.857 days (since Sat Sep 11 17:16:20 2004)

Nmap run completed -- 1 IP address (1 host up) scanned in 5.190 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, 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)
```

These tools reveal a great deal 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 to medium sized businesses (SMBs) that want to expand without paying the high costs associated with enterprise-level, dedicated digital circuits.

To address this need, *Virtual Private Networks* (VPNs) were developed. Following the same functional principles as dedicated circuits, VPNs allow for secured digital communication between two parties (or networks), creating a Wide Area Network (WAN) from existing *Local Area Networks* (LANs). Where it differs from frame relay or ATM is in its transport medium. VPNs transmit over IP using datagrams as the transport layer, making it a secure conduit through the Internet to an intended destination. Most free software VPN implementations incorporate open standard encryption methods 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* (or Internet Protocol Security) implementation. These VPN solutions, regardless if hardware or software based, 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). The data is encrypted and is enclosed with decryption and handling instruction called the *Encapsulating Security Payload* (ESP). The receiving VPN router strips the header information, decrypts the data, and routes it to its intended destination (either a workstation or node on a network). Using a network-to-network connection, the receiving node on the local network receives the packets decrypted and ready for processing. The encryption/decryption process in a network-to-network VPN connection is transparent to a local node.

With such a heightened level of security, a cracker must not only intercept a packet, but decrypt the packet as well. Intruders who employ a man-in-the-middle attack between a server and client must also have access to at least one of the private keys for authenticating sessions. Because they employ several layers of authentication and encryption, VPNs are a secure and effective means to connect multiple remote nodes to act as a unified Intranet.

6.1. VPNs and Red Hat Enterprise Linux

Red Hat Enterprise Linux users have various options in terms of implementing a software solution to securely connect to their WAN. *Internet Protocol Security*, or IPsec is the supported VPN implementation for Red Hat Enterprise Linux that sufficiently addresses the usability needs of organizations with branch offices or remote users.

6.2. IPsec

Red Hat Enterprise Linux supports IPsec for connecting remote hosts and networks to each other using a secure tunnel on a common carrier network such as the Internet. IPsec can be implemented using a host-to-host (one computer workstation to another) or network-to-network (one LAN/WAN to another). The IPsec implementation in Red Hat Enterprise Linux uses *Internet Key Exchange* (*IKE*), which is a protocol implemented by the Internet Engineering Task Force (IETF) to be used for mutual authentication and secure associations between connecting systems.

An IPsec connection is split into two logical phases. In phase 1, an IPsec node initializes the connection with the remote node or network. The remote node/network checks the requesting node's credentials and both parties negotiate the authentication method for the connection. On Red Hat Enterprise Linux systems, an IPsec connection uses the *pre-shared key* method of IPsec node authentication. In a pre-shared key IPsec connection, both hosts must use the same key in order to move to the second phase of the IPsec connection.

Phase 2 of the IPsec connection is where the *security association* (SA) is created between IPsec nodes. This phase establishes an SA database with configuration information, such as the encryption method, secret session key exchange parameters, and more. This phase manages the actual IPsec connection between remote nodes and networks.

The Red Hat Enterprise Linux implementation of IPsec uses IKE for sharing keys between hosts across the Internet. The **racoon** keying daemon handles the IKE key distribution and exchange.

6.3. IPsec Installation

Implementing IPsec requires that the **ipsec-tools** RPM package be installed on all IPsec hosts (if using a host-to-host configuration) or routers (if using a network-to-network configuration). The RPM package contains essential libraries, daemons, and configuration files to aid in setup of the IPsec connection, including:

- **/sbin/setkey** — manipulates the key management and security attributes of IPsec in the kernel. This executable is controlled by the **racoon** key management daemon. For more information on **setkey**, refer to the **setkey**(8) man page.
- **/sbin/racoon** — the IKE key management daemon, used to manage and control security associations and key sharing between IPsec-connected systems. This daemon can be configured by editing the **/etc/racoon/racoon.conf** file. For more information about **racoon**, refer to the **racoon**(8) man page.
- **/etc/racoon/racoon.conf** — the **racoon** daemon configuration file used to configure various aspects of the IPsec connection, including authentication methods and encryption algorithms used in the connection. For a complete listing of directives available, refer to the **racoon.conf**(5) man page.

Configuring IPsec on Red Hat Enterprise Linux can be done via the **Network Administration Tool** or by manually editing networking and IPsec configuration files. For more information about using the **Network Administration Tool**, refer to the *System Administrators Guide*.

To connect two network-connected hosts via IPsec, refer to Section 6.4, “IPsec Host-to-Host Configuration”. To connect one LAN/WAN to another via IPsec, refer to Section 6.5, “IPsec Network-to-Network configuration”.

6.4. IPsec Host-to-Host Configuration

IPsec can be configured to connect one desktop or workstation to another by way of a host-to-host connection. This type of connection uses the network to which each host is connected to create the secure tunnel to each other. The requirements of a host-to-host connection are minimal, as is the configuration of IPsec on each host. The hosts need only a dedicated connection to a carrier network (such as the Internet) and Red Hat Enterprise Linux to create the IPsec connection.

The first step in creating a connection is to gather system and network information from each workstation. For a host-to-host connection, you need the following information:

- The IP address for both hosts
- A unique name to identify the IPsec connection and distinguish it from other devices or connections (for example, **ipsec0**)
- A fixed encryption key or one automatically generated by **racoon**
- A pre-shared authentication key that is used to initiate the connection and exchange encryption keys during the session

For example, suppose Workstation A and Workstation B want to connect to each other through an IPsec tunnel. They want to connect using a pre-shared key with the value of **foobarbaz** and the users agree to let **racoon** automatically generate and share an authentication key between each host. Both host users decide to name their connections **ipsec0**.

The following is the **ifcfg** file for Workstation A for a host-to-host IPsec connection with Workstation B (the unique name to identify the connection in this example is ***ipsec0***, so the resulting file is named **/etc/sysconfig/network-scripts/ifcfg-ipsec0**):

```
DST=X.X.X.X
TYPE=IPSEC
ONBOOT=yes
IKE_METHOD=PSK
```

Workstation A would replace ***X.X.X.X*** with the IP address of Workstation B, while Workstation B replaces ***X.X.X.X*** with the IP address of Workstation A. The connection is set to initiate upon boot-up (**ONBOOT=yes**) and uses the pre-shared key method of authentication (**IKE_METHOD=PSK**).

The following is the content of the pre-shared key file (called **/etc/sysconfig/network-scripts/keys-ipsec0**) that both workstations need to authenticate each other. The contents of this file should be identical on both workstations and only the root user should be able to read or write this file.

```
IKE_PSK=foobarbaz
```

Important

To change the **keys-ipsec0** file so that only the root user can read or edit the file, perform the following command after creating the file:

```
chmod 600 /etc/sysconfig/network-scripts/keys-ipsec0
```

To change the authentication key at any time, edit the **keys-ipsec0** file on both workstations. *Both keys must be identical for proper connectivity*.

The next example shows the specific configuration for the phase 1 connection to the remote host. The file is named ***X.X.X.X*.conf** (***X.X.X.X*** is replaced with the IP address of the remote IPsec router). Note that this file is automatically generated once the IPsec tunnel is activated and should not be edited directly.

```
;
remote X.X.X.X
{
        exchange_mode aggressive, main;
        my_identifier address;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 2 ;
        }
}
```

The default phase 1 configuration file created when an IPsec connection is initialized contains the following statements used by the Red Hat Enterprise Linux implementation of IPsec:

remote *X.X.X.X*

Specifies that the subsequent stanzas of this configuration file applies only to the remote node identified by the ***X.X.X.X*** IP address.

exchange_mode aggressive

The default configuration for IPsec on Red Hat Enterprise Linux uses an aggressive authentication mode, which lowers the connection overhead while allowing configuration of several IPsec connections with multiple hosts.

my_identifier address

Defines the identification method to be used when authenticating nodes. Red Hat Enterprise Linux uses IP addresses to identify nodes.

encryption_algorithm 3des

Defines the encryption cipher used during authentication. By default, *Triple Data Encryption Standard* (3DES) is used.

hash_algorithm sha1;

Specifies the hash algorithm used during phase 1 negotiation between nodes. By default, Secure Hash Algorithm version 1 is used.

authentication_method pre_shared_key

Defines the authentication method used during node negotiation. Red Hat Enterprise Linux by default uses pre-shared keys for authentication.

dh_group 2

Specifies the Diffie-Hellman group number for establishing dynamically-generated session keys. By default, the 1024-bit group is used.

The **/etc/racoon/racoon.conf** files should be identical on all IPsec nodes *except* for the **include "/etc/racoon/*X.X.X.X*.conf"** statement. This statement (and the file it references) is generated when the IPsec tunnel is activated. For Workstation A, the ***X.X.X.X*** in the **include** statement is

Workstation B's IP address. The opposite is true of Workstation B. The following shows a typical **racoon.conf** file when IPsec connection is activated.

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and entries.

path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

sainfo anonymous
{
 pfs_group 2;
 lifetime time 1 hour ;
 encryption_algorithm 3des, blowfish 448, rijndael ;
 authentication_algorithm hmac_sha1, hmac_md5 ;
 compression_algorithm deflate ;
}
include "/etc/racoon/X.X.X.X.conf"
```

This default **racoon.conf** file includes defined paths for IPsec configuration, pre-shared key files, and certificates. The fields in **sainfo anonymous** describe the phase 2 SA between the IPsec nodes — the nature of the IPsec connection (including the supported encryption algorithms used) and the method of exchanging keys. The following list defines the fields of phase 2:

sainfo anonymous

Denotes that SA can anonymously initialize with any peer insofar as the IPsec credentials match.

pfs_group 2

Defines the Diffie-Hellman key exchange protocol, which determines the method in which the IPsec nodes establish a mutual temporary session key for the second phase of IPsec connectivity. By default, the Red Hat Enterprise Linux implementation of IPsec uses group 2 (or **modp1024**) of the Diffie-Hellman cryptographic key exchange groups. Group 2 uses a 1024-bit modular exponentiation that prevents attackers from decrypting previous IPsec transmissions even if a private key is compromised.

lifetime time 1 hour

This parameter specifies the life cycle of an SA and can be quantified either by time or by bytes of data. The Red Hat Enterprise Linux implementation of IPsec specifies a one hour lifetime.

encryption_algorithm 3des, blowfish 448, rijndael

Specifies the supported encryption ciphers for phase 2. Red Hat Enterprise Linux supports 3DES, 448-bit Blowfish, and Rijndael (the cipher used in the *Advanced Encryption Standard*, or AES).

authentication_algorithm hmac_sha1, hmac_md5

Lists the supported hash algorithms for authentication. Supported modes are sha1 and md5 hashed message authentication codes (HMAC).

compression_algorithm deflate

Defines the Deflate compression algorithm for IP Payload Compression (IPCOMP) support, which allows for potentially faster transmission of IP datagrams over slow connections.

To start the connection, either reboot the workstation or execute the following command as root on each host:

```
/sbin/ifup ipsec0
```

To test the IPsec connection, run the **tcpdump** utility to view the network packets being transfered between the hosts (or networks) and verify that they are encrypted via IPsec. The packet should include an AH header and should be shown as ESP packets. ESP means it is encrypted. For example:

```
17:13:20.617872 pinky.example.com > ijin.example.com: \
     AH(spi=0x0aaa749f,seq=0x335): ESP(spi=0x0ec0441e,seq=0x335) (DF)
```

6.5. IPsec Network-to-Network configuration

IPsec can also be configured to connect an entire network (such as a LAN or WAN) to a remote network by way of a network-to-network connection. A network-to-network connection requires the setup of IPsec routers on each side of the connecting networks to transparently process and route information from one node on a LAN to a node on a remote LAN. Figure 6.1, “A Network-to-network IPsec tunneled connection” shows a network-to-network IPsec tunneled connection.

Figure 6.1. A Network-to-network IPsec tunneled connection

This diagram shows two separate LANs separated by the Internet. These LANs use IPsec routers to authenticate and initiate a connection using a secure tunnel through the Internet. Packets that are intercepted in transit would require brute-force decryption in order to crack the cipher protecting the packets between these LANs. The process of communicating from one node on the 192.168.1.0/24 IP range to another on 192.168.2.0/24 is completely transparent to the nodes as the processing, encryption/decryption, and routing of the IPsec packets are completely handled by the IPsec router.

The information needed for a network-to-network connection include:

- The externally-accessible IP addresses of the dedicated IPsec routers
- The network address ranges of the LAN/WAN served by the IPsec routers (such as 192.168.0.0/24 or 10.0.1.0/24)
- The IP addresses of the gateway devices that route the data from the network nodes to the Internet
- A unique name to identify the IPsec connection and distinguish it from other devices or connections (for example, **ipsec0**)
- A fixed encryption key or one automatically generated by **racoon**
- A pre-shared authentication key that initiates the connection and exchange encryption keys during the session

For example, suppose LAN A (lana.example.com) and LAN B (lanb.example.com) want to connect to each other through an IPsec tunnel. The network address for LAN A is in the 192.168.1.0/24 range, while LAN B uses the 192.168.2.0/24 range. The gateway IP address is 192.168.1.254 for LAN A and 192.168.2.254 for LAN B. The IPsec routers are separate from each LAN gateway and uses two network

devices: eth0 is assigned to an externally-accessible static IP address which accesses the Internet, while eth1 acts as a routing point to process and transmit LAN packets from one network node to the remote network nodes.

The IPsec connection between each network uses a pre-shared key with the value of **r3dh4tl1nux**, and the administrators of A and B agree to let **racoon** automatically generate and share an authentication key between each IPsec router. The administrator of LAN A decides to name the IPsec connection **ipsec0**, while the administrator of LAN B names the IPsec connection **ipsec1**..

The following example are the contents the **ifcfg** file for a network-to-network IPsec connection for LAN A. The unique name to identify the connection in this example is ***ipsec0***, so the resulting file is named **/etc/sysconfig/network-scripts/ifcfg-ipsec0**.

```
TYPE=IPSEC
ONBOOT=yes
IKE_METHOD=PSK
SRCGW=192.168.1.254
DSTGW=192.168.2.254
SRCNET=192.168.1.0/24
DSTNET=192.168.2.0/24
DST=X.X.X.X
```

The connection is set to initiate upon boot-up (**ONBOOT=yes**) and uses the pre-shared key method of authentication (**IKE_METHOD=PSK**). The administrator for LAN A enters the destination gateway, which is the gateway for LAN B (**DSTGW=192.168.2.254**) as well as the source gateway, which is the gateway IP address for LAN A (**SRCGW=192.168.1.254**). The administrator then enters the destination network, which is the network range for LAN B (**DSTNET=192.168.2.0/24**) as well as the source network (**SRCNET=192.168.1.0/24**). Finally, the administrator enters the destination IP address, which is the externally-accessible IP address for LAN B (***X.X.X.X***).

The following example is the content of the pre-shared key file called **/etc/sysconfig/network-scripts/keys-ipsec*X*** (where ***X*** is 0 for LAN A and 1 for LAN B) that both networks use to authenticate each other. The contents of this file should be identical and only the root user should be able to read or write this file.

```
IKE_PSK=r3dh4tl1nux
```

Important

To change the **keys-ipsec*X*** file so that only the root user can read or edit the file, perform the following command after creating the file:

```
chmod 600 /etc/sysconfig/network-scripts/keys-ipsec1
```

To change the authentication key at any time, edit the **keys-ipsec*X*** file on both IPsec routers. *Both keys must be identical for proper connectivity*.

The following example is the contents of the **/etc/racoon/racoon.conf** configuration file for the IPsec connection. Note that the **include** line at the bottom of the file is automatically generated and only appears if the IPsec tunnel is running.

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and entries.

path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

sainfo anonymous
{
 pfs_group 2;
 lifetime time 1 hour ;
 encryption_algorithm 3des, blowfish 448, rijndael ;
 authentication_algorithm hmac_sha1, hmac_md5 ;
 compression_algorithm deflate ;
}
include "/etc/racoon/X.X.X.X.conf"
```

The following is the specific configuration for the connection to the remote network. The file is named ***X.X.X.X*.conf** (replace ***X.X.X.X*** with the IP address of the remote IPsec router). Note that this file is automatically generated once the IPsec tunnel is activated and should not be edited directly.

```
;
remote X.X.X.X
{
        exchange_mode aggressive, main;
        my_identifier address;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 2 ;
        }
}
```

Prior to starting the IPsec connection, IP forwarding should be enabled in the kernel. As root at a shell prompt, enable IP forwarding:

1. Edit **/etc/sysctl.conf** and set **net.ipv4.ip_forward** to **1**.
2. Execute the following command to enable the change:

```
sysctl -p /etc/sysctl.conf
```

To start the IPsec connection, either reboot the IPsec routers or execute the following command as root on each router:

```
/sbin/ifup ipsec0
```

The connections are activated, and both LAN A and B are able to communicate with each other. The routes are created automatically via the initialization script called by running **ifup** on the IPsec connection. To show a list of routes for the network, run the following command:

```
/sbin/ip route list
```

To test the IPsec connection, run the **tcpdump** utility on the externally-routable device (eth0 in this example) to view the network packets being transfered between the hosts (or networks) and verify that

they are encrypted via IPsec. For example, to check the IPsec connectivity of LAN A, type the following:

```
tcpdump -n -i eth0 host lana.example.com
```

The packet should include an AH header and should be shown as ESP packets. ESP means it is encrypted. For example (back slashes denote a continuation of one line):

```
12:24:26.155529 lanb.example.com > lana.example.com: AH(spi=0x021c9834,seq=0x358):
\
   lanb.example.com > lana.example.com: ESP(spi=0x00c887ad,seq=0x358) (DF) \
   (ipip-proto-4)
```

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 Enterprise Linux includes several powerful tools to assist administrators and security engineers with network-level access control issues.

Along with VPN solutions, such as IPsec (discussed in Chapter 6, *Virtual Private Networks*), firewalls are one of the core components of a 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, Nokia, and Sonicwall. 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, “Firewall Types” details three common types of firewalls and how they function:

Table 7.1. Firewall Types

Method	Description	Advantages	Disadvantages
NAT	*Network Address Translation* (NAT) places private IP subnetworks behind one or a small pool of public 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) simplifies 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	A packet filtering firewall reads 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 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
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 frequently-accessed data locally 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	· 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

	logged and monitored closely, allowing tighter control over resource utilization on the network	are passed through one source rather than directly from a client to a remote service

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`** utility.

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 *Reference Guide*.

7.2. 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 **`ip6tables`** services should be turned off to use the **`iptables`** service with the following commands:

```
service ip6tables stop
chkconfig 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 iptables on
```

The syntax of **`iptables`** is separated into tiers. The main tier is the *chain*. A chain specifies the state at which a packet is manipulated. The usage is as follows:

```
iptables -A chain -j target
```

The **-A** option appends a rule at the end of an existing ruleset. The ***chain*** is the name of the chain for a rule. 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. The **-j *target*** option specifies the location in the **iptables** ruleset where this particular rule should *jump*. Some built in targets are ACCEPT, DROP, and REJECT.

New chains (also called user-defined chains) can be created by using the **-N** option. Creating a new chain is useful for customizing granular or elaborate rules.

7.2.1. Basic Firewall Policies

Establishing basic firewall policies creates a foundation for building more detailed, user-defined rules. **iptables** uses policies (**-P**) to create default rules. Security-minded administrators usually elect to drop all packets as a policy and only allow specific packets on a case-by-case basis. The following rules block all incoming and outgoing packets on a network gateway:

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

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. To do this, use the following rule:

```
iptables -P FORWARD DROP
```

After setting the policy chains, you can create new rules for your particular network and security requirements. The following sections outline some rules you may implement in the course of building your **iptables** firewall.

7.2.2. Saving and Restoring `iptables` Rules

Firewall rules are only valid for the time the computer is on; so, if the system is rebooted, the rules are automatically flushed and reset. To save the rules so that they are loaded later, use the following command:

```
/sbin/service iptables save
```

The rules are stored in the file **/etc/sysconfig/iptables** and are applied whenever the service is started or restarted, including when the machine is rebooted.

7.3. Common `iptables` 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. However, with a default policy set to block all incoming, outgoing, and forwarded packets, it is impossible for the firewall/gateway and internal LAN users to communicate with each other or with external resources. To allow users to perform network-related functions and use networking applications, administrators must open certain ports for communication.

For example, to allow access to port 80 *on the firewall*, append the following rule:

```
 iptables -A INPUT -p tcp -m tcp --sport 80 -j ACCEPT iptables -A OUTPUT -p
tcp -m tcp --dport 80 -j ACCEPT
```

This allows regular Web browsing from websites that communicate via port 80. To allow access to secure websites (such as https://www.example.com/), you must open port 443, as well.

```
 iptables -A INPUT -p tcp -m tcp --sport 443 -j ACCEPT iptables -A OUTPUT -p
tcp -m tcp --dport 443 -j ACCEPT
```

Important

When creating an **iptables** ruleset, it is critical to remember that order is important. For example, if one chain that specifies that any packets from the local 192.168.100.0/24 subnet be dropped, and then another chain is appended (**-A**) to allow packets from 192.168.100.13 (which is within the dropped restricted subnet), then the appended rule is ignored. You must set a rule to allow 192.168.100.13 first, and then set a drop rule on the subnet.
To arbitrarily insert a rule in an existing chain of rules, use **-I**, followed by the chain in which to insert the rule, and a rule number (1,2,3,...,n) for where the rule should reside. For example:

```
 iptables -I INPUT 1 -i lo -p all -j ACCEPT
```

The rule is inserted as the first rule in the INPUT chain to allow local loopback device traffic.

There may be times when you require remote access to the LAN from outside the LAN. Secure services such as SSH, can be used for encrypted remote connection to LAN services. For administrators with PPP-based resources (such as modem banks or bulk ISP accounts), dial-up access can be 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 configure **iptables** to accept connections from remote SSH clients. For example, to allow remote SSH access, the following rules may be used:

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p udp --sport 22 -j ACCEPT
```

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

These rules allow incoming and outbound access for an individual system, such as a single PC directly connected to the Internet or a firewall/gateway. However, they do not allow nodes behind the firewall/gateway to access these services. To allow LAN access to these services, you can use NAT with **iptables** filtering rules.

7.4. 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 limited public IP addresses to every node on the LAN. Using private IP address is the common way to allow all nodes on a LAN to properly access internal and external network services. Edge routers (such as firewalls) can receive incoming transmissions from the Internet and route the packets to the intended LAN node. At the same time, firewall/gateways 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 can be implemented to prevent aberrant usage of network resources.

The **FORWARD** policy allows an administrator to control where packets can be routed within a LAN. For example, to allow forwarding for the entire LAN (assuming the firewall/gateway is assigned an internal IP address on eth1), the following rules can be set:

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

This rule gives systems behind the firewall/gateway access to the internal network. The gateway routes packets from one LAN node to its intended destination node, passing all packets through its **eth1** device.

Note

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

```
sysctl -w net.ipv4.ip_forward=1
```

If this command is run via shell prompt, then the setting is not remembered after a reboot. You can permanently set forwarding by editing the **/etc/sysctl.conf** file. Find and edit the following line, replacing **0** with **1**:

```
net.ipv4.ip_forward = 0
```

Execute the following command to enable the change to the **sysctl.conf** file:

```
sysctl -p /etc/sysctl.conf
```

Accepting forwarded packets via the firewall's internal IP device allows LAN nodes to communicate with each other; however they still are not allowed to communicate externally to the Internet. To allow LAN nodes with private IP addresses to communicate with external public networks, configure the firewall for *IP masquerading*, which masks requests from LAN nodes with the IP address of the firewall's external device (in this case, eth0):

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

The rule uses the NAT packet matching table (**-t nat**) and specifies the built-in POSTROUTING chain for NAT (**-A POSTROUTING**) on the firewall's external networking device (**-o eth0**). POSTROUTING allows packets to be altered as they are leaving the firewall's external device. The **-j MASQUERADE** target is specified to mask the private IP address of a node with the external IP address of the firewall/gateway.

If you have a server on your internal network that you want make available externally, you can use the **-j DNAT** target of the PREROUTING chain in NAT to specify a destination IP address and port where incoming packets requesting a connection to your internal service can be forwarded. For example, if you

wanted to forward incoming HTTP requests to your dedicated Apache HTTP Server server system at 172.31.0.23, run the following command:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
      --to 172.31.0.23:80
```

This rule specifies that the NAT table use the built-in PREROUTING chain to forward incoming HTTP requests exclusively to the listed destination IP address of 172.31.0.23.

Note

If you have a default policy of DROP in your FORWARD chain, you must append a rule to allow forwarding of incoming HTTP requests so that destination NAT routing can be possible. To do this, run the following command:

```
iptables -A FORWARD -i eth0 -p tcp --dport 80 -d 172.31.0.23 -j ACCEPT
```

This rule allows forwarding of incoming HTTP requests from the firewall to its intended destination of the Apache HTTP Server server behind the firewall.

7.4.1. DMZs and `iptables`

`iptables` rules can be set to route traffic to certain machines, such as a dedicated HTTP or FTP server, in a *demilitarized zone* (DMZ) — a special local subnetwork dedicated to providing services on a public carrier such as the Internet. For example, to set a rule for routing incoming HTTP requests to a dedicated HTTP server at 10.0.4.2 (outside of the 192.168.1.0/24 range of the LAN), NAT calls a **`PREROUTING`** table to forward the packets to their proper destination:

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

With this command, all HTTP connections to port 80 from the outside of the LAN are 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. If the HTTP server is configured to accept secure connections, then port 443 must be forwarded as well.

7.5. Viruses and Spoofed IP Addresses

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 terminology). 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
iptables -A FORWARD -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
```

You can also block outside connections that attempt to spoof private IP address ranges to infiltrate your LAN. For example, if your 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 to the external-facing device (eth0) is rejected automatically.

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

Note

There is a distinction between the **DROP** and **REJECT** targets when dealing with *appended* rules. The **REJECT** target denies access and returns a **connection refused** error to users who attempt to connect to the service. The **DROP** target, as the name implies, drops the packet without any warning. 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.

7.6. iptables and Connection Tracking

iptables includes a module that allows administrators to inspect and restrict connections to services available on an internal network using a method called *connection tracking*. Connection tracking stores connections in a table, which allows administrators to allow or deny access based on the following connection states:

- **NEW** — A packet requesting a new connection, such as an HTTP request.
- **ESTABLISHED** — A packet that is part of an existing connection.
- **RELATED** — A packet that is requesting a new connection but is part of an existing connection, such as passive FTP connections where the connection port is 20, but the transfer port can be any unused port 1024 or higher.
- **INVALID** — A packet that is not part of any connections in the connection tracking table.

You can use the stateful functionality of **iptables** connection tracking with any network protocol, even if the protocol itself is stateless (such as UDP). The following example shows a rule that uses connection tracking to forward only the packets that are associated with an established connection:

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

7.7. 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 Enterprise 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 **iptables** services must be turned off to use the **ip6tables** service exclusively:

```
service iptables stop
chkconfig iptables off
```

To make **ip6tables** start by default whenever the system is booted, change the 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/.

7.8. Additional Resources

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

7.8.1. Installed Documentation

- The *Reference Guide* has a comprehensive chapter on **iptables**, including definitions for all command options.
- The **iptables** man page contains a brief summary of the various options, as well.
- A list of common services and their port numbers can be found in Appendix C, *Common Ports* and in **/etc/services**.

7.8.2. Useful Websites

- http://www.netfilter.org/ — The official homepage of the Netfilter and **iptables** project.
- http://www.tldp.org/ — The Linux Documentation Project contains several useful guides relating to firewall creation and administration.
- http://www.iana.org/assignments/port-numbers — The official list of registered and common service ports as assigned by the Internet Assigned Numbers Authority.

7.8.3. Related Documentation

- *Red Hat Linux Firewalls*, by Bill McCarty; Red Hat Press — a comprehensive reference to building network and server firewalls using open source packet filtering technology such as Netfilter and **iptables**. It includes such topics as analyzing firewall logs, developing firewall rules, and customizing your firewall with graphical tools such as **lokkit**.
- *Linux Firewalls*, by Robert Ziegler; New Riders Press — 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.

Part III. Assessing Your Security

This part provides an overview of the theory and practice of security assessment. From network monitors to cracking tools, an administrator can learn more about securing a system and a network by cracking into it.

Chapter 8. Vulnerability Assessment

Given time, resources, and motivation, a cracker can break into nearly any system. At the end of the day, all of the security procedures and technologies currently available cannot guarantee that any systems are safe from intrusion. Routers help secure gateways to the Internet. Firewalls help secure the edge of the network. Virtual Private Networks safely pass data in an encrypted stream. Intrusion detection systems 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 corporate resources can be quite complex. Due to this complexity, it is often 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.

8.1. Thinking Like the Enemy

Suppose that you administer an enterprise network. Such networks are commonly comprised of operating systems, applications, servers, network monitors, firewalls, intrusion detection systems, and more. Now imagine trying to keep current with each 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 large 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, you must think like a cracker and gauge the security of your 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 exploits it.

A vulnerability assessment is an internal audit of your network and system security; the results of which indicate the confidentiality, integrity, and availability of your network (as explained in Section 1.1.4, “Standardizing Security”). Typically, vulnerability assessment starts with a reconnaissance phase, during which important data regarding the target systems and resources is gathered. This phase leads to the system readiness phase, whereby the target is essentially checked for all known vulnerabilities. The readiness 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 (or mitigating the risk 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 closed and locked. You would also check every window, making sure that they closed completely and latch correctly. This same concept applies to systems, networks, and electronic data. Malicious users are the thieves and vandals of your data. Focus on their tools, mentality, and motivations, and you can then react swiftly to their actions.

8.2. Defining Assessment and Testing

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 — publicly-routable IP addresses, systems on your *DMZ*, external interfaces of your firewall, and more. DMZ stands for "demilitarized zone", which corresponds to a computer or small subnetwork that sits between a trusted internal network, such as a corporate private LAN, and an untrusted external network, such as the public Internet. Typically, the DMZ contains devices accessible to Internet traffic, such as Web (HTTP) servers, FTP servers, SMTP (e-mail) servers and DNS servers.

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, user-level access controls, authentication procedures for internal resources, and more). Typically, there are many more resources when looking around inside as most systems are internal to a company. Once you set yourself outside of the company, you immediately are given an untrusted status. The systems and resources available to you externally are usually very 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 is used for 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 are 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 a vulnerability assessment and a penetration test is defined, take the findings of the assessment and review them carefully before conducting a penetration test as part of your new best practices approach.

Warning

Attempting to exploit vulnerabilities on production resources can have adverse effects to the productivity and efficiency of your systems and network.

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

- Creates proactive focus on information security
- Finds potential exploits before crackers find them

- Results in systems being kept up to date and patched
- Promotes growth and aids in developing staff expertise
- Abates Financial loss and negative publicity

8.2.1. Establishing a Methodology

To aid in the selection of tools for a 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 help determine not only which tools to select but also the manner in which they are used.

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

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

8.3. Evaluating the Tools

An assessment can start by using some form of an information gathering tool. When assessing the entire network, map the layout first to find the hosts that are running. Once located, examine each host individually. Focusing on these hosts requires 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 the protocols used). Some tools are free; others are not. Some tools are intuitive and easy to use, while others are cryptic and poorly documented but have features that other tools do not.

Finding the right tools may be a daunting task and 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. Additionally, 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.

8.3.1. Scanning Hosts with Nmap

Nmap is a popular tool included in Red Hat Enterprise Linux that can be used to determine the layout of a network. 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 provides a detailed description of its options and usage. Administrators can use Nmap on a 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 allows Nmap to attempt to identify the operating system running on a particular host. Nmap is a good foundation for establishing a policy of using secure services and stopping unused services.

8.3.1.1. Using Nmap

Nmap can be run from a shell prompt by typing the **nmap** command followed by the hostname or IP address of the machine 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:

```
Starting nmap V. 3.50 ( 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
443/tcp    open        https
515/tcp    open        printer
950/tcp    open        oftep-rpc
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 71.825 seconds
```

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

For more information about using Nmap, refer to the official homepage at the following URL:

http://www.insecure.org/

8.3.2. 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 frequently updated and 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 Enterprise 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 the following URL:

http://www.nessus.org/

8.3.3. Nikto

Nikto is an excellent common gateway interface (CGI) script scanner. Nikto not only checks for CGI vulnerabilities but does so in an evasive manner, so as to elude intrusion detection systems. It comes with thorough documentation which should be carefully reviewed prior to running the program. If you have Web servers serving up CGI scripts, Nikto can be an excellent resource for checking the security of these servers.

Note

Nikto is not included with Red Hat Enterprise 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 Nikto can be found at the following URL:

http://www.cirt.net/code/nikto.shtml

8.3.4. VLAD the Scanner

VLAD is a vulnerabilities scanner developed by the RAZOR team at Bindview, Inc., which 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 Enterprise 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 RAZOR team website at the following URL:

http://www.bindview.com/Support/Razor/Utilities/

8.3.5. Anticipating Your Future Needs

Depending upon your target and resources, there are many tools available. There are tools for wireless networks, Novell networks, Windows systems, Linux systems, and more. Another essential part of performing assessments may include reviewing physical security, personnel screening, or voice/PBX network assessment. New concepts, such as *war walking* — scanning the perimeter of your enterprise's physical structures for wireless network vulnerabilities — are some emerging concepts that you can investigate and, if needed, incorporate into your assessments. Imagination and exposure are the only limits of planning and conducting vulnerability assessments.

Part IV. Intrusions and Incident Response

It is inevitable that a network falls to intrusion or malicious use of network resources. This part discusses some proactive measures an administrator can take to prevent security breaches, such as forming an emergency response team capable of quickly and effectively responding to security issues. This part also details the steps an administrator can take to collect and analyze evidence of a security breach after the fact.

Chapter 9. Intrusion Detection

Valuable property needs to be protected from the prospect of theft and destruction. Some 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 ensure 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 personal to 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 a 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.

9.1. 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 an IDS detects anomalies can vary widely; however, the ultimate aim of any IDS is to catch perpetrators in the act before they do real damage to resources.

An IDS protects a system from attack, misuse, and compromise. It can also monitor network activity, audit network and system configurations 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.

9.1.1. IDS Types

Understanding what an IDS is, and the functions it provides, is key in determining what type is appropriate to include in a computer security policy. This section discusses 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-based* 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 combine 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.

The most common types of IDSes referred to in the security field are known as *host-based* and *network-based* IDSes. A host-based IDS is the most comprehensive of the two, which involves implementing a detection system on each individual host. Regardless of which network environment the host resides on, it is still protected. A network-based IDS funnels packets through a single device before being sent to specific hosts. Network-based IDSes are often regarded as less comprehensive since many hosts in a mobile environment make it unavailable for reliable network packet screening and protection.

9.2. 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 (for example, minor printer messages versus major kernel warnings). The **syslog** command is available when installing the **sysklogd** package, which is included with Red Hat Enterprise Linux. This package provides system logging and kernel message trapping. The host-based IDS filters logs (which, in the case of some network and kernel event logs, can be quite verbose), analyzes them, re-tags the anomalous messages with its own system of severity rating, and collects them in its own specialized log for administrator analysis.

A host-based IDS can also verify the data integrity of important files and executables. It checks a database of sensitive files (and any files added by the administrator) and creates a *checksum* of each file with a message-file digest utility such as **md5sum** (128-bit algorithm) or **sha1sum** (160-bit algorithm). The host-based 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 file checksums do not match, the IDS alerts the administrator by email or cellular pager. This is the process used by Tripwire, which is discussed in Section 9.2.1, “Tripwire”.

9.2.1. Tripwire

Tripwire is the most popular host-based IDS for Linux. Tripwire, Inc., the developers of Tripwire, opened the software source code for the Linux version and licensed it under the terms of the GNU General Public License. Tripwire is available from http://www.tripwire.org/.

Note

Tripwire is not included with Red Hat Enterprise 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.

9.2.2. 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 RPM options that can verify file integrity on a Red Hat Enterprise Linux system. Refer to the *System Administrators Guide* for complete information about using RPM.

Important

Some of the commands in the following list require the importation of the Red Hat GPG public key into the system's RPM keyring. This key verifies that packages installed on the system contain an Red Hat package signature, which ensures that the packages originated from Red Hat. The key can be imported by issuing the following command as root (substituting ***<version>*** with the version of RPM installed on the system):

```
rpm --import /usr/share/doc/rpm-<version>/RPM-GPG-KEY
```

rpm -V *package_name*

The **-V** option verifies the files in the installed package called ***package_name***. If it shows no output and exits, this means that none of the files have been modified in any way since the last time the RPM database was updated. If there is an error, such as the following

```
S.5....T c /bin/ps
```

then the file has been modified in some way and you must assess whether to keep the file (such as with modified configuration files in the **/etc/** directory) or delete the file and reinstall the package that contains it. The following list defines the elements of the 8-character string (**S.5....T** in the above example) that notifies of a verification failure.

- **.** — The test has passed this phase of verification
- **?** — The test has found a file that could not be read, which is most likely a file permission issue
- **S** — The test has encountered a file that that is smaller or larger than it was when originally installed on the system
- **5** — The test has found a file whose md5 checksum does not match the original checksum of the file when first installed
- **M** — The test has detected a file permission or file type error on the file
- **D** — The test has encountered a device file mismatch in major/minor number
- **L** — The test has found a symbolic link that has been changed to another file path
- **U** — The test has found a file that had its user ownership changed
- **G** — The test has found a file that had its group ownership changed
- **T** — The test has encountered **mtime** verification errors on the file

rpm -Va

The **-Va** option 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

The **-Vf** option verifies individual files in an installed package. This can be useful when performing a quick verification of a suspect file.

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

The **-K** option is useful for checking the md5 checksum and the GPG signature of an RPM package file. This is useful for checking whether a package about to be installed is signed by Red Hat or any organization for which you have the GPG public key imported into a GPG keyring. A package that has not been properly signed triggers 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 not approved by Red Hat, Inc and could contain malicious code.

RPM can be a powerful tool, as evidenced by its many verification tools for installed packages and RPM package files. It is strongly recommended that the contents of the RPM database directory (**/var/lib/rpm/**) be backed up to read-only media, such as CD-ROM, after installation of Red Hat Enterprise Linux. Doing so allows verification of 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 the results.

9.2.3. 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 regarding installation and configuration.

Note

These applications are not included with Red Hat Enterprise 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://sourceforge.net/projects/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.

9.3. 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 cellular pager 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 activity and successfully tag suspect transmissions 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 passed on the network. Promiscuous mode can be set through the **ifconfig** command, such as the following:

```
ifconfig eth0 promisc
```

Running **ifconfig** with no options reveals that **eth0** is now in promiscuous (**PROMISC**) 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 Enterprise Linux), we can see the large amounts of traffic flowing throughout a network:

```
tcpdump: listening on eth0
02:05:53.702142 pinky.example.com.ha-cluster > \
 heavenly.example.com.860:  udp 92 (DF)
02:05:53.702294 heavenly.example.com.860 > \
 pinky.example.com.ha-cluster:  udp 32 (DF)
02:05:53.702360 pinky.example.com.55828 > dns1.example.com.domain: \
 PTR? 192.35.168.192.in-addr.arpa. (45) (DF)
02:05:53.702706 ns1.example.com.domain > pinky.example.com.55828: \
 6077 NXDomain* 0/1/0 (103) (DF)
02:05:53.886395 shadowman.example.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.example.com.netbios-ns > 172.16.59.255.netbios-ns:\
 NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
02:05:56.323715 pinky.example.com.1013 > heavenly.example.com.860:\
 udp 56 (DF)
02:05:56.323882 heavenly.example.com.860 > pinky.example.com.1013:\
 udp 28 (DF)
```

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

9.3.1. Snort

While **tcpdump** is a useful auditing tool, it is not considered a true IDS because it does not analyze and flag packets for anomalies. Instead, **tcpdump** prints *all* packet information to the screen or to a log file without any analysis. A proper IDS analyzes the packets, tags potentially malicious packet transmissions, and stores them in a formatted log.

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, in addition to its functionality, is 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 can benefit. This community ethic of sharing has developed Snort into one of the most up-to-date and robust network-based IDSes available.

Note

Snort is not included with Red Hat Enterprise 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 10. 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.

10.1. Defining Incident Response

An incident response is an expedited reaction to a security issue or occurrence. Pertaining to information security, an example would be a security team's actions against 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 while attempting to guarantee data integrity.

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

Research into past internal and external security breaches shows that some companies go of business as a result of a serious breach of security. A breach can result in resources rendered unavailable and data being either stolen or corrupted. But one cannot overlook issues that are difficult to calculate financially, such as bad publicity. To gain an accurate idea of how important an efficient incident response is, an organization must calculate the cost of the actual security breach as well as the financial effects of the negative publicity over, in the short and long term.

10.2. Creating an Incident Response Plan

It is important that an *incident response plan* is formulated, supported throughout the organization, and is regularly tested. A good incident response plan can minimize not only the affects of the actual security breach, but it may also reduce the negative publicity.

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 occurs. Do not think of a system as weak and vulnerable; it is important to realize that given enough time and resources, someone can break into even the most security-hardened system or network. You do not need to look any further than the *Security Focus* website, http://www.securityfocus.com/, for updated and detailed information concerning recent security breaches and vulnerabilities, such as the frequent defacement of corporate webpages or the 2002 attacks on the root DNS nameservers [7].

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 itself can be separated into four phases:

- Immediate action to stop or minimize the incident
- Investigation of the incident
- Restoration of affected resources
- Reporting the incident to the proper channels

An incident response must be decisive and executed quickly. Because there is little room for error, it is critical that practice emergencies are staged and response times measured. This way it is possible to

develop a methodology that fosters speed and accuracy, minimizing the impact of resource unavailability and potential damage in the event of an actual system compromise.

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

- A team of in-house experts (a *Computer Emergency Response Team*)
- A legally reviewed and approved strategy
- Financial support from the company
- Executive/upper management support
- A feasible and tested action plan
- Physical resources, such as redundant storage, standby systems, and backup services

10.2.1. The Computer Emergency Response Team (CERT)

The Computer Emergency Response Team (CERT) is a group of in-house experts who are prepared to act quickly in the event of a catastrophic computer event. 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 accept the responsibility that is required of them to respond to an emergency at any hour.

CERT teams typically include system and network administrators as well as information security experts. System administrators provide the knowledge and expertise of system resources, including data backups, backup hardware available for use, and more. Network administrators provide their knowledge of network protocols and the ability to re-route network traffic dynamically. Information security personnel are useful for thoroughly tracking and tracing security issues as well as performing a *post-mortem (after the attack)* analysis of compromised systems.

Although it may not always be feasible, 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, if only one person owns the key to data safety and integrity, then the entire enterprise becomes helpless in that one person's absence.

10.2.2. Legal Considerations

Some important aspects of an incident response to consider include legal ramifications. Security plans should be developed with members of legal staff or some form of general counsel. Just as every company should have their own corporate security policy, every company should have 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, is dictated by (or in conjunction with) legal counsel. General counsel can alert technical staff of the legal ramifications of security breaches; the hazards of leaking a client's personal, medical, or financial records; and the importance of restoring service in mission-critical environments such as hospitals and banks.

10.3. 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 an active implementation can result in poor response time and downtime in the event of a breach. This is where practice exercises become invaluable. Unless something is brought to attention before the plan is actively set in production, the implementation should be agreed upon by all directly connected parties and executed with confidence.

If a breach is detected and the CERT team is present for quick reaction, potential responses can vary. The team can decide to disable the network connections, disconnect the affected systems, patch the exploit, and then reconnect quickly without further, potential complications. The team can also watch the perpetrators and track their actions. The team could even redirect the perpetrator to a *honeypot* — a system or segment of a network containing intentionally false data — used to track incursion safely and without disruption to production resources.

Responding to an incident should also be accompanied by information gathering whenever 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 are great resources in tracking such anomalies in a system. System administrators know what processes should and should not appear when running **top** or **ps**. Network administrators are aware of what normal network traffic should look like when running **snort** or even **tcpdump**. These team members should know their systems and should be able to spot an anomaly more quickly than someone unfamiliar with the infrastructure.

10.4. Investigating the Incident

Investigating a computer breach is like investigating a crime scene. Detectives collect evidence, note any strange clues, and take inventory on loss and damage. An analysis of a computer compromise can either be done as the attack is happening or post-mortem.

Although it is unwise to trust any system log files on an exploited system, there are other forensic utilities to aid in the 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 file system information, and recover deleted files whenever possible.

It is also a good idea to record of all of the investigatory actions executed on a compromised system by using the **script** command, as in the following example:

```
script -q <file-name>
```

Replace **<*file-name*>** with file name for the **script** log. Always save the log file on media other than the hard drive of the compromised system — a floppy disk or CD-ROM works particularly well for this purpose.

By recording all your actions, an audit trail is created that may prove valuable if the attacker is ever caught.

10.4.1. 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 **coreutils** package in Red Hat Enterprise Linux to create a monolithic image of an exploited system as evidence in an investigation or for comparison with trusted images. Suppose there is a single hard drive from a system you want to image. Attach that drive as a slave to the system and then use **dd** to create the image file, such as the following:

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

This command creates a single file named **image1** using a 1k block size for speed. The **conv=noerror,sync** options force **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.

10.4.2. 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 Enterprise 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 10.1, “File Auditing Tools” details some commands for file auditing and management. It also lists some examples that can be used to properly identify files and file attributes (such as permissions and access dates) to allow the collection of further evidence or items for analysis. These tools, when combined with intrusion detection systems, firewalls, hardened services, and other security measures, can help reduce the amount of potential damage when an attack occurs.

Note

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

Table 10.1. File Auditing Tools

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	Finds useful string (text) information inside files and directories as well as reveals permissions, script changes, file attributes, and more. Used mostly as a piped command of for commands like **ls**, **ps**, or **ifconfig**.	**ps auxw |grep /bin**
strings	Prints the strings of printable characters within 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 file.	**strings /bin/ps |grep 'mail'**
file	Determines the characteristics of files based on format, encoding, linked-libraries (if any), and file type (binary, text, and more). It is useful for determining whether an executable such as **/bin/ls** has been modified using static libraries, which is a sure sign that the executable has been replaced with one installed by a malicious user.	**file /bin/ls**
find	Searches directories for particular files. It is a useful tool for searching the directory structure by keyword, date and time of access, permissions, and more. It can also be useful for administrators that perform general system audits of particular directories or files.	**find -atime +12 -name *log* -perm u+rw**
stat	Displays file status information, including time last accessed, permissions, UID and GID bit settings, and more. It can be useful for checking when a breached system executable was last used or modified.	**stat /bin/netstat**
md5sum	Calculates the 128-bit checksum using the md5 hash algorithm. Use this command to create a text file that lists all crucial executables that are	**md5sum /usr/bin/gdm >>md5sum.txt**

	often 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.

10.5. Restoring and Recovering Resources

While an incident response is in progress, the CERT team should be investigating while working toward data and system recovery. Unfortunately, it is the nature of the breach which dictates the course of recovery. Having backups or offline, redundant systems during this time is invaluable.

To recover systems, the response team must bring any downed systems or applications back online, 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. Only the most recent and pertinent data needs to be imported. This ready-made system should be kept isolated from the rest of the network. If a compromise occurs and the backup system is a part of the network, then the purpose of having a backup system is defeated.

System recovery can be a tedious process. In many instances there are two courses of action from which to choose. Administrators can perform a clean re-installation of the operating system on each affected system followed by restoration of all applications and data. Alternatively, administrators can patch the offending vulnerabilities and bring the affected system back into production.

10.5.1. Reinstalling the System

Performing a clean re-installation ensures that the affected system is cleansed of any trojans, backdoors, or malicious processes. Re-installation also ensures that any data (if restored from a trusted backup source) is cleared of any malicious modifications. 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, system downtime is greatly reduced.

10.5.2. Patching the System

Patching affected systems is a more dangerous course of action and should be undertaken with great caution. The problem with patching a system instead of reinstalling is determining whether or not a given system is *cleansed* of trojans, security holes, and corrupted data. Most *rootkits* (programs or packages that a cracker uses to gain root access to a system), trojan system commands, and shell environments are designed to hide malicious activities from cursory audits. If the patch approach is taken, only trusted binaries should be used (for example, from a mounted, read-only CD-ROM).

10.6. 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 and report all issues 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 an enterprise employs, a post-mortem analysis may be required. Even if it is not a functional requirement to a compromise analysis, a post-mortem can prove invaluable in helping to learn how a cracker thinks and how the systems are structured so that future compromises can be prevented.

[7] http://www.gcn.com/21_32/web/20404-1.html

Part V. Appendixes

This part discusses some of the most common ways an intruder can breach computer systems or intercept data in transit. This part also details some of the most commonly used services and their associated port numbers, which can be useful to administrators looking to mitigate the risks of being cracked.

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 the *Security Guide* is to explain how to secure Red Hat Enterprise Linux, a more detailed examination of hardware and physical network security is beyond the scope of this document. However, this chapter presents a brief overview of establishing security policies with respect to hardware and physical networks. Important factors to consider include how computing needs and connectivity requirements fit into the overall security strategy. The following explains some of these factors in detail.

- *Computing* involves more than just workstations running desktop software. Modern organizations require massive computational power and highly-available services, which can include mainframes, compute or application 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.
- *Connectivity* is the method by which an administrator intends to connect disparate resources to a network. An administrator may use Ethernet (hubbed or switched CAT-5/RJ-45 cabling), token ring, 10-base-2 coaxial cable, or even 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 allows an easier administrative process if security issues arise.

From these general considerations, administrators can get a better view of implementation. The design of a computing environment can then be based on both organizational needs and security considerations — an implementation that evenly assesses both factors.

A.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 resources provided, distance between nodes, and transmission medium. Depending upon the needs of the organization that the network services, there are several choices available for network implementation. Each topology has unique advantages and security issues that network architects should regard when designing their network layout.

A.1.1. Physical Topologies

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

A.1.1.1. Ring Topology

The *Ring* topology connects each node using 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 or 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).

A.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.

A.1.1.3. Star Topology

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

A.1.2. Transmission Considerations

Section A.1.1.3, “Star Topology” introduced the concept of broadcast and switched networking. There are several factors to consider when evaluating the type of networking hardware suitable and secure enough for your network environment. The following distinguishes these two distinct forms of networking.

In a broadcast network, a node will send a packet that is received by every other node until the intended recipient accepts the packet. Every node in the network can 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, creates a *direct* connection between the sending and recipient nodes. 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, where each node receives every packet on the way 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 can store such information in text files and, perhaps, even send the information to arbitrary sources (for example, the malicious user's email address.)

A switched network requires a network switch, a specialized piece of hardware that 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, D-Link, SMC, and Netgear offer various types of switches with features such as 10/100-Base-T compatibility, gigabit Ethernet support, and IPv6 networking.

A.1.3. Wireless Networks

An emerging issue for enterprises today is that of mobility. Remote workers, field technicians, and executives require portable solutions, such as 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 currently approved IEEE standard is 802.11g for wireless networking, while 802.11a and 802.11b are legacy standards. The 802.11g standard is backwards-compatible with 802.11b, but is incompatible with 802.11a.

The 802.11b and 802.11g specifications are actually a group of standards governing wireless communication and access control on the unlicensed 2.4GHz radio-frequency (RF) spectrum (802.11a uses the 5GHz spectrum). These specifications have been approved as standards by the IEEE, and several vendors market 802.11***x*** products and services. 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 requiring broadband Internet access to conduct business remotely.

The 802.11***x*** specifications allow 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 dedicated 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 allow wireless nodes to connect to the WAP as if it were on the Ethernet-based network. The WAP effectively acts as a bridge between the nodes connected to it and the rest of the network.

A.1.3.1. 802.11*x* 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 802.11***x*** network, many administrators fail to exercise even the most basic security precautions. Since all 802.11***x*** networking is done using high-band RF signals, the data transmitted is easily accessible to any user with a compatible 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 Equivalent Privacy (WEP) protocol, which is an RC4-based 64- or 128-bit encrypted key shared between each node or between the WAP and the node. This key encrypts transmissions and decrypts 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). However, enabling WEP on a wireless network can greatly reduce the possibility of data interception.

Red Hat Enterprise Linux supports various 802.11***x*** products from several vendors. The **Network Administration Tool** includes a facility for configuring wireless NICs and WEP security. For information about using the **Network Administration Tool**, refer to the *System Administrators Guide*.

Relying on WEP, however, is still not a sufficiently sound means of protection against determined malicious users. There are specialized utilities specifically designed to crack the RC4 WEP encryption algorithm protecting a wireless network and to expose 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 connectivity by restricting it only to SSH or VPN connections, which introduce 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-bit DES algorithm encryption (3DES), or proprietary algorithms of even greater strength. Administrators who apply these policies should restrict plain text protocols such as Telnet or FTP, as passwords and data can be exposed using any of the aforementioned attacks.

A recent method of security and authentication that has been adopted by wireless networking equipment manufacturers is *Wi-fi Protected Access* (WPA). Administrators can configure WPA on their network by using an authentication server that manages keys for clients accessing the wireless network. WPA improves upon WEP encryption by using *Temporal Key Integrity Protocol* (TKIP), which is a method of using a shared key and associating it with the MAC address of the wireless network card installed on the client system. The value of the shared key and MAC address is then processed by an *initialization vector* (IV), which is used to generate a key that encrypts each data packet. The IV changes the key

each time a packet is transferred, preventing most common wireless network attacks.

However, WPA using TKIP is thought of as a temporary solution. Solutions using stronger encryption ciphers (such as AES) are under development, and have the potential to improve wireless network security in the enterprise.

For more information about 802.11 standards, refer to the following URL:

```
http://standards.ieee.org/getieee802/802.11.html
```

A.1.4. Network Segmentation and DMZs

For administrators who want 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 the hardening of hosts and applications are effective ways to deter casual intruders. However, determined crackers can find ways into the internal network if the services they have cracked reside on the same network segment. The externally accessible services should reside on what the security industry regards as a *demilitarized zone* (DMZ), a logical network segment where inbound traffic from the Internet would only be able to access those services and are not permitted to access the internal network. This is effective in that, even if 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 using dedicated hardware firewalls allow for complex routing and forwarding rules. Administrators can use these policies to segment inbound traffic to specific services at specified addresses and ports while allowing only LAN access to internal services, which can prevent IP spoofing exploits. For more information about implementing **`iptables`**, refer to Chapter 7, *Firewalls*.

A.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 is just as important as an external strategy. This section explains some of the common steps administrators and users can take to safeguard their systems from internal exploitation.

Employee workstations, for the most part, are not as likely to be targets for remote attacks, 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 use a BIOS that controls system resources on the hardware level. Workstation users can set administrative passwords within the BIOS to prevent malicious users from accessing or booting 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 among 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 accessing the hard drive to read its contents. In these cases, it is recommended that workstations have locks to restrict access to internal hardware. Specialized security devices, such as lockable steel cables, can be attached to PC and laptop chassis to prevent theft, as well as locks on the chassis itself to prevent internal access. This type of 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) tampering.

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.

Common Exploits and Attacks

Table B.1, “Common Exploits” 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.

Table B.1. Common Exploits

Exploit	Description	Notes
Null or Default Passwords	Leaving administrative passwords blank or using a default password set by the product vendor. This is most common in hardware such as routers and firewalls, though some services that run on Linux can contain default administrator passwords (though Red Hat Enterprise 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 user accounts in a rush and leave the password null, a perfect entrypoint for malicious users who discover the account.
Default Shared Keys	Secure services sometimes package default security keys for development or evaluation testing purposes. If these keys are left unchanged and are placed in a production environment on the Internet, *all* users with the same default keys have access to that shared-key resource, and any sensitive information contained in it.	Most common in wireless access points and preconfigured secure server appliances. CIPE (refer to Chapter 6, *Virtual Private Networks*) contains a sample static key that must be changed before deployment in 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 horse 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 recommended when compared to PKI or other forms of encrypted authentication used in **ssh** or SSL/TLS.
Eavesdropping	Collecting data that passes between two active nodes on a network by eavesdropping on 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. Preventive measures include services with cryptographic key exchange, one-time passwords, or encrypted authentication to prevent password snooping; strong encryption during transmission is 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 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 execution and even interactive 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 overflows*, where attackers crash a service using arbitary values that fill the memory buffer of an application, giving the attacker an interactive command prompt from which they may execute arbitrary commands) can give complete administrative control to an attacker. Administrators should make sure that services do not run as the root user, and should stay vigilant of patches and errata updates for applications from vendors or security organizations such as CERT and CVE.
Application Vulnerabilities	Attackers find faults in desktop and workstation applications (such as e-mail clients) and execute arbitrary code, implant trojan horses 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 as workers do not have the expertise or experience to prevent or detect a compromise; it is imperative to inform individuals of the risks they are taking when they install unauthorized software or open unsolicited email attachments. Safeguards can be implemented such that email client software does not automatically open or execute

		attachments. Additionally, the automatic update of workstation software via Red Hat Network or other system management services can alleviate the burdens of multi-seat security deployments.
Denial of Service (DoS) Attacks	Attacker or group of attackers coordinate against an organization's network or server resources by sending unauthorized packets to the target host (either server, router, or workstation). This forces the resource to become unavailable to legitimate users.	The most reported DoS case in the US occurred in 2000. Several highly-trafficked commercial and government sites were rendered unavailable by a coordinated ping flood attack using several compromised systems with high bandwidth connections acting as *zombies*, or redirected broadcast nodes. Source packets are usually forged (as well as rebroadcasted), making investigation as to the true source of the attack difficult. Advances in ingress filtering (IETF rfc2267) using **`iptables`** and Network IDSes such as **`snort`** assist administrators in tracking down and preventing distributed DoS attacks.

Common Ports

The following tables list the most common communication ports used by services, daemons, and programs included in Red Hat Enterprise Linux. This listing can also be found in the **/etc/services** file. For the official list of Well Known, Registered, and Dynamic ports as designated by the Internet Assigned Numbers Authority (IANA), refer to the following URL:

http://www.iana.org/assignments/port-numbers

Note

The ***Layer***, where listed, denotes whether the service or protocol uses TCP or UDP for transport. If not listed, the service/protocol can use both TCP and UDP.

Table C.1, “Well Known Ports” lists the Well Known Ports as defined by IANA and is used by Red Hat Enterprise Linux as default communication ports for various services, including FTP, SSH, and Samba.

Table C.1. Well Known Ports

Port # / Layer	Name	Comment
1	tcpmux	TCP port service multiplexer
5	rje	Remote Job Entry
7	echo	Echo service
9	discard	Null service for connection testing
11	systat	System Status service for listing connected ports
13	daytime	Sends date and time to requesting host
17	qotd	Sends quote of the day to connected host
18	msp	Message Send Protocol
19	chargen	Character Generation service; sends endless stream of characters
20	ftp-data	FTP data port
21	ftp	File Transfer Protocol (FTP) port; sometimes used by File Service Protocol (FSP)
22	ssh	Secure Shell (SSH) service
23	telnet	The Telnet service
25	smtp	Simple Mail Transfer Protocol (SMTP)
37	time	Time Protocol
39	rlp	Resource Location Protocol
42	nameserver	Internet Name Service
43	nicname	WHOIS directory service
49	tacacs	Terminal Access Controller Access Control System for TCP/IP based authentication and access
50	re-mail-ck	Remote Mail Checking Protocol
53	domain	domain name services (such as BIND)
63	whois++	WHOIS++, extended WHOIS services
67	bootps	Bootstrap Protocol (BOOTP) services; also used by Dynamic Host Configuration Protocol (DHCP) services
68	bootpc	Bootstrap (BOOTP) client; also used by Dynamic Host Configuration Protocol (DHCP) clients
69	tftp	Trivial File Transfer Protocol (TFTP)
70	gopher	Gopher Internet document search and retrieval
71	netrjs-1	Remote Job Service
72	netrjs-2	Remote Job Service
73	netrjs-3	Remote Job Service
73	netrjs-4	Remote Job Service
79	finger	Finger service for user contact information
80	http	HyperText Transfer Protocol (HTTP) for World Wide Web (WWW) services
88	kerberos	Kerberos network authentication system
95	supdup	Telnet protocol extension
101	hostname	Hostname services on SRI-NIC machines
102/tcp	iso-tsap	ISO Development Environment (ISODE) network applications

105	csnet-ns	Mailbox nameserver; also used by CSO nameserver
107	rtelnet	Remote Telnet
109	pop2	Post Office Protocol version 2
110	pop3	Post Office Protocol version 3
111	sunrpc	Remote Procedure Call (RPC) Protocol for remote command execution, used by Network Filesystem (NFS)
113	auth	Authentication and Ident protocols
115	sftp	Simple File Transfer Protocol services
117	uucp-path	Unix-to-Unix Copy Protocol (UUCP) Path services
119	nntp	Network News Transfer Protocol (NNTP) for the USENET discussion system
123	ntp	Network Time Protocol (NTP)
137	netbios-ns	NETBIOS Name Service used in Red Hat Enterprise Linux by Samba
138	netbios-dgm	NETBIOS Datagram Service used in Red Hat Enterprise Linux by Samba
139	netbios-ssn	NETBIOS Session Service used in Red Hat Enterprise Linux by Samba
143	imap	Internet Message Access Protocol (IMAP)
161	snmp	Simple Network Management Protocol (SNMP)
162	snmptrap	Traps for SNMP
163	cmip-man	Common Management Information Protocol (CMIP)
164	cmip-agent	Common Management Information Protocol (CMIP)
174	mailq	MAILQ email transport queue
177	xdmcp	X Display Manager Control Protocol (XDMCP)
178	nextstep	NeXTStep window server
179	bgp	Border Gateway Protocol
191	prospero	Prospero distributed filesystem services
194	irc	Internet Relay Chat (IRC)
199	smux	SNMP UNIX Multiplexer
201	at-rtmp	AppleTalk routing
202	at-nbp	AppleTalk name binding
204	at-echo	AppleTalk echo
206	at-zis	AppleTalk zone information
209	qmtp	Quick Mail Transfer Protocol (QMTP)
210	z39.50	NISO Z39.50 database
213	ipx	Internetwork Packet Exchange (IPX), a datagram protocol commonly used in Novell Netware environments
220	imap3	Internet Message Access Protocol version 3
245	link	LINK / 3-DNS iQuery service
347	fatserv	FATMEN file and tape management server
363	rsvp_tunnel	RSVP Tunnel
369	rpc2portmap	Coda file system portmapper
370	codaauth2	Coda file system authentication services
372	ulistproc	UNIX LISTSERV

389	ldap	Lightweight Directory Access Protocol (LDAP)
427	svrloc	Service Location Protocol (SLP)
434	mobileip-agent	Mobile Internet Protocol (IP) agent
435	mobilip-mn	Mobile Internet Protocol (IP) manager
443	https	Secure Hypertext Transfer Protocol (HTTP)
444	snpp	Simple Network Paging Protocol
445	microsoft-ds	Server Message Block (SMB) over TCP/IP
464	kpasswd	Kerberos password and key changing services
468	photuris	Photuris session key management protocol
487	saft	Simple Asynchronous File Transfer (SAFT) protocol
488	gss-http	Generic Security Services (GSS) for HTTP
496	pim-rp-disc	Rendezvous Point Discovery (RP-DISC) for Protocol Independent Multicast (PIM) services
500	isakmp	Internet Security Association and Key Management Protocol (ISAKMP)
535	iiop	Internet Inter-Orb Protocol (IIOP)
538	gdomap	GNUstep Distributed Objects Mapper (GDOMAP)
546	dhcpv6-client	Dynamic Host Configuration Protocol (DHCP) version 6 client
547	dhcpv6-server	Dynamic Host Configuration Protocol (DHCP) version 6 Service
554	rtsp	Real Time Stream Control Protocol (RTSP)
563	nntps	Network News Transport Protocol over Secure Sockets Layer (NNTPS)
565	whoami	whoami user ID listing
587	submission	Mail Message Submission Agent (MSA)
610	npmp-local	Network Peripheral Management Protocol (NPMP) local / Distributed Queueing System (DQS)
611	npmp-gui	Network Peripheral Management Protocol (NPMP) GUI / Distributed Queueing System (DQS)
612	hmmp-ind	HyperMedia Management Protocol (HMMP) Indication / DQS
631	ipp	Internet Printing Protocol (IPP)
636	ldaps	Lightweight Directory Access Protocol over Secure Sockets Layer (LDAPS)
674	acap	Application Configuration Access Protocol (ACAP)
694	ha-cluster	Heartbeat services for High-Availability Clusters
749	kerberos-adm	Kerberos version 5 (v5) 'kadmin' database administration
750	kerberos-iv	Kerberos version 4 (v4) services
765	webster	Network Dictionary
767	phonebook	Network Phonebook
873	rsync	rsync file transfer services
992	telnets	Telnet over Secure Sockets Layer (TelnetS)
993	imaps	Internet Message Access Protocol over Secure Sockets Layer (IMAPS)
994	ircs	Internet Relay Chat over Secure Sockets Layer (IRCS)

995	pop3s	Post Office Protocol version 3 over Secure Sockets Layer (POP3S)

Table C.2, “UNIX Specific Ports” lists UNIX-specific ports and cover services ranging from email to authentication and more. Names enclosed in brackets (for example, [***service***]) are either daemon names for the service or common alias(es).

Table C.2. UNIX Specific Ports

Port # / Layer	Name	Comment
512/tcp	exec	Authentication for remote process execution
512/udp	biff [comsat]	Asynchrous mail client (biff) and service (comsat)
513/tcp	login	Remote Login (rlogin)
513/udp	who [whod]	whod user logging daemon
514/tcp	shell [cmd]	Remote shell (rshell) and remote copy (rcp) with no logging
514/udp	syslog	UNIX system logging service
515	printer [spooler]	Line printer (lpr) spooler
517/udp	talk	Talk remote calling service and client
518/udp	ntalk	Network talk (ntalk) remote calling service and client
519	utime [unixtime]	UNIX time (utime) protocol
520/tcp	efs	Extended Filename Server (EFS)
520/udp	router [route, routed]	Routing Information Protocol (RIP)
521	ripng	Routing Information Protocol for Internet Protocol version 6 (IPv6)
525	timed [timeserver]	Time daemon (timed)
526/tcp	tempo [newdate]	Tempo
530/tcp	courier [rpc]	Courier Remote Procedure Call (RPC) protocol
531/tcp	conference [chat]	Internet Relay Chat
532	netnews	Netnews newsgroup service
533/udp	netwall	Netwall for emergency broadcasts
540/tcp	uucp [uucpd]	UNIX-to-UNIX copy services
543/tcp	klogin	Kerberos version 5 (v5) remote login
544/tcp	kshell	Kerberos version 5 (v5) remote shell
548	afpovertcp	Appletalk Filing Protocol (AFP) over Transmission Control Protocol (TCP)
556	remotefs [rfs_server, rfs]	Brunhoff's Remote Filesystem (RFS)

Table C.3, “Registered Ports” lists ports submitted by the network and software community to the IANA for formal registration into the port number list.

Table C.3. Registered Ports

Port # / Layer	Name	Comment
1080	socks	SOCKS network application proxy services
1236	bvcontrol [rmtcfg]	Remote configuration server for Gracilis Packeten network switches [a]
1300	h323hostcallsc	H.323 telecommunication Host Call Secure
1433	ms-sql-s	Microsoft SQL Server
1434	ms-sql-m	Microsoft SQL Monitor
1494	ica	Citrix ICA Client
1512	wins	Microsoft Windows Internet Name Server
1524	ingreslock	Ingres Database Management System (DBMS) lock services
1525	prospero-np	Prospero non-privileged
1645	datametrics [old-radius]	Datametrics / old radius entry
1646	sa-msg-port [oldradacct]	sa-msg-port / old radacct entry
1649	kermit	Kermit file transfer and management service
1701	l2tp [l2f]	Layer 2 Tunneling Protocol (LT2P) / Layer 2 Forwarding (L2F)
1718	h323gatedisc	H.323 telecommunication Gatekeeper Discovery
1719	h323gatestat	H.323 telecommunication Gatekeeper Status
1720	h323hostcall	H.323 telecommunication Host Call setup
1758	tftp-mcast	Trivial FTP Multicast
1759/udp	mtftp	Multicast Trivial FTP (MTFTP)
1789	hello	Hello router communication protocol
1812	radius	Radius dial-up authentication and accounting services
1813	radius-acct	Radius Accounting
1911	mtp	Starlight Networks Multimedia Transport Protocol (MTP)
1985	hsrp	Cisco Hot Standby Router Protocol
1986	licensedaemon	Cisco License Management Daemon
1997	gdp-port	Cisco Gateway Discovery Protocol (GDP)
2049	nfs [nfsd]	Network File System (NFS)
2102	zephyr-srv	Zephyr distributed messaging Server
2103	zephyr-clt	Zephyr client
2104	zephyr-hm	Zephyr host manager
2401	cvspserver	Concurrent Versions System (CVS) client/server operations
2430/tcp	venus	Venus cache manager for Coda file system (codacon port)
2430/udp	venus	Venus cache manager for Coda file system (callback/wbc interface)
2431/tcp	venus-se	Venus Transmission Control Protocol (TCP) side effects
2431/udp	venus-se	Venus User Datagram Protocol (UDP) side effects
2432/udp	codasrv	Coda file system server port
2433/tcp	codasrv-se	Coda file system TCP side effects
2433/udp	codasrv-se	Coda file system UDP SFTP side effect

2600	hpstgmgr [zebrasrv]	Zebra routing [b]
2601	discp-client [zebra]	discp client; Zebra integrated shell
2602	discp-server [ripd]	discp server; Routing Information Protocol daemon (ripd)
2603	servicemeter [ripngd]	Service Meter; RIP daemon for IPv6
2604	nsc-ccs [ospfd]	NSC CCS; Open Shortest Path First daemon (ospfd)
2605	nsc-posa	NSC POSA; Border Gateway Protocol daemon (bgpd)
2606	netmon [ospf6d]	Dell Netmon; OSPF for IPv6 daemon (ospf6d)
2809	corbaloc	Common Object Request Broker Architecture (CORBA) naming service locator
3130	icpv2	Internet Cache Protocol version 2 (v2); used by Squid proxy caching server
3306	mysql	MySQL database service
3346	trnsprntproxy	Transparent proxy
4011	pxe	Pre-execution Environment (PXE) service
4321	rwhois	Remote Whois (rwhois) service
4444	krb524	Kerberos version 5 (v5) to version 4 (v4) ticket translator
5002	rfe	Radio Free Ethernet (RFE) audio broadcasting system
5308	cfengine	Configuration engine (Cfengine)
5999	cvsup [CVSup]	CVSup file transfer and update tool
6000/tcp	x11 [X]	X Window System services
7000	afs3-fileserver	Andrew File System (AFS) file server
7001	afs3-callback	AFS port for callbacks to cache manager
7002	afs3-prserver	AFS user and group database
7003	afs3-vlserver	AFS volume location database
7004	afs3-kaserver	AFS Kerberos authentication service
7005	afs3-volser	AFS volume management server
7006	afs3-errors	AFS error interpretation service
7007	afs3-bos	AFS basic overseer process
7008	afs3-update	AFS server-to-server updater
7009	afs3-rmtsys	AFS remote cache manager service
9876	sd	Session Director for IP multicast conferencing
10080	amanda	Advanced Maryland Automatic Network Disk Archiver (Amanda) backup services
11371	pgpkeyserver	Pretty Good Privacy (PGP) / GNU Privacy Guard (GPG) public keyserver
11720	h323callsigalt	H.323 Call Signal Alternate
13720	bprd	Veritas NetBackup Request Daemon (bprd)
13721	bpdbm	Veritas NetBackup Database Manager (bpdbm)
13722	bpjava-msvc	Veritas NetBackup Java / Microsoft Visual C++ (MSVC) protocol
13724	vnetd	Veritas network utility
13782	bpcd	Veritas NetBackup

13783	vopied	Veritas VOPIE authentication daemon
22273	wnn6 [wnn4]	Kana/Kanji conversion system [c]
26000	quake	Quake (and related) multi-player game servers
26208	wnn6-ds	Wnn6 Kana/Kanji server
33434	traceroute	Traceroute network tracking tool

[a] Comment from **/etc/services**: "Port 1236 is registered as `bvcontrol', but is also used by the Gracilis Packeten remote config server. The official name is listed as the primary name, with the unregistered name as an alias."
[b] Comment from **/etc/services**: "Ports numbered 2600 through 2606 are used by the zebra package without being registered. The primary names are the registered names, and the unregistered names used by zebra are listed as aliases."
[c] Comment from **/etc/services**: "This port is registered as wnn6, but also used under the unregistered name 'wnn4' by the FreeWnn package."

Table C.4, “Datagram Deliver Protocol Ports” is a listing of ports related to the Datagram Delivery Protocol (DDP) used on AppleTalk networks.

Table C.4. Datagram Deliver Protocol Ports

Port # / Layer	Name	Comment
1/ddp	rtmp	Routing Table Management Protocol
2/ddp	nbp	Name Binding Protocol
4/ddp	echo	AppleTalk Echo Protocol
6/ddp	zip	Zone Information Protocol

Table C.5, “Kerberos (Project Athena/MIT) Ports” is a listing of ports related to the Kerberos network authentication protocol. Where noted, v5 refers to the Kerberos version 5 protocol. Note that these ports are not registered with the IANA.

Table C.5. Kerberos (Project Athena/MIT) Ports

Port # / Layer	Name	Comment
751	kerberos_master	Kerberos authentication
752	passwd_server	Kerberos Password (kpasswd) server
754	krb5_prop	Kerberos v5 slave propagation
760	krbupdate [kreg]	Kerberos registration
1109	kpop	Kerberos Post Office Protocol (KPOP)
2053	knetd	Kerberos de-multiplexor
2105	eklogin	Kerberos v5 encrypted remote login (rlogin)

Table C.6, “Unregistered Ports” is a listing of unregistered ports that are used by services and protocols that may be installed on your Red Hat Enterprise Linux system, or that is necessary for communication between Red Hat Enterprise Linux and other operating systems.

Table C.6. Unregistered Ports

Port # / Layer	Name	Comment
15/tcp	netstat	Network Status (netstat)
98/tcp	linuxconf	Linuxconf Linux administration tool
106	poppassd	Post Office Protocol password change daemon (POPPASSD)
465/tcp	smtps	Simple Mail Transfer Protocol over Secure Sockets Layer (SMTPS)
616/tcp	gii	Gated (routing daemon) Interactive Interface
808	omirr [omirrd]	Online Mirror (Omirr) file mirroring services
871/tcp	supfileserv	Software Upgrade Protocol (SUP) server
901/tcp	swat	Samba Web Administration Tool (SWAT)
953	rndc	Berkeley Internet Name Domain version 9 (BIND 9) remote configuration tool
1127/tcp	supfiledbg	Software Upgrade Protocol (SUP) debugging
1178/tcp	skkserv	Simple Kana to Kanji (SKK) Japanese input server
1313/tcp	xtel	French Minitel text information system
1529/tcp	support [prmsd, gnatsd]	GNATS bug tracking system
2003/tcp	cfinger	GNU finger
2150	ninstall	Network Installation Service
2988	afbackup	afbackup client-server backup system
3128/tcp	squid	Squid Web proxy cache
3455	prsvp	RSVP port
5432	postgres	PostgreSQL database
4557/tcp	fax	FAX transmission service (old service)
4559/tcp	hylafax	HylaFAX client-server protocol (new service)
5232	sgi-dgl	SGI Distributed Graphics Library
5354	noclog	NOCOL network operation center logging daemon (noclogd)
5355	hostmon	NOCOL network operation center host monitoring
5680/tcp	canna	Canna Japanese character input interface
6010/tcp	x11-ssh-offset	Secure Shell (SSH) X11 forwarding offset
6667	ircd	Internet Relay Chat daemon (ircd)
7100/tcp	xfs	X Font Server (XFS)
7666/tcp	tircproxy	Tircproxy IRC proxy service
8008	http-alt	Hypertext Tranfer Protocol (HTTP) alternate
8080	webcache	World Wide Web (WWW) caching service
8081	tproxy	Transparent Proxy
9100/tcp	jetdirect [laserjet, hplj]	Hewlett-Packard (HP) JetDirect network printing service
9359	mandelspawn [mandelbrot]	Parallel mandelbrot spawning program for the X Window System
10081	kamanda	Amanda backup service over Kerberos
10082/tcp	amandaidx	Amanda index server

10083/tcp	amidxtape	Amanda tape server
20011	isdnlog	Integrated Services Digital Network (ISDN) logging system
20012	vboxd	ISDN voice box daemon (vboxd)
22305/tcp	wnn4_Kr	kWnn Korean input system
22289/tcp	wnn4_Cn	cWnn Chinese input system
22321/tcp	wnn4_Tw	tWnn Chinese input system (Taiwan)
24554	binkp	Binkley TCP/IP Fidonet mailer daemon
27374	asp	Address Search Protocol
60177	tfido	Ifmail FidoNet compatible mailer service
60179	fido	FidoNet electronic mail and news network

Revision History

Revision 2-9.402	**Fri Oct 25 2013**	**Rüdiger Landmann**
Rebuild with Publican 4.0.0		
Revision 2-9	**2012-07-18**	**Anthony Towns**
Rebuild for Publican 3.0		
Revision 1-0	**Wed Sep 17 2008**	**Don Domingo**
migrated to new automated build system		

Index

Symbols

A

B

C

D

E

F

S