Red Hat Enterprise Linux 6
Security Guide

A Guide to Securing Red Hat Enterprise Linux
Edition 4

Red Hat Engineering Content Services

Red Hat Enterprise Linux 6 Security Guide

A Guide to Securing Red Hat Enterprise Linux
Edition 4

Red Hat Engineering Content Services

Legal Notice

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

Preface

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 submenu 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. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product **Red Hat Enterprise Linux 6**.

When submitting a bug report, be sure to mention the manual's identifier: doc-*Security_Guide* and version number: **`6.5`**.

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

Chapter 1. Security Overview

Due to the increased reliance on powerful, networked computers to help run businesses and keep track of our personal information, entire 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 their organization. Because most organizations are increasingly dynamic in nature, their workers are accessing critical company IT resources locally and remotely, hence the need for secure computing environments has become more pronounced.

Unfortunately, many organizations (as well as individual users) regard security as more of an afterthought, a process that is overlooked in favor of increased power, productivity, convenience, ease of use, and budgetary concerns. Proper security implementation is often enacted postmortem — *after* an unauthorized intrusion has already occurred. Taking the correct measures prior to connecting a site to an untrusted network, such as the Internet, is an effective means of thwarting many attempts at intrusion.

Note

This document makes several references to files in the **`/lib`** directory. When using 64-bit systems, some of the files mentioned may instead be located in **`/lib64`**.

1.1. Introduction to Security

1.1.1. 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 critical 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), return on investment (ROI), and quality of service (QoS). Using these metrics, industries can calculate aspects such as data integrity and high-availability (HA) as part of their planning and process management costs. In some industries, such as electronic commerce, the availability and trustworthiness of data can mean the difference between success and failure.

1.1.1.1. How did Computer Security come about?

Information security has evolved over the years due to the increasing reliance on public networks not to disclose personal, financial, and other restricted information. There are numerous instances such as the Mitnick [1] and the Vladimir Levin [2] cases that prompted organizations across all industries to re-think the way they handle information, including its 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 transactions, 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 (IP) 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.

1.1.1.2. 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 attackers using specially created, widely available programs that scanned vulnerable network servers, installed client applications called *Trojans* on the servers. Then they 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, regardless of the purpose of the packets or where they were sent to.*will possibly be removed*

In 2007, a data breach exploiting the widely-known weaknesses of the Wired Equivalent Privacy (WEP) wireless encryption protocol resulted in the theft from a global financial institution of over 45 million credit card numbers.

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 who make up the organization) conducts business is paramount to implementing a proper security plan.

1.1.1.3. 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.

1.1.2. SELinux

Red Hat Enterprise Linux includes an enhancement to the Linux kernel called SELinux, which implements a Mandatory Access Control (MAC) architecture that provides a fine-grained level of control over files, processes, users and applications in the system. Detailed discussion of SELinux is beyond the scope of this document; however, for more information on SELinux and its use in Red Hat Enterprise Linux, see the *Red Hat Enterprise Linux SELinux User Guide.* For more information on configuring and running services that are protected by SELinux, see the *SELinux Managing Confined Services Guide*. Other available resources for SELinux are listed in Chapter 9, *References*.

1.1.3. 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.

1.1.3.1. 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)

1.1.3.2. 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

1.1.3.3. Administrative Controls

Administrative controls define the human factors of security. They involve all levels of personnel within an organization and determine 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

1.1.4. Conclusion

Now that you have learned about the origins, reasons, and aspects of security, you will find it easier to 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.

1.2. Vulnerability Assessment

Given time, resources, and motivation, an attacker can break into nearly any system. All of the security procedures and technologies currently available cannot guarantee that any systems are completely 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.

1.2.1. Thinking Like the Enemy

Suppose that you administer an enterprise network. Such networks commonly comprise operating systems, applications, servers, network monitors, firewalls, intrusion detection systems, and more. Now imagine trying to keep current with each of those. 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 an attacker 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 an attacker 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 1.1.1.3, “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.

1.2.2. 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 attacker's viewpoint. You see what an attacker 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 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 the two types of vulnerability assessments. Being internal to your company gives you more privileges than an outsider. In most organizations, security is configured to keep intruders out. Very little is done to secure the internals of the organization (such as departmental firewalls, user-level access controls, and authentication procedures for internal resources). Typically, there are many more resources when looking around inside as most systems are internal to a company. Once you are outside the company, your status is untrusted. 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 undertaken to check 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 on the system shows an overview, which can turn up false positives and false negatives. A false positive is a result, where the tool finds vulnerabilities which in reality do not exist. A false negative is when it omits actual vulnerabilities.

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 false positives, or, even worse, false negatives.

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

Do not attempt to exploit vulnerabilities on production systems. Doing so can have adverse effects on 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 attackers 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.

1.2.2.1. 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.owasp.org/ *The Open Web Application Security Project*

1.2.3. 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. Read documentation that comes with the tool (for example, in a README file or a manual page). For more information, search articles, step-by-step guides, or even mailing lists specific to a tool on the Internet.

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

1.2.3.1. Scanning Hosts with Nmap

Nmap is a popular tool 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 manual page is included that provides detailed descriptions 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 restricting unused services.

To install Nmap, run the **`yum install nmap`** command as the root user.

1.2.3.1.1. 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 <hostname>
```

For example, to scan a machine with hostname **foo.example.com**, type the following at a shell prompt:

```
~]$ nmap foo.example.com
```

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

```
Interesting ports on foo.example.com:
Not shown: 1710 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
53/tcp  open   domain
80/tcp  open   http
113/tcp closed auth
```

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/

1.2.3.2. 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

> The Nessus client and server software requires a subscription to use. 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/

1.2.3.3. 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.

More information about Nikto can be found at the following URL:

http://cirt.net/nikto2

1.2.3.4. 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. Concepts such as *war walking* and *wardriving*, which involves scanning the perimeter of your enterprise's physical structures for wireless network vulnerabilities, are some concepts that you should investigate and, if needed, incorporate into your assessments. Imagination and exposure are the only limits of planning and conducting vulnerability assessments.

1.3. 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. However, before detailing these issues, the terminology used when identifying an attacker must be defined.

1.3.1. 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.

1.3.1.1. Hats of Hackers

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 *gray 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 gray hat hacker can be thought of as a white hat hacker who wears a black hat at times to accomplish his own agenda.

Gray 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.

1.3.2. Threats to Network Security

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

1.3.2.1. 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.

1.3.2.1.1. 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.

1.3.2.1.2. 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.

1.3.3. 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 attacker to steal or manipulate at will. The following sections detail some of the main issues.

1.3.3.1. Unused Services and Open Ports

A full installation of Red Hat Enterprise Linux 7 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 attackers. Refer To Section 2.2, “Server Security” for information on closing ports and disabling unused services.

1.3.3.2. Inattentive Administration

Administrators who fail to patch their systems are one of the greatest threats to server security. According to the *SysAdmin, Audit, Network, 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. 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 attacker 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.

1.3.3.3. 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, an attacker 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 an attacker 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 an attacker 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 ASCII or DBM (ASCII-derived) database. An attacker 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 Section 2.2, “Server Security” for more information about setting up services in a safe manner.

1.3.4. 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 attackers. 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.

1.3.4.1. 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 2.1.3, “Password Security”.

1.3.4.2. 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 user names 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.

Section 2.1, “Workstation Security” discusses in more detail what steps administrators and home users should take to limit the vulnerability of computer workstations.

1.4. Common Exploits and Attacks

Table 1.1, “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 1.1. 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, but some services that run on Linux can contain default administrator passwords as well (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 those that bundle services (such as UNIX and Windows.) Administrators sometimes create privileged user accounts in a rush and leave the password null, creating a perfect entry point 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 that it contains.	Most common in wireless access points and preconfigured secure server appliances.
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 sequence numbers to coordinate a connection to target systems, but several tools are available to assist attackers 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 attacker has used an active attack (such as IP spoofing or man-in-the-middle) to compromise a system on the LAN. Preventative measures include services

		with cryptographic key exchange, one-time passwords, or encrypted authentication to prevent password snooping; strong encryption during transmission 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 arbitrary 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.	Source packets are usually forged (as well as rebroadcast), making investigation as to the true source of the attack difficult. Advances in ingress filtering (IETF rfc2267) using **`iptables`** and Network Intrusion Detection Systems such as **`snort`** assist administrators in tracking down and preventing distributed DoS attacks.

1.5. 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 a Red Hat Enterprise Linux distribution that is currently supported, Red Hat is committed to releasing updated packages that fix the vulnerability as soon as is 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 and tested and released as an errata update. However, if an announcement does not include a patch, a developer first 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 affected packages as soon as possible to minimize the amount of time the system is potentially vulnerable.

1.5.1. 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 and to check the signature of the package to verify its integrity.

Note

Red Hat Enterprise Linux includes a convenient panel icon that displays visible alerts when there is an update available.

1.5.2. Verifying Signed Packages

All Red Hat Enterprise Linux packages are signed with the Red Hat *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) locks the package while the public key unlocks and verifies the package. If the public key distributed by Red Hat Enterprise Linux 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 6 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 a Red Hat installation CD-ROM or DVD.

Assuming the disc is mounted in **`/mnt/cdrom`**, use the following command as the root user to import it into the *keyring* (a database of trusted keys on the system):

```
~]# rpm --import /mnt/cdrom/RPM-GPG-KEY
```

Now, the Red Hat GPG key is located in the **`/etc/pki/rpm-gpg/`** directory.

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

```
~]# rpm -qa gpg-pubkey*
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
Name        : gpg-pubkey                   Relocations: (not relocatable)
Version     : 2fa658e0                          Vendor: (none)
Release     : 45700c69                     Build Date: Fri 07 Oct 2011 02:04:51
PM CEST
Install Date: Fri 07 Oct 2011 02:04:51 PM CEST      Build Host: localhost
Group       : Public Keys                  Source RPM: (none)
[output truncated]
```

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 original source of the packages. To verify all the downloaded packages at once, issue the following command:

```
~]# rpm -K /root/updates/*.rpm
alsa-lib-1.0.22-3.el6.x86_64.rpm: rsa sha1 (md5) pgp md5 OK
alsa-utils-1.0.21-3.el6.x86_64.rpm: rsa sha1 (md5) pgp md5 OK
aspell-0.60.6-12.el6.x86_64.rpm: rsa sha1 (md5) pgp md5 OK
```

For each package, if the GPG key verifies successfully, the command returns **`gpg OK`**. If it does not, 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 verification 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.

Alternatively, you may use the Yum utility to verify signed packages. Yum provides secure package management by enabling GPG signature verification on GPG-signed packages to be turned on for all package repositories (that is, package sources), or for individual repositories. When signature verification is enabled, Yum will refuse to install any packages not GPG-signed with the correct key for that repository. This means that you can trust that the RPM packages you download and install on your system are from a trusted source, such as Red Hat, and were not modified during transfer.

In order to have automatic GPG signature verification enabled when installing or updating packages via Yum, ensure you have the following option defined under the **`[main]`** section of your **`/etc/yum.conf`** file:

```
gpgcheck=1
```

1.5.3. Installing Signed Packages

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

```
rpm -Uvh <package>…
```

For example, to install all packages in a new directory, called **updates/**, under the **/tmp/** directory, run:

```
~]# rpm -Uvh /tmp/updates/*.rpm
Preparing...                ################################################## [100%]
   1:alsa-lib               ################################################## [ 33%]
   2:alsa-utils             ################################################## [ 67%]
   3:aspell                 ################################################## [100%]
```

For kernel packages, as root use the command in the following form:

```
rpm -ivh <kernel-package>
```

For example, to install *kernel-2.6.32-220.el6.x86_64.rpm*, type the following at a shell prompt:

```
~]# rpm -ivh /tmp/updates/kernel-2.6.32-220.el6.x86_64.rpm
Preparing...                ################################################## [100%]
   1:kernel                 ################################################## [100%]
```

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>
```

For instance, to remove *kernel-2.6.32-206.el6.x86_64*, type:

```
~]# rpm -e kernel-2.6.32-206.el6.x86_64
```

Alternatively, to install packages with Yum, run, as root, the following command:

```
~]# yum install kernel-2.6.32-220.el6.x86_64.rpm
```

To install local packages with Yum, run, as root, the following command:

```
~]# yum localinstall /root/updates/emacs-23.1-21.el6_2.3.x86_64.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 1.5.4, “Applying the Changes” for general instructions about applying the changes made by an errata update.

1.5.4. Applying the Changes

After downloading and installing security errata and updates, 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 required, or available to the system administrator.

Applications

User-space applications are any programs that 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:

```
lsof <path>
```

For example, to determine which running applications link against the **libwrap.so** library, type:

```
~]# lsof /lib64/libwrap.so*
COMMAND     PID      USER   FD    TYPE DEVICE SIZE/OFF   NODE NAME
sshd      13600 root mem     REG  253,0    43256 400501
/lib64/libwrap.so.0.7.6
sshd      13603 juan mem     REG  253,0    43256 400501
/lib64/libwrap.so.0.7.6
gnome-set 14898 juan mem     REG  253,0    43256 400501
/lib64/libwrap.so.0.7.6
metacity  14925 juan mem     REG  253,0    43256 400501
/lib64/libwrap.so.0.7.6
[output truncated]
```

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:

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

Replace ***<service-name>*** with the name of the service, such as **sshd**.

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** or **pgrep** 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:

```
~]# pgrep -l imap
1439 imapd
1788 imapd
1793 imapd
```

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

```
kill <PID>
```

If this fails to terminate the session, use the following command instead:

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

In the previous examples, replace ***<PID>*** with the process identification number (found in the second column of the **`pgrep -l`** command) for an IMAP session.

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

```
~]# killall imapd
```

[1] http://law.jrank.org/pages/3791/Kevin-Mitnick-Case-1999.html

[2] http://www.livinginternet.com/i/ia_hackers_levin.htm

Chapter 2. Securing Your Network

2.1. 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. A computer network is only as secure as its weakest node.

2.1.1. 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?

2.1.2. 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 obtaining root privileges through single user mode. The security measures you should take to protect against such attacks depends both on the sensitivity of the information on the workstation and the location of the machine.

For example, if a machine is used in a trade show and contains no sensitive information, then 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.

If the workstation is located in a place where only authorized or trusted people have access, however, then securing the BIOS or the boot loader may not be necessary.

2.1.2.1. BIOS Passwords

The two primary reasons for password protecting the BIOS of a computer are [3]:

1. *Preventing Changes to BIOS Settings* — If an intruder has access to the BIOS, they can set it to boot from a CD-ROM or a flash drive. This makes it possible for an intruder 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.

2.1.2.1.1. 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.

2.1.2.2. Boot Loader Passwords

The primary reasons for password protecting a Linux boot loader are as follows:

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.

   > **Warning**
   >
   > Protecting access to single user mode with a password by editing the ***SINGLE*** parameter in the **/etc/sysconfig/init** file is not recommended. An attacker can bypass the password by specifying a custom initial command (using the ***init=*** parameter) on the kernel command line in GRUB. It is recommended to password-protect the GRUB boot loader as specified in Section 2.1.2.2.1, “Password Protecting GRUB”.

2. *Preventing Access to the GRUB Console* — If the machine uses GRUB as its boot loader, an attacker can use the GRUB editor interface to change its configuration or to gather information using the **cat** command.
3. *Preventing Access to Insecure Operating Systems* — If it is a dual-boot system, an attacker can select an operating system at boot time (for example, DOS), which ignores access controls and file permissions.

Red Hat Enterprise Linux 6 ships with the GRUB boot loader on the x86 platform. For a detailed look at GRUB, see the *Red Hat Enterprise Linux Installation Guidei*.

2.1.2.2.1. Password Protecting GRUB

You can configure GRUB to address the first two issues listed in Section 2.1.2.2, “Boot Loader Passwords” by adding a password directive to its configuration file. To do this, first choose a strong password, open a shell, log in as root, and then type the following command:

```
/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`** [4].

The next time the system boots, the GRUB menu prevents 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 an insecure 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 operating system that you want to secure, and add a line with the **`lock`** directive immediately 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>
```

2.1.2.2.2. Disabling Interactive Startup

Pressing the **`I`** key at the beginning of the boot sequence allows you to start up your system interactively. During an interactive startup, the system prompts you to start up each service one by one. However, this may allow an attacker who gains physical access to your system to disable the security-related services and gain access to the system.

To prevent users from starting up the system interactively, as root, disable the ***`PROMPT`*** parameter in the **`/etc/sysconfig/init`** file:

```
PROMPT=no
```

2.1.3. Password Security

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

For security purposes, the installation program configures the system to use *Secure Hash Algorithm 512* (*SHA512*) and shadow passwords. It is highly recommended that you do not alter these settings.

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.

In addition to format and storage considerations 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.

2.1.3.1. Creating Strong Passwords

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

- *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:

  - svjetionik23
  - bienvenido1
  - 1AufWiedersehen
- *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* — Avoid using any personal information in your passwords. 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.

The following guidelines will help you to create a strong password:

- *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 that meets all of the criteria for good passwords while avoiding the traits of a bad one. Fortunately, there are some steps you can take to generate an easily-remembered, secure password.

2.1.3.1.1. Secure Password Creation Methodology

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

- Think of an easily-remembered 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.

2.1.3.2. Creating User Passwords Within an Organization

If an organization has a large number of users, 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.

2.1.3.2.1. 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 Modules* (*PAM*) aware and therefore checks to see if the password is too short or otherwise easy to crack. This check is performed using the **pam_cracklib.so** PAM module. In Red Hat Enterprise Linux, the **pam_cracklib** PAM module can be used to check a password's strength against a set of rules. It can be stacked alongside other PAM modules in the **password** component of the**/etc/pam.d/passwd** file to configure a custom set of rules for user login. The **pam_cracklib**'s routine consists of two parts: it checks whether the password provided is found in a dictionary, and, if that's not the case, it continues with a number of additional checks. For a complete list of these checks, refer to the **pam_cracklib(8)** manual page.

> **Example 2.1. Configuring password strength-checking with pam_cracklib**
>
> To require a password with a minimum length of 8 characters, including all four classes of characters, add the following line to the **password** section of the **/etc/pam.d/passwd** file:
>
> ```
> password   required     pam_cracklib.so retry=3 minlen=8 minclass=4
> ```
>
> To set a password strength-check for consecutive or repetitive characters, add the following line to the **password** section of the **/etc/pam.d/passwd** file:
>
> ```
> password   required     pam_cracklib.so retry=3 maxsequence=3 maxrepeat=3
> ```
>
> In this example, the password entered cannot contain more than 3 consecutive characters, such as "abcd" or "1234". Additionally, the number of identical consecutive characters is limited to 3.

Note

As these checks are not performed for the root user, he can set any password for a regular user, despite the warning messages.

Since PAM is customizable, it is possible to add more 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://uw714doc.sco.com/en/SEC_pam/pam-6.html. For more information about PAM, refer to the *Managing Single Sign-On and Smart Cards* guide.

The password check that is performed at the time of their creation does not discover bad passwords as effectively as running a password cracking program against the passwords.

Many password cracking programs are available 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:

- ***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**.
- ***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.

2.1.3.2.2. Passphrases

Passphrases and passwords are the cornerstone to security in most of today's systems. Unfortunately, techniques such as biometrics and two-factor authentication have not yet become mainstream in many systems. If passwords are going to be used to secure a system, then the use of passphrases should be considered. Passphrases are longer than passwords and provide better protection than a password even when implemented with non-standard characters such as numbers and symbols.

2.1.3.2.3. 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 specified period (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.

> **Important**
>
> Shadow passwords must be enabled to use the **`chage`** command. For more information, see the *Red Hat Enterprise Linux 6 Deployment Guide*.

The **`-M`** option of the **`chage`** command specifies the maximum number of days the password is valid. For example, to set a user's password to expire in 90 days, use 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).

For more information on the options available with the **`chage`** command, refer to the table below.

Table 2.1. chage command line options

Option	Description
`-d` ***days***	Specifies the number of days since January 1, 1970 the password was changed.
`-E` ***date***	Specifies the date on which the account is locked, in the format YYYY-MM-DD. Instead of the date, the number of days since January 1, 1970 can also be used.
`-I` ***days***	Specifies the number of inactive days after the password expiration before locking the account. If the value is **`0`**, the account is not locked after the password expires.
`-l`	Lists current account aging settings.
`-m` ***days***	Specify the minimum number of days after which the user must change passwords. If the value is **`0`**, the password does not expire.
`-M` ***days***	Specify the maximum number of days for which the password is valid. When the number of days specified by this option plus the number of days specified with the **`-d`** option is less than the current day, the user must change passwords before using the account.
`-W` ***days***	Specifies the number of days before the password expiration date to warn the user.

You can also use the **`chage`** command in interactive mode to modify multiple password aging and account details. Use the following command to enter interactive mode:

```
chage <username>
```

The following is a sample interactive session using this command:

```
~]# chage juan
Changing the aging information for juan
Enter the new value, or press ENTER for the default
Minimum Password Age [0]: 10
Maximum Password Age [99999]: 90
Last Password Change (YYYY-MM-DD) [2006-08-18]:
Password Expiration Warning [7]:
Password Inactive [-1]:
Account Expiration Date (YYYY-MM-DD) [1969-12-31]:
```

You can configure a password to expire the first time a user logs in. This forces users to change passwords immediately.

1. Set up an initial password. There are two common approaches to this step: you can either assign a default password, or you can use a null password.

   To assign a default password, type the following at a shell prompt as **`root`**:

```
passwd username
```

To assign a null password instead, use the following command:

```
passwd -d username
```

Avoid using null passwords whenever possible

Using a null password, while convenient, is a highly insecure practice, as any third party can log in first and access the system using the insecure username. Always make sure that the user is ready to log in before unlocking an account with a null password.

2. Force immediate password expiration by running the following command as **`root`**:

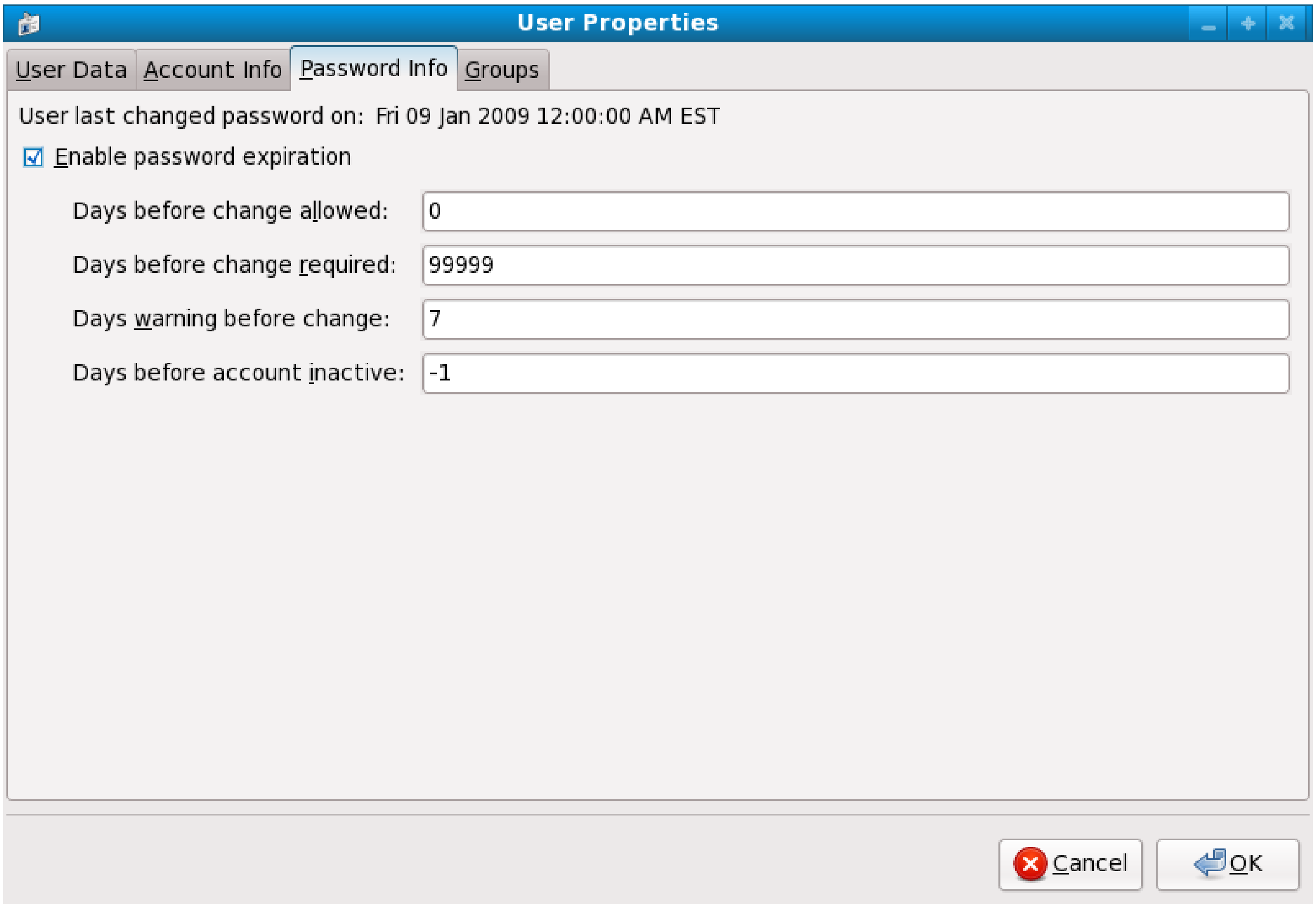
   ```
   chage -d 0 username
   ```

   This command sets the value for the date the password was last changed to the epoch (January 1, 1970). This value forces immediate password expiration no matter what password aging policy, if any, is in place.

Upon the initial log in, the user is now prompted for a new password.

You can also use the graphical **User Manager** application to create password aging policies, as follows. Note: you need Administrator privileges to perform this procedure.

1. Click the **System** menu on the Panel, point to **Administration** and then click **Users and Groups** to display the User Manager. Alternatively, type the command **`system-config-users`** at a shell prompt.
2. Click the **`Users`** tab, and select the required user in the list of users.
3. Click **`Properties`** on the toolbar to display the User Properties dialog box (or choose **Properties** on the **File** menu).
4. Click the **`Password Info`** tab, and select the check box for **`Enable password expiration`**.
5. Enter the required value in the **`Days before change required`** field, and click **`OK`**.



User last changed password on: Fri 09 Jan 2009 12:00:00 AM EST

☑ Enable password expiration

Days before change allowed: 0

Days before change required: 99999

Days warning before change: 7

Days before account inactive: -1

Figure 2.1. Specifying password aging options

screenshot needs to be updated

2.1.4. Locking Inactive Accounts

The **pam_lastlog** PAM module is used to lock out users who have not logged in recently enough, or to display information about the last login attempt of a user. The module does not perform a check on the root account, so it is never locked out.

The **lastlog** command displays the last login of the user, as opposed to the **last** command, which displays all current and previous login sessions. The commands read respectively from the **/var/log/lastlog** and **/var/log/wtmp** files where the data is stored in binary format.

- To display the number of failed login attempts prior to the last successful login of a user, add, as root, the following line to the **session** section in the **/etc/pam.d/login** file:

```
session     optional      pam_lastlog.so silent noupdate showfailed
```

Account locking due to inactivity can be configured to work for the console, GUI, or both:

- To lock out an account after 10 days of inactivity, add, as root, the following line to the **auth** section of the **/etc/pam.d/login** file:

```
auth   required   pam_lastlog.so inactive=10
```

- To lock out an account for the GNOME desktop environment, add, as root, the following line to the

auth section of the **/etc/pam.d/gdm** file:

```
auth  required  pam_lastlog.so inactive=10
```

> **Note**
>
> Note that for other desktop environments, the respective files of those environments should be edited.

2.1.5. Customizing Access Control

The **pam_access** PAM module allows an administrator to customize access control based on login names, host or domain names, or IP addresses. By default, the module reads the access rules from the **/etc/security/access.conf** file if no other is specified. For a complete description of the format of these rules, refer to the **access.conf(5)** manual page. By default, in Red Hat Enterprise Linux, **pam_access** is included in the **/etc/pam.d/crond** and **/etc/pam.d/atd** files.

To deny the user john from accessing system from the console and the graphic desktop environment, follow these steps:

1. Include the following line in the **account** section of both **/etc/pam.d/login** and **/etc/pam.d/gdm-*** files:

   ```
   account     required     pam_access.so
   ```

2. Specify the following rule in the **/etc/security/access.conf** file:

   ```
   - : john : ALL
   ```

   This rule prohibits all logins from user john from any location.

To grant access to all users attempting to log in via SSH except the user john from the 1.2.3.4 IP address, follow these steps:

1. Include the following line in the **account** section of **/etc/pam.d/sshd**:

   ```
   account     required     pam_access.so
   ```

2. Specify the following rule in the /etc/security/access.conf file:

   ```
   + : ALL EXCEPT john : 1.2.3.4
   ```

In order to limit access from other services, the **pam_access** module should be required in the respective file in the **/etc/pam.d** directory.

It is possible to call the **pam_access** module for all services that call the system wide PAM configuration files (***-auth** files in the **/etc/pam.d** directory) using the following command:

```
authconfig --enablepamaccess --update
```

Alternatively, you can enable the **pam_access** module using the Authentication Configuration utility. To start this utility, select **System** → **Administration** → **Authentication** from the top menu. From the

Advanced Options tab, check the "enable local access control option". This will add the **`pam_access`** module to the systemwide PAM configuration.

2.1.6. Time-based Restriction of Access

The **`pam_time`** PAM module is used to restrict access during a certain time of the day. It can also be configured to control access based on specific days of a week, user name, usage of a system service, and more. By default, the module reads the access rules from the **`/etc/security/time.conf`** file. For a complete description of the format of these rules, refer to the **`time.conf(5)`** manual page.

To restrict all users except the root user from logging in from 05:30 PM to 08:00 AM on Monday till Friday and Saturday and Sunday, follow these steps:

1. Include the following line in the account section of the **`/etc/pam.d/login`** file:

   ```
   account     required     pam_time.so
   ```

2. Specify the following rule in the **`/etc/security/time.conf`** file:

   ```
   login ; ALL ; !root ; tty* ; !Wk1730-0800
   ```

To allow user john to use the SSH service during working hours and working days only (starting with Monday), follow these steps:

1. Add the following line to the **`/etc/pam.d/sshd file:`**

   ```
   account     required     pam_time.so
   ```

2. Specify the following rule in the **`/etc/security/time.conf`** file:

   ```
   sshd ; john ; tty* ; Wk0800-1730
   ```

> **Note**
>
> For these configurations to be applied to the desktop environment, the **`pam_time`** module should be included in the corresponding files in the **`/etc/pam.d`** directory.

2.1.7. Applying Account Limits

The **`pam_limits`** PAM module is used to:

- apply limits to user login sessions, such as maximum simultaneous login sessions per user,
- specify limits to be set by the **ulimit** utility,
- and specify priority to be set by the **nice** utility.

By default, the rules are read from the**`/etc/security/limits.conf`** file. For a complete description of the format of these rules, refer to the **`limits.conf(5)`** manual page. Additionally, you can create individual configuration files in the **`/etc/security/limits.d`** directory specifically for certain applications or services. By default, the **`pam_limits`** module is included in a number of files in the**`/etc/pam.d/`** directory. A default limit of user processes is defined in the **`/etc/security/limits.d/90-nproc.conf`** file to prevent malicious denial of service attacks, such as fork bombs. To change the default limit of user processes to 50, change the value in the

/etc/security/limits.d/90-nproc.conf, following the format in the file:

```
* soft nproc 50
```

> **Example 2.2. Specifying a maximum number of logins per user**
>
> 1. To set a maximum number of simultaneous logins for each user in a group called **office**, specify the following rule in the **/etc/security/limits.conf** file:
>
>    ```
>    @office - maxlogins 4
>    ```
>
> 2. The following line should be present by default in **/etc/pam.d/system-auth**. If not, add it manually.
>
>    ```
>    session  required  pam_limits.so
>    ```

2.1.8. 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 an **s** in the owner section of a long format listing, as in the following example:

```
~]$ ls -l /bin/su
-rwsr-xr-x. 1 root root 34904 Mar 10  2011 /bin/su
```

> **Note**
>
> The **s** may be upper case or lower case. If it appears as upper case, it means that the underlying permission bit has not been set.

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 (refer to *Managing Single Sign-On and Smart Cards* for more information 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 privileges. As a result, system administrators must decide how much access the users on their network should receive.

2.1.8.1. Allowing Root Access

If the users within an organization are trusted and computer-literate, 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 to resolve issues. Even worse, they might open up security holes without knowing it.

- *Running Insecure Services* — Users with root access might run insecure servers on their machine, such as FTP or Telnet, potentially putting usernames and passwords at risk. These services transmit this information over the network in plain text.
- *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.
- *Keeping the audit trail intact* — Because the root account is often shared by multiple users, so that multiple system administrators can maintain the system, it is impossible to figure out which of those users was root at a given time. When using separate logins, the account a user logs in with, as well as a unique number for session tracking purposes, is put into the task structure, which is inherited by every process that the user starts. When using concurrent logins, the unique number can be used to trace actions to specific logins. When an action generates an audit event, it is recorded with the login account and the session associated with that unique number. Use the **`aulast`** command to view these logins and sessions. The **`--proof`** option of the **`aulast`** command can be used suggest a specific **`ausearch`** query to isolate auditable events generated by a particular session.

2.1.8.2. 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 2.1.2.2, “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 2.2. 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 2.3. 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 2.4. 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 *Using Pluggable Authentication Modules (PAM)* in the *Red Hat Enterprise Linux Managing Single Sign-On and Smart Cards* guide.

Table 2.5. 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.

2.1.8.3. Enabling Automatic Logouts

When the user is logged in as **root**, an unattended login session may pose a significant security risk. To reduce this risk, you can configure the system to automatically log out idle users after a fixed period of time:

1. Make sure the *screen* package is installed. You can do so by running the following command as **root**:

   ```
   yum install screen
   ```

   For more information on how to install packages in Red Hat Enterprise Linux, see *Red Hat Enterprise Linux 6 Deployment Guide*.

2. As **root**, add the following line at the beginning of the **/etc/profile** file to make sure the processing of this file cannot be interrupted:

   ```
   trap "" 1 2 3 15
   ```

3. Add the following lines at the end of the **/etc/profile** file to start a **screen** session each time a user logs in to a virtual console or remotely:

   ```
   SCREENEXEC="screen"
   if [ -w $(tty) ]; then
     trap "exec $SCREENEXEC" 1 2 3 15
     echo -n 'Starting session in 10 seconds'
     sleep 10
     exec $SCREENEXEC
   fi
   ```

   Note that each time a new session starts, a message will be displayed and the user will have to wait ten seconds. To adjust the time to wait before starting a session, change the value after the **sleep** command.

4. Add the following lines to the **/etc/screenrc** configuration file to close the **screen** session

after a given period of inactivity:

```
idle 120 quit autodetach off
```

This will set the time limit to 120 seconds. To adjust this limit, change the value after the **idle** directive.

Alternatively, you can configure the system to only lock the session by using the following lines instead:

```
idle 120 lockscreen autodetach off
```

This way, a password will be required to unlock the session.

The changes take effect the next time a user logs in to the system.

2.1.8.4. Limiting Root Access

Rather than completely denying access to the root user, the administrator may want to allow access only via setuid programs, such as **su** or **sudo**. For more information on **su** and **sudo**, refer to the *Red Hat Enterprise Linux 6 Deployment Guide* and the **su(1)** and **sudo(8)** man pages.

2.1.8.5. Account Locking

In Red Hat Enterprise Linux 6, the **pam_faillock** PAM module allows system administrators to lock out user accounts after a specified number of failed attempts. Limiting user login attempts serves mainly as a security measure that aims to prevent possible brute force attacks targeted to obtain a user's account password.

With the **pam_faillock** module, failed login attempts are stored in a separate file for each user in the **/var/run/faillock** directory.

Note

The order of lines in the failed attempt log files is important. Any change in this order can lock all user accounts, including the root user account when the **even_deny_root** option is used.

Follow these steps to configure account locking:

1. To lock out any non-root user after three unsuccessful attempts and unlock that user after 10 minutes, add the following lines to the **auth** section of the **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth** files:

```
auth        required      pam_faillock.so preauth silent audit deny=3
unlock_time=600
auth        sufficient    pam_unix.so nullok try_first_pass
auth        [default=die] pam_faillock.so authfail audit deny=3
unlock_time=600
```

2. Add the following line to the **account** section of both files specified in the previous step:

```
account     required     pam_faillock.so
```

3. To apply account locking for the root user as well, add the **even_deny_root** option to the

pam_faillock entries in the **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth** files:

```
auth        required      pam_faillock.so preauth silent audit deny=3
even_deny_root unlock_time=600
auth        sufficient    pam_unix.so nullok try_first_pass
auth        [default=die] pam_faillock.so authfail audit deny=3
even_deny_root unlock_time=600
auth        sufficient    pam_faillock.so authsucc audit deny=3
even_deny_root unlock_time=600
```

When user **john** attempts to log in for the fourth time after failing to log in three times previously, his account is locked upon the fourth attempt:

```
[yruseva@localhost ~]$ su - john
Account locked due to 3 failed logins
su: incorrect password
```

To disable a user from locking out even after multiple failed logins add the below line just above the "first call of" pam_faillock in both **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth**. Also replace **user1**, **user2**, **user3** with the actual user names.

```
auth [success=1 default=ignore] pam_succeed_if.so user in user1:user2:user3
```

To view the number of failed attempts per user, run, as root, the following command:

```
[root@localhost ~]# faillock
john:
When                Type  Source                                           Valid
2013-03-05 11:44:14 TTY   pts/0                                                V
```

To unlock a user's account, run, as root, the following command:

```
faillock --user <username> --reset
```

When modifying authentication configuration using the **authconfig** utility, the **system-auth** and **password-auth** files are overwritten with the settings from the **authconfig** utility. In order to use the configuration files and **authconfig** simultaneously, you must configure account locking using the following steps:

1. Create the following symbolic links:

   ```
   ~]# ln -s /etc/pam.d/system-auth /etc/pam.d/system-auth-local
   ~]# ln -s /etc/pam.d/password-auth /etc/pam.d/password-auth-local
   ```

2. The **/etc/pam.d/system-auth-local** file should contain the following lines:

```
auth        required       pam_faillock.so preauth silent audit deny=3
unlock_time=600 include system-auth-ac
auth        [default=die]  pam_faillock.so authfail silent audit deny=3
unlock_time=600

account     required       pam_faillock.so
account     include        system-auth-ac

password    include        system-auth-ac

session     include        system-auth-ac
```

3. The **/etc/pam.d/password-auth-local** file should contain the following lines:

```
auth        required       pam_faillock.so preauth silent audit deny=3
unlock_time=600 include password-auth-ac
auth        [default=die]  pam_faillock.so authfail silent audit deny=3
unlock_time=600

account     required       pam_faillock.so
account     include        password-auth-ac

password    include        system-auth-ac

session     include        system-auth-ac
```

For more information on various **pam_faillock** configuration options, refer to the **pam_faillock(8)** man page.

2.1.9. Session Locking

Users may need to leave their workstation unattended for a number of reasons during everyday operation. This could present an opportunity for an attacker to physically access the machine, especially in environments with insufficient physical security measures (see Section 1.1.3.1, “Physical Controls”). Laptops are especially exposed since their mobility interferes with physical security. You can alleviate these risks by using session locking features which prevent access to the system until a correct password is entered.

> **Note**
>
> The main advantage of locking the screen instead of logging out is that a lock allows the user's processes (such as file transfers) to continue running. Logging out would stop these processes.

2.1.9.1. Locking GNOME Using gnome-screensaver-command

The default desktop environment for Red Hat Enterprise Linux 6, GNOME, includes a feature which allows users to lock their screen at any time. There are several ways to activate the lock:

- Press the key combination specified in **System** → **Preferences** → **Keyboard Shortcuts** → **Desktop** → **Lock screen**. The default combination is **Ctrl**+**Alt**+**L**.
- Select **System** → **Lock screen** on the panel.
- Execute the following command from a command line interface:

```
gnome-screensaver-command -l
```

All of the techniques described have the same result: the screen saver is activated and the screen is locked. Users can then press any key to deactivate the screen saver, enter their password and continue working.

Keep in mind that this function requires the **`gnome-screensaver`** process to be running. You can check whether this is the case by using any command which provides information about processes. For example, execute the following command from the terminal:

```
pidof gnome-screensaver
```

If the **`gnome-screensaver`** process is currently running, a number denoting its identification number (PID) will be displayed on the screen after executing the command. If the process is not currently running, the command will provide no output at all.

Refer to the **`gnome-screensaver-command(1)`** man page for additional information.

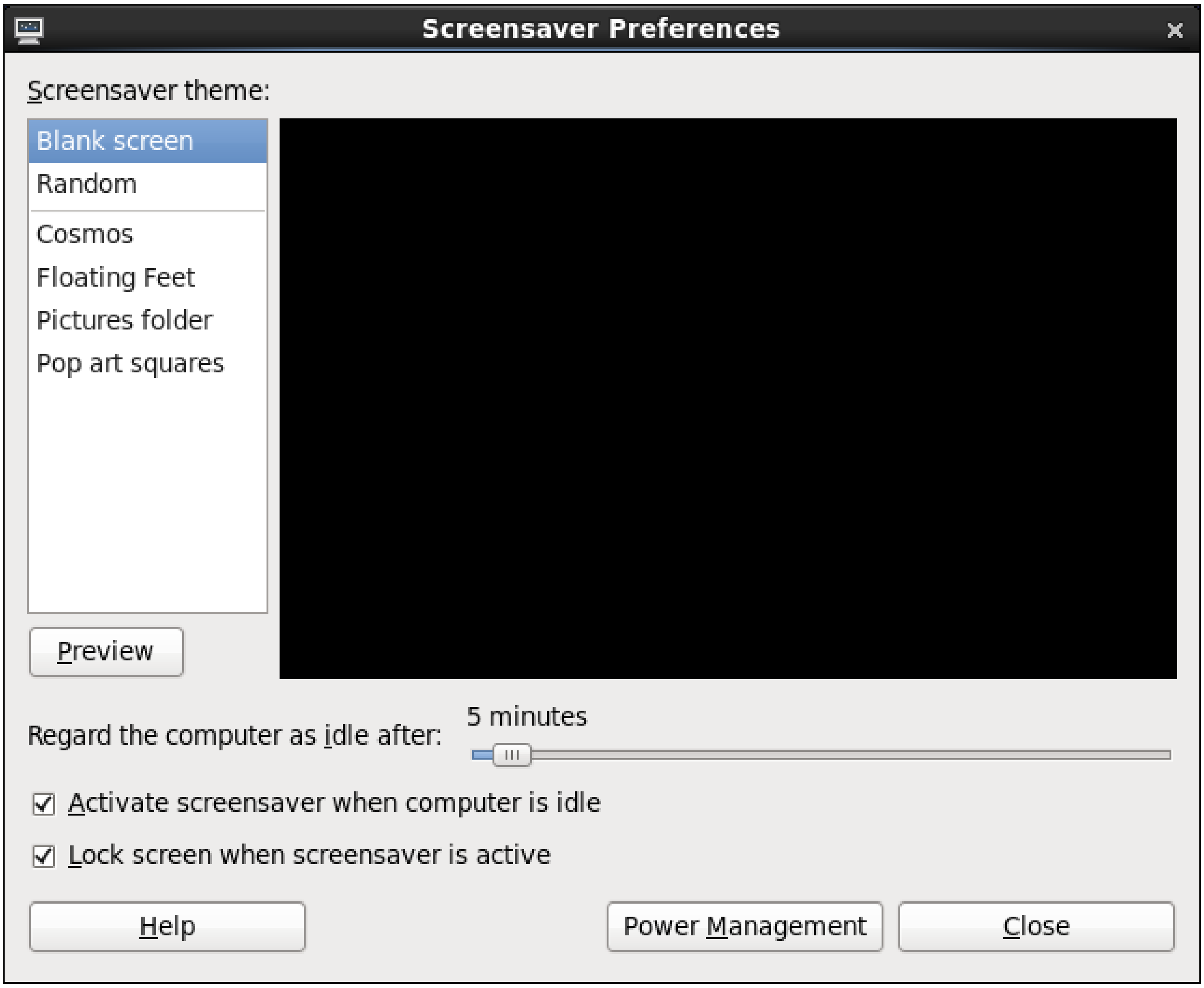
Important

The means of locking the screen described above rely on manual activation. Administrators should therefore advise their users to lock their computers every time they leave them unattended, even if only for a short period of time.

2.1.9.1.1. Automatic Lock on Screen Saver Activation

As the name **`gnome-screensaver-command`** suggests, the locking functionality is tied to GNOME's screen saver. It is possible to tie the lock to the screen saver's activation, locking the workstation every time it is left unattended for a set period of time. This function is activated by default with a five minute timeout.

To change the automatic locking settings, select **System** → **Preferences** → **Screensaver** on the main panel. This opens a window which allows setting the timeout period (the **`Regard the computer as idle after`** slider) and activating or deactivating the automatic lock (the **`Lock screen when screensaver is active`** check box).



Regard the computer as idle after: 5 minutes

☑ Activate screensaver when computer is idle

☑ Lock screen when screensaver is active

Figure 2.2. Changing the screen saver preferences

> **Note**
>
> Disabling the **`Activate screensaver when computer is idle`** option in the **`Screensaver Preferences`** dialog prevents the screen saver from starting automatically. Automatic locking is therefore disabled as well, but it is still possible to lock the workstation manually using the techniques described in Section 2.1.9.1, "Locking GNOME Using gnome-screensaver-command".

2.1.9.1.2. Remote Session Locking

You can also lock a GNOME session remotely using **ssh** as long as the target workstation accepts connections over this protocol. To remotely lock the screen on a machine you have access to, execute the following command:

```
ssh -X <username>@<server> "export DISPLAY=:0; gnome-screensaver-command -l"
```

Replace ***<username>*** with your user name and ***<server>*** with the IP address of the workstation you wish to lock.

Refer to Section 3.2.2, “Secure Shell” for more information regarding **ssh**.

2.1.9.2. Locking Virtual Consoles Using vlock

Users may also need to lock a virtual console. This can be done using a utility called **vlock**. To install this utility, execute the following command as root:

```
~]# yum install vlock
```

After installation, any console session can be locked using the **vlock** command without any additional parameters. This locks the currently active virtual console session while still allowing access to the others. To prevent access to all virtual consoles on the workstation, execute the following:

```
vlock -a
```

In this case, **vlock** locks the currently active console and the **-a** option prevents switching to other virtual consoles.

Refer to the **vlock(1)** man page for additional information.

> **Important**
>
> There are several known issues relevant to the version of **vlock** currently available for Red Hat Enterprise Linux 6:
>
> - The program does not currently allow unlocking consoles using the root password. Additional information can be found in BZ#895066.
> - Locking a console does not clear the screen and scrollback buffer, allowing anyone with physical access to the workstation to view previously issued commands and any output displayed in the console. Refer to BZ#807369 for more information.

2.1.10. Available Network Services

While user access to administrative controls is an important issue for system administrators within an organization, monitoring 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 6 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 a potential avenue of attack.

2.1.10.1. 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 render a system unusable as it tries to log and answer each request.
- *Distributed Denial of Service Attack (DDoS)* — A type of DoS attack which uses multiple compromised machines (often numbering in the thousands or more) to direct a coordinated attack on a service, flooding it with requests and making it unusable.
- *Script Vulnerability Attacks* — If a server is using scripts to execute server-side actions, as Web servers commonly do, a cracker can attack 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 that 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 Intel® 64 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 buffer overflow exploits.

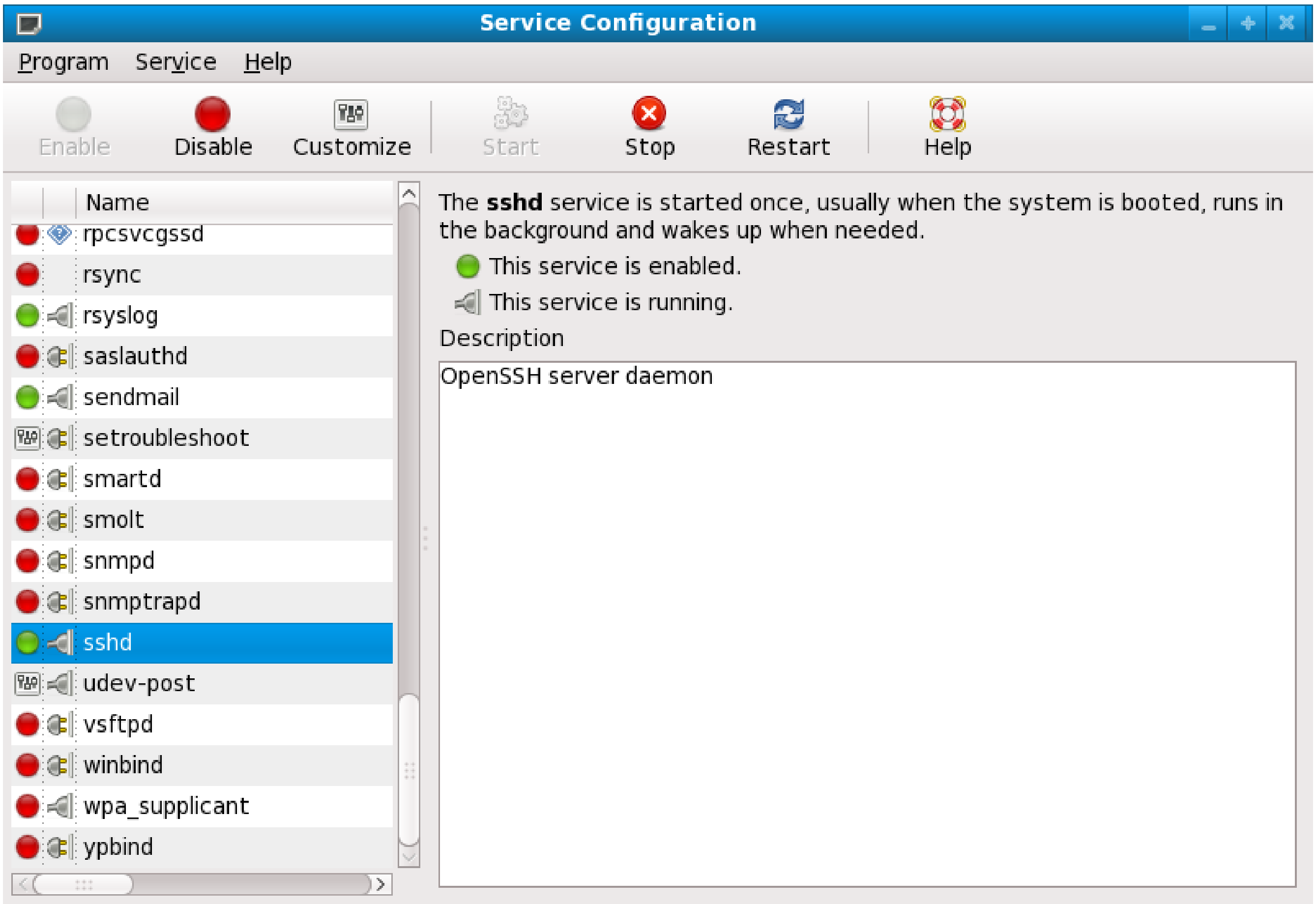
> **Important**
>
> To limit exposure to attacks over the network, disable all services that are unused.

2.1.10.2. 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 alternative print server.
- **`xinetd`** — A super server that controls connections to a range of subordinate servers, such as **`gssftp`** and **`telnet`**.
- **`sendmail`** — The Sendmail *Mail Transport Agent* (MTA) 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 avoid taking any risks. 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.



Name
rpcsvcgssd
rsync
rsyslog
saslauthd
sendmail
setroubleshoot
smartd
smolt
snmpd
snmptrapd
sshd
udev-post
vsftpd
winbind
wpa_supplicant
ypbind

The **sshd** service is started once, usually when the system is booted, runs in the background and wakes up when needed.

● This service is enabled.

This service is running.

Description

OpenSSH server daemon

Figure 2.3. Services Configuration Tool

If unsure of the purpose for a particular service, the **Services Configuration Tool** has a description field, illustrated in Figure 2.3, "Services Configuration Tool", that provides additional information.

Checking which network services are available to start at boot time is not sufficient. It is recommended to also check which ports are open and listening. Refer to Section 2.2.9, "Verifying Which Ports Are Listening" for more information.

2.1.10.3. Insecure Services

Potentially, any network service is insecure. This is why turning off unused services is so important. Exploits for services are routinely revealed and patched, making it very important to regularly update packages associated with any network service. Refer to Section 1.5, "Security Updates" for more information.

Some network protocols are inherently more insecure than others. These include any services that:

- *Transmit 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.
- *Transmit Sensitive Data Over a Network Unencrypted* — Many protocols transmit data over the network unencrypted. These protocols include Telnet, FTP, HTTP, and SMTP. Many network file systems, such as NFS and SMB, also transmit information over the network unencrypted. It is the user's responsibility when using these protocols to limit what type of data is transmitted.

  Remote memory dump services, like **`netdump`**, transmit 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 include **rlogin**, **rsh**, **telnet**, and **vsftpd**.

All remote login and shell programs (**rlogin**, **rsh**, and **telnet**) should be avoided in favor of SSH. Refer to Section 2.1.12, “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 2.2.6, “Securing FTP” for more information about securing FTP servers.

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

- **finger**
- **authd** (this was called **identd** in previous Red Hat Enterprise Linux releases.)
- **netdump**
- **netdump-server**
- **nfs**
- **rwhod**
- **sendmail**
- **smb** (Samba)
- **yppasswdd**
- **ypserv**
- **ypxfrd**

More information on securing network services is available in Section 2.2, “Server Security”.

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

2.1.11. Personal Firewalls

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

Important

Configure the necessary services and implement a firewall *before* connecting to the Internet or any other network that you do not trust.

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, ensure that you block access to ports not in use when configuring a firewall, while not blocking access to ports used by configured services.

For most users, the best tool for configuring a simple firewall is the graphical firewall configuration tool which ships with Red Hat Enterprise Linux: the **Firewall Configuration Tool** (**system-config-firewall**). This tool creates broad **iptables** rules for a general-purpose firewall using a control panel interface.

Refer to Section 2.8.2, “Basic Firewall Configuration” for more information about using this application and its available options.

For advanced users and server administrators, manually configuring a firewall with **`iptables`** is preferable. Refer to Section 2.8, “Firewalls” for more information. Refer to Section 2.8.9, “IPTables” for a comprehensive guide to the **`iptables`** command.

2.1.12. 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 6 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.

Refer to Section 3.2.2, “Secure Shell” for more information regarding OpenSSH.

> **Important**
>
> Although the **`sshd`** service is inherently secure, the service *must* be kept up-to-date to prevent security threats. Refer to Section 1.5, “Security Updates” for more information.

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.

2.2. Server Security

When a system is used as a server on a public network, it becomes a target for attacks. Hardening the system and locking down services is therefore 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.

2.2.1. 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 server 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 Section 2.8, "Firewalls" for more information about implementing firewalls with iptables commands.

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

2.2.1.1. Enhancing Security With TCP Wrappers

TCP Wrappers are capable of much more than denying access to services. This section illustrates how they can be used to send connection banners, warn of attacks from particular hosts, and enhance logging functionality. Refer to the **hosts_options** man page for information about the TCP Wrapper functionality and control language. Refer to the **xinetd.conf** man page available online at http://linux.die.net/man/5/xinetd.conf for available flags, which act as options you can apply to a service.

2.2.1.1.1. TCP Wrappers and Connection Banners

Displaying a suitable banner when users connect to a service is a good way to let potential attackers know that the system administrator is being vigilant. You can also control what information about the system is presented to users. 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 have same name as the daemon. For this example, the file is called **/etc/banners/vsftpd** and contains the following lines:

```
220-Hello, %c
220-All activity on ftp.example.com is logged.
220-Inappropriate use will result in your access privileges being removed.
```

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.

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

```
vsftpd : ALL : banners /etc/banners/
```

2.2.1.1.2. TCP Wrappers and Attack Warnings

If a particular host or network has been detected attacking the server, TCP Wrappers can be used to warn the administrator of subsequent attacks from that host or network using the **spawn** directive.

In this example, assume that a cracker from the 206.182.68.0/24 network has been detected attempting to attack the server. Place the following line in the **/etc/hosts.deny** file to deny any connection attempts from that network, and to log the attempts to a special file:

```
ALL : 206.182.68.0 : spawn /bin/echo `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**
>
> Because the **spawn** directive executes any shell command, it is a good idea to 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.

2.2.1.1.3. 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 using the **severity** option.

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

2.2.1.2. Enhancing Security With xinetd

This section focuses on using **xinetd** to set a trap service and using it to control resource levels available to any given **xinetd** service. Setting resource limits for services can help thwart *Denial of Service* (DoS) attacks. Refer to the man pages for **xinetd** and **xinetd.conf** for a list of available options.

2.2.1.2.1. 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 period or until **xinetd** is restarted. You can do this using the **SENSOR** attribute. This is an easy way to block hosts attempting to scan the ports on 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:

```
deny_time       = 30
```

This denies any further connection attempts to that port by that host 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
```

This enables the trap itself.

While using **SENSOR** is a good way to detect and stop connections from undesirable 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.

2.2.1.2.2. Controlling Server Resources

Another important feature of **xinetd** is its ability to set resource limits for services under its control.

It does this using the following directives:

- **cps = <number_of_connections> <wait_period>** — Limits the rate of incoming connections. This directive takes two arguments:
  - **<number_of_connections>** — The number of connections per second to handle. If the rate of incoming connections is higher than this, the service is temporarily disabled. The default value is fifty (50).
  - **<wait_period>** — The number of seconds to wait before re-enabling the service after it has been disabled. The default interval is ten (10) seconds.
- **instances = <number_of_connections>** — Specifies the total number of connections allowed to a service. This directive accepts either an integer value or **UNLIMITED**.
- **per_source = <number_of_connections>** — Specifies the number of connections allowed to a service by each host. This directive accepts either an integer value or **UNLIMITED**.
- **rlimit_as = <number[K|M]>** — Specifies 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>** — Specifies 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 single **xinetd** service from overwhelming the system, resulting in a denial of service.

2.2.2. 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 an NFSv2 or NFSv3 server, then **portmap** is required, and the following section applies.

If running RPC services, follow these basic rules.

2.2.2.1. 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 by DNS poisoning and other methods.

2.2.2.2. 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. The first allows TCP connections to the port 111 (used by the **`portmap`** service) from the 192.168.0.0/24 network. The second allows TCP connections to the same port from the localhost. This 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 Section 2.8, “Firewalls” for more information about implementing firewalls with iptables commands.

2.2.3. Securing NIS

The *Network Information Service* (NIS) 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.

A 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 somewhat insecure by today's standards. It has no host authentication mechanisms and transmits all of its information over the network unencrypted, including password hashes. As a result, extreme care must be taken when setting up a network that uses NIS. This is further complicated by the fact that the default configuration of NIS is inherently insecure.

It is recommended that anyone planning to implement a NIS server first secure the **`portmap`** service as outlined in Section 2.2.2, “Securing Portmap”, then address the following issues, such as network planning.

2.2.3.1. Carefully Plan the Network

Because NIS transmits sensitive information unencrypted over the network, it is important the service be run behind a firewall and on a segmented and secure network. Whenever NIS information is transmitted over an insecure network, it risks being intercepted. Careful network design can help prevent severe security breaches.

2.2.3.2. Use a Password-like NIS Domain Name and Hostname

Any machine within a 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 a 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.

2.2.3.3. Edit the /var/yp/securenets File

If the **/var/yp/securenets** file is blank or does not exist (as is the case after a default installation), NIS listens to all networks. 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 appropriate network.

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

```
255.255.255.0     192.168.0.0
```

Warning

Never start a 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.

2.2.3.4. 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 then be used 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
```

This means that the server only allows connections to ports 834 and 835 if the requests come from the 192.168.0.0/24 network, regardless of the protocol.

Note

Refer to Section 2.8, “Firewalls” for more information about implementing firewalls with iptables commands.

2.2.3.5. Use Kerberos Authentication

One of the issues to consider 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 a NIS domain and sniffs network traffic, they can collect usernames and password hashes. 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. Refer to *Managing Single Sign-On and Smart Cards* for more information about Kerberos.

2.2.4. Securing NFS

Important

The version of NFS included in Red Hat Enterprise Linux 6, NFSv4, no longer requires the **`portmap`** service as outlined in Section 2.2.2, “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 6 supports NFSv2 and NFSv3, both of which utilize **`portmap`**.

2.2.4.1. Carefully Plan the Network

NFSv2 and NFSv3 traditionally passed data insecurely. All versions of NFS now have the ability to authenticate (and optionally encrypt) ordinary file system operations using Kerberos. Under NFSv4 all operations can use Kerberos; under v2 or v3, file locking and mounting still do not use it. When using

NFSv4.0, delegations may be turned off if the clients are behind NAT or a firewall. Refer to the section on pNFS in the *Storage Administration Guide* for information on the use of NFSv4.1 to allow delegations to operate through NAT and firewalls.

2.2.4.2. Securing NFS Mount Options

The use of the **mount** command in the **/etc/fstab** file is explained in the Storage Administration Guide. From a security administration point of view it is worthwhile to note that the NFS mount options can also be specified in **/etc/nfsmount.conf**, which can be used to set custom default options.

2.2.4.2.1. Review the NFS Server

Warning

Only export entire file systems. Exporting a subdirectory of a file system can be a security issue. It is possible in some cases for a client to "break out" of the exported part of the file system and get to unexported parts (see the section on subtree checking in the **exports(5)** man page.

Use the **ro** option to export the file system as read-only whenever possible to reduce the number of users able to write to the mounted file system. Only use the **rw** option when specifically required. Refer to the man **exports(5)** page for more information. Allowing write access increases the risk from symlink attacks for example. This includes temporary directories such as **/tmp** and **/usr/tmp**.

Where directories must be mounted with the **rw** option avoid making them world-writable whenever possible to reduce risk. Exporting home directories is also viewed as a risk as some applications store passwords in clear text or weakly encrypted. This risk is being reduced as application code is reviewed and improved. Some users do not set passwords on their SSH keys so this too means home directories present a risk. Enforcing the use of passwords or using Kerberos would mitigate that risk.

Restrict exports only to clients that need access. Use the **showmount -e** command on an NFS server to review what the server is exporting. Do not export anything that is not specifically required.

Do not use the **no_root_squash** option and review existing installations to make sure it is not used. Refer to Section 2.2.4.4, “Do Not Use the **no_root_squash** Option” for more information.

The **secure** option is the server-side export option used to restrict exports to “reserved” ports. By default, the server allows client communication only from “reserved” ports (ports numbered less than 1024), because traditionally clients have only allowed “trusted” code (such as in-kernel NFS clients) to use those ports. However, on many networks it is not difficult for anyone to become root on some client, so it is rarely safe for the server to assume that communication from a reserved port is privileged. Therefore the restriction to reserved ports is of limited value; it is better to rely on Kerberos, firewalls, and restriction of exports to particular clients.

Most clients still do use reserved ports when possible. However, reserved ports are a limited resource, so clients (especially those with a large number of NFS mounts) may choose to use higher-numbered ports as well. Linux clients may do this using the “noresvport” mount option. If you wish to allow this on an export, you may do so with the “insecure” export option.

It is good practice not to allow users to login to a server. While reviewing the above settings on an NFS server conduct a review of who and what can access the server.

2.2.4.2.2. Review the NFS Client

Use the **nosuid** option to disallow the use of a **setuid** program. The **nosuid** option disables the **set-**

`user-identifier` or **`set-group-identifier`** bits. This prevents remote users from gaining higher privileges by running a setuid program. Use this option on the client and the server side.

The **`noexec`** option disables all executable files on the client. Use this to prevent users from inadvertently executing files placed in the file system being shared. The **`nosuid`** and **`noexec`** options are standard options for most, if not all, file systems.

Use the **`nodev`** option to prevent “device-files” from being processed as a hardware device by the client.

The **`resvport`** option is a client-side mount option and **`secure`** is the corresponding server-side export option (see explanation above). It restricts communication to a "reserved port". The reserved or "well known" ports are reserved for privileged users and processes such as the root user. Setting this option causes the client to use a reserved source port to communicate with the server.

All versions of NFS now support mounting with Kerberos authentication. The mount option to enable this is: **`sec=krb5`**.

NFSv4 supports mounting with Kerberos using **`krb5i`** for integrity and **`krb5p`** for privacy protection. These are used when mounting with **`sec=krb5`**, but need to be configured on the NFS server. Refer to the man page on exports (**`man 5 exports`**) for more information.

The NFS man page (**`man 5 nfs`**) has a “SECURITY CONSIDERATIONS” section which explains the security enhancements in NFSv4 and contains all the NFS specific mount options.

2.2.4.3. Beware of Syntax Errors

The NFS server determines which file systems to export and which hosts to export these directories to by consulting 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)
```

The following 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>
```

2.2.4.4. Do Not Use the `no_root_squash` Option

By default, NFS shares change the root user to the **`nfsnobody`** user, an unprivileged user account. This changes the owner of all root-created files to **`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 applications infected by Trojans for other users to inadvertently execute.

2.2.4.5. NFS Firewall Configuration

The ports used for NFS are assigned dynamically by rpcbind, which can cause problems when creating firewall rules. To simplify this process, use the */etc/sysconfig/nfs* file to specify which ports are to be used:

- **`MOUNTD_PORT`** — TCP and UDP port for mountd (rpc.mountd)
- **`STATD_PORT`** — TCP and UDP port for status (rpc.statd)
- **`LOCKD_TCPPORT`** — TCP port for nlockmgr (rpc.lockd)
- **`LOCKD_UDPPORT`** — UDP port nlockmgr (rpc.lockd)

Port numbers specified must not be used by any other service. Configure your firewall to allow the port numbers specified, as well as TCP and UDP port 2049 (NFS).

Run the **`rpcinfo -p`** command on the NFS server to see which ports and RPC programs are being used.

2.2.5. 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. A large number of options and techniques are available to secure the Apache HTTP Server — too numerous to delve into deeply here. The following section briefly explains good practices when running the Apache HTTP Server.

Always verify that any scripts running on the system work as intended *before* putting them into production. Also, ensure that only the root user has write permissions to any directory containing scripts or CGIs. To do this, run the following commands as the root user:

```
chown root <directory_name>
```

```
chmod 755 <directory_name>
```

System administrators should be careful when using the following configuration options (configured in **`/etc/httpd/conf/httpd.conf`**):

`FollowSymLinks`

This directive is enabled by default, so 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 `/`.

`Indexes`

This directive is enabled by default, but may not be desirable. To prevent visitors from browsing files on the server, remove this directive.

`UserDir`

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.

`ServerTokens`

The **`ServerTokens`** directive controls the server response header field which is sent back to clients. It includes various information which can be customized using the following parameters:

- **`ServerTokens Full`** (default option) — provides all available information (OS type and used modules), for example:

  ```
  Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2
  ```

- **`ServerTokens Prod`** or **`ServerTokens ProductOnly`** — provides the following information:

  ```
  Apache
  ```

- **`ServerTokens Major`** — provides the following information:

  ```
  Apache/2
  ```

- **`ServerTokens Minor`** — provides the following information:

  ```
  Apache/2.0
  ```

- **`ServerTokens Min`** or **`ServerTokens Minimal`** — provides the following information:

  ```
  Apache/2.0.41
  ```

- **`ServerTokens OS`** — provides the following information:

  ```
  Apache/2.0.41 (Unix)
  ```

It is recommended to use the **`ServerTokens Prod`** option so that a possible attacker does not gain any valuable information about your system.

Important

Do not remove the **`IncludesNoExec`** directive. By default, the *Server-Side Includes* (SSI) module cannot execute commands. It is recommended that you do not change this setting unless absolutely necessary, as it could, potentially, enable an attacker to execute commands on the system.

Removing httpd Modules

In certain scenarios, it is beneficial to remove certain **`httpd`** modules to limit the functionality of the HTTP Server. To do so, simply comment out the entire line which loads the module you wish to remove in the **`/etc/httpd/conf/httpd.conf`** file. For example, to remove the proxy module, comment out the following line by prepending it with a hash sign:

```
#LoadModule proxy_module modules/mod_proxy.so
```

Note that the **/etc/httpd/conf.d/** directory contains configuration files which are used to load modules as well.

httpd and SELinux

For information regarding the Apache HTTP Server and SELinux, refer to the *Managing Confined Services Guide*.

2.2.6. Securing FTP

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

Red Hat Enterprise Linux provides three FTP servers.

- **gssftpd** — A Kerberos-aware **xinetd**-based FTP daemon that does not transmit 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.

2.2.6.1. 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 2.2.1.1.1, “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 2.2.1.1.1, "TCP Wrappers and Connection Banners".

2.2.6.2. 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 establishes a directory tree 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.

Warning

If enabling anonymous access to an FTP server, be aware of where sensitive data is stored.

2.2.6.2.1. Anonymous Upload

To allow anonymous users to upload files, it is recommended you create a write-only directory within **/var/ftp/pub/**. To do this, run the following command as root:

```
~]# mkdir /var/ftp/pub/upload
```

Next, change the permissions so that anonymous users cannot view the contents of the directory:

```
~]# chmod 730 /var/ftp/pub/upload
```

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

```
~]# ls -ld /var/ftp/pub/upload
drwx-wx---. 2 root ftp 4096 Nov 14 22:57 /var/ftp/pub/upload
```

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
```

2.2.6.3. User Accounts

Because FTP transmits 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 all user accounts in **vsftpd**, add the following directive to **/etc/vsftpd/vsftpd.conf**:

```
local_enable=NO
```

2.2.6.3.1. Restricting User Accounts

To disable FTP access for specific accounts or specific groups of accounts, such as the root user and those with **sudo** privileges, the easiest way is to use a PAM list file as described in Section 2.1.8.2, "Disallowing 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`**

2.2.6.4. Use TCP Wrappers To Control Access

Use TCP Wrappers to control access to either FTP daemon as outlined in Section 2.2.1.1, "Enhancing Security With TCP Wrappers".

2.2.7. Securing Postfix

Postfix is a Mail Transfer Agent (MTA) that uses the Simple Mail Transfer 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.

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

2.2.7.1. 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. The effectiveness of such attacks can be limited by adding and setting limits of the following directives in **`/etc/postfix/main.cf`**.

- **`smtpd_client_connection_rate_limit`** — The maximum number of connection attempts any client is allowed to make to this service per time unit (described below). The default value is 0, which means a client can make as many connections per time unit as Postfix can accept. By default, clients in trusted networks are excluded.
- **`anvil_rate_time_unit`** — This time unit is used for rate limit calculations. The default value is 60 seconds.
- **`smtpd_client_event_limit_exceptions`** — Clients that are excluded from the connection and rate limit commands. By default, clients in trusted networks are excluded.
- **`smtpd_client_message_rate_limit`** — The maximum number of message deliveries a client is allowed to request per time unit (regardless of whether or not Postfix actually accepts those messages).
- **`default_process_limit`** — The default maximum number of Postfix child processes that provide a given service. This limit can be overruled for specific services in the **`master.cf`** file. By default the value is 100.
- **`queue_minfree`** — The minimum amount of free space in bytes in the queue file system that is needed to receive mail. This is currently used by the Postfix SMTP server to decide if it will accept any mail at all. By default, the Postfix SMTP server rejects **`MAIL FROM`** commands when the amount of free space is less than 1.5 times the message_size_limit. To specify a higher minimum free space limit, specify a queue_minfree value that is at least 1.5 times the message_size_limit. By default the queue_minfree value is 0.
- **`header_size_limit`** — The maximum amount of memory in bytes for storing a message header. If a header is larger, the excess is discarded. By default the value is 102400.
- **`message_size_limit`** — The maximum size in bytes of a message, including envelope information. By default the value is 10240000.

2.2.7.2. NFS and Postfix

Never put the mail spool directory, **`/var/spool/postfix/`**, 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.

Note

With NFSv4 using Kerberos, this is not the case, since the **SECRPC_GSS** kernel module does not utilize UID-based authentication. However, it is still considered good practice *not* to put the mail spool directory on NFS shared volumes.

2.2.7.3. Mail-only Users

To help prevent local user exploits on the Postfix server, it is best for mail users to only access the Postfix 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).

2.2.7.4. Disable Postfix Network Listening

By default, Postfix is set up to only listen to the local loopback address. You can verify this by viewing the file **/etc/postfix/main.cf**.

View the file **/etc/postfix/main.cf** to ensure that only the following ***inet_interfaces*** line appears:

```
inet_interfaces = localhost
```

This ensures that Postfix only accepts mail messages (such as cron job reports) from the local system and not from the network. This is the default setting and protects Postfix from a network attack.

For removal of the localhost restriction and allowing Postfix to listen on all interfaces the ***inet_interfaces* = *all*** setting can be used.

2.2.8. Securing Sendmail

Sendmail is a Mail Transfer Agent (MTA) that uses the Simple Mail Transfer 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.

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

2.2.8.1. 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 is 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 a single message.

2.2.8.2. 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.

Note

With NFSv4 using Kerberos, this is not the case, since the **`SECRPC_GSS`** kernel module does not utilize UID-based authentication. However, it is still considered good practice *not* to put the mail spool directory on NFS shared volumes.

2.2.8.3. 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).

2.2.8.4. Disable Sendmail Network Listening

By default, Sendmail is set up to only listen to the local loopback address. You can verify this by viewing the file **`/etc/mail/sendmail.mc`** to ensure that the following line appears:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

This ensures that Sendmail only accepts mail messages (such as cron job reports) from the local system and not from the network. This is the default setting and protects Sendmail from a network attack.

For removal of the localhost restriction, the Addr=127.0.0.1 string needs to be removed. Changing Sendmail's configuration requires installing the *sendmail-cf* package, then editing the **`.mc`** file, running **`/etc/mail/make`** and finally restarting **sendmail**. The **`.cf`** configuration file will be regenerated. Note that the system clock must be correct and working and that there must not be any system clock time shifts between these actions in order for the configuration file to be automatically regenerated.

2.2.9. Verifying Which Ports Are Listening

Unnecessary open ports should be avoided because it increases the attack surface of your system. If after the system has been in service you find unexpected open ports in listening state, that might be signs of intrusion and it should be investigated.

Issue the following command, as root, from the console to determine which ports are listening for connections from the network:

```
~]# netstat -tanp | grep LISTEN
tcp        0      0 0.0.0.0:45876               0.0.0.0:*
LISTEN      1193/rpc.statd
tcp        0      0 192.168.122.1:53           0.0.0.0:*
LISTEN      1241/dnsmasq
tcp        0      0 127.0.0.1:631               0.0.0.0:*
LISTEN      1783/cupsd
tcp        0      0 127.0.0.1:25                0.0.0.0:*
LISTEN      7696/sendmail
tcp        0      0 0.0.0.0:111                 0.0.0.0:*
LISTEN      1167/rpcbind
tcp        0      0 127.0.0.1:30003             0.0.0.0:*
LISTEN      1118/tcsd
tcp        0      0 :::631                      :::*
LISTEN      1/init
tcp        0      0 :::35018                    :::*
LISTEN      1193/rpc.statd
tcp        0      0 :::111                      :::*
LISTEN      1167/rpcbind
```

Review the output of the command with the services needed on the system, turn off what is not specifically required or authorized, repeat the check. Proceed then to make external checks using **nmap** from another system connected via the network to the first system. This can be used verify the rules in **iptables**. Make a scan for every IP address shown in the **netstat** output (except for localhost 127.0.0.0 or ::1 range) from an external system. Use the **-6** option for scanning an IPv6 address. See **man nmap(1)** for more information.

The following is an example of the command to be issued from the console of another system to determine which ports are listening for TCP connections from the network:

```
~]# nmap -sT -O 192.168.122.1
```

Refer to the man pages for **netstat**, **nmap**, and **services** for more information.

2.2.10. Disable Source Routing

Source routing is an Internet Protocol mechanism that allows an IP packet to carry information, a list of addresses, that tells a router the path the packet must take. There is also an option to record the hops as the route is traversed. The list of hops taken, the "route record", provides the destination with a return path to the source. This allows the source (the sending host) to specify the route, loosely or strictly, ignoring the routing tables of some or all of the routers. It can allow a user to redirect network traffic for malicious purposes. Therefore, source-based routing should be disabled.

The **accept_source_route** option causes network interfaces to accept packets with the *Strict Source Route* (SSR) or *Loose Source Routing* (LSR) option set. The acceptance of source routed packets is controlled by **sysctl** settings. Issue the following command as root to drop packets with the SSR or LSR option set:

```
~]# /sbin/sysctl -w net.ipv4.conf.all.accept_source_route=0
```

Disabling the forwarding of packets should also be done in conjunction with the above when possible (disabling forwarding may interfere with virtualization). Issue the commands listed below as root:

These commands disable forwarding of IPv4 and IPv6 packets on all interfaces.

```
~]# /sbin/sysctl -w net.ipv4.conf.all.forwarding=0
```

```
~]# /sbin/sysctl -w net.ipv6.conf.all.forwarding=0
```

These commands disable forwarding of all multicast packets on all interfaces.

```
~]# /sbin/sysctl -w net.ipv4.conf.all.mc_forwarding=0
```

```
~]# /sbin/sysctl -w net.ipv6.conf.all.mc_forwarding=0
```

Accepting ICMP redirects has few legitimate uses. Disable the acceptance and sending of ICMP redirected packets unless specifically required.

These commands disable acceptance of all ICMP redirected packets on all interfaces.

```
~]# /sbin/sysctl -w net.ipv4.conf.all.accept_redirects=0
```

```
~]# /sbin/sysctl -w net.ipv6.conf.all.accept_redirects=0
```

This command disables acceptance of secure ICMP redirected packets on all interfaces.

```
~]# /sbin/sysctl -w net.ipv4.conf.all.secure_redirects=0
```

This command disables acceptance of all IPv4 ICMP redirected packets on all interfaces.

```
~]# /sbin/sysctl -w net.ipv4.conf.all.send_redirects=0
```

There is only a directive to disable sending of IPv4 redirected packets. Refer to *RFC4294* for an explanation of “IPv6 Node Requirements” which resulted in this difference between IPv4 and IPv6.

In order to make the settings permanent they must be added to **/etc/sysctl.conf**.

Refer to the sysctl man page, **sysctl(8)**, for more information. Refer to *RFC791* for an explanation of the Internet options related to source based routing and its variants.

Warning

Ethernet networks provide additional ways to redirect traffic, such as ARP or MAC address spoofing, unauthorized DHCP servers, and IPv6 router or neighbor advertisements. In addition, unicast traffic is occasionally broadcast, causing information leaks. These weaknesses can only be addressed by specific countermeasures implemented by the network operator. Host-based countermeasures are not fully effective.

2.2.11. Reverse Path Filtering

Reverse path filtering is used to prevent packets which arrived via one interface from leaving via a different interface. When outgoing routes and incoming routes are different it is sometimes referred to as “asymmetric routing”. Routers often route packets this way but most hosts should not need to do this. Exceptions are such applications as sending traffic out over one link and receiving traffic over another link from a different service provider. For example, using leased lines in combination with xDSL, or Satellite links with 3G modems. If such a scenario is applicable to you then turning off reverse path filtering on the incoming interface is necessary. In short, unless you know that it is required, it is best disabled as it prevents users spoofing IP addresses from local subnets and reduces the opportunity for DDoS attacks.

Note

Red Hat Enterprise Linux 6 (unlike Red Hat Enterprise Linux 5) defaults to using Strict Reverse Path filtering. Red Hat Enterprise Linux 6 follows the “Strict Reverse Path” recommendation from RFC 3704, Ingress Filtering for Multihomed Networks. This currently only applies to IPv4 in Red Hat Enterprise Linux 6.

Warning

If forwarding is enabled, then Reverse Path Filtering should only be disabled if there are other means for source address validation (such as iptables rules for example).

rp_filter

Reverse Path Filter is enabled by means of the **rp_filter** directive. The **rp_filter** option is used to direct the kernel to select from one of three modes.

It takes the following form when setting the default behavior:

```
~]# /sbin/sysctl -w net.ipv4.conf.default.rp_filter=INTEGER
```

where ***INTEGER*** is one of the following:

- **0** — No source validation.
- **1** — Strict mode as defined in RFC3704.
- **2** — Loose mode as defined in RFC3704.

The setting can be overridden per network interface using **net.ipv4.*interface*.rp_filter**. To make these settings persistent across reboot, modify the **/etc/sysctl.conf** file.

2.2.11.1. Additional Resources

The following are resources which explain more about Reverse Path Filtering.

- **Installed Documentation**

  usr/share/doc/kernel-doc-*version*/Documentation/networking/ip-sysctl.txt - This file contains a complete list of files and options available in the **/proc/sys/net/ipv4/** directory.

- **Useful Websites**

  https://access.redhat.com/knowledge/solutions/53031 - The Red Hat Knowledgebase article about **rp_filter**.

  Refer to *RFC3704.* for an explanation of Ingress Filtering for Multihomed Networks.

2.3. Single Sign-on (SSO)

The Red Hat Enterprise Linux SSO functionality reduces the number of times Red Hat Enterprise Linux desktop users have to enter their passwords. Several major applications leverage the same underlying authentication and authorization mechanisms so that users can log in to Red Hat Enterprise Linux from the log-in screen, and then not need to re-enter their passwords. These applications are detailed below.

For more information on Pluggable Authentication Modules, refer to Red Hat Enterprise Linux 6 Managing Single Sign-On and Smart Cards guide.

2.4. Pluggable Authentication Modules (PAM)

Pluggable authentication modules are a common framework for authentication and security. Both of Red Hat Enterprise Linux's single sign-on methods — Kerberos and smart cards — depend on underlying PAM configuration.

For more information on Pluggable Authentication Modules, refer to Chapter 2., "Using Pluggable Authentication Modules (PAM)" of the *Red Hat Enterprise Linux 6 Managing Single Sign-On and Smart Cards* guide.

2.5. Kerberos

Maintaining system security and integrity within a network is critical, and it encompasses every user, application, service, and server within the network infrastructure. It requires an understanding of everything that is running on the network and the manner in which these services are used. At the core of maintaining this security is maintaining access to these applications and services and enforcing that access.

Kerberos provides a mechanism that allows both users and machines to identify themselves to network and receive defined, limited access to the areas and services that the administrator configured. Kerberos authenticates entities by verifying their identity, and Kerberos also secures this authenticating data so that it cannot be accessed and used or tampered with by an outsider.

For more information on Pluggable Authentication Modules, refer to Chapter 3., "Using Kerberos" of the *Red Hat Enterprise Linux 6 Managing Single Sign-On and Smart Cards* guide.

2.6. TCP Wrappers and xinetd

Controlling access to network services is one of the most important security tasks facing a server administrator. Red Hat Enterprise Linux provides several tools for this purpose. For example, an **iptables**-based firewall filters out unwelcome network packets within the kernel's network stack. For network services that utilize it, *TCP Wrappers* add an additional layer of protection by defining which hosts are or are not allowed to connect to "*wrapped*" network services. One such wrapped network service is the **xinetd** *super server*. This service is called a super server because it controls connections to a subset of network services and further refines access control.

Figure 2.4, “Access Control to Network Services” is a basic illustration of how these tools work together to protect network services.

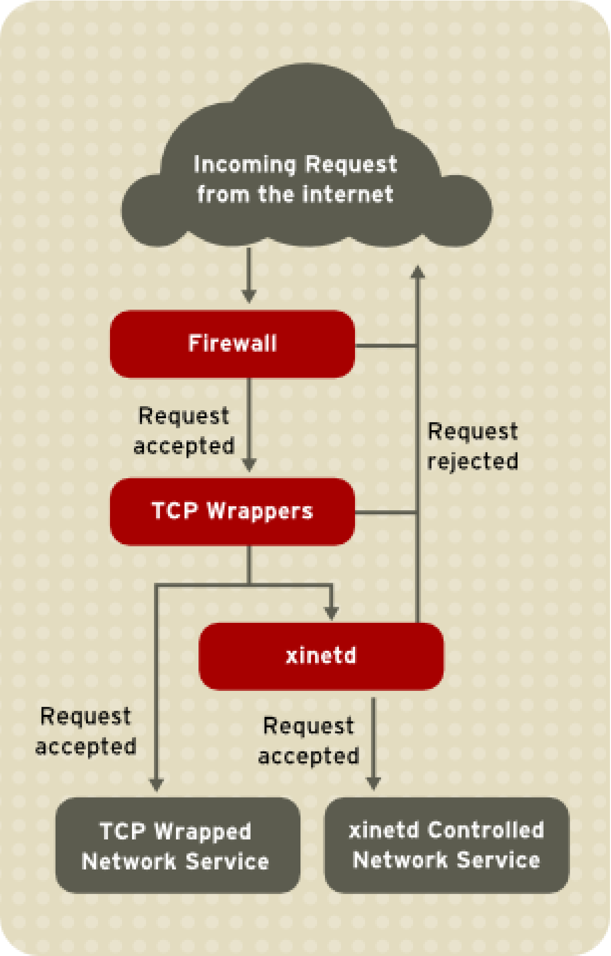

Incoming Request from the internet
Firewall
Request accepted
Request rejected
TCP Wrappers
xinetd
Request accepted
Request accepted
TCP Wrapped Network Service
xinetd Controlled Network Service

Figure 2.4. Access Control to Network Services

For more information about using forewalls with **`iptables`**, refer to Section 2.8.9, “IPTables”.

2.6.1. TCP Wrappers

The TCP Wrappers packages (*tcp_wrappers* and *tcp_wrappers-libs*) are installed by default and provide host-based access control to network services. The most important component within the package is the **`/lib/libwrap.so`** or **`/lib64/libwrap.so`** library. In general terms, a TCP-wrapped service is one that has been compiled against the **`libwrap.so`** library.

When a connection attempt is made to a TCP-wrapped service, the service first references the host's access files (**`/etc/hosts.allow`** and **`/etc/hosts.deny`**) to determine whether or not the client is allowed to connect. In most cases, it then uses the syslog daemon (**`syslogd`**) to write the name of the requesting client and the requested service to **`/var/log/secure`** or **`/var/log/messages`**.

If a client is allowed to connect, TCP Wrappers release control of the connection to the requested service and take no further part in the communication between the client and the server.

In addition to access control and logging, TCP Wrappers can execute commands to interact with the client before denying or releasing control of the connection to the requested network service.

Because TCP Wrappers are a valuable addition to any server administrator's arsenal of security tools, most network services within Red Hat Enterprise Linux are linked to the **`libwrap.so`** library. Such applications include **`/usr/sbin/sshd`**, **`/usr/sbin/sendmail`**, and **`/usr/sbin/xinetd`**.

Note

To determine if a network service binary is linked to **`libwrap.so`**, type the following command as the root user:

```
ldd <binary-name> | grep libwrap
```

Replace ***`<binary-name>`*** with the name of the network service binary. If the command returns straight to the prompt with no output, then the network service is *not* linked to **`libwrap.so`**. The following example indicates that **`/usr/sbin/sshd`** is linked to **`libwrap.so`**:

```
~]# ldd /usr/sbin/sshd | grep libwrap
        libwrap.so.0 => /lib/libwrap.so.0 (0x00655000)
```

2.6.1.1. Advantages of TCP Wrappers

TCP Wrappers provide the following advantages over other network service control techniques:

- *Transparency to both the client and the wrapped network service* — Both the connecting client and the wrapped network service are unaware that TCP Wrappers are in use. Legitimate users are logged and connected to the requested service while connections from banned clients fail.
- *Centralized management of multiple protocols* — TCP Wrappers operate separately from the network services they protect, allowing many server applications to share a common set of access control configuration files, making for simpler management.

2.6.2. TCP Wrappers Configuration Files

To determine if a client is allowed to connect to a service, TCP Wrappers reference the following two files, which are commonly referred to as *hosts access* files:

- **`/etc/hosts.allow`**
- **`/etc/hosts.deny`**

When a TCP-wrapped service receives a client request, it performs the following steps:

1. *It references **`/etc/hosts.allow`*** — The TCP-wrapped service sequentially parses the **`/etc/hosts.allow`** file and applies the first rule specified for that service. If it finds a matching rule, it allows the connection. If not, it moves on to the next step.
2. *It references **`/etc/hosts.deny`*** — The TCP-wrapped service sequentially parses the **`/etc/hosts.deny`** file. If it finds a matching rule, it denies the connection. If not, it grants access to the service.

The following are important points to consider when using TCP Wrappers to protect network services:

- Because access rules in **`hosts.allow`** are applied first, they take precedence over rules specified in **`hosts.deny`**. Therefore, if access to a service is allowed in **`hosts.allow`**, a rule denying access to that same service in **`hosts.deny`** is ignored.
- The rules in each file are read from the top down and the first matching rule for a given service is the only one applied. The order of the rules is extremely important.
- If no rules for the service are found in either file, or if neither file exists, access to the service is granted.

- TCP-wrapped services do not cache the rules from the hosts access files, so any changes to **`hosts.allow`** or **`hosts.deny`** take effect immediately, without restarting network services.

> **Warning**
>
> If the last line of a hosts access file is not a newline character (created by pressing the **Enter** key), the last rule in the file fails and an error is logged to either **`/var/log/messages`** or **`/var/log/secure`**. This is also the case for a rule that spans multiple lines without using the backslash character. The following example illustrates the relevant portion of a log message for a rule failure due to either of these circumstances:
>
> ```
> warning: /etc/hosts.allow, line 20: missing newline or line too long
> ```

2.6.2.1. Formatting Access Rules

The format for both **`/etc/hosts.allow`** and **`/etc/hosts.deny`** is identical. Each rule must be on its own line. Blank lines or lines that start with a hash (#) are ignored.

Each rule uses the following basic format to control access to network services:

```
<daemon list> : <client list> [: <option> : <option> : …]
```

- ***`<daemon list>`*** — A comma-separated list of process names (*not* service names) or the **`ALL`** wildcard. The daemon list also accepts operators (refer to Section 2.6.2.1.4, “Operators”) to allow greater flexibility.
- ***`<client list>`*** — A comma-separated list of hostnames, host IP addresses, special patterns, or wildcards which identify the hosts affected by the rule. The client list also accepts operators listed in Section 2.6.2.1.4, “Operators” to allow greater flexibility.
- ***`<option>`*** — An optional action or colon-separated list of actions performed when the rule is triggered. Option fields support expansions, launch shell commands, allow or deny access, and alter logging behavior.

> **Note**
>
> More information on some of the terms above can be found elsewhere in this guide:
>
> - Section 2.6.2.1.1, “Wildcards”
> - Section 2.6.2.1.2, “Patterns”
> - Section 2.6.2.2.4, “Expansions”
> - Section 2.6.2.2, “Option Fields”

The following is a basic sample hosts access rule:

```
vsftpd : .example.com
```

This rule instructs TCP Wrappers to watch for connections to the FTP daemon (**`vsftpd`**) from any host in the **`example.com`** domain. If this rule appears in **`hosts.allow`**, the connection is accepted. If this rule appears in **`hosts.deny`**, the connection is rejected.

The next sample hosts access rule is more complex and uses two option fields:

```
sshd : .example.com  \
 : spawn /bin/echo `/bin/date` access denied>>/var/log/sshd.log \
 : deny
```

Note that each option field is preceded by the backslash (\). Use of the backslash prevents failure of the rule due to length.

This sample rule states that if a connection to the SSH daemon (**sshd**) is attempted from a host in the **example.com** domain, execute the **echo** command to append the attempt to a special log file, and deny the connection. Because the optional **deny** directive is used, this line denies access even if it appears in the **hosts.allow** file. Refer to Section 2.6.2.2, “Option Fields” for a more detailed look at available options.

2.6.2.1.1. Wildcards

Wildcards allow TCP Wrappers to more easily match groups of daemons or hosts. They are used most frequently in the client list field of access rules.

The following wildcards are available:

- **ALL** — Matches everything. It can be used for both the daemon list and the client list.
- **LOCAL** — Matches any host that does not contain a period (.), such as localhost.
- **KNOWN** — Matches any host where the hostname and host address are known or where the user is known.
- **UNKNOWN** — Matches any host where the hostname or host address are unknown or where the user is unknown.
- **PARANOID** — A reverse DNS lookup is done on the source IP address to obtain the host name. Then a DNS lookup is performed to resolve the IP address. If the two IP addresses do not match the connection is dropped and the logs are updated

> **Important**
>
> The **KNOWN**, **UNKNOWN**, and **PARANOID** wildcards should be used with care, because they rely on a functioning DNS server for correct operation. Any disruption to name resolution may prevent legitimate users from gaining access to a service.

2.6.2.1.2. Patterns

Patterns can be used in the client field of access rules to more precisely specify groups of client hosts.

The following is a list of common patterns for entries in the client field:

- *Hostname beginning with a period (.)* — Placing a period at the beginning of a hostname matches all hosts sharing the listed components of the name. The following example applies to any host within the **example.com** domain:

  ```
  ALL : .example.com
  ```

- *IP address ending with a period (.)* — Placing a period at the end of an IP address matches all hosts sharing the initial numeric groups of an IP address. The following example applies to any host within the **192.168.x.x** network:

```
ALL : 192.168.
```

- *IP address/netmask pair* — Netmask expressions can also be used as a pattern to control access to a particular group of IP addresses. The following example applies to any host with an address range of **192.168.0.0** through **192.168.1.255**:

```
ALL : 192.168.0.0/255.255.254.0
```

Important

When working in the IPv4 address space, the address/prefix length (*prefixlen*) pair declarations (CIDR notation) are not supported. Only IPv6 rules can use this format.

- *[IPv6 address]/prefixlen pair* — [net]/prefixlen pairs can also be used as a pattern to control access to a particular group of IPv6 addresses. The following example would apply to any host with an address range of **3ffe:505:2:1::** through **3ffe:505:2:1:ffff:ffff:ffff:ffff**:

```
ALL : [3ffe:505:2:1::]/64
```

- *The asterisk (*)* — Asterisks can be used to match entire groups of hostnames or IP addresses, as long as they are not mixed in a client list containing other types of patterns. The following example would apply to any host within the **example.com** domain:

```
ALL : *.example.com
```

- *The slash (/)* — If a client list begins with a slash, it is treated as a file name. This is useful if rules specifying large numbers of hosts are necessary. The following example refers TCP Wrappers to the ***/etc/telnet.hosts*** file for all Telnet connections:

```
in.telnetd : /etc/telnet.hosts
```

Other, less used patterns are also accepted by TCP Wrappers. Refer to the **hosts_access** man 5 page for more information.

Warning

Be very careful when using hostnames and domain names. Attackers can use a variety of tricks to circumvent accurate name resolution. In addition, disruption to DNS service prevents even authorized users from using network services. It is, therefore, best to use IP addresses whenever possible.

2.6.2.1.3. Portmap and TCP Wrappers

Portmap's implementation of TCP Wrappers does not support host look-ups, which means **portmap** can not use hostnames to identify hosts. Consequently, access control rules for portmap in **hosts.allow** or **hosts.deny** must use IP addresses, or the keyword **ALL**, for specifying hosts.

Changes to **portmap** access control rules may not take effect immediately. You may need to restart the

portmap service.

Widely used services, such as NIS and NFS, depend on **portmap** to operate, so be aware of these limitations.

2.6.2.1.4. Operators

At present, access control rules accept one operator, **EXCEPT**. It can be used in both the daemon list and the client list of a rule.

The **EXCEPT** operator allows specific exceptions to broader matches within the same rule.

In the following example from a **hosts.allow** file, all **example.com** hosts are allowed to connect to all services except **cracker.example.com**:

```
ALL : .example.com EXCEPT cracker.example.com
```

In another example from a **hosts.allow** file, clients from the **192.168.0.*x*** network can use all services except for FTP:

```
ALL EXCEPT vsftpd : 192.168.0.
```

> **Note**
>
> Organizationally, it is often easier to avoid using **EXCEPT** operators. This allows other administrators to quickly scan the appropriate files to see what hosts are allowed or denied access to services, without having to sort through **EXCEPT** operators.

2.6.2.2. Option Fields

In addition to basic rules that allow and deny access, the Red Hat Enterprise Linux implementation of TCP Wrappers supports extensions to the access control language through *option fields*. By using option fields in hosts access rules, administrators can accomplish a variety of tasks such as altering log behavior, consolidating access control, and launching shell commands.

2.6.2.2.1. Logging

Option fields let administrators easily change the log facility and priority level for a rule by using the **severity** directive.

In the following example, connections to the SSH daemon from any host in the **example.com** domain are logged to the default **authpriv** **syslog** facility (because no facility value is specified) with a priority of **emerg**:

```
sshd : .example.com : severity emerg
```

It is also possible to specify a facility using the **severity** option. The following example logs any SSH connection attempts by hosts from the **example.com** domain to the **local0** facility with a priority of **alert**:

```
sshd : .example.com : severity local0.alert
```

Note

In practice, this example does not work until the syslog daemon (**syslogd**) is configured to log to the **local0** facility. Refer to the **syslog.conf** man page for information about configuring custom log facilities.

2.6.2.2.2. Access Control

Option fields also allow administrators to explicitly allow or deny hosts in a single rule by adding the **allow** or **deny** directive as the final option.

For example, the following two rules allow SSH connections from **client-1.example.com**, but deny connections from **client-2.example.com**:

```
sshd : client-1.example.com : allow
sshd : client-2.example.com : deny
```

By allowing access control on a per-rule basis, the option field allows administrators to consolidate all access rules into a single file: either **hosts.allow** or **hosts.deny**. Some administrators consider this an easier way of organizing access rules.

2.6.2.2.3. Shell Commands

Option fields allow access rules to launch shell commands through the following two directives:

- **spawn** — Launches a shell command as a child process. This directive can perform tasks like using **/usr/sbin/safe_finger** to get more information about the requesting client or create special log files using the **echo** command.

  In the following example, clients attempting to access Telnet services from the **example.com** domain are quietly logged to a special file:

  ```
  in.telnetd : .example.com \
   : spawn /bin/echo `/bin/date` from %h>>/var/log/telnet.log \
   : allow
  ```

- **twist** — Replaces the requested service with the specified command. This directive is often used to set up traps for intruders (also called "honey pots"). It can also be used to send messages to connecting clients. The **twist** directive must occur at the end of the rule line.

  In the following example, clients attempting to access FTP services from the **example.com** domain are sent a message using the **echo** command:

  ```
  vsftpd : .example.com \
   : twist /bin/echo "421 This domain has been black-listed. Access denied!"
  ```

For more information about shell command options, refer to the **hosts_options** man page.

2.6.2.2.4. Expansions

Expansions, when used in conjunction with the **spawn** and **twist** directives, provide information about the client, server, and processes involved.

The following is a list of supported expansions:

- **%a** — Returns the client's IP address.
- **%A** — Returns the server's IP address.
- **%c** — Returns a variety of client information, such as the username and hostname, or the username and IP address.
- **%d** — Returns the daemon process name.
- **%h** — Returns the client's hostname (or IP address, if the hostname is unavailable).
- **%H** — Returns the server's hostname (or IP address, if the hostname is unavailable).
- **%n** — Returns the client's hostname. If unavailable, **unknown** is printed. If the client's hostname and host address do not match, **paranoid** is printed.
- **%N** — Returns the server's hostname. If unavailable, **unknown** is printed. If the server's hostname and host address do not match, **paranoid** is printed.
- **%p** — Returns the daemon's process ID.
- **%s** —Returns various types of server information, such as the daemon process and the host or IP address of the server.
- **%u** — Returns the client's username. If unavailable, **unknown** is printed.

The following sample rule uses an expansion in conjunction with the **spawn** command to identify the client host in a customized log file.

When connections to the SSH daemon (**sshd**) are attempted from a host in the **example.com** domain, execute the **echo** command to log the attempt, including the client hostname (by using the **%h** expansion), to a special file:

```
sshd : .example.com  \
 : spawn /bin/echo `/bin/date` access denied to %h>>/var/log/sshd.log \
 : deny
```

Similarly, expansions can be used to personalize messages back to the client. In the following example, clients attempting to access FTP services from the **example.com** domain are informed that they have been banned from the server:

```
vsftpd : .example.com \
: twist /bin/echo "421 %h has been banned from this server!"
```

For a full explanation of available expansions, as well as additional access control options, refer to section 5 of the man pages for **hosts_access** (**man 5 hosts_access**) and the man page for **hosts_options**.

Refer to Section 2.6.5, “Additional Resources” for more information about TCP Wrappers.

2.6.3. xinetd

The **xinetd** daemon is a TCP-wrapped *super service* which controls access to a subset of popular network services, including FTP, IMAP, and Telnet. It also provides service-specific configuration options for access control, enhanced logging, binding, redirection, and resource utilization control.

When a client attempts to connect to a network service controlled by **xinetd**, the super service receives the request and checks for any TCP Wrappers access control rules.

If access is allowed, **xinetd** verifies that the connection is allowed under its own access rules for that service. It also checks that the service is able to have more resources assigned to it and that it is not in breach of any defined rules.

If all these conditions are met (that is, access is allowed to the service; the service has not reached its resource limit; and the service is not in breach of any defined rule), **xinetd** then starts an instance of the requested service and passes control of the connection to it. After the connection has been established, **xinetd** takes no further part in the communication between the client and the server.

2.6.4. xinetd Configuration Files

The configuration files for **xinetd** are as follows:

- **/etc/xinetd.conf** — The global **xinetd** configuration file.
- **/etc/xinetd.d/** — The directory containing all service-specific files.

2.6.4.1. The /etc/xinetd.conf File

The **/etc/xinetd.conf** file contains general configuration settings which affect every service under **xinetd**'s control. It is read when the **xinetd** service is first started, so for configuration changes to take effect, you need to restart the **xinetd** service. The following is a sample **/etc/xinetd.conf** file:

```
defaults
{
  instances                = 60
  log_type                  = SYSLOG authpriv
  log_on_success           = HOST PID
  log_on_failure           = HOST
  cps                       = 25 30
}
includedir /etc/xinetd.d
```

These lines control the following aspects of **xinetd**:

- **instances** — Specifies the maximum number of simultaneous requests that **xinetd** can process.
- **log_type** — Configures **xinetd** to use the **authpriv** log facility, which writes log entries to the **/var/log/secure** file. Adding a directive such as **FILE /var/log/xinetdlog** would create a custom log file called **xinetdlog** in the **/var/log/** directory.
- **log_on_success** — Configures **xinetd** to log successful connection attempts. By default, the remote host's IP address and the process ID of the server processing the request are recorded.
- **log_on_failure** — Configures **xinetd** to log failed connection attempts or if the connection was denied.
- **cps** — Configures **xinetd** to allow no more than 25 connections per second to any given service. If this limit is exceeded, the service is retired for 30 seconds.
- **includedir** **/etc/xinetd.d/** — Includes options declared in the service-specific configuration files located in the **/etc/xinetd.d/** directory. Refer to Section 2.6.4.2, “The /etc/xinetd.d/ Directory” for more information.

> **Note**
>
> Often, both the **log_on_success** and **log_on_failure** settings in **/etc/xinetd.conf** are further modified in the service-specific configuration files. More information may therefore appear in a given service's log file than the **/etc/xinetd.conf** file may indicate. Refer to Section 2.6.4.3.1, “Logging Options” for further information.

2.6.4.2. The /etc/xinetd.d/ Directory

The **/etc/xinetd.d/** directory contains the configuration files for each service managed by **xinetd** and the names of the files are correlated to the service. As with **xinetd.conf**, this directory is read only when the **xinetd** service is started. For any changes to take effect, the administrator must restart the **xinetd** service.

The format of files in the **/etc/xinetd.d/** directory use the same conventions as **/etc/xinetd.conf**. The primary reason the configuration for each service is stored in a separate file is to make customization easier and less likely to affect other services.

To gain an understanding of how these files are structured, consider the **/etc/xinetd.d/krb5-telnet** file:

```
service telnet
{
  flags           = REUSE
  socket_type     = stream
  wait            = no
  user            = root
  server          = /usr/kerberos/sbin/telnetd
  log_on_failure  += USERID
  disable         = yes
}
```

These lines control various aspects of the **telnet** service:

- **service** — Specifies the service name, usually one of those listed in the **/etc/services** file.
- **flags** — Sets any of a number of attributes for the connection. **REUSE** instructs **xinetd** to reuse the socket for a Telnet connection.

  Note

  The **REUSE** flag is deprecated. All services now implicitly use the **REUSE** flag.

- **socket_type** — Sets the network socket type to **stream**.
- **wait** — Specifies whether the service is single-threaded (**yes**) or multi-threaded (**no**).
- **user** — Specifies which user ID the process runs under.
- **server** — Specifies which binary executable to launch.
- **log_on_failure** — Specifies logging parameters for **log_on_failure** in addition to those already defined in **xinetd.conf**.
- **disable** — Specifies whether the service is disabled (**yes**) or enabled (**no**).

Refer to the **xinetd.conf** man page for more information about these options and their usage.

2.6.4.3. Altering xinetd Configuration Files

A range of directives are available for services protected by **xinetd**. This section highlights some of the more commonly used options.

2.6.4.3.1. Logging Options

The following logging options are available for both **/etc/xinetd.conf** and the service-specific configuration files within the **/etc/xinetd.d/** directory.

The following is a list of some of the more commonly used logging options:

- **ATTEMPT** — Logs the fact that a failed attempt was made (**log_on_failure**).
- **DURATION** — Logs the length of time the service is used by a remote system (**log_on_success**).
- **EXIT** — Logs the exit status or termination signal of the service (**log_on_success**).
- **HOST** — Logs the remote host's IP address (**log_on_failure** and **log_on_success**).
- **PID** — Logs the process ID of the server receiving the request (**log_on_success**).
- **USERID** — Logs the remote user using the method defined in RFC 1413 for all multi-threaded stream services (**log_on_failure** and **log_on_success**).

For a complete list of logging options, refer to the **xinetd.conf** man page.

2.6.4.3.2. Access Control Options

Users of **xinetd** services can choose to use the TCP Wrappers hosts access rules, provide access control via the **xinetd** configuration files, or a mixture of both. Refer to Section 2.6.2, "TCP Wrappers Configuration Files" for more information about TCP Wrappers hosts access control files.

This section discusses using **xinetd** to control access to services.

Note

> Unlike TCP Wrappers, changes to access control only take effect if the **xinetd** administrator restarts the **xinetd** service.
> Also, unlike TCP Wrappers, access control through **xinetd** only affects services controlled by **xinetd**.

The **xinetd** hosts access control differs from the method used by TCP Wrappers. While TCP Wrappers places all of the access configuration within two files, **/etc/hosts.allow** and **/etc/hosts.deny**, **xinetd**'s access control is found in each service's configuration file in the **/etc/xinetd.d/** directory.

The following hosts access options are supported by **xinetd**:

- **only_from** — Allows only the specified hosts to use the service.
- **no_access** — Blocks listed hosts from using the service.
- **access_times** — Specifies the time range when a particular service may be used. The time range must be stated in 24-hour format notation, HH:MM-HH:MM.

The **only_from** and **no_access** options can use a list of IP addresses or host names, or can specify an entire network. Like TCP Wrappers, combining **xinetd** access control with the enhanced logging configuration can increase security by blocking requests from banned hosts while verbosely recording each connection attempt.

For example, the following **/etc/xinetd.d/telnet** file can be used to block Telnet access from a particular network group and restrict the overall time range that even allowed users can log in:

```
service telnet
{
  disable          = no
  flags            = REUSE
  socket_type      = stream
  wait             = no
  user             = root
  server           = /usr/kerberos/sbin/telnetd
  log_on_failure  += USERID
  no_access        = 172.16.45.0/24
  log_on_success  += PID HOST EXIT
  access_times     = 09:45-16:15
}
```

In this example, when a client system from the **172.16.45.0/24** network, such as **172.16.45.2**, tries to access the Telnet service, it receives the following message:

```
Connection closed by foreign host.
```

In addition, their login attempts are logged in **/var/log/messages** as follows:

```
Sep  7 14:58:33 localhost xinetd[5285]: FAIL: telnet address from=172.16.45.107
Sep  7 14:58:33 localhost xinetd[5283]: START: telnet pid=5285 from=172.16.45.107
Sep  7 14:58:33 localhost xinetd[5283]: EXIT: telnet status=0 pid=5285
duration=0(sec)
```

When using TCP Wrappers in conjunction with **xinetd** access controls, it is important to understand the relationship between the two access control mechanisms.

The following is the sequence of events followed by **xinetd** when a client requests a connection:

1. The **xinetd** daemon accesses the TCP Wrappers hosts access rules using a **libwrap.so** library call. If a deny rule matches the client, the connection is dropped. If an allow rule matches the client, the connection is passed to **xinetd**.
2. The **xinetd** daemon checks its own access control rules both for the **xinetd** service and the requested service. If a deny rule matches the client, the connection is dropped. Otherwise, **xinetd** starts an instance of the requested service and passes control of the connection to that service.

Important

Care should be taken when using TCP Wrappers access controls in conjunction with **xinetd** access controls. Misconfiguration can cause undesirable effects.

2.6.4.3.3. Binding and Redirection Options

The service configuration files for **xinetd** support binding the service to an IP address and redirecting incoming requests for that service to another IP address, hostname, or port.

Binding is controlled with the **bind** option in the service-specific configuration files and links the service to one IP address on the system. When this is configured, the **bind** option only allows requests to the correct IP address to access the service. You can use this method to bind different services to different network interfaces based on requirements.

This is particularly useful for systems with multiple network adapters or with multiple IP addresses. On such a system, insecure services (for example, Telnet), can be configured to listen only on the interface connected to a private network and not to the interface connected to the Internet.

The **redirect** option accepts an IP address or hostname followed by a port number. It configures the service to redirect any requests for this service to the specified host and port number. This feature can be used to point to another port number on the same system, redirect the request to a different IP address on the same machine, shift the request to a totally different system and port number, or any combination of these options. A user connecting to a certain service on a system may therefore be rerouted to another system without disruption.

The **xinetd** daemon is able to accomplish this redirection by spawning a process that stays alive for the duration of the connection between the requesting client machine and the host actually providing the service, transferring data between the two systems.

The advantages of the **bind** and **redirect** options are most clearly evident when they are used together. By binding a service to a particular IP address on a system and then redirecting requests for this service to a second machine that only the first machine can see, an internal system can be used to provide services for a totally different network. Alternatively, these options can be used to limit the exposure of a particular service on a multi-homed machine to a known IP address, as well as redirect any requests for that service to another machine especially configured for that purpose.

For example, consider a system that is used as a firewall with this setting for its Telnet service:

```
service telnet
{
  socket_type  = stream
  wait   = no
  server   = /usr/kerberos/sbin/telnetd
  log_on_success  += DURATION USERID
  log_on_failure  += USERID
  bind                    = 123.123.123.123
  redirect                = 10.0.1.13 23
}
```

The **bind** and **redirect** options in this file ensure that the Telnet service on the machine is bound to the external IP address (**123.123.123.123**), the one facing the Internet. In addition, any requests for Telnet service sent to **123.123.123.123** are redirected via a second network adapter to an internal IP address (**10.0.1.13**) that only the firewall and internal systems can access. The firewall then sends the communication between the two systems, and the connecting system thinks it is connected to **123.123.123.123** when it is actually connected to a different machine.

This feature is particularly useful for users with broadband connections and only one fixed IP address. When using Network Address Translation (NAT), the systems behind the gateway machine, which are using internal-only IP addresses, are not available from outside the gateway system. However, when certain services controlled by **xinetd** are configured with the **bind** and **redirect** options, the gateway machine can act as a proxy between outside systems and a particular internal machine configured to provide the service. In addition, the various **xinetd** access control and logging options are also available for additional protection.

2.6.4.3.4. Resource Management Options

The **xinetd** daemon can add a basic level of protection from Denial of Service (DoS) attacks. The following is a list of directives which can aid in limiting the effectiveness of such attacks:

- **`per_source`** — Defines the maximum number of instances for a service per source IP address. It accepts only integers as an argument and can be used in both **`xinetd.conf`** and in the service-specific configuration files in the **`xinetd.d/`** directory.
- **`cps`** — Defines the maximum number of connections per second. This directive takes two integer arguments separated by white space. The first argument is the maximum number of connections allowed to the service per second. The second argument is the number of seconds that **`xinetd`** must wait before re-enabling the service. It accepts only integers as arguments and can be used in either the **`xinetd.conf`** file or the service-specific configuration files in the **`xinetd.d/`** directory.
- **`max_load`** — Defines the CPU usage or load average threshold for a service. It accepts a floating point number argument.

  The load average is a rough measure of how many processes are active at a given time. See the **`uptime`**, **`who`**, and **`procinfo`** commands for more information about load average.

There are more resource management options available for **`xinetd`**. Refer to the **`xinetd.conf`** man page for more information.

2.6.5. Additional Resources

More information about TCP Wrappers and **`xinetd`** is available from system documentation and on the Internet.

2.6.5.1. Installed TCP Wrappers Documentation

The documentation on your system is a good place to start looking for additional configuration options for TCP Wrappers, **`xinetd`**, and access control.

- **`/usr/share/doc/tcp_wrappers-<version>/`** — This directory contains a **`README`** file that discusses how TCP Wrappers work and the various hostname and host address spoofing risks that exist.
- **`/usr/share/doc/xinetd-<version>/`** — This directory contains a **`README`** file that discusses aspects of access control and a **`sample.conf`** file with various ideas for modifying service-specific configuration files in the **`/etc/xinetd.d/`** directory.
- TCP Wrappers and **`xinetd`**-related man pages — A number of man pages exist for the various applications and configuration files involved with TCP Wrappers and **`xinetd`**. The following are some of the more important man pages:

  Server Applications

  - **`man xinetd`** — The man page for **`xinetd`**.

  Configuration Files

  - **`man 5 hosts_access`** — The man page for the TCP Wrappers hosts access control files.
  - **`man hosts_options`** — The man page for the TCP Wrappers options fields.
  - **`man xinetd.conf`** — The man page listing **`xinetd`** configuration options.

2.6.5.2. Useful TCP Wrappers Websites

- http://www.docstoc.com/docs/2133633/An-Unofficial-Xinetd-Tutorial — A thorough tutorial that discusses many different ways to optimize default **`xinetd`** configuration files to meet specific security goals.

2.6.5.3. Related Books

- *Hacking Linux Exposed* by Brian Hatch, James Lee, and George Kurtz; Osbourne/McGraw-Hill — An excellent security resource with information about TCP Wrappers and **`xinetd`**.

2.7. Virtual Private Networks (VPNs)

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 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. VPN implementations in hardware and software usually implement the IETF **`IPsec`** standards for establishing and authenticating VPN connections. Encryption of data is also standardized in the IETF **`IPsec`** standards, although FIPS regulations restrict which options can be used.

Some organizations employ hardware VPN solutions to augment security, while others use software or protocol-based implementations. Several vendors provide hardware VPN solutions, such as Cisco, Nortel, IBM, and Checkpoint. Free software based VPN solutions, such as Openswan, utilize the standardized *Internet Protocol Security* (IPsec) implementation. These VPN solutions, irrespective of whether they are hardware or software based, act as specialized routers that exist between the **`IP`** connection from one office to another.

2.7.1. How Does a VPN Work?

When a packet is transmitted from a client, it sends it through the VPN router or gateway, which adds authentication and optionally encrypts the data. IPsec has various modes. The most commonly used configuration uses *Tunnel Mode* with *Encapsulating Security Payload* (ESP). Some proprietary VPN clients use *Transport Mode* with ESP, despite the difficulties when Transport Mode is deployed with NAT. The deployment of *Authentication Header* (AH) is uncommon these days and is mostly used between core routers where the overhead of ESP is undesirable and encryption is not required. ESP as deployed by Red Hat Enterprise Linux always includes authenticated headers, although older versions and other racoon software based implementations are sometimes mistakenly deployed with ESP + AH, resulting in a double authentication layer.

The receiving VPN router strips the header information, decrypts the data, and routes it to its intended destination (either a workstation or other node on a network). Using a network-to-network connection, the receiving node on the local network receives the packets already decrypted and ready for processing. The encryption and decryption process in a network-to-network VPN connection is transparent to a local node.

With such a heightened level of security, an attacker must not only be able to intercept a packet, but needs to be able to decrypt the packet as well. Intruders who employ a man-in-the-middle attack between a server and client need access to at least one of the private keys for authenticating sessions. Additionally, obtaining such a private key only compromises current traffic and not traffic sent encrypted in the past. Because they employ several layers of authentication and encryption, VPNs are a secure and effective means of connecting multiple remote nodes to act as a unified intranet.

2.7.2. Openswan

Openswan is an open source, user space **`IPsec`** implementation available in Red Hat Enterprise Linux. It employs the key establishment protocol IKE (Internet Key Exchange) v1 and v2, implemented as a user-level daemon. Manual key establishment is also possible via **`ip xfrm`** commands, however this is not recommended. **Openswan** interfaces with the Linux kernel using netlink to transfer the encryption keys. Packet encryption and decryption happen in the Linux kernel.

Cryptographic Support

Openswan uses the NSS (Network Security Services) cryptographic library, which is required for FIPS security compliance. More information on the FIPS (Federal Information Processing Standard) can be found in Section 8.2, “Federal Information Processing Standard (FIPS)”.

Installation

Run the **`yum install openswan`** command to install **Openswan**.

2.7.2.1. Configuration

Locations

This section lists and explains important directories and files used for configuring **Openswan**.

- **`/etc/ipsec.d`** - main directory. Stores **Openswan** related files.
- **`/etc/ipsec.conf`** - master configuration file. Further **`*.conf`** configuration files can be created in **`/etc/ipsec.d`** for individual configurations.
- **`/etc/ipsec.secrets`** - master secrets file. Further **`*.secrets`** files can be created in **`/etc/ipsec.d`** for individual configurations.
- **`/etc/ipsec.d/cert*.db`** - Certificate database files. The old default NSS database file is **`cert8.db`**. From Red Hat Enterprise Linux 6 onwards, NSS sqlite databases are used in the **`cert9.db`** file.
- **`/etc/ipsec.d/key*.db`** - Key database files. The old default NSS database file is **`key3.db`**. From Red Hat Enterprise Linux 6 onwards, NSS sqlite databases are used in the **`key4.db`** file.
- **`/etc/ipsec.d/cacerts`** - The old Location for Certificate Authority (CA) certificates. It is recommend to import CA certificates into the NSS database files. This location will be obsoleted in the next version of Red Hat Enterprise Linux.
- **`/etc/ipsec.d/certs`** - The old location for machine certificates. It is recommend to import machine certificates into the NSS database files. This location will be obsoleted in the next version of Red Hat Enterprise Linux.
- **`/etc/ipsec.d/policies`** - Opportunistic Encryption groups policies. These are not in use in Red Hat Enterprise Linux 6.
- **`/etc/ipsec.d/nsspassword`** - NSS password file. This file does not exist by default, and is required if the NSS database in use is created with a password.

Global Configuration Parameters

This section lists some of the configuration options available for the global **`config setup`** section in **`/etc/ipsec.conf`**.

- **`protostack`** - defines which protocol stack is used. The default option in Red Hat Enterprise Linux 6 is *netkey*. Other valid values are *klips* and *mast*. These alternative **`IPsec`** kernel stacks require a third-party kernel module which is not supported by Red Hat Enterprise Linux. Alternative stacks are mostly in use with embedded Linux devices.

- **`nat_traversal`** - defines if NAT-traversal (RFC 3947) for connections are accepted. Default is yes.
- **`dumpdir`** - defines the location for core dump files. Changing this will require changing the SELinux policies to match the new location.
- **`nhelpers`** - defines the number of helper threads used for cryptographic operations.
- **`virtual_private`** - subnets allowed for the client connection. Ranges that may exist behind a NAT router through which a client connects. This should contain the RFC 1918 address space with an exception for any LAN range used by the server.
- **`plutorestartoncrash`** - set to yes by default.
- **`plutostderrlog`** - When enabled, the log file name to use instead of the syslog facility.

Connection Specific Parameters

This section lists some of the configuration options available per connection in a **`conn`** section.

- **`auto`** - defines the startup profile for this connection. The default is *ignore*. Other valid values are *start* (initiate), *add* (respond-only) and *route* (Initiate on-demand).
- **`type`** - defines the type of tunnel policy used. The default is *tunnel* for Tunnel Mode. Other valid values are *transport* for Transport Mode (used with L2TP) and *passthrough* which bypasses **`IPsec`** completely.
- **`authby`** - defines the type of authentication used. The default is *rsasig* for raw RSA keys. . Other valid values are *secret* for Pre-Shared Key (PSK) and *never* for pass-through connections.
- **`rekey`** - defines whether the connection should rekey when it reaches its expiry time. The default is *yes* and is usually used for static VPNs. The value *no* is mostly used for supporting dynamic VPN clients.

Further details about **Openswan** configuration can be found in the **`ipsec.conf(5)`** and **`ipsec.secrets(5)`** manual pages.

2.7.2.2. Commands

This section provides examples of commands used for **Openswan**. Note that except for checking the status of a service, these commands must be run as root.

Service Commands for Openswan

-
  ```
  service ipsec start
  ```

-
  ```
  service ipsec stop
  ```

-
  ```
  service ipsec status
  ```

Loading and Unloading a Connection

-
  ```
  ipsec auto --add connection_name
  ```

-
  ```
  ipsec auto --delete connection_name
  ```

Initiating, On-demand and Terminating a Connection

-
  ```
  ipsec auto --up connection_name
  ```

-
  ```
  ipsec auto --route connection_name
  ```

-
  ```
  ipsec auto --down connection_name
  ```

Checking the IPsec Kernel Subsystem

-
  ```
  ip xfrm policy
  ```

-
  ```
  ip xfrm state
  ```

-
  ```
  ip -s xfrm monitor
  ```

Checking the IPsec User Space Subsystem

-
  ```
  ipsec auto --status
  ```

-
  ```
  ipsec verify
  ```

Raw RSA for IPsec

To enable an **IPsec** host to use raw RSA keys (without a Certificate Authority), the NSS database must be initialized and a new raw RSA key generated.

> **Note**
>
> If the **Openswan IPsec** service has been started before initializing the NSS database, you must shutdown the **IPsec** service and remove the ***.db** files in **/etc/ipsec.d/** before you can add raw RSA keys or certificates to the NSS database.

-
  ```
  certutil -N -d /etc/ipsec.d
  ```

> **Note**
>
> Commands that generate raw RSA keys depend on the system's entropy. Depending on system activity and key size, these commands can take several minutes to complete.

- Generate raw RSA keys:

  -
    ```
    ipsec newhostkey --configdir /etc/ipsec.d --output /etc/ipsec.d/name-of-file.secret
    ```

- List raw RSA keys for use in **IPsec** configuration files

  -
    ```
    ipsec showhostkey --left | --right
    ```

- List raw RSA keys for publication in DNS

-
  ```
  ipsec showhostkey --ipseckey
  ```

Configuring Certificates with IPsec

Creating and maintaining a Certificate Agency should be done on a separate machine. Only the created PKCS #12 files need to be transferred off this machine onto the **`IPsec`** servers. Exporting the PKCS #12 (**`.p12`**) files requires the CA administrator to use a password to encrypt the **`.p12`** file. Importing these files requires the system administrator to know this password. This allows the **`.p12`** files themselves to be safe to email, provided the password is shared via a different method, such as a phone call.

Create a directory in your home directory to hold the certificates and initialize the database:

```
mkdir ~/CAstore
```

```
certutil -N -d ~/CAstore
```

Create a (self-signed) CA certificate:

```
certutil -S -k rsa -n ca.example.com -s \ "CN=ca.example.com" -w 12 -t
"C,C,C" -x -d ~/CAstore
```

Create a host certificate signed by this CA:

```
certutil -S -k rsa -c ca.example.com -n west.example.com -s
"CN=west.example.com" -w 12 -t "u,u,u" -d ~/CAstore
```

Create an export file in PKCS #12 format for transfer to a target machine:

```
pk12util -o west.p12 -n west.example.com -d ~/CAstore
```

Import an exported machine certificate to the **`IPsec`** NSS database:

```
pk12util -i west.example.com.p12 -d /etc/ipsec.d
```

List the contents of the NSS database (**`IPsec`** gateway or CA):

```
certutil -L -d /etc/ipsec.d
```

```
certutil -L -d ~/CAstore
```

2.7.3. IPsec VPN Using Openswan

Installing Openswan

To install **Openswan**, issue the following command as root:

```
~]# yum install openswan
```

You will be asked to accept the installation of two keys as well as the package.

Initialize the *network security services* (NSS) database by issuing the following command as root:

```
~]# certutil -N -d /etc/ipsec.d/
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
Re-enter password:
```

If you do not wish to use a password for NSS, just press **`Enter`** twice when prompted for the password. If you do enter a password then you will have to re-enter it every time **Openswan** is started, such as every time the system is booted.

Restore default SELinux security contexts:

```
~]# restorecon -Rv /etc/ipsec.d
```

To start the **`ipsec`** daemon provided by **Openswan**, issue the following command as root:

```
~]# service ipsec start
ipsec_setup: Starting Openswan IPsec U2.6.32/K2.6.32-358.el6.x86_64...
```

To ensure that **Openswan** will start when the system starts, issue the following command as root:

```
~]# chkconfig ipsec on
```

Configure any intermediate as well as host-based firewalls to permit the **`ipsec`** service. See Section 2.8.1, “Netfilter and IPTables” for information on firewalls and allowing specific services to pass through. **Openswan** requires the firewall to allow the following packets:

- **`UDP`** port 500 for the **`Internet Key Exchange`** (IKE) protocol
- **`UDP`** port 4500 for **`IKE NAT-Traversal`**
- Protocol 50 for **`Encapsulated Security Payload`** (ESP) **`IPsec`** packets
- Protocol 51 for **`Authenticated Header`** (AH) **`IPsec`** packets (uncommon)

We present three examples of using **Openswan** to set up an **`IPsec`** VPN. The first example is for connecting two hosts together so that they may communicate securely. The second example is connecting two sites together to form one network. The third example is supporting roaming users, known as *road warriors* in this context.

2.7.4. VPN Configurations Using Openswan

Openswan does not use the terms “source” or “destination”. Instead, it uses the terms “left” and “right” to refer to end points (the hosts). This allows the same configuration to be used on both end points in most cases, although most administrators use “left” for the local host and “right” for the remote host. The keywords containing “left” and “right” are arbitrary choices. You can interpret this literally that left is the host on the left of your network diagram. **Openswan** will auto-detect whether it is “left” or “right”. In examples, the host referred to as “left” by the **Openswan** commands is also given a host name to match. Usually “west” and “east” are used in the host name to match “left” and “right” respectively. Throughout this guide, we will use **`west.example.com`** as “left” and **`east.example.com`** as “right”.

There are three commonly used methods for authentication of endpoints:

- *Pre-Shared Keys* (PSK) is the simplest authentication method. PSK's should consist of random characters and have a length of at least 20 characters. Due to the dangers of non-random and short

PSKs, this method is not available when the system is running in FIPS mode

- Raw RSA keys are commonly used for static host-to-host or subnet-to-subnet **IPsec** configurations. The hosts are manually configured with each other's public RSA key. This method does not scale well when dozens or more hosts all need to setup **IPsec** tunnels to each other.
- **X.509 certificates** are commonly used for large scale deployments where there are many hosts that need to connect to a common **IPsec** gateway. A central *certificate authority* (CA) is used to sign RSA certificates for hosts or users. This central CA is responsible for relaying trust, including the revocations of individual hosts or users.

2.7.5. Host-To-Host VPN Using Openswan

To configure **Openswan** to create a host-to-host **IPsec** VPN, between two hosts referred to as "left" and "right", enter the following commands as root on the host called "left" to create a new raw RSA keypair:

```
~]# ipsec newhostkey --configdir /etc/ipsec.d \
        --output /etc/ipsec.d/ipsec.secrets --bits 4096
Generated RSA key pair using the NSS database
```

This generates an RSA keypair for the host. The process of generating RSA keys can take many minutes, especially on virtual machines with low entropy.

To view the public key, issue the following command as root, on the host referred to as "left":

```
~]# ipsec showhostkey --left
ipsec showhostkey nss directory showhostkey: /etc/ipsec.d
# rsakey AQO+NIez4
leftrsasigkey=0sAQO+NIez4bxcib5FPjT3jF3S6Mrz9NACaD5B4wPXFuhxQmy6c8GNX1A9yB0vvLWon
[...] W8rQIf4NrL6eGd5r9HwIPT7
```

You will need this key to add to the configuration file as explained below.

Enter the following commands as root on the host referred to as "right":

```
~]# ipsec newhostkey --configdir /etc/ipsec.d \
        --output /etc/ipsec.d/ipsec.secrets --bits 4096
Generated RSA key pair using the NSS database
```

To view the public key, issue the following command as root on the host referred to as "right":

```
~]# ipsec showhostkey --right
# rsakey AQO3fwC6n
rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
```

You will need this key to add to the configuration file.

To make a configuration file for this host-to-host tunnel, the lines **leftrsasigkey=** and **rightrsasigkey=** from above, are added to a custom configuration file placed in the ***/etc/ipsec.d/*** directory. To enable **Openswan** to read the custom configurations files, use an editor running as root to edit the main configuration file, ***/etc/ipsec.conf***, and enable the following line by removing the # comment character so that it looks as follows:

```
include /etc/ipsec.d/*.conf
```

Using an editor running as root, create a file with a suitable name in the following format:

```
/etc/ipsec.d/my_host-to-host.conf
```

Edit the file as follows:

```
conn mytunnel
    leftid=@west.example.com
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east.example.com
    right=192.1.2.45
    rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    authby=rsasig
    # load and initiate automatically
    auto=start
```

You can use the identical configuration file on both left and right hosts. They will auto-detect if they are “left” or “right”. Ensure the **leftrsasigkey** value is obtained from the “left” host and the **rightrsasigkey** value is obtained from the “right” host.

Ensure **ipsec** has been started:

```
~]# service ipsec start
ipsec_setup: Starting Openswan IPsec U2.6.32/K2.6.32-412.el6.x86_64...
ipsec_setup: /usr/libexec/ipsec/addconn Non-fips mode set in
/proc/sys/crypto/fips_enabled
```

Issue the following command as root to load the **IPsec** tunnel:

```
~]# ipsec auto --add mytunnel
/usr/libexec/ipsec/addconn Non-fips mode set in /proc/sys/crypto/fips_enabled
```

To bring up the tunnel, issue the following command as root, on both left and right hosts:

```
~]# ipsec auto --up mytunnel
104 "mytunnel" #1: STATE_MAIN_I1: initiate
003 "mytunnel" #1: received Vendor ID payload [Dead Peer Detection]
106 "mytunnel" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "mytunnel" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "mytunnel" #1: received Vendor ID payload [CAN-IKEv2]
004 "mytunnel" #1: STATE_MAIN_I4: ISAKMP SA established {auth=OAKLEY_RSA_SIG
cipher=aes_128 prf=oakley_sha group=modp2048}
117 "mytunnel" #2: STATE_QUICK_I1: initiate
004 "mytunnel" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel mode
{ESP=>0x63ad7e17 <0x4841b647 xfrm=AES_128-HMAC_SHA1 NATOA=none NATD=none
DPD=none}
```

2.7.5.1. Verify Host-To-Host VPN Using Openswan

The **IKE** negotiation takes place on **UDP** port 500. **IPsec** packets show up as **Encapsulated Security Payload** (ESP) packets. When the VPN connection needs to pass through a NAT router, the **ESP** packets are encapsulated in **UDP** packets on port 4500.

To verify that packets are being sent via the VPN tunnel issue a command as root in the following format:

```
~]# tcpdump -n -i interface esp and udp port 500 and udp port 4500
00:32:32.632165 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1a), length
132
00:32:32.632592 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1a), length
132
00:32:32.632592 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489, seq 7,
length 64
00:32:33.632221 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1b), length
132
00:32:33.632731 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1b), length
132
00:32:33.632731 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489, seq 8,
length 64
00:32:34.632183 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1c), length
132
00:32:34.632607 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1c), length
132
00:32:34.632607 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489, seq 9,
length 64
00:32:35.632233 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1d), length
132
00:32:35.632685 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1d), length
132
00:32:35.632685 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489, seq 10,
length 64
```

Where ***interface*** is the interface known to carry the traffic. To end the capture with **tcpdump**, press **Ctrl**+**C**.

Note

The **tcpdump** commands interacts a little unexpectedly with **IPsec**. It only sees the outgoing encrypted packet, not the outgoing plaintext packet. It does see the encrypted incoming packet, as well as the decrypted incoming packet. If possible, run **tcpdump** on a router between the two machines and not on one of the endpoints itself.

2.7.6. Site-to-Site VPN Using Openswan

In order for **Openswan** to create a site-to-site **IPsec** VPN, joining together two networks, an **IPsec** tunnel is created between two hosts, endpoints, which are configured to permit traffic from one or more subnets to pass through. They can therefore be thought of as gateways to the remote portion of the network. The configuration of the site-to-site VPN differs from the host-to-host VPN only in that one or more network or subnet (address) ranges must be specified in the configuration file.

To configure **Openswan** to create a site-to-site **IPsec** VPN, first configure a host-to-host **IPsec** VPN as described in Section 2.7.5, “Host-To-Host VPN Using Openswan” and then copy or move the file to a file with suitable name such as **/etc/ipsec.d/my_site-to-site.conf**. Using an editor running as root, edit the custom configuration file **/etc/ipsec.d/my_site-to-site.conf** as follows:

```
conn mysubnet
     also=mytunnel
     leftsubnet=192.0.1.0/24
     rightsubnet=192.0.2.0/24

conn mysubnet6
     also=mytunnel
     connaddrfamily=ipv6
     leftsubnet=2001:db8:0:1::/64
     rightsubnet=2001:db8:0:2::/64

conn mytunnel
    auto=start
    leftid=@west.example.com
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east.example.com
    right=192.1.2.45
    rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    authby=rsasig
```

To bring the tunnels up, restart **Openswan** or manually load and initiate all the connections using the following commands as root:

```
~]# ipsec auto --add mysubnet
/usr/libexec/ipsec/addconn Non-fips mode set in /proc/sys/crypto/fips_enabled
```

```
~]# ipsec auto --add mysubnet6
/usr/libexec/ipsec/addconn Non-fips mode set in /proc/sys/crypto/fips_enabled
```

```
~]# ipsec auto --add mytunnel
/usr/libexec/ipsec/addconn Non-fips mode set in /proc/sys/crypto/fips_enabled
```

```
~]# ipsec auto --up mysubnet
104 "mysubnet" #1: STATE_MAIN_I1: initiate
003 "mysubnet" #1: received Vendor ID payload [Dead Peer Detection]
106 "mysubnet" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "mysubnet" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "mysubnet" #1: received Vendor ID payload [CAN-IKEv2]
004 "mysubnet" #1: STATE_MAIN_I4: ISAKMP SA established {auth=OAKLEY_RSA_SIG
cipher=aes_128 prf=oakley_sha group=modp2048}
117 "mysubnet" #2: STATE_QUICK_I1: initiate
004 "mysubnet" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel mode
{ESP=>0x9414a615 <0x1a8eb4ef xfrm=AES_128-HMAC_SHA1 NATOA=none NATD=none
DPD=none}
```

```
~]# ipsec auto --up mysubnet6
117 "mysubnet" #2: STATE_QUICK_I1: initiate
004 "mysubnet" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel mode
{ESP=>0x06fe2099 <0x75eaa862 xfrm=AES_128-HMAC_SHA1 NATOA=none NATD=none
DPD=none}
```

```
~]# ipsec auto --up mytunnel
117 "mysubnet" #2: STATE_QUICK_I1: initiate
004 "mysubnet" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel mode
{ESP=>0x06fe209a <0x75eaa863 xfrm=AES_128-HMAC_SHA1 NATOA=none NATD=none
DPD=none}
```

2.7.6.1. Verify Site-to-Site VPN Using Openswan

Verifying that packets are being sent via the VPN tunnel is the same procedure as explained in Section 2.7.5.1, “Verify Host-To-Host VPN Using Openswan”.

2.7.7. Site-to-Site Single Tunnel VPN Using Openswan

Often, when a site-to-site tunnel is built, the gateways need to communicate to each other using their internal **IP** addresses instead of their public **IP** addresses. This can be accomplished using a single tunnel. If the left host, with host name **west**, has internal **IP** address **192.0.1.254** and the right host, with host name **east**, has internal **IP** address **192.0.2.254**, one could use the following configuration using a single tunnel:

```
conn mysubnet
     leftid=@west.example.com
     leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
     left=192.1.2.23
     leftsourceip=192.0.1.254
     leftsubnet=192.0.1.0/24
     rightid=@east.example.com
     rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
     right=192.1.2.45
     rightsourceip=192.0.2.254
     rightsubnet=192.0.2.0/24
     auto=start
     authby=rsasig
```

2.7.8. Subnet Extrusion Using Openswan

Often **IPsec** is deployed in a hub-and-spoke architecture. Each leaf node has an **IP** range that is part of a larger range. Leaves communicate with each other via the hub. This is called *subnet extrusion*. In the example below we configure the head office with **10.0.0.0/8** and two branches that use a smaller **/24** subnet.

At the head office:

```
conn branch1
    left=1.2.3.4
    leftid=@headoffice
    leftsubnet=0.0.0.0/0
    leftrsasigkey=0sA[...]
    #
    right=5.6.7.8
    rightid=@branch1
    righsubnet=10.0.1.0/24
    rightrsasigkey=0sAXXXX[...]
    #
    auto=start
    authby=rsasigkey

conn branch2
    left=1.2.3.4
    leftid=@headoffice
    leftsubnet=0.0.0.0/0
    leftrsasigkey=0sA[...]
    #
    right=10.11.12.13
    rightid=@branch2
    righsubnet=10.0.2.0/24
    rightrsasigkey=0sAYYYY[...]
    #
    auto=start
    authby=rsasigkey
```

At the “branch1” office we use the same connection. Additionally we use a pass-through connection to exclude our local LAN traffic from being sent through the tunnel:

```
conn branch1
    left=1.2.3.4
    leftid=@headoffice
    leftsubnet=0.0.0.0/0
    leftrsasigkey=0sA[...]
    #
    right=10.11.12.13
    rightid=@branch2
    righsubnet=10.0.1.0/24
    rightrsasigkey=0sAYYYY[...]
    #
    auto=start
    authby=rsasigkey

conn passthrough
    left=1.2.3.4
    right=0.0.0.0
    leftsubnet=10.0.1.0/24
    rightsubnet=10.0.1.0/24
    authby=never
    type=passthrough
    auto=route
```

2.7.9. Road Warrior Application Using Openswan

Road Warriors are traveling users with mobile clients with a dynamically assigned **IP** address, such as laptops. These are authenticated using certificates.

On the server:

```
conn roadwarriors
    left=1.2.3.4
    # if access to the LAN is given, enable this
    #leftsubnet=10.10.0.0/16
    leftcert=gw.example.com
    leftid=%fromcert
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # allow clients to be behind a NAT router
    rightsubnet=vhost:%priv,%no
    authby=rsasigkey
    # load connection, don't initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=30
    dpdtimeout=120
    dpdaction=%clear
```

On the mobile client, the Road Warrior's device, we need to use a slight variation of the above connection:

```
conn roadwarriors
    # pick up our dynamic IP
    left=%defaultroute
    leftcert=myname.example.com
    leftid=%fromcert
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this
    #rightsubnet=10.10.0.0/16
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasigkey
    # Initiate connection
    auto=start
```

2.7.10. Additional Resources

The following sources of information provide additional resources regarding **Openswan** and the **ipsec** daemon.

2.7.10.1. Installed Documentation

- **ipsec(8)** man page — Describes command options for **ipsec**.
- **ipsec.conf(5)** man page — Contains information to configure **ipsec**.
- **ipsec.secrets(5)** man page — Contains information to configure **ipsec**.
- **ipsec_auto(8)** man page — Describes the use of the **auto** command line client for manipulating automatically-keyed **Openswan IPsec** connections.
- **ipsec_rsasigkey(8)** man page — Describes the tool used to generate RSA signature keys.
- **/usr/share/doc/openswan-*version*/README.nss** from the *openswan-doc* package — Describes the commands for using raw RSA keys and certificates with the NSS crypto library used with the **Openswan pluto** daemon.

The **openswan-doc** package can be installed via **yum install** for additional documentation.

2.7.10.2. Useful Websites

http://www.openswan.org
The website of the upstream project.

http://www.mozilla.org/projects/security/pki/nss/
Network Security Services (NSS) project.

2.8. 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 tools to assist administrators and security engineers with network-level access control issues.

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 stand-alone hardware solutions, such as firewall appliances by Cisco, Nokia, and Sonicwall. Vendors such as Checkpoint, McAfee, and Symantec have also developed proprietary software firewall solutions for home and business markets.

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 2.6, “Firewall Types” details three common types of firewalls and how they function:

Table 2.6. 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. The Linux kernel has built-in NAT functionality through the Netfilter kernel subsystem.	Can be configured transparently to machines on a LAN. Protection of many machines and services behind one or more external IP addresses 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 through 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. This helps to reduce bandwidth consumption. Proxy services can be logged and monitored closely, allowing tighter control over resource utilization on the network.	Proxies are often application-specific (HTTP, Telnet, etc.), or protocol-restricted (most proxies work with TCP-connected services only). Application services cannot run behind a proxy, so your application servers must use a separate form of network security. Proxies can become a network bottleneck, as all requests and transmissions are passed through one source rather than directly from a client to a remote service.

2.8.1. 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 using the **`iptables`** tool.

2.8.1.1. IPTables Overview

The power and flexibility of Netfilter is implemented using the **`iptables`** administration tool, a command line tool similar in syntax to its predecessor, **`ipchains`**, which Netfilter/iptables replaced in the Linux kernel 2.4 and above.

`iptables` uses the Netfilter subsystem to enhance network connection, inspection, and processing. **`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, refer to Section 2.8.9, “IPTables”.

2.8.2. Basic Firewall Configuration

Just as a firewall in a building attempts to prevent a fire from spreading, a computer firewall attempts to prevent malicious software from spreading to your computer. It also helps to prevent unauthorized users from accessing your computer.

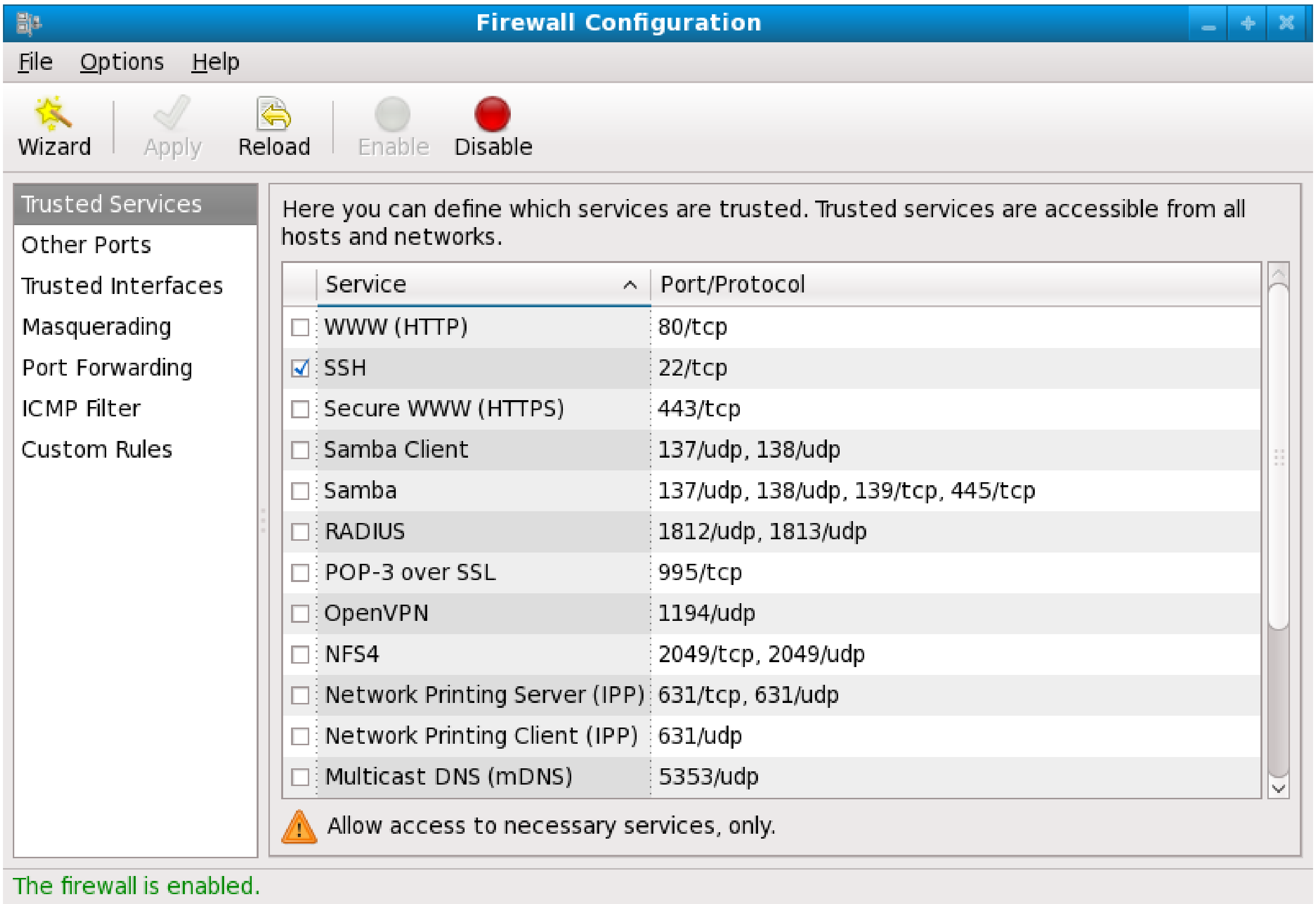
In a default Red Hat Enterprise Linux installation, a firewall exists between your computer or network and any untrusted networks, for example the Internet. It determines which services on your computer remote users can access. A properly configured firewall can greatly increase the security of your system. It is recommended that you configure a firewall for any Red Hat Enterprise Linux system with an Internet connection.

2.8.2.1. Firewall Configuration Tool

During the **`Firewall Configuration`** screen of the Red Hat Enterprise Linux installation, you were given the option to enable a basic firewall as well as to allow specific devices, incoming services, and ports.

After installation, you can change this preference by using the **Firewall Configuration Tool**.

To start this application, either select **System** → **Administration** → **Firewall** from the panel, or type **`system-config-firewall`** at a shell prompt.



Here you can define which services are trusted. Trusted services are accessible from all hosts and networks.

	Service	Port/Protocol
□	WWW (HTTP)	80/tcp
☑	SSH	22/tcp
□	Secure WWW (HTTPS)	443/tcp
□	Samba Client	137/udp, 138/udp
□	Samba	137/udp, 138/udp, 139/tcp, 445/tcp
□	RADIUS	1812/udp, 1813/udp
□	POP-3 over SSL	995/tcp
□	OpenVPN	1194/udp
□	NFS4	2049/tcp, 2049/udp
□	Network Printing Server (IPP)	631/tcp, 631/udp
□	Network Printing Client (IPP)	631/udp
□	Multicast DNS (mDNS)	5353/udp

Allow access to necessary services, only.

The firewall is enabled.

Figure 2.5. Firewall Configuration Tool

Note

The **Firewall Configuration Tool** only configures a basic firewall. If the system needs more complex rules, refer to Section 2.8.9, “IPTables” for details on configuring specific **`iptables`** rules.

As of **Red Hat Enterprise Linux 6.5**, the **`iptables`** and **`ip6tables`** services now provide the ability to assign a fallback firewall configuration if the default configuration cannot be applied. If application of the firewall rules from **`/etc/sysconfig/iptables`** fails, the fallback file is applied if it exists. The fallback file is named **`/etc/sysconfig/iptables.fallback`** and uses the same file format as **`/etc/sysconfig/iptables`**. If application of the fallback file also fails, there is no further fallback. To create a fallback file, use the standard firewall configuration tool and rename or copy the file to the fallback file.

For the **`ip6tables`** service, replace all occurrences of **`iptables`** with **`ip6tables`** in the above examples.

2.8.2.2. Enabling and Disabling the Firewall

Select one of the following options for the firewall:

- **`Disabled`** — Disabling the firewall provides complete access to your system and does no security checking. This should only be selected if you are running on a trusted network (not the Internet) or need to configure a custom firewall using the **`iptables`** command line tool.

Warning

Firewall configurations and any customized firewall rules are stored in the **`/etc/sysconfig/iptables`** file. If you choose **`Disabled`** and click **`OK`**, these configurations and firewall rules will be lost.

- **`Enabled`** — This option configures the system to reject incoming connections that are not in response to outbound requests, such as DNS replies or DHCP requests. If access to services running on this machine is needed, you can choose to allow specific services through the firewall.

  If you are connecting your system to the Internet, but do not plan to run a server, this is the safest choice.

2.8.2.3. Trusted Services

Enabling options in the **`Trusted services`** list allows the specified service to pass through the firewall.

`WWW (HTTP)`

The HTTP protocol is used by Apache (and by other Web servers) to serve web pages. If you plan on making your Web server publicly available, select this check box. This option is not required for viewing pages locally or for developing web pages. This service requires that the **`httpd`** package be installed.

Enabling **`WWW (HTTP)`** will not open a port for HTTPS, the SSL version of HTTP. If this service is required, select the **`Secure WWW (HTTPS)`** check box.

`FTP`

The FTP protocol is used to transfer files between machines on a network. If you plan on making your FTP server publicly available, select this check box. This service requires that the **`vsftpd`** package be installed.

`SSH`

Secure Shell (SSH) is a suite of tools for logging into and executing commands on a remote machine. To allow remote access to the machine via SSH, select this check box. This service requires that the **`openssh-server`** package be installed.

`Telnet`

Telnet is a protocol for logging into remote machines. Telnet communications are unencrypted and provide no security from network snooping. Allowing incoming Telnet access is not recommended. To allow remote access to the machine via telnet, select this check box. This service requires that the **`telnet-server`** package be installed.

`Mail (SMTP)`

SMTP is a protocol that allows remote hosts to connect directly to your machine to deliver mail. You do not need to enable this service if you collect your mail from your ISP's server using POP3 or IMAP, or if you use a tool such as **`fetchmail`**. To allow delivery of mail to your machine, select this check box. Note that an improperly configured SMTP server can allow remote machines to use your server to send spam.

`NFS4`

The Network File System (NFS) is a file sharing protocol commonly used on *NIX systems. Version 4 of this protocol is more secure than its predecessors. If you want to share files or directories on your system with other network users, select this check box.

`Samba`

Samba is an implementation of Microsoft's proprietary SMB networking protocol. If you need to share files, directories, or locally-connected printers with Microsoft Windows machines, select this check box.

2.8.2.4. Other Ports

The **Firewall Configuration Tool** includes an **`Other ports`** section for specifying custom IP ports as being trusted by **`iptables`**. For example, to allow IRC and Internet printing protocol (IPP) to pass through the firewall, add the following to the **`Other ports`** section:

`194:tcp,631:tcp`

2.8.2.5. Saving the Settings

Click **`OK`** to save the changes and enable or disable the firewall. If **`Enable firewall`** was selected, the options selected are translated to **`iptables`** commands and written to the **`/etc/sysconfig/iptables`** file. The **`iptables`** service is also started so that the firewall is activated immediately after saving the selected options. If **`Disable firewall`** was selected, the **`/etc/sysconfig/iptables`** file is removed and the **`iptables`** service is stopped immediately.

The selected options are also written to the **`/etc/sysconfig/system-config-firewall`** file so that the settings can be restored the next time the application is started. Do not edit this file manually.

Even though the firewall is activated immediately, the **`iptables`** service is not configured to start automatically at boot time. Refer to Section 2.8.2.6, "Activating the IPTables Service" for more information.

2.8.2.6. Activating the IPTables Service

The firewall rules are only active if the **`iptables`** service is running. To manually start the service, use the following command as the root user:

```
~]# service iptables restart
iptables: Applying firewall rules:                         [  OK  ]
```

To ensure that **`iptables`** starts when the system is booted, use the following command:

```
~]# chkconfig --level 345 iptables on
```

2.8.3. Using IPTables

The first step in using **`iptables`** is to start the **`iptables`** service. Use the following command as the root user to start the **`iptables`** service:

```
~]# service iptables restart
iptables: Applying firewall rules:                         [  OK  ]
```

Note

> The **ip6tables** service can be turned off if you intend to use the **iptables** service only. If you deactivate the **ip6tables** service, remember to deactivate the IPv6 network also. Never leave a network device active without the matching firewall.

To force **iptables** to start by default when the system is booted, use the following command as the root user:

```
~]# chkconfig --level 345 iptables on
```

This forces **iptables** to start whenever the system is booted into runlevel 3, 4, or 5.

2.8.3.1. IPTables Command Syntax

The following sample **iptables** command illustrates the basic command syntax:

```
iptables -A <chain> -j <target>
```

The **-A** option specifies that the rule be appended to *<chain>*. Each chain is comprised of one or more *rules*, and is therefore also known as a *ruleset*.

The three built-in chains are INPUT, OUTPUT, and FORWARD. These chains are permanent and cannot be deleted. The chain specifies the point at which a packet is manipulated.

The **-j *<target>*** option specifies the target of the rule; i.e., what to do if the packet matches the rule. Examples of built-in targets are ACCEPT, DROP, and REJECT.

Refer to the **iptables** man page for more information on the available chains, options, and targets.

2.8.3.2. Basic Firewall Policies

Establishing basic firewall policies creates a foundation for building more detailed, user-defined rules.

Each **iptables** chain is comprised of a default policy, and zero or more rules which work in concert with the default policy to define the overall ruleset for the firewall.

The default policy for a chain can be either DROP or ACCEPT. Security-minded administrators typically implement a default policy of DROP, and only allow specific packets on a case-by-case basis. For example, the following policies block all incoming and outgoing packets on a network gateway:

```
~]# iptables -P INPUT DROP
~]# iptables -P OUTPUT DROP
```

It is also 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
```

When you have established the default policies for each chain, you can create and save further rules for your particular network and security requirements.

The following sections describe how to save **`iptables`** rules and outline some of the rules you might implement in the course of building your **`iptables`** firewall.

2.8.3.3. Saving and Restoring IPTables Rules

Changes to **`iptables`** are transitory; if the system is rebooted or if the **`iptables`** service is restarted, the rules are automatically flushed and reset. To save the rules so that they are loaded when the **`iptables`** service is started, use the following command as the root user:

```
~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

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

2.8.4. Common IPTables Filtering

Preventing remote attackers from accessing a LAN is one of the most important aspects of network security. 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 to 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 --dport 80 -j ACCEPT
```

This allows users to browse websites that communicate using the standard port 80. To allow access to secure websites (for example, https://www.example.com/), you also need to provide access to port 443, as follows:

```
~]# iptables -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

Important

When creating an **`iptables`** ruleset, order is important.
If a rule specifies that any packets from the 192.168.100.0/24 subnet be dropped, and this is followed by a rule that allows packets from 192.168.100.13 (which is within the dropped subnet), then the second rule is ignored.
The rule to allow packets from 192.168.100.13 must precede the rule that drops the remainder of the subnet.
To insert a rule in a specific location in an existing chain, use the **`-I`** option. For example:

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

This 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. Secure services, for example SSH, can be used for encrypted remote connection to LAN services.

Administrators with PPP-based resources (such as modem banks or bulk ISP accounts), dial-up access can be used to securely circumvent firewall barriers. Because they are direct connections, modem connections are typically behind a firewall/gateway.

For remote users with broadband connections, however, special cases can be made. You can configure **`iptables`** to accept connections from remote SSH clients. For example, the following rules allow remote SSH access:

```
~]# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
~]# iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

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 *Network Address Translation* (NAT) with **`iptables`** filtering rules.

2.8.5. `FORWARD` and NAT Rules

Most ISPs provide only a limited number of publicly routable IP addresses to the organizations they serve.

Administrators must, therefore, find alternative ways to share access to Internet services without giving public IP addresses to every node on the LAN. Using private IP addresses is the most common way of allowing 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, firewalls/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 abnormal usage of network resources.

The **`FORWARD`** chain 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), use the following rules:

```
~]# 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. This prevents machines that run Red Hat Enterprise Linux from functioning as dedicated edge routers. To enable IP forwarding, use the following command as the root user:

```
~]# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

This configuration change is only valid for the current session; it does not persist beyond a reboot or network service restart. To permanently set IP forwarding, edit the **/etc/sysctl.conf** file as follows:
Locate the following line:

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

Edit it to read as follows:

```
net.ipv4.ip_forward = 1
```

As the root user, run the following command to enable the change to the **sysctl.conf** file:

```
~]# sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
[output truncated]
```

2.8.5.1. Postrouting and IP Masquerading

Accepting forwarded packets via the firewall's internal IP device allows LAN nodes to communicate with each other; however they still cannot 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
```

This 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.

2.8.5.2. Prerouting

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 want to forward incoming HTTP requests to your dedicated Apache HTTP Server at 172.31.0.23, use the following command as the root user:

```
~]# 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 forward all incoming HTTP requests so that destination NAT routing is possible. To do this, use the following command as the root user:

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

This rule forwards all incoming HTTP requests from the firewall to the intended destination; the Apache HTTP Server behind the firewall.

2.8.5.3. DMZs and IPTables

You can create **`iptables`** rules to route traffic to certain machines, such as a dedicated HTTP or FTP server, in a *demilitarized zone* (DMZ). A DMZ is 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 uses the **`PREROUTING`** table to forward the packets to the appropriate 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 outside of the LAN are routed to the HTTP server on a network separate 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.

2.8.6. Malicious Software 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 applications or programs such as Trojans, worms, and other client/server viruses from contacting their server.

For example, some Trojans 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 them can effectively diminish the chances that potentially infected nodes on your network independently communicate with their remote master servers.

The following rules drop all TCP traffic that attempts to use port 31337:

```
~]# 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, you can design a rule that instructs 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.

2.8.7. IPTables and Connection Tracking

You can inspect and restrict connections to services based on their *connection state.* A module within **iptables** uses a method called *connection tracking* to store information about incoming connections. You can 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. For example, FTP uses port 21 to establish a connection, but data is transferred on a different port (typically port 20).
- **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
```

2.8.8. IPv6

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 carrier networks that are IPv6 aware are therefore 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. In Red Hat Enterprise Linux 6, both IPv4 and IPv6 services are enabled by default.

The **ip6tables** command syntax is identical to **iptables** in every aspect except that it supports 128-bit addresses. For example, use the following command to enable SSH connections on an IPv6-aware network server:

```
~]# 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/.

2.8.9. IPTables

Included with Red Hat Enterprise Linux are advanced tools for network *packet filtering* — the process of controlling network packets as they enter, move through, and exit the network stack within the kernel. Kernel versions prior to 2.4 relied on **ipchains** for packet filtering and used lists of rules applied to packets at each step of the filtering process. The 2.4 kernel introduced **iptables** (also called *netfilter*), which is similar to **ipchains** but greatly expands the scope and control available for filtering network packets.

This chapter focuses on packet filtering basics, explains various options available with **iptables** commands, and explains how filtering rules can be preserved between system reboots.

Important

The default firewall mechanism in the 2.4 and later kernels is **iptables**, but **iptables** cannot be used if **ipchains** is already running. If **ipchains** is present at boot time, the kernel issues an error and fails to start **iptables**.

The functionality of **ipchains** is not affected by these errors.

2.8.9.1. Packet Filtering

The Linux kernel uses the **Netfilter** facility to filter packets, allowing some of them to be received by or pass through the system while stopping others. This facility is built in to the Linux kernel, and has five built-in *tables* or *rules lists*, as follows:

- **filter** — The default table for handling network packets.
- **nat** — Used to alter packets that create a new connection and used for *Network Address Translation* (*NAT*).
- **mangle** — Used for specific types of packet alteration.
- **raw** — Used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target.
- **security** — Used for Mandatory Access Control (MAC) networking rules, such as those enabled by the SECMARK and CONNSECMARK targets.

Each table has a group of built-in *chains*, which correspond to the actions performed on the packet by **netfilter**.

The built-in chains for the **filter** table are as follows:

- *INPUT* — Applies to network packets that are targeted for the host.
- *OUTPUT* — Applies to locally-generated network packets.
- *FORWARD* — Applies to network packets routed through the host.

The built-in chains for the **`nat`** table are as follows:

- *PREROUTING* — Applies to network packets when they arrive.
- *OUTPUT* — Applies to locally-generated network packets before they are sent out.
- *POSTROUTING* — Applies to network packets before they are sent out.

The built-in chains for the **`mangle`** table are as follows:

- *INPUT* — Applies to network packets targeted for the host.
- *OUTPUT* — Applies to locally-generated network packets before they are sent out.
- *FORWARD* — Applies to network packets routed through the host.
- *PREROUTING* — Applies to incoming network packets before they are routed.
- *POSTROUTING* — Applies to network packets before they are sent out.

The built-in chains for the **`raw`** table are as follows:

- *OUTPUT* — Applies to locally-generated network packets before they are sent out.
- *PREROUTING* — Applies to incoming network packets before they are routed.

The built-in chains for the **`security`** table are as follows:

- *INPUT* — Applies to network packets targeted for the host.
- *OUTPUT* — Applies to locally-generated network packets before they are sent out.
- *FORWARD* — Applies to network packets routed through the host.

Every network packet received by or sent from a Linux system is subject to at least one table. However, a packet may be subjected to multiple rules within each table before emerging at the end of the chain. The structure and purpose of these rules may vary, but they usually seek to identify a packet coming from or going to a particular IP address, or set of addresses, when using a particular protocol and network service. The following image outlines how the flow of packets is examined by the **`iptables`** subsystem:

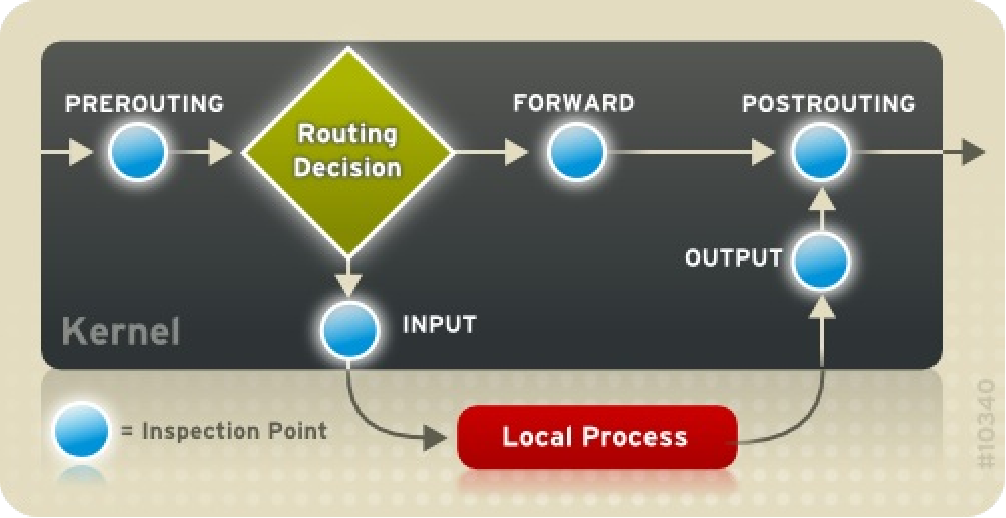


Figure 2.6. Packet filtering in IPTables

Important

By default, firewall rules are saved in the **/etc/sysconfig/iptables** or **/etc/sysconfig/ip6tables** files.
The **iptables** service starts before any DNS-related services when a Linux system is booted. This means that firewall rules can only reference numeric IP addresses (for example, 192.168.0.1). Domain names (for example, host.example.com) in such rules produce errors.

Regardless of their destination, when packets match a particular rule in one of the tables, a *target* or action is applied to them. If the rule specifies an **ACCEPT** target for a matching packet, the packet skips the rest of the rule checks and is allowed to continue to its destination. If a rule specifies a **DROP** target, that packet is refused access to the system and nothing is sent back to the host that sent the packet. If a rule specifies a **QUEUE** target, the packet is passed to user-space. If a rule specifies the optional **REJECT** target, the packet is dropped, but an error packet is sent to the packet's originator.

Every chain has a default policy to **ACCEPT**, **DROP**, **REJECT**, or **QUEUE**. If none of the rules in the chain apply to the packet, then the packet is dealt with in accordance with the default policy.

The **iptables** command configures these tables, as well as sets up new tables if necessary.

Note

The netfilter modules are not loaded by default. Therefore a user will not see all of them by looking in the **/proc/** directory as it only shows what is being used or has been loaded already. This means that there is no way to see what features of netfilter are available before you attempt to use it.

2.8.9.2. Command Options for IPTables

Rules for filtering packets are created using the **iptables** command. The following aspects of the packet are most often used as criteria:

- *Packet Type* — Specifies the type of packets the command filters.
- *Packet Source/Destination* — Specifies which packets the command filters based on the source or destination of the packet.
- *Target* — Specifies what action is taken on packets matching the above criteria.

Refer to Section 2.8.9.2.4, “IPTables Match Options” and Section 2.8.9.2.5, “Target Options” for more information about specific options that address these aspects of a packet.

The options used with specific **iptables** rules must be grouped logically, based on the purpose and conditions of the overall rule, for the rule to be valid. The remainder of this section explains commonly-used options for the **iptables** command.

2.8.9.2.1. Structure of IPTables Command Options

Many **iptables** commands have the following structure:

```
iptables [-t <table-name>] <command> <chain-name> \
  <parameter-1> <option-1> \
  <parameter-n> <option-n>
```

<table-name> — Specifies which table the rule applies to. If omitted, the **filter** table is used.

<command> — Specifies the action to perform, such as appending or deleting a rule.

<chain-name> — Specifies the chain to edit, create, or delete.

<parameter>-<option> pairs — Parameters and associated options that specify how to process a packet that matches the rule.

The length and complexity of an **iptables** command can change significantly, based on its purpose.

For example, a command to remove a rule from a chain can be very short:

```
iptables -D <chain-name> <line-number>
```

In contrast, a command that adds a rule which filters packets from a particular subnet using a variety of specific parameters and options can be rather long. When constructing **iptables** commands, it is important to remember that some parameters and options require further parameters and options to construct a valid rule. This can produce a cascading effect, with the further parameters requiring yet more parameters. Until every parameter and option that requires another set of options is satisfied, the rule is not valid.

Type **iptables -h** to view a comprehensive list of **iptables** command structures.

2.8.9.2.2. Command Options

Command options instruct **iptables** to perform a specific action. Only one command option is allowed per **iptables** command. With the exception of the help command, all commands are written in upper-case characters.

The **iptables** command options are as follows:

- **-A** — Appends the rule to the end of the specified chain. Unlike the **-I** option described below, it does not take an integer argument. It always appends the rule to the end of the specified chain.
- **-D <integer> | <rule>** — Deletes a rule in a particular chain by number (such as **5** for the fifth rule in a chain), or by rule specification. The rule specification must exactly match an existing rule.
- **-E** — Renames a user-defined chain. A user-defined chain is any chain other than the default, pre-existing chains. (Refer to the **-N** option, below, for information on creating user-defined chains.) This is a cosmetic change and does not affect the structure of the table.

> **Note**
>
> If you attempt to rename one of the default chains, the system reports a **Match not found** error. You cannot rename the default chains.

- **-F** — Flushes the selected chain, which effectively deletes every rule in the chain. If no chain is specified, this command flushes every rule from every chain.
- **-h** — Provides a list of command structures, as well as a quick summary of command parameters and options.
- **-I [<integer>]** — Inserts the rule in the specified chain at a point specified by a user-defined integer argument. If no argument is specified, the rule is inserted at the top of the chain.

Important

> As noted above, the order of rules in a chain determines which rules apply to which packets. This is important to remember when adding rules using either the `-A` or `-I` option.
>
> This is especially important when adding rules using the `-I` with an integer argument. If you specify an existing number when adding a rule to a chain, `iptables` adds the new rule *before* (or above) the existing rule.

- `-L` — Lists all of the rules in the chain specified after the command. To list all rules in all chains in the default `filter` table, do not specify a chain or table. Otherwise, the following syntax should be used to list the rules in a specific chain in a particular table:

  ```
  iptables -L <chain-name> -t <table-name>
  ```

  Additional options for the `-L` command option, which provide rule numbers and allow more verbose rule descriptions, are described in Section 2.8.9.2.6, “Listing Options”.
- `-N` — Creates a new chain with a user-specified name. The chain name must be unique, otherwise an error message is displayed.
- `-P` — Sets the default policy for the specified chain, so that when packets traverse an entire chain without matching a rule, they are sent to the specified target, such as ACCEPT or DROP.
- `-R` — Replaces a rule in the specified chain. The rule's number must be specified after the chain's name. The first rule in a chain corresponds to rule number one.
- `-X` — Deletes a user-specified chain. You cannot delete a built-in chain.
- `-Z` — Sets the byte and packet counters in all chains for a table to zero.

2.8.9.2.3. IPTables Parameter Options

Certain `iptables` commands, including those used to add, append, delete, insert, or replace rules within a particular chain, require various parameters to construct a packet filtering rule.

- `-c` — Resets the counters for a particular rule. This parameter accepts the `PKTS` and `BYTES` options to specify which counter to reset.
- `-d` — Sets the destination hostname, IP address, or network of a packet that matches the rule. When matching a network, the following IP address/netmask formats are supported:
  - ***N.N.N.N/M.M.M.M*** — Where ***N.N.N.N*** is the IP address range and ***M.M.M.M*** is the netmask.
  - ***N.N.N.N/M*** — Where ***N.N.N.N*** is the IP address range and ***M*** is the bitmask.
- `-f` — Applies this rule only to fragmented packets.

  You can use the exclamation point character (`!`) option before this parameter to specify that only unfragmented packets are matched.

Note

> Distinguishing between fragmented and unfragmented packets is desirable, despite fragmented packets being a standard part of the IP protocol.
>
> Originally designed to allow IP packets to travel over networks with differing frame sizes, these days fragmentation is more commonly used to generate DoS attacks using malformed packets. It's also worth noting that IPv6 disallows fragmentation entirely.

- **`-i`** — Sets the incoming network interface, such as **`eth0`** or **`ppp0`**. With **`iptables`**, this optional parameter may only be used with the INPUT and FORWARD chains when used with the **`filter`** table and the PREROUTING chain with the **`nat`** and **`mangle`** tables.

  This parameter also supports the following special options:

  - Exclamation point character (**`!`**) — Reverses the directive, meaning any specified interfaces are excluded from this rule.
  - Plus character (**`+`**) — A wildcard character used to match all interfaces that match the specified string. For example, the parameter **`-i eth+`** would apply this rule to any Ethernet interfaces but exclude any other interfaces, such as **`ppp0`**.

  If the **`-i`** parameter is used but no interface is specified, then every interface is affected by the rule.

- **`-j`** — Jumps to the specified target when a packet matches a particular rule.

  The standard targets are **`ACCEPT`**, **`DROP`**, **`QUEUE`**, and **`RETURN`**.

  Extended options are also available through modules loaded by default with the Red Hat Enterprise Linux **`iptables`** RPM package. Valid targets in these modules include **`LOG`**, **`MARK`**, and **`REJECT`**, among others. Refer to the **`iptables`** man page for more information about these and other targets.

  This option can also be used to direct a packet matching a particular rule to a user-defined chain outside of the current chain so that other rules can be applied to the packet.

  If no target is specified, the packet moves past the rule with no action taken. The counter for this rule, however, increases by one.

- **`-o`** — Sets the outgoing network interface for a rule. This option is only valid for the OUTPUT and FORWARD chains in the **`filter`** table, and the POSTROUTING chain in the **`nat`** and **`mangle`** tables. This parameter accepts the same options as the incoming network interface parameter (**`-i`**).

- **`-p <protocol>`** — Sets the IP protocol affected by the rule. This can be either **`icmp`**, **`tcp`**, **`udp`**, or **`all`**, or it can be a numeric value, representing one of these or a different protocol. You can also use any protocols listed in the **`/etc/protocols`** file.

  The "**`all`**" protocol means the rule applies to every supported protocol. If no protocol is listed with this rule, it defaults to "**`all`**".

- **`-s`** — Sets the source for a particular packet using the same syntax as the destination (**`-d`**) parameter.

2.8.9.2.4. IPTables Match Options

Different network protocols provide specialized matching options which can be configured to match a particular packet using that protocol. However, the protocol must first be specified in the **`iptables`** command. For example, **`-p`** ***`<protocol-name>`*** enables options for the specified protocol. Note that you can also use the protocol ID, instead of the protocol name. Refer to the following examples, each of which have the same effect:

```
~]# iptables -A INPUT -p icmp --icmp-type any -j ACCEPT
~]# iptables -A INPUT -p 5813 --icmp-type any -j ACCEPT
```

Service definitions are provided in the **`/etc/services`** file. For readability, it is recommended that you use the service names rather than the port numbers.

Warning

Secure the `/etc/services` file to prevent unauthorized editing. If this file is editable, crackers can use it to enable ports on your machine you have otherwise closed. To secure this file, run the following commands as root:

```
~]# chown root.root /etc/services
~]# chmod 0644 /etc/services
~]# chattr +i /etc/services
```

This prevents the file from being renamed, deleted or having links made to it.

2.8.9.2.4.1. TCP Protocol

These match options are available for the TCP protocol (`-p tcp`):

- `--dport` — Sets the destination port for the packet.

  To configure this option, use a network service name (such as www or smtp); a port number; or a range of port numbers.

  To specify a range of port numbers, separate the two numbers with a colon (`:`). For example: `-p tcp --dport 3000:3200`. The largest acceptable valid range is `0:65535`.

  Use an exclamation point character (`!`) after the `--dport` option to match all packets that *do not* use that network service or port.

  To browse the names and aliases of network services and the port numbers they use, view the `/etc/services` file.

  The `--destination-port` match option is synonymous with `--dport`.

- `--sport` — Sets the source port of the packet using the same options as `--dport`. The `--source-port` match option is synonymous with `--sport`.

- `--syn` — Applies to all TCP packets designed to initiate communication, commonly called *SYN packets*. Any packets that carry a data payload are not touched.

  Use an exclamation point character (`!`) before the `--syn` option to match all non-SYN packets.

- `--tcp-flags <tested flag list> <set flag list>` — Allows TCP packets that have specific bits (flags) set, to match a rule.

  The `--tcp-flags` match option accepts two parameters. The first parameter is the mask; a comma-separated list of flags to be examined in the packet. The second parameter is a comma-separated list of flags that must be set for the rule to match.

  The possible flags are:

  - `ACK`
  - `FIN`
  - `PSH`
  - `RST`
  - `SYN`
  - `URG`
  - `ALL`
  - `NONE`

  For example, an `iptables` rule that contains the following specification only matches TCP packets that have the SYN flag set and the ACK and FIN flags not set:

`--tcp-flags ACK,FIN,SYN SYN`

Use the exclamation point character (**`!`**) after the **`--tcp-flags`** to reverse the effect of the match option.

- **`--tcp-option`** — Attempts to match with TCP-specific options that can be set within a particular packet. This match option can also be reversed by using the exclamation point character (**`!`**) after the option.

2.8.9.2.4.2. UDP Protocol

These match options are available for the UDP protocol (**`-p udp`**):

- **`--dport`** — Specifies the destination port of the UDP packet, using the service name, port number, or range of port numbers. The **`--destination-port`** match option is synonymous with **`--dport`**.
- **`--sport`** — Specifies the source port of the UDP packet, using the service name, port number, or range of port numbers. The **`--source-port`** match option is synonymous with **`--sport`**.

For the **`--dport`** and **`--sport`** options, to specify a range of port numbers, separate the two numbers with a colon (:). For example: **`-p tcp --dport 3000:3200`**. The largest acceptable valid range is **`0:65535`**.

2.8.9.2.4.3. ICMP Protocol

The following match option is available for the Internet Control Message Protocol (ICMP) (**`-p icmp`**):

- **`--icmp-type`** — Sets the name or number of the ICMP type to match with the rule. A list of valid ICMP names can be retrieved by typing the **`iptables -p icmp -h`** command.

2.8.9.2.4.4. Additional Match Option Modules

Additional match options are available through modules loaded by the **`iptables`** command.

To use a match option module, load the module by name using the **`-m`** ***`<module-name>`***, where ***`<module-name>`*** is the name of the module.

Many modules are available by default. You can also create modules to provide additional functionality.

The following is a partial list of the most commonly used modules:

- **`limit`** module — Places limits on how many packets are matched to a particular rule.

  When used in conjunction with the **`LOG`** target, the **`limit`** module can prevent a flood of matching packets from filling up the system log with repetitive messages or using up system resources.

  Refer to Section 2.8.9.2.5, “Target Options” for more information about the **`LOG`** target.

  The **`limit`** module enables the following options:

  - **`--limit`** — Sets the maximum number of matches for a particular time period, specified as a ***`<value>/<period>`*** pair. For example, using **`--limit 5/hour`** allows five rule matches per hour.

    Periods can be specified in seconds, minutes, hours, or days.

    If a number and time modifier are not used, the default value of **`3/hour`** is assumed.
  - **`--limit-burst`** — Sets a limit on the number of packets able to match a rule at one time.

    This option is specified as an integer and should be used in conjunction with the **`--limit`** option.

    If no value is specified, the default value of five (5) is assumed.
- **`state`** module — Enables state matching.

The **state** module enables the following options:

  - **--state** — match a packet with the following connection states:
    - **ESTABLISHED** — The matching packet is associated with other packets in an established connection. You need to accept this state if you want to maintain a connection between a client and a server.
    - **INVALID** — The matching packet cannot be tied to a known connection.
    - **NEW** — The matching packet is either creating a new connection or is part of a two-way connection not previously seen. You need to accept this state if you want to allow new connections to a service.
    - **RELATED** — The matching packet is starting a new connection related in some way to an existing connection. An example of this is FTP, which uses one connection for control traffic (port 21), and a separate connection for data transfer (port 20).

    These connection states can be used in combination with one another by separating them with commas, such as **-m state --state INVALID,NEW**.

- **mac** module — Enables hardware MAC address matching.

  The **mac** module enables the following option:

  - **--mac-source** — Matches a MAC address of the network interface card that sent the packet. To exclude a MAC address from a rule, place an exclamation point character (**!**) after the **--mac-source** match option.

Refer to the **iptables** man page for more match options available through modules.

2.8.9.2.5. Target Options

When a packet has matched a particular rule, the rule can direct the packet to a number of different targets which determine the appropriate action. Each chain has a default target, which is used if none of the rules on that chain match a packet or if none of the rules which match the packet specify a target.

The following are the standard targets:

- ***<user-defined-chain>*** — A user-defined chain within the table. User-defined chain names must be unique. This target passes the packet to the specified chain.
- **ACCEPT** — Allows the packet through to its destination or to another chain.
- **DROP** — Drops the packet without responding to the requester. The system that sent the packet is not notified of the failure.
- **QUEUE** — The packet is queued for handling by a user-space application.
- **RETURN** — Stops checking the packet against rules in the current chain. If the packet with a **RETURN** target matches a rule in a chain called from another chain, the packet is returned to the first chain to resume rule checking where it left off. If the **RETURN** rule is used on a built-in chain and the packet cannot move up to its previous chain, the default target for the current chain is used.

In addition, extensions are available which allow other targets to be specified. These extensions are called target modules or match option modules and most only apply to specific tables and situations. Refer to Section 2.8.9.2.4.4, “Additional Match Option Modules” for more information about match option modules.

Many extended target modules exist, most of which only apply to specific tables or situations. Some of the most popular target modules included by default in Red Hat Enterprise Linux are:

- **LOG** — Logs all packets that match this rule. Because the packets are logged by the kernel, the **/etc/syslog.conf** file determines where these log entries are written. By default, they are placed in the `/var/log/messages` file.

Additional options can be used after the `LOG` target to specify the way in which logging occurs:

- `--log-level` — Sets the priority level of a logging event. Refer to the `syslog.conf` man page for a list of priority levels.
- `--log-ip-options` — Logs any options set in the header of an IP packet.
- `--log-prefix` — Places a string of up to 29 characters before the log line when it is written. This is useful for writing syslog filters for use in conjunction with packet logging.

> **Note**
>
> Due to an issue with this option, you should add a trailing space to the ***`log-prefix`*** value.

- `--log-tcp-options` — Logs any options set in the header of a TCP packet.
- `--log-tcp-sequence` — Writes the TCP sequence number for the packet in the log.

- `REJECT` — Sends an error packet back to the remote system and drops the packet.

  The `REJECT` target accepts `--reject-with <type>` (where ***<type>*** is the rejection type) allowing more detailed information to be returned with the error packet. The message `port-unreachable` is the default error type given if no other option is used. Refer to the `iptables` man page for a full list of ***<type>*** options.

Other target extensions, including several that are useful for IP masquerading using the `nat` table, or with packet alteration using the `mangle` table, can be found in the `iptables` man page.

2.8.9.2.6. Listing Options

The default list command, `iptables -L [<chain-name>]`, provides a very basic overview of the default filter table's current chains. Additional options provide more information:

- `-v` — Displays verbose output, such as the number of packets and bytes each chain has processed, the number of packets and bytes each rule has matched, and which interfaces apply to a particular rule.
- `-x` — Expands numbers into their exact values. On a busy system, the number of packets and bytes processed by a particular chain or rule may be abbreviated to `Kilobytes`, `Megabytes`, or `Gigabytes`. This option forces the full number to be displayed.
- `-n` — Displays IP addresses and port numbers in numeric format, rather than the default hostname and network service format.
- `--line-numbers` — Lists rules in each chain next to their numeric order in the chain. This option is useful when attempting to delete the specific rule in a chain or to locate where to insert a rule within a chain.
- `-t <table-name>` — Specifies a table name. If omitted, defaults to the filter table.

2.8.9.3. Saving IPTables Rules

Rules created with the `iptables` command are stored in memory. If the system is restarted before saving the `iptables` rule set, all rules are lost. For netfilter rules to persist through a system reboot, they need to be saved. To save netfilter rules, type the following command as root:

```
~]# /sbin/service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

This executes the **iptables** init script, which runs the **/sbin/iptables-save** program and writes the current **iptables** configuration to **/etc/sysconfig/iptables**. The existing **/etc/sysconfig/iptables** file is saved as **/etc/sysconfig/iptables.save**.

The next time the system boots, the **iptables** init script reapplies the rules saved in **/etc/sysconfig/iptables** by using the **/sbin/iptables-restore** command.

While it is always a good idea to test a new **iptables** rule before committing it to the **/etc/sysconfig/iptables** file, it is possible to copy **iptables** rules into this file from another system's version of this file. This provides a quick way to distribute sets of **iptables** rules to multiple machines.

You can also save the **iptables** rules to a separate file for distribution, backup, or other purposes. To do so, run the following command as root:

```
iptables-save > <filename>
```

… where ***<filename>*** is a user-defined name for your ruleset.

Important

If distributing the **/etc/sysconfig/iptables** file to other machines, type **/sbin/service iptables reload** or **/sbin/service iptables restart** for the new rules to take effect. It is better to use the **reload** command because there is no period of time without a firewall in place. See the description of the **reload** command in Section 2.8.9.4, “IPTables Control Scripts”. For **IPv6**, substitute **ip6tables** for **iptables** in the **/sbin/service** commands listed in this section. For more information about **IPv6** and netfilter, refer to Section 2.8.9.5, “IPTables and IPv6”.

Note

Note the difference between the **iptables** *command* (**/sbin/iptables**), which is used to manipulate the tables and chains that constitute the **iptables** functionality, and the **iptables** *service* (**/sbin/service iptables**), which is used to enable and disable the **iptables** service itself.

2.8.9.4. IPTables Control Scripts

There are two basic methods for controlling **iptables** in Red Hat Enterprise Linux:

- **Firewall Configuration Tool** (**system-config-firewall**) — A graphical interface for creating, activating, and saving basic firewall rules. Refer to Section 2.8.2, “Basic Firewall Configuration” for more information.
- **/sbin/service iptables *<option>*** — Used to manipulate various functions of **iptables** using its initscript. The following options are available:
  - **start** — If a firewall is configured (that is, **/etc/sysconfig/iptables** exists), all running **iptables** are stopped completely and then started using the **/sbin/iptables-restore** command. This option only works if the **ipchains** kernel module is *not* loaded. To check if this module is loaded, type the following command as root:

```
~]# lsmod | grep ipchains
```

If this command returns no output, it means the module is not loaded. If necessary, use the **/sbin/rmmod** command to remove the module.

- **stop** — If a firewall is running, the firewall rules in memory are flushed, and all **iptables** modules and helpers are unloaded.

  If the **IPTABLES_SAVE_ON_STOP** directive in the **/etc/sysconfig/iptables-config** configuration file is changed from its default value to **yes**, current rules are saved to **/etc/sysconfig/iptables** and any existing rules are moved to the file **/etc/sysconfig/iptables.save**.

  Refer to Section 2.8.9.4.1, “IPTables Control Scripts Configuration File” for more information about the **iptables-config** file.

- **reload** — If a firewall is running, the firewall rules are reloaded from the configuration file. The **reload** command does not unload helpers that have been in use before, but will add new helpers that have been added to IPTABLES_MODULES (for **IPv4**) and IP6TABLES_MODULES (for **IPv6**). The advantage of not flushing the current firewall rules is that if the new rules can not be applied, because of an error in the rules, the old rules are still in place.

- **restart** — If a firewall is running, the firewall rules in memory are flushed, and the firewall is started again if it is configured in **/etc/sysconfig/iptables**. This option only works if the **ipchains** kernel module is not loaded.

  If the **IPTABLES_SAVE_ON_RESTART** directive in the **/etc/sysconfig/iptables-config** configuration file is changed from its default value to **yes**, current rules are saved to **/etc/sysconfig/iptables** and any existing rules are moved to the file **/etc/sysconfig/iptables.save**.

  Refer to Section 2.8.9.4.1, “IPTables Control Scripts Configuration File” for more information about the **iptables-config** file.

- **status** — Displays the status of the firewall and lists all active rules.

  The default configuration for this option displays IP addresses in each rule. To display domain and hostname information, edit the **/etc/sysconfig/iptables-config** file and change the value of **IPTABLES_STATUS_NUMERIC** to **no**. Refer to Section 2.8.9.4.1, “IPTables Control Scripts Configuration File” for more information about the **iptables-config** file.

- **panic** — Flushes all firewall rules. The policy of all configured tables is set to **DROP**.

  This option could be useful if a server is known to be compromised. Rather than physically disconnecting from the network or shutting down the system, you can use this option to stop all further network traffic but leave the machine in a state ready for analysis or other forensics.

- **save** — Saves firewall rules to **/etc/sysconfig/iptables** using **iptables-save**. Refer to Section 2.8.9.3, “Saving IPTables Rules” for more information.

Note

To use the same initscript commands to control netfilter for IPv6, substitute **ip6tables** for **iptables** in the **/sbin/service** commands listed in this section. For more information about IPv6 and netfilter, refer to Section 2.8.9.5, “IPTables and IPv6”.

2.8.9.4.1. IPTables Control Scripts Configuration File

The behavior of the **iptables** initscripts is controlled by the **/etc/sysconfig/iptables-config** configuration file. The following is a list of directives contained in this file:

- **IPTABLES_MODULES** — Specifies a space-separated list of additional **iptables** modules to load when a firewall is activated. These can include connection tracking and NAT helpers.
- **IPTABLES_MODULES_UNLOAD** — Unloads modules on restart and stop. This directive accepts the following values:
  - **yes** — The default value. This option must be set to achieve a correct state for a firewall restart or stop.
  - **no** — This option should only be set if there are problems unloading the netfilter modules.
- **IPTABLES_SAVE_ON_STOP** — Saves current firewall rules to **/etc/sysconfig/iptables** when the firewall is stopped. This directive accepts the following values:
  - **yes** — Saves existing rules to **/etc/sysconfig/iptables** when the firewall is stopped, moving the previous version to the **/etc/sysconfig/iptables.save** file.
  - **no** — The default value. Does not save existing rules when the firewall is stopped.
- **IPTABLES_SAVE_ON_RESTART** — Saves current firewall rules when the firewall is restarted. This directive accepts the following values:
  - **yes** — Saves existing rules to **/etc/sysconfig/iptables** when the firewall is restarted, moving the previous version to the **/etc/sysconfig/iptables.save** file.
  - **no** — The default value. Does not save existing rules when the firewall is restarted.
- **IPTABLES_SAVE_COUNTER** — Saves and restores all packet and byte counters in all chains and rules. This directive accepts the following values:
  - **yes** — Saves the counter values.
  - **no** — The default value. Does not save the counter values.
- **IPTABLES_STATUS_NUMERIC** — Outputs IP addresses in numeric form instead of domain or hostnames. This directive accepts the following values:
  - **yes** — The default value. Returns only IP addresses within a status output.
  - **no** — Returns domain or hostnames within a status output.

2.8.9.5. IPTables and IPv6

If the **iptables-ipv6** package is installed, netfilter in Red Hat Enterprise Linux can filter the next-generation IPv6 Internet protocol. The command used to manipulate the IPv6 netfilter is **ip6tables**.

Most directives for this command are identical to those used for **iptables**, except the **nat** table is not yet supported. This means that it is not yet possible to perform IPv6 network address translation tasks, such as masquerading and port forwarding.

Rules for **ip6tables** are saved in the **/etc/sysconfig/ip6tables** file. Previous rules saved by the **ip6tables** initscripts are saved in the **/etc/sysconfig/ip6tables.save** file.

Configuration options for the **ip6tables** init script are stored in **/etc/sysconfig/ip6tables-config**, and the names for each directive vary slightly from their **iptables** counterparts.

For example, for the **iptables-config** directive **IPTABLES_MODULES** the equivalent in the **ip6tables-config** file is **IP6TABLES_MODULES**.

2.8.9.6. 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.

2.8.9.6.1. Useful Firewall Websites

- http://www.netfilter.org/ — The home of the netfilter/iptables project. Contains assorted information about **`iptables`**, including a FAQ addressing specific problems and various helpful guides by Rusty Russell, the Linux IP firewall maintainer. The HOWTO documents on the site cover subjects such as basic networking concepts, kernel packet filtering, and NAT configurations.
- 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.

2.8.9.6.2. 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 topics that cover analyzing firewall logs, developing firewall rules, and customizing your firewall using various graphical tools.
- *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.

2.8.9.6.3. Installed IP Tables Documentation

- **`man iptables`** — Contains a description of **`iptables`** as well as a comprehensive list of targets, options, and match extensions.

[3] 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.

[4] GRUB also accepts unencrypted passwords, but it is recommended that an MD5 hash be used for added security.

Chapter 3. Encryption

There are two main types of data that must be protected: data at rest and data in motion. These different types of data are protected in similar ways using similar technology but the implementations can be completely different. No single protective implementation can prevent all possible methods of compromise as the same information may be at rest and in motion at different points in time.

3.1. Data at Rest

Data at rest is data that is stored on a hard drive, tape, CD, DVD, disk, or other media. This information's biggest threat comes from being physically stolen. Laptops in airports, CDs going through the mail, and backup tapes that get left in the wrong places are all examples of events where data can be compromised through theft. If the data is encrypted on the media, it lowers the chances of the data being accessed.

3.1.1. Full Disk Encryption

Full disk or partition encryption is one of the best ways of protecting your data. Not only is each file protected but also the temporary storage that may contain parts of these files is also protected. Full disk encryption will protect all of your files so you do not have to worry about selecting what you want to protect and possibly missing a file.

Red Hat Enterprise Linux 6 natively supports LUKS Encryption. LUKS bulk encrypts your hard drive partitions so that while your computer is off, your data is protected. This will also protect your computer from attackers attempting to use single-user-mode to login to your computer or otherwise gain access.

Full disk encryption solutions like LUKS only protect the data when your computer is off. Once the computer is on and LUKS has decrypted the disk, the files on that disk are available to anyone who would normally have access to them. To protect your files when the computer is on, use full disk encryption in combination with another solution such as file based encryption. Also remember to lock your computer whenever you are away from it. A passphrase protected screen saver set to activate after a few minutes of inactivity is a good way to keep intruders out. For more information on LUKS, refer to Section 3.2.4, “LUKS Disk Encryption”.

3.1.2. File Based Encryption

GnuPG (GPG) is an open source version of PGP that allows you to sign and/or encrypt a file or an email message. This is useful to maintain integrity of the message or file and also protects the confidentiality of the information contained within the file or email. In the case of email, GPG provides dual protection. Not only can it provide Data at Rest protection but also Data in Motion protection once the message has been sent across the network.

File based encryption is used to protect the contents of files on mobile storage devices, such as CDs, flash drives, or external hard drives. Some file based encryption solutions may leave remnants of the encrypted files that an attacker who has physical access to your computer can recover under some circumstances. To protect the contents of these files from attackers who may have access to your computer, use file based encryption combined with another solution such as full disk encryption.

3.2. Data in Motion

Data in motion is data that is being transmitted over a network. The biggest threats to data in motion are interception and alteration. Your username and password should never be transmitted over a network without protection as it could be intercepted and used by someone else to impersonate you or gain access to sensitive information. Encrypting the network session ensures a higher security level for data in motion.

Data in motion is particularly vulnerable to attackers because the attacker does not have to be near the computer in which the data is being stored rather they only have to be somewhere along the path. Encryption tunnels can protect data along the path of communications.

3.2.1. Virtual Private Networks

Virtual Private Networks (VPN) provide encrypted tunnels between computers or networks of computers across all ports. With a VPN in place, all network traffic from the client is forwarded to the server through the encrypted tunnel. This means that the client is logically on the same network as the server it is connected to via the VPN. VPNs are very common and are simple to use and setup.

3.2.2. Secure Shell

Secure Shell (SSH) is a powerful network protocol used to communicate with another system over a secure channel. The transmissions over SSH are encrypted and protected from interception. Cryptographic log-on can also be utilized to provide a better authentication method over traditional usernames and passwords.

SSH is very easy to activate. By starting the **sshd** service, the system begins to accept connections and will allow access to the system when a correct username and password is provided during the connection process. The standard TCP port for the SSH service is 22, however this can be changed by modifying the configuration file */etc/ssh/sshd_config* and restarting the service. This file also contains other configuration options for SSH.

Secure Shell (SSH) also provides encrypted tunnels between computers but only using a single port. Port forwarding can be done over an SSH tunnel and traffic will be encrypted as it passes over that tunnel but using port forwarding is not as fluid as a VPN.

3.2.2.1. SSH Cryptographic Login

SSH supports the use of cryptographic keys to login to a computer. This is much more secure than using a password and if setup properly could be considered multifactor authentication.

A configuration change must occur before cryptographic logon can occur. In the file **/etc/ssh/sshd_config** uncomment and modify the following lines so they appear as such:

```
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

The first line tells the SSH program to allow public key authentication. The second line points to a file in the home directory where the public key of authorized key pairs exists on the system.

The next thing to do is to generate the SSH key pairs on the client you will use to connect to the system. The command **ssh-keygen** will generate an RSA 2048-bit key set for logging into the system. The keys are stored, by default, in the **~/.ssh** directory. You can utilize the switch **-b** to modify the bit-strength of the key. Using 2048-bit is normally sufficient.

In your **~/.ssh** directory you should see the two keys you just created. If you accepted the defaults when running the **ssh-keygen** then your keys are named **id_rsa** and **id_rsa.pub**, the private and public keys. You should always protect the private key from exposure. The public key, however, needs to be transferred over to the system you are going to login to. Once you have it on your system the easiest way to add the key to the approved list is by:

```
$ cat id_rsa.pub >> ~/.ssh/authorized_keys
```

This will append the public key to the authorized_key file. The SSH application will check this file when you attempt to login to the computer.

Similarly to passwords and any other authentication mechanism, you should change your SSH keys regularly. When you do, make sure you clean out any unused keys from the **`authorized_key`** file.

3.2.3. OpenSSL Intel AES-NI Engine

The Intel Advanced Encryption Standard (AES) New Instructions (AES-NI) engine is available for certain Intel processors, and allows for extremely fast hardware encryption and decryption.

> **Note**
>
> For a list of Intel processors that support the AES-NI engine, refer to: Intel's ARK.

The AES-NI engine is automatically enabled if the detected processor is among the supported ones.

To check if the engine is enabled, run the following command as the root user:

```
openssl engine -c -tt
```

To test its speed, run the following command as root:

```
openssl speed aes-128-cbc
```

To test the speed of OpenSSH you can run a command like the following:

```
~]# dd if=/dev/zero count=100 bs=1M | ssh -c aes128-cbc localhost "cat
>/dev/null"
root@localhost's password:
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 4.81868 s, 21.8 MB/s
```

You can find out more about the AES-NI engine at the following URL: http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/.

3.2.4. LUKS Disk Encryption

Linux Unified Key Setup-on-disk-format (or LUKS) allows you to encrypt partitions on your Linux computer. This is particularly important when it comes to mobile computers and removable media. LUKS allows multiple user keys to decrypt a master key which is used for the bulk encryption of the partition.

Overview of LUKS

What LUKS does

- LUKS encrypts entire block devices and is therefore well-suited for protecting the contents of mobile devices such as removable storage media or laptop disk drives.
- The underlying contents of the encrypted block device are arbitrary. This makes it useful for encrypting **`swap`** devices. This can also be useful with certain databases that use specially formatted block devices for data storage.
- LUKS uses the existing device mapper kernel subsystem.
- LUKS provides passphrase strengthening which protects against dictionary attacks.

- LUKS devices contain multiple key slots, allowing users to add backup keys/passphrases.

What LUKS does *not* do:

- LUKS is not well-suited for applications requiring many (more than eight) users to have distinct access keys to the same device.
- LUKS is not well-suited for applications requiring file-level encryption.

3.2.4.1. LUKS Implementation in Red Hat Enterprise Linux

Red Hat Enterprise Linux 6 utilizes LUKS to perform file system encryption. By default, the option to encrypt the file system is unchecked during the installation. If you select the option to encrypt your hard drive, you will be prompted for a passphrase that will be asked every time you boot the computer. This passphrase "unlocks" the bulk encryption key that is used to decrypt your partition. If you choose to modify the default partition table you can choose which partitions you want to encrypt. This is set in the partition table settings.

The default cipher used for LUKS (refer to **`cryptsetup --help`**) is aes-cbc-essiv:sha256. Note that the installation program, Anaconda, uses by default the AES cipher in XTS mode, aes-xts-plain64. The default key size for LUKS is 256 bits. The default key size for LUKS with Anaconda (XTS mode) is 512 bits.

Warning

Changing the default cryptographic attributes can affect your system's performance and expose your system to various security risks. You should not change the default cryptographic attributes of your system without good knowledge of cryptography and understanding to the capabilities of the used cipher combinations.

Red Hat strongly recommends using the default ciphers. If you need to use another than the default cipher, you can initialize your partition with the **`--cipher`** and **`--key-size`** options. The syntax of the command is the following:

```
cryptsetup --verify-passphrase --cipher <cipher>-<mode>-<iv> --key-size <key-
size> luksFormat <device>
```

where ***<cipher>-<mode>-<iv>*** is a string representing the used cipher. The string consists of three parts: a block cipher, block cipher mode, and an initial vector (IV).

A block cipher is a deterministic algorithm that operates on data blocks and allows encryption and decryption of bulk data. Block ciphers that are available on Red Hat Enterprise Linux are:

- AES — Advanced Encryption Standard, a 128-bit symmetric block cipher using encryption keys with lengths of 128, 192, and 256 bits; for more information, refer to the FIPS PUB 197.
- Twofish — A 128-bit block cipher operating with encryption keys of the range from 128 bits to 256 bits.
- Serpent — A 128-bit block cipher operating with 128-bit, 192-bit and 256-bit encryption keys.
- cast5 — A 64-bit Feistel cipher supporting encryption keys of the range from 40 to 128 bits; for more information, refer to the RFC 2144.
- cast6 — A 128-bit Feistel cipher using 128-bit, 160-bit, 192-bit, 224-bit, or 256-bit encryption keys; for

more information, refer to the RFC 2612.

Block cipher mode describes a way the block cipher is repeatedly applied on bulk data in order to encrypt or decrypt the data securely. The following modes can be used:

- CBC — Cipher Block Chaining; for more information, refer to the NIST SP 800-38A.
- XTS — XEX Tweakable Block Cipher with Ciphertext Stealing; for more information, refer to the IEEE 1619, or NIST SP 800-38E.
- CTR — Counter; for more information, refer to the NIST SP 800-38A.
- ECB — Electronic Codebook; for more information, refer to the NIST SP 800-38A.
- CFB — Cipher Feedback; for more information, refer to the NIST SP 800-38A.
- OFB — Output Feedback; for more information, refer to the NIST SP 800-38A.

An initial vector is a block of data used for ciphertext randomization. IV ensures that repeated encryption of the same plain text provides different ciphertext output. IV must not be reused with the same encryption key. For ciphers in CBC mode, IV **must be unpredictable**, otherwise the system could become vulnerable to certain watermarking attacks (see LUKS/cryptsetup FAQ for more information). Red Hat recommends using the following IV with AES:

- ESSIV — Encrypted Salt-Sector Initialization Vector - This IV should be used for ciphers in CBC mode. You should use the default hash: sha256.
- plain64 (or plain) — IV sector offset - This IV should be used for ciphers in XTS mode.

You may also specify the length of the used encryption key. The size of the key depends on the used combination of the block cipher and block cipher mode. If you do not specify the key length, LUKS will use the default value for the given combination. For example: if you decide to use a 128-bit key for AES in CBC mode, LUKS will encrypt your partition using the AES-128 implementation, while specifying a 512-bit key for AES in XTS mode means that the AES-256 implementation will be used. Note that XTS mode operates with two keys, the first is determined for tweakable encryption and the second for regular encryption.

3.2.4.2. Manually Encrypting Directories

Warning

Following this procedure will remove all data on the partition that you are encrypting. You WILL lose all your information! Make sure you backup your data to an external source before beginning this procedure!

1. Enter runlevel 1 by typing the following at a shell prompt as root:

   ```
   telinit 1
   ```

2. Unmount your existing `/home`:

   ```
   umount /home
   ```

3. If the command in the previous step fails, use `fuser` to find processes hogging `/home` and kill them:

   ```
   fuser -mvk /home
   ```

4. Verify `/home` is no longer mounted:

   ```
   grep home /proc/mounts
   ```

5. Fill your partition with random data:

   ```
   shred -v --iterations=1 /dev/VG00/LV_home
   ```

   This command proceeds at the sequential write speed of your device and may take some time to complete. It is an important step to ensure no unencrypted data is left on a used device, and to obfuscate the parts of the device that contain encrypted data as opposed to just random data.
6. Initialize your partition:

   ```
   cryptsetup --verbose --verify-passphrase luksFormat /dev/VG00/LV_home
   ```

7. Open the newly encrypted device:

   ```
   cryptsetup luksOpen /dev/VG00/LV_home home
   ```

8. Make sure the device is present:

   ```
   ls -l /dev/mapper | grep home
   ```

9. Create a file system:

   ```
   mkfs.ext3 /dev/mapper/home
   ```

10. Mount the file system:

    ```
    mount /dev/mapper/home /home
    ```

11. Make sure the file system is visible:

    ```
    df -h | grep home
    ```

12. Add the following to the `/etc/crypttab` file:

    ```
    home /dev/VG00/LV_home none
    ```

13. Edit the `/etc/fstab` file, removing the old entry for `/home` and adding the following line:

    ```
    /dev/mapper/home /home ext3 defaults 1 2
    ```

14. Restore default SELinux security contexts:

    ```
    /sbin/restorecon -v -R /home
    ```

15. Reboot the machine:

    ```
    shutdown -r now
    ```

16. The entry in the `/etc/crypttab` makes your computer ask your `luks` passphrase on boot.
17. Log in as root and restore your backup.

You now have an encrypted partition for all of your data to safely rest while the computer is off.

3.2.4.3. Add a new passphrase to an existing device

Use the following command to add a new passphrase to an existing device:

```
cryptsetup luksAddKey <device>
```

After being prompted for any one of the existing passprases for authentication, you will be prompted to enter the new passphrase.

3.2.4.4. Remove a passphrase from an existing device

Use the following command to remove a passphrase from an existing device:

```
cryptsetup luksRemoveKey <device>
```

You will be prompted for the passphrase you wish to remove and then for any one of the remaining passphrases for authentication.

3.2.4.5. Creating Encrypted Block Devices in Anaconda

You can create encrypted devices during system installation. This allows you to easily configure a system with encrypted partitions.

To enable block device encryption, check the **Encrypt System** check box when selecting automatic partitioning or the **Encrypt** checkbox when creating an individual partition, software RAID array, or logical volume. After you finish partitioning, you will be prompted for an encryption passphrase. This passphrase will be required to access the encrypted devices. If you have pre-existing LUKS devices and provided correct passphrases for them earlier in the install process the passphrase entry dialog will also contain a check box. Checking this checkbox indicates that you would like the new passphrase to be added to an available slot in each of the pre-existing encrypted block devices.

Note

Checking the **Encrypt System** checkbox on the **Automatic Partitioning** screen and then choosing **Create custom layout** does not cause any block devices to be encrypted automatically.

Note

You can use a **kickstart** file to set a separate passphrase for each new encrypted block device. Also, kickstart allows you to specify a different type of encryption if the Anaconda default cipher, aes-xts-plain64, does not suit you. In dependencies on a device you want to encrypt, you can specify the **--cipher=<*cipher-string*>** along with the **autopart**, **part**, **partition**, **logvol**, and **raid** directives. This option has to be used together with the **--encrypted** option, otherwise it has no effect. For more information about the **<*cipher-string*>** format and possible cipher combinations, refer to Section 3.2.4.1, “LUKS Implementation in Red Hat Enterprise Linux”. For more information about kickstart configuration, refer to the Red Hat Enterprise Linux Installation Guide.

3.2.4.6. Additional Resources

For additional information on LUKS or encrypting hard drives under Red Hat Enterprise Linux, visit one of the following links:

- LUKS home page
- LUKS/cryptsetup FAQ
- LUKS - Linux Unified Key Setup Wikipedia Article
- HOWTO: Creating an encrypted Physical Volume (PV) using a second hard drive and pvmove

3.2.5. Using GNU Privacy Guard (GPG)

GPG is used to identify yourself and authenticate your communications, including those with people you do not know. GPG allows anyone reading a GPG-signed email to verify its authenticity. In other words, GPG allows someone to be reasonably certain that communications signed by you actually are from you. GPG is useful because it helps prevent third parties from altering code or intercepting conversations and altering the message.

3.2.5.1. Creating GPG Keys in GNOME

To create a GPG Key in GNOME, follow these steps:

1. Install the **Seahorse** utility, which makes GPG key management easier:

   ```
   ~]# yum install seahorse
   ```

2. To create a key, from the **Applications** → **Accessories** menu select **Passwords and Encryption Keys**, which starts the application **Seahorse**.
3. From the **File** menu select **New** and then **PGP Key**. Then click **Continue**.
4. Type your full name, email address, and an optional comment describing who you are (for example: John C. Smith, jsmith@example.com, Software Engineer). Click **Create**. A dialog is displayed asking for a passphrase for the key. Choose a strong passphrase but also easy to remember. Click **OK** and the key is created.

> **Warning**
>
> If you forget your passphrase, you will not be able to decrypt the data.

To find your GPG key ID, look in the **Key ID** column next to the newly created key. In most cases, if you are asked for the key ID, prepend **0x** to the key ID, as in **0x6789ABCD**. You should make a backup of your private key and store it somewhere secure.

3.2.5.2. Creating GPG Keys in KDE

To create a GPG Key in KDE, follow these steps:

1. Start the KGpg program from the main menu by selecting **Applications** → **Utilities** → **Encryption Tool**. If you have never used KGpg before, the program walks you through the process of creating your own GPG keypair.
2. A dialog box appears prompting you to create a new key pair. Enter your name, email address, and an optional comment. You can also choose an expiration time for your key, as well as the key strength (number of bits) and algorithms.
3. Enter your passphrase in the next dialog box. At this point, your key appears in the main **KGpg**

window.

Warning

If you forget your passphrase, you will not be able to decrypt the data.

To find your GPG key ID, look in the **`Key ID`** column next to the newly created key. In most cases, if you are asked for the key ID, prepend **`0x`** to the key ID, as in **`0x6789ABCD`**. You should make a backup of your private key and store it somewhere secure.

3.2.5.3. Creating GPG Keys Using the Command Line

1. Use the following shell command:

   ```
   ~]$ gpg2 --gen-key
   ```

   This command generates a key pair that consists of a public and a private key. Other people use your public key to authenticate and/or decrypt your communications. Distribute your public key as widely as possible, especially to people who you know will want to receive authentic communications from you, such as a mailing list.
2. A series of prompts directs you through the process. Press the **`Enter`** key to assign a default value if desired. The first prompt asks you to select what kind of key you prefer:

   ```
   Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   Your selection?
   ```

   In almost all cases, the default is the correct choice. An RSA/RSA key allows you not only to sign communications, but also to encrypt files.
3. Choose the key size:

   ```
   RSA keys may be between 1024 and 4096 bits long.
   What keysize do you want? (2048)
   ```

   Again, the default, 2048, is sufficient for almost all users, and represents an extremely strong level of security.
4. Choose when the key will expire. It is a good idea to choose an expiration date instead of using the default, which is **`none`**. If, for example, the email address on the key becomes invalid, an expiration date will remind others to stop using that public key.

   ```
   Please specify how long the key should be valid.
   0 = key does not expire
   d = key expires in n days
   w = key expires in n weeks
   m = key expires in n months
   y = key expires in n years
   key is valid for? (0)
   ```

   Entering a value of **`1y`**, for example, makes the key valid for one year. (You may change this expiration date after the key is generated, if you change your mind.)

5. Before the **gpg2** application asks for signature information, the following prompt appears:

   ```
   Is this correct (y/N)?
   ```

   Enter **y** to finish the process.

6. Enter your name and email address for your GPG key. Remember this process is about authenticating you as a real individual. For this reason, include your real name. If you choose a bogus email address, it will be more difficult for others to find your public key. This makes authenticating your communications difficult. If you are using this GPG key for self-introduction on a mailing list, for example, enter the email address you use on that list.

   Use the comment field to include aliases or other information. (Some people use different keys for different purposes and identify each key with a comment, such as "Office" or "Open Source Projects.")

7. At the confirmation prompt, enter the letter **O** to continue if all entries are correct, or use the other options to fix any problems. Finally, enter a passphrase for your secret key. The **gpg2** program asks you to enter your passphrase twice to ensure you made no typing errors.

8. Finally, **gpg2** generates random data to make your key as unique as possible. Move your mouse, type random keys, or perform other tasks on the system during this step to speed up the process. Once this step is finished, your keys are complete and ready to use:

   ```
   pub  1024D/1B2AFA1C 2005-03-31 John Q. Doe <jqdoe@example.com>
   Key fingerprint = 117C FE83 22EA B843 3E86  6486 4320 545E 1B2A FA1C
   sub  1024g/CEA4B22E 2005-03-31 [expires: 2006-03-31]
   ```

9. The key fingerprint is a shorthand "signature" for your key. It allows you to confirm to others that they have received your actual public key without any tampering. You do not need to write this fingerprint down. To display the fingerprint at any time, use this command, substituting your email address:

   ```
   ~]$ gpg2 --fingerprint jqdoe@example.com
   ```

   Your "GPG key ID" consists of 8 hex digits identifying the public key. In the example above, the GPG key ID is **1B2AFA1C**. In most cases, if you are asked for the key ID, prepend **0x** to the key ID, as in **0x6789ABCD**.

Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

3.2.5.4. About Public Key Encryption

1. Wikipedia - Public Key Cryptography
2. HowStuffWorks - Encryption

Chapter 4. General Principles of Information Security

The following general principals provide an overview of good security practices:

- Encrypt all data transmitted over networks to help prevent man-in-the-middle attacks and eavesdropping. It is important to encrypt authentication information, such as passwords.
- Minimize the amount of software installed and running services.
- Use security-enhancing software and tools, for example, Security-Enhanced Linux (SELinux) for Mandatory Access Control (MAC), Netfilter iptables for packet filtering (firewall), and the GNU Privacy Guard (GPG) for encrypting files.
- If possible, run each network service on a separate system to minimize the risk of one compromised service being used to compromise other services.
- Maintain user accounts: create and enforce a strong password policy; delete unused user accounts.
- Routinely review system and application logs. By default, security-relevant system logs are written to **`/var/log/secure`** and **`/var/log/audit/audit.log`**. Note: sending logs to a dedicated log server helps prevent attackers from easily modifying local logs to avoid detection.
- Never log in as the root user unless absolutely necessary. It is recommended that administrators use **`sudo`** to execute commands as root when required. Users capable of running **`sudo`** are specified in **`/etc/sudoers`**. Use the **`visudo`** utility to edit **`/etc/sudoers`**.

4.1. Tips, Guides, and Tools

The United States' National Security Agency (NSA) provides hardening guides and tips for many different operating systems, to help government agencies, businesses, and individuals secure their systems against attack. The following guides (in PDF format) provide guidance for Red Hat Enterprise Linux 6:

- Hardening Tips for the Red Hat Enterprise Linux 5
- Guide to the Secure Configuration of Red Hat Enterprise Linux 5

Note

References to Red Hat Enterprise Linux 5 hardening guides are provided in this document until hardening guides for Red Hat Enterprise Linux 6 can be made available. In the meantime, please note that the hardening guides for Red Hat Enterprise Linux 5 may not apply completely to Red Hat Enterprise Linux 6.

The Defense Information Systems Agency (DISA) provides documentation, checklists, and tests to help secure your system (Information Assurance Support Environment).

Chapter 5. Secure Installation

Security begins with the first time you put that CD or DVD into your disk drive to install Red Hat Enterprise Linux. Configuring your system securely from the beginning makes it easier to implement additional security settings later.

5.1. Disk Partitions

Red Hat recommends creating separate partitions for /boot, /, /home, /tmp, and /var/tmp. If the root partition (/) becomes corrupt your data could be lost forever. By using a separate partition there is slightly more protection against data loss. You can also target this partition for frequent backups. The reasons for each are different and we will address each partition.

/boot - This partition is the first partition that is read by the system during boot up. The boot loader and kernel images that are used to boot your system into Red Hat Enterprise Linux are stored in this partition. This partition should not be encrypted. If this partition is included in / and that partition is encrypted or otherwise becomes unavailable then your system will not be able to boot.

/home - When user data (/home) is stored in / instead of in a separate partition, the partition can fill up causing the operating system to become unstable. Also, when upgrading your system to the next version of Red Hat Enterprise Linux it is a lot easier when you can keep your data in the /home partition as it will not be overwritten during installation.

/tmp and /var/tmp - Both the /tmp and the /var/tmp directories are used to store data that does not need to be stored for a long period of time. However if a lot of data floods one of these directories it can consume all of your storage space. If this happens and these directories are stored within / then your system could become unstable and crash. For this reason, moving these directories into their own partitions is a good idea.

5.2. Utilize LUKS Partition Encryption

During the installation process, an option to encrypt partitions is presented to you. The user must supply a passphrase. This passphrase will be used as a key to unlock the bulk encryption key, which is used to secure the partition's data.

Chapter 6. Software Maintenance

Software maintenance is extremely important to maintaining a secure system. It is vital to patch software as soon as it becomes available in order to prevent attackers from using known holes to infiltrate your system.

6.1. Install Minimal Software

It is a recommended practice to install only the packages you will use because each piece of software on your computer could possibly contain a vulnerability. If you are installing from the DVD media take the opportunity to select exactly what packages you want to install during the installation. When you find you need another package, you can always add it to the system later.

For more information on minimal installation, refer to the *"Package Group Selection"* section of the *Red Hat Enterprise Linux 6 Installation Guide*. A minimal installation can also be performed via a kickstart file using the **`--nobase`** option. For more information, refer to the *"Package Selection"* section of the *Red Hat Enterprise Linux 6 Installation Guide*.

6.2. Plan and Configure Security Updates

All software contains bugs. Often, these bugs can result in a vulnerability that can expose your system to malicious users. Unpatched systems are a common cause of computer intrusions. You should have a plan to install security patches in a timely manner to close those vulnerabilities so they can not be exploited.

For home users, security updates should be installed as soon as possible. Configuring automatic installation of security updates is one way to avoid having to remember, but does carry a slight risk that something can cause a conflict with your configuration or with other software on the system.

For business or advanced home users, security updates should be tested and scheduled for installation. Additional controls will need to be used to protect the system during the time between the patch release and its installation on the system. These controls would depend on the exact vulnerability, but could include additional firewall rules, the use of external firewalls, or changes in software settings.

6.3. Adjusting Automatic Updates

Red Hat Enterprise Linux is configured to apply all updates on a daily schedule. If you want to change how your system installs updates, you must do so via **`Software Update Preferences`**. You can change the schedule, the type of updates to apply, or to notify you of available updates.

In GNOME, you can find controls for your updates at: **System** → **Preferences** → **Software Updates**. In KDE, it is located at: **Applications** → **Settings** → **Software Updates**.

6.4. Install Signed Packages from Well Known Repositories

Software packages are published through repositories. All well known repositories support package signing. Package signing uses public key technology to prove that the package that was published by the repository has not been changed since the signature was applied. This provides some protection against installing software that may have been maliciously altered after the package was created but before you downloaded it.

Using too many repositories, untrustworthy repositories, or repositories with unsigned packages has a higher risk of introducing malicious or vulnerable code into your system. Use caution when adding repositories to yum/software update.

Chapter 7. System Auditing

The Linux Audit system provides a way to track security-relevant information on your system. Based on pre-configured rules, Audit generates log entries to record as much information about the events that are happening on your system as possible. This information is crucial for mission-critical environments to determine the violator of the security policy and the actions they performed. Audit does not provide additional security to your system; rather, it can be used to discover violations of security policies used on your system. These violations can further be prevented by additional security measures such as SELinux.

The following list summarizes some of the information that Audit is capable of recording in its log files:

- Date and time, type, and outcome of an event.
- Sensitivity labels of subjects and objects.
- Association of an event with the identity of the user who triggered the event.
- All modifications to Audit configuration and attempts to access Audit log files.
- All uses of authentication mechanisms, such as SSH, Kerberos, and others.
- Changes to any trusted database, such as **`/etc/passwd`**.
- Attempts to import or export information into or from the system.
- Include or exclude events based on user identity, subject and object labels, and other attributes.

The use of the Audit system is also a requirement for a number of security-related certifications. Audit is designed to meet or exceed the requirements of the following certifications or compliance guides:

- Controlled Access Protection Profile (CAPP)
- Labeled Security Protection Profile (LSPP)
- Rule Set Base Access Control (RSBAC)
- National Industrial Security Program Operating Manual (NISPOM)
- Federal Information Security Management Act (FISMA)
- Payment Card Industry — Data Security Standard (PCI-DSS)
- Security Technical Implementation Guides (STIG)

Audit has also been:

- Evaluated by National Information Assurance Partnership (NIAP) and Best Security Industries (BSI).
- Certified to LSPP/CAPP/RSBAC/EAL4+ on Red Hat Enterprise Linux 5.
- Certified to Operating System Protection Profile / Evaluation Assurance Level 4+ (OSPP/EAL4+) on Red Hat Enterprise Linux 6.

Use Cases

Watching file access

Audit can track whether a file or a directory has been accessed, modified, executed, or the file's attributes have been changed. This is useful, for example, to detect access to important files and have an Audit trail available in case one of these files is corrupted.

Monitoring system calls

Audit can be configured to generate a log entry every time a particular system call is used. This can be used, for example, to track changes to the system time by monitoring the **`settimeofday`**, **`clock_adjtime`**, and other time-related system calls.

Recording commands run by a user

Because Audit can track whether a file has been executed, a number of rules can be defined to record every execution of a particular command. For example, a rule can be defined for every executable in the `/bin` directory. The resulting log entries can then be searched by user ID to generate an audit trail of executed commands per user.

Recording security events

The `pam_faillock` authentication module is capable of recording failed login attempts. Audit can be set up to record failed login attempts as well, and provides additional information about the user who attempted to log in.

Searching for events

Audit provides the **ausearch** utility, which can be used to filter the log entries and provide a complete audit trail based on a number of conditions.

Running summary reports

The **aureport** utility can be used to generate, among other things, daily reports of recorded events. A system administrator can then analyze these reports and investigate suspicious activity furthermore.

Monitoring network access

The **iptables** and **ebtables** utilities can be configured to trigger Audit events, allowing system administrators to monitor network access.

Note

System performance may be affected depending on the amount of information that is collected by Audit.

7.1. Audit System Architecture

The Audit system consists of two main parts: the user-space applications and utilities, and the kernel-side system call processing. The kernel component receives system calls from user-space applications and filters them through one of the three filters: *user*, *task*, or *exit*. Once a system call passes through one of these filters, it is sent through the *exclude* filter, which, based on the Audit rule configuration, sends it to the Audit daemon for further processing. Figure 7.1, “Audit system architecture” illustrates this process.

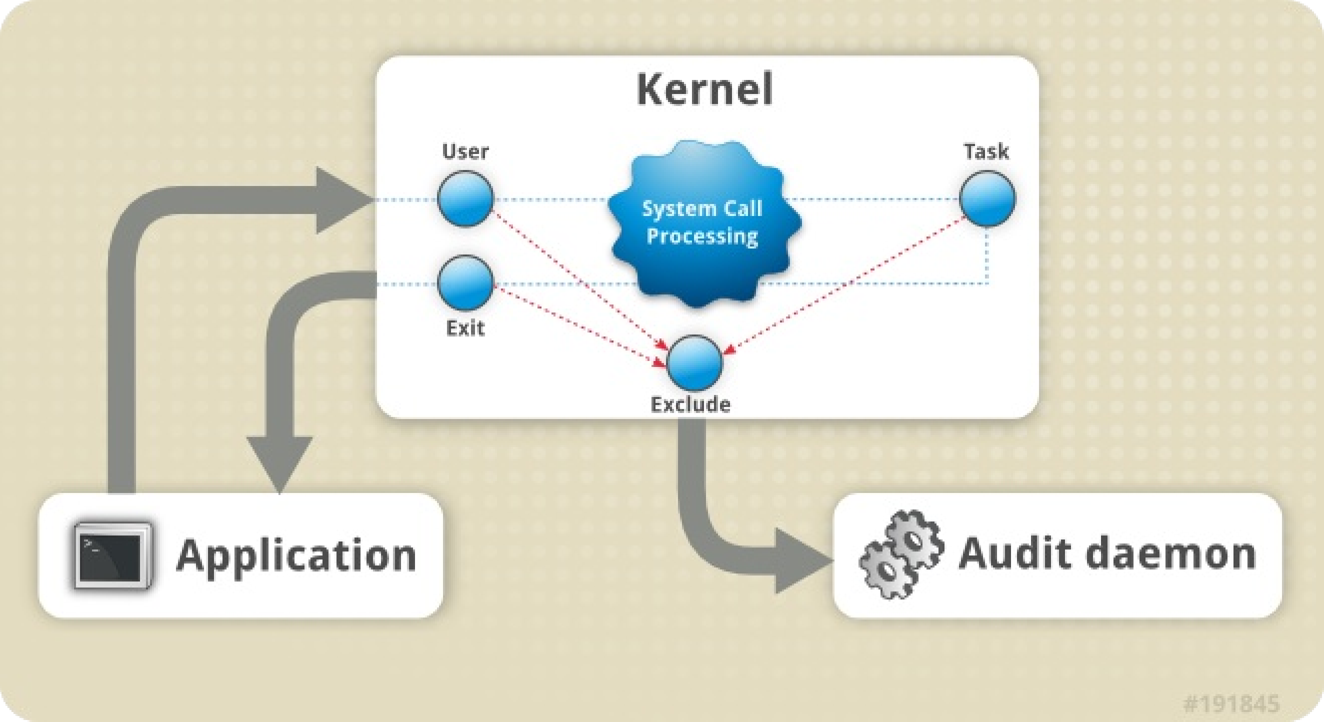


Figure 7.1. Audit system architecture

The user-space Audit daemon collects the information from the kernel and creates log file entries in a log file. Other Audit user-space utilities interact with the Audit daemon, the kernel Audit component, or the Audit log files:

- **audisp** — the Audit dispatcher daemon interacts with the Audit daemon and sends events to other applications for further processing. The purpose of this daemon is to provide a plug-in mechanism so that real-time analytical programs can interact with Audit events.
- **auditctl** — the Audit control utility interacts with the kernel Audit component to control a number of settings and parameters of the event generation process.
- The remaining Audit utilities take the contents of the Audit log files as input and generate output based on user's requirements. For example, the **aureport** utility generates a report of all recorded events.

7.2. Installing the *audit* Packages

In order to use the Audit system, you must have the *audit* packages installed on your system. The *audit* packages (*audit* and *audit-libs*) are installed by default on Red Hat Enterprise Linux 6. If you do not have these packages installed, execute the following command as the root user to install them:

```
~]# yum install audit
```

7.3. Configuring the `audit` Service

The Audit daemon can be configured in the **`/etc/audit/auditd.conf`** configuration file. This file consists of configuration parameters that modify the behavior of the Audit daemon. Any empty lines or any text following a hash sign (#) is ignored. A complete listing of all configuration parameters and their explanation can be found in the audit.conf(5) man page.

7.3.1. Configuring auditd for a CAPP Environment

The default **auditd** configuration should be suitable for most environments. However, if your environment has to meet the criteria set by the *Controlled Access Protection Profile* (CAPP), which is a part of the Common Criteria certification, the Audit daemon must be configured with the following settings:

- The directory that holds the Audit log files (usually **/var/log/audit/**) should reside on a separate partition. This prevents other processes from consuming space in this directory, and provides accurate detection of the remaining space for the Audit daemon.
- The ***max_log_file*** parameter, which specifies the maximum size of a single Audit log file, must be set to make full use of the available space on the partition that holds the Audit log files.
- The ***max_log_file_action*** parameter, which decides what action is taken once the limit set in ***max_log_file*** is reached, should be set to **keep_logs** to prevent Audit log files from being overwritten.
- The ***space_left*** parameter, which specifies the amount of free space left on the disk for which an action that is set in the ***space_left_action*** parameter is triggered, must be set to a number that gives the administrator enough time to respond and free up disk space. The ***space_left*** value depends on the rate at which the Audit log files are generated.
- It is recommended to set the ***space_left_action*** parameter to **email** or **exec** with an appropriate notification method.
- The ***admin_space_left*** parameter, which specifies the absolute minimum amount of free space for which an action that is set in the ***admin_space_left_action*** parameter is triggered, must be set to a value that leaves enough space to log actions performed by the administrator.
- The ***admin_space_left_action*** parameter must be set to **single** to put the system into single-user mode and allow the administrator to free up some disk space.
- The ***disk_full_action*** parameter, which specifies an action that is triggered when no free space is available on the partition that holds the Audit log files, must be set to **halt** or **single**. This ensures that the system is either shut down or operating in single-user mode when Audit can no longer log events.
- The ***disk_error_action***, which specifies an action that is triggered in case an error is detected on the partition that holds the Audit log files, must be set to **syslog**, **single**, or **halt**, depending on your local security policies regarding the handling of hardware malfunctions.
- The ***flush*** configuration parameter must be set to **sync** or **data**. These parameters assure that all Audit event data is fully synchronized with the log files on the disk.

The remaining configuration options should be set according to your local security policy.

7.4. Starting the audit Service

Once **auditd** is properly configured, start the service to collect Audit information and store it in the log files. Execute the following command as the root user to start **auditd**:

```
~]# service auditd start
```

Optionally, you can configure **auditd** to start at boot time using the following command as the root user:

```
~]# chkconfig auditd on
```

A number of other actions can be performed on **auditd** using the **service auditd *action*** command, where ***action*** can be one of the following:

- **stop** — stops **auditd**.
- **restart** — restarts **auditd**.
- **reload** or **force-reload** — reloads the configuration of **auditd** from the **/etc/audit/auditd.conf** file.
- **rotate** — rotates the log files in the **/var/log/audit/** directory.
- **resume** — resumes logging of Audit events after it has been previously suspended, for example, when there is not enough free space on the disk partition that holds the Audit log files.
- **condrestart** or **try-restart** — restarts **auditd** only if it is already running.
- **status** — displays the running status of **auditd**.

7.5. Defining Audit Rules

The Audit system operates on a set of rules that define what is to be captured in the log files. There are three types of Audit rules that can be specified:

- Control rules — allow the Audit system's behavior and some of its configuration to be modified.
- File system rules — also known as file watches, allow the auditing of access to a particular file or a directory.
- System call rules — allow logging of system calls that any specified program makes.

Audit rules can be specified on the command line with the **auditctl** utility (note that these rules are not persistent across reboots), or written in the **/etc/audit/audit.rules** file. The following two sections summarize both approaches to defining Audit rules.

7.5.1. Defining Audit Rules with the auditctl Utility

Note

All commands, which interact with the Audit service and the Audit log files require root privileges. Ensure you execute these commands as the root user.

The **auditctl** command allows you to control the basic functionality of the Audit system and to define rules that decide which Audit events are logged.

Defining Control Rules

The following are some of the control rules that allow you to modify the behavior of the Audit system:

-b

sets the maximum amount of existing Audit buffers in the kernel, for example:

```
~]# auditctl -b 8192
```

-f

sets the action that is performed when a critical error is detected, for example:

```
~]# auditctl -f 2
```

The above configuration triggers a kernel panic in case of a critical error.

-e

enables and disables the Audit system or locks its configuration, for example:

```
~]# auditctl -e 2
```

The above command locks the Audit configuration.

-r

sets the rate of generated messages per second, for example:

```
~]# auditctl -r 0
```

The above configuration sets no rate limit on generated messages.

-s

reports the status of the Audit system, for example:

```
~]# auditctl -s
AUDIT_STATUS: enabled=1 flag=2 pid=0 rate_limit=0 backlog_limit=8192
lost=259 backlog=0
```

-l

lists all currently loaded Audit rules, for example:

```
~]# auditctl -l
LIST_RULES: exit,always watch=/etc/localtime perm=wa key=time-change
LIST_RULES: exit,always watch=/etc/group perm=wa key=identity
LIST_RULES: exit,always watch=/etc/passwd perm=wa key=identity
LIST_RULES: exit,always watch=/etc/gshadow perm=wa key=identity
 ⋮
```

-D

deletes all currently loaded Audit rules, for example:

```
~]# auditctl -D
No rules
```

Defining File System Rules

To define a file system rule, use the following syntax:

```
auditctl -w path_to_file -p permissions -k key_name
```

where:

- ***path_to_file*** is the file or directory that is audited.

- ***permissions*** are the permissions that are logged:
  - **r** — read access to a file or a directory.
  - **w** — write access to a file or a directory.
  - **x** — execute access to a file or a directory.
  - **a** — change in the file's or directory's attribute.
- ***key_name*** is an optional string that helps you identify which rule or a set of rules generated a particular log entry.

Example 7.1. File System Rules

To define a rule that logs all write access to, and every attribute change of, the **/etc/passwd** file, execute the following command:

```
~]# auditctl -w /etc/passwd -p wa -k passwd_changes
```

Note that the string following the **-k** option is arbitrary.

To define a rule that logs all write access to, and every attribute change of, all the files in the **/etc/selinux/** directory, execute the following command:

```
~]# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

To define a rule that logs the execution of the **/sbin/insmod** command, which inserts a module into the Linux kernel, execute the following command:

```
~]# auditctl -w /sbin/insmod -p x -k module_insertion
```

Defining System Call Rules

To define a system call rule, use the following syntax:

```
auditctl -a action,filter -S system_call -F field=value -k key_name
```

where:

- ***action*** and ***filter*** specify when a certain event is logged. ***action*** can be either **always** or **never**. ***filter*** specifies which kernel rule-matching filter is applied to the event. The rule-matching filter can be one of the following: **task**, **exit**, **user**, and **exclude**. For more information about these filters, refer to the beginning of Section 7.1, “Audit System Architecture”.
- ***system_call*** specifies the system call by its name. A list of all system calls can be found in the **/usr/include/asm/unistd_64.h** file. Several system calls can be grouped into one rule, each specified after the **-S** option.
- ***field***=***value*** specifies additional options that furthermore modify the rule to match events based on a specified architecture, group ID, process ID, and others. For a full listing of all available field types and their values, refer to the auditctl(8) man page.
- ***key_name*** is an optional string that helps you identify which rule or a set of rules generated a particular log entry.

Example 7.2. System Call Rules

To define a rule that creates a log entry every time the **adjtimex** or **settimeofday** system calls are used by a program, and the system uses the 64-bit architecture, execute the following command:

```
~]# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k
time_change
```

To define a rule that creates a log entry every time a file is deleted or renamed by a system user whose ID is 500 or larger (the **-F auid!=4294967295** option is used to exclude users whose login UID is not set), execute the following command:

```
~]# auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F
auid>=500 -F auid!=4294967295 -k delete
```

It is also possible to define a file system rule using the system call rule syntax. The following command creates a rule for system calls that is analogous to the **-w /etc/shadow -p wa** file system rule:

```
~]# auditctl -a always,exit -F path=/etc/shadow -F perm=wa
```

7.5.2. Defining Persistent Audit Rules and Controls in the `/etc/audit/audit.rules` File

To define Audit rules that are persistent across reboots, you must include them in the **/etc/audit/audit.rules** file. This file uses the same **auditctl** command line syntax to specify the rules. Any empty lines or any text following a hash sign (**#**) is ignored.

The **auditctl** command can also be used to read rules from a specified file with the **-R** option, for example:

```
~]# auditctl -R /usr/share/doc/audit-version/stig.rules
```

Defining Control Rules

A file can contain only the following control rules that modify the behavior of the Audit system: **-b**, **-D**, **-e**, **-f**, and **-r**. For more information on these options, refer to Section 7.5.1, “Defining Control Rules”.

Example 7.3. Control rules in `audit.rules`

```
# Delete all previous rules
-D

# Set buffer size
-b 8192

# Make the configuration immutable -- reboot is required to change audit rules
-e 2

# Panic when a failure occurs
-f 2

# Generate at most 100 audit messages per second
-r 100
```

Defining File System and System Call Rules

File system and system call rules are defined using the **`auditctl`** syntax. The examples in Section 7.5.1, “Defining Audit Rules with the **auditctl** Utility” can be represented with the following rules file:

Example 7.4. File system and system call rules in `audit.rules`

```
-w /etc/passwd -p wa -k passwd_changes
-w /etc/selinux/ -p wa -k selinux_changes
-w /sbin/insmod -p x -k module_insertion

-a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
-a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=500 -F
auid!=4294967295 -k delete
```

Preconfigured Rules Files

In the **`/usr/share/doc/audit-`*`version`*`/`** directory, the *audit* package provides a set of pre-configured rules files according to various certification standards:

- **`nispom.rules`** — Audit rule configuration that meets the requirements specified in Chapter 8 of the National Industrial Security Program Operating Manual.
- **`capp.rules`** — Audit rule configuration that meets the requirements set by Controlled Access Protection Profile (CAPP), which is a part of the Common Criteria certification.
- **`lspp.rules`** — Audit rule configuration that meets the requirements set by Labeled Security Protection Profile (LSPP), which is a part of the Common Criteria certification.
- **`stig.rules`** — Audit rule configuration that meets the requirements set by Security Technical Implementation Guides (STIG).

To use these configuration files, create a backup of your original **`/etc/audit/audit.rules`** file and copy the configuration file of your choice over the **`/etc/audit/audit.rules`** file:

```
~]# cp /etc/audit/audit.rules /etc/audit/audit.rules_backup
~]# cp /usr/share/doc/audit-version/stig.rules /etc/audit/audit.rules
```

7.6. Understanding Audit Log Files

By default, the Audit system stores log entries in the **`/var/log/audit/audit.log`** file; if log rotation is enabled, rotated **`audit.log`** files are stored in the same directory.

The following Audit rule logs every attempt to read or modify the **`/etc/ssh/sshd_config`** file:

```
-w /etc/ssh/sshd_config -p warx -k sshd_config
```

If the **`auditd`** daemon is running, running the following command creates a new event in the Audit log file:

```
~]# cat /etc/ssh/sshd_config
```

This event in the **`audit.log`** file looks as follows:

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2 success=no
exit=-13 a0=7fffd19c5592 a1=0 a2=7fffd19c4b50 a3=a items=1 ppid=2686 pid=3538
auid=500 uid=500 gid=500 euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500
tty=pts0 ses=1 comm="cat" exe="/bin/cat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="sshd_config"
type=CWD msg=audit(1364481363.243:24287):  cwd="/home/shadowman"
type=PATH msg=audit(1364481363.243:24287): item=0 name="/etc/ssh/sshd_config"
inode=409248 dev=fd:00 mode=0100600 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:etc_t:s0
```

The above event consists of three records (each starting with the **`type=`** keyword), which share the same time stamp and serial number. Each record consists of several ***`name`*`=`*`value`*** pairs separated by a white space or a comma. A detailed analysis of the above event follows:

First Record

`type=SYSCALL`

The **`type`** field contains the type of the record. In this example, the **`SYSCALL`** value specifies that this record was triggered by a system call to the kernel.

For a list of all possible type values and their explanations, refer to Section B.2, “Audit Record Types”.

`msg=audit(1364481363.243:24287):`

The **`msg`** field records:

- a time stamp and a unique ID of the record in the form **`audit(`*`time_stamp`*`:`*`ID`*`)`**. Multiple records can share the same time stamp and ID if they were generated as part of the same Audit event.
- various event-specific ***`name`*`=`*`value`*** pairs provided by the kernel or user space applications.

`arch=c000003e`

The **`arch`** field contains information about the CPU architecture of the system. The value,

c000003e, is encoded in hexadecimal notation. When searching Audit records with the **ausearch** command, use the **-i** or **--interpret** option to automatically convert hexadecimal values into their human-readable equivalents. The **c000003e** value is interpreted as **x86_64**.

syscall=2

The **syscall** field records the type of the system call that was sent to the kernel. The value, **2**, can be matched with its human-readable equivalent in the **/usr/include/asm/unistd_64.h** file. In this case, **2** is the **open** system call. Note that the **ausyscall** utility allows you to convert system call numbers to their human-readable equivalents. Use the **ausyscall --dump** command to display a listing of all system calls along with their numbers. For more information, refer to the ausyscall(8) man page.

success=no

The **success** field records whether the system call recorded in that particular event succeeded or failed. In this case, the call did not succeed.

exit=-13

The **exit** field contains a value that specifies the exit code returned by the system call. This value varies for different system call. You can interpret the value to its human-readable equivalent with the following command: **ausearch --interpret --exit -13** (assuming your Audit log contains an event that failed with exit code **-13**).

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

The **a0** to **a3** fields record the first four arguments, encoded in hexadecimal notation, of the system call in this event. These arguments depend on the system call that is used; they can be interpreted by the **ausearch** utility.

items=1

The **items** field contains the number of path records in the event.

ppid=2686

The **ppid** field records the Parent Process ID (PPID). In this case, **2686** was the PPID of the **bash** process.

pid=3538

The **pid** field records the Process ID (PID). In this case, **3538** was the PID of the **cat** process.

auid=500

The **auid** field records the Audit user ID, that is the loginuid. This ID is assigned to a user upon login and is inherited by every process even when the user's identity changes (for example, by switching user accounts with the **su - john** command).

uid=500

The **uid** field records the user ID of the user who started the analyzed process. The user ID can be interpreted into user names with the following command: **`ausearch -i --uid`** ***`UID`***. In this case, **`500`** is the user ID of user **`shadowman`**.

`gid=500`

The **`gid`** field records the group ID of the user who started the analyzed process.

`euid=500`

The **`euid`** field records the effective user ID of the user who started the analyzed process.

`suid=500`

The **`suid`** field records the set user ID of the user who started the analyzed process.

`fsuid=500`

The **`fsuid`** field records the file system user ID of the user who started the analyzed process.

`egid=500`

The **`egid`** field records the effective group ID of the user who started the analyzed process.

`sgid=500`

The **`sgid`** field records the set group ID of the user who started the analyzed process.

`fsgid=500`

The **`fsgid`** field records the file system group ID of the user who started the analyzed process.

`tty=pts0`

The **`tty`** field records the terminal from which the analyzed process was invoked.

`ses=1`

The **`ses`** field records the session ID of the session from which the analyzed process was invoked.

`comm="cat"`

The **`comm`** field records the command-line name of the command that was used to invoke the analyzed process. In this case, the **`cat`** command was used to trigger this Audit event.

`exe="/bin/cat"`

The **`exe`** field records the path to the executable that was used to invoke the analyzed process.

`subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`

The **`subj`** field records the SELinux context with which the analyzed process was labeled at the time of execution.

`key="sshd_config"`

The **`key`** field records the administrator-defined string associated with the rule that generated this event in the Audit log.

Second Record

`type=CWD`

In the second record, the **`type`** field value is **`CWD`** — current working directory. This type is used to record the working directory from which the process that invoked the system call specified in the first record was executed.

The purpose of this record is to record the current process's location in case a relative path winds up being captured in the associated PATH record. This way the absolute path can be reconstructed.

`msg=audit(1364481363.243:24287)`

The **`msg`** field holds the same time stamp and ID value as the value in the first record.

`cwd="/home/shadowman"`

The **`cwd`** field contains the path to the directory in which the system call was invoked.

Third Record

`type=PATH`

In the third record, the **`type`** field value is **`PATH`**. An Audit event contains a **`PATH`**-type record for every path that is passed to the system call as an argument. In this Audit event, only one path (**`/etc/ssh/sshd_config`**) was used as an argument.

`msg=audit(1364481363.243:24287):`

The **`msg`** field holds the same time stamp and ID value as the value in the first and second record.

`item=0`

The **`item`** field indicates which item, of the total number of items referenced in the **`SYSCALL`** type record, the current record is. This number is zero-based; a value of **`0`** means it is the first item.

`name="/etc/ssh/sshd_config"`

The **`name`** field records the full path of the file or directory that was passed to the system call as an argument. In this case, it was the **`/etc/ssh/sshd_config`** file.

`inode=409248`

The **`inode`** field contains the inode number associated with the file or directory recorded in this event. The following command displays the file or directory that is associated with the **`409248`** inode number:

```
~]# find / -inum 409248 -print
/etc/ssh/sshd_config
```

dev=fd:00

The **dev** field specifies the minor and major ID of the device that contains the file or directory recorded in this event. In this case, the value represents the **/dev/fd/0** device.

mode=0100600

The **mode** field records the file or directory permissions, encoded in numerical notation. In this case, **0100600** can be interpreted as **-rw-------**, meaning that only the root user has read and write permissions to the **/etc/ssh/sshd_config** file.

ouid=0

The **ouid** field records the object owner's user ID.

ogid=0

The **ogid** field records the object owner's group ID.

rdev=00:00

The **rdev** field contains a recorded device identifier for special files only. In this case, it is not used as the recorded file is a regular file.

obj=system_u:object_r:etc_t:s0

The **obj** field records the SELinux context with which the recorded file or directory was labeled at the time of execution.

The Audit event analyzed above contains only a subset of all possible fields that an event can contain. For a list of all event fields and their explanation, refer to Section B.1, “Audit Event Fields”. For a list of all event types and their explanation, refer to Section B.2, “Audit Record Types”.

Example 7.5. Additional audit.log events

The following Audit event records a successful start of the **auditd** daemon. The **ver** field shows the version of the Audit daemon that was started.

```
type=DAEMON_START msg=audit(1363713609.192:5426): auditd start, ver=2.2
format=raw kernel=2.6.32-358.2.1.el6.x86_64 auid=500 pid=4979
subj=unconfined_u:system_r:auditd_t:s0 res=success
```

The following Audit event records a failed attempt of user with UID of 500 to log in as the root user.

```
type=USER_AUTH msg=audit(1364475353.159:24270): user pid=3280 uid=500 auid=500
ses=1 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:authentication acct="root" exe="/bin/su" hostname=? addr=?
terminal=pts/0 res=failed'
```

7.7. Searching the Audit Log Files

The **ausearch** utility allows you to search Audit log files for specific events. By default, **ausearch** searches the **`/var/log/audit/audit.log`** file. You can specify a different file using the **`ausearch`** ***`options`*** **`-if`** ***`file_name`*** command. Supplying multiple options in one **`ausearch`** command is equivalent to using the *AND* operator.

> **Example 7.6. Using `ausearch` to search Audit log files**
>
> To search the **`/var/log/audit/audit.log`** file for failed login attempts, use the following command:
>
> ```
> ~]# ausearch --message USER_LOGIN --success no --interpret
> ```
>
> To search for all account, group, and role changes, use the following command:
>
> ```
> ~]# ausearch -m ADD_USER -m DEL_USER -m ADD_GROUP -m USER_CHAUTHTOK -m
> DEL_GROUP -m CHGRP_ID -m ROLE_ASSIGN -m ROLE_REMOVE -i
> ```
>
> To search for all logged actions performed by a certain user, using the user's login ID (**`auid`**), use the following command:
>
> ```
> ~]# ausearch -au 500 -i
> ```
>
> To search for all failed system calls from yesterday up until now, use the following command:
>
> ```
> ~]# ausearch --start yesterday --end now -m SYSCALL -sv no -i
> ```

For a full listing of all **`ausearch`** options, refer to the ausearch(8) man page.

7.8. Creating Audit Reports

The **aureport** utility allows you to generate summary and columnar reports on the events recorded in Audit log files. By default, all **`audit.log`** files in the **`/var/log/audit/`** directory are queried to create the report. You can specify a different file to run the report against using the **`aureport`** ***`options`*** **`-if`** ***`file_name`*** command.

Example 7.7. Using `aureport` to generate Audit reports

To generate a report for logged events in the past three days excluding the current example day, use the following command:

```
~]# aureport --start 04/08/2013 00:00:00 --end 04/11/2013 00:00:00
```

To generate a report of all executable file events, use the following command:

```
~]# aureport -x
```

To generate a summary of the executable file event report above, use the following command:

```
~]# aureport -x --summary
```

To generate a summary report of failed events for all users, use the following command:

```
~]# aureport -u --failed --summary -i
```

To generate a summary report of all failed login attempts per each system user, use the following command:

```
~]# aureport --login --summary -i
```

To generate a report from an **`ausearch`** query that searches all file access events for user **`500`**, use the following command:

```
~]# ausearch --start today --loginuid 500 --raw | aureport -f --summary
```

To generate a report of all Audit files that are queried and the time range of events they include, use the following command:

```
~]# aureport -t
```

For a full listing of all **`aureport`** options, refer to the aureport(8) man page.

7.9. Additional Resources

For more information about the Audit system, refer to the following sources.

Online Sources

- The Linux Audit system project page: http://people.redhat.com/sgrubb/audit/.
- Article *Investigating kernel Return Codes with the Linux Audit System* in the Hack In the Box magazine: http://magazine.hackinthebox.org/issues/HITB-Ezine-Issue-005.pdf.

Installed Documentation

Documentation provided by the *audit* package can be found in the **`/usr/share/doc/audit-`*`version`*`/`** directory.

Manual Pages

- audispd.conf(5)
- auditd.conf(5)
- ausearch-expression(5)
- audit.rules(7)
- audispd(8)
- auditctl(8)
- auditd(8)
- aulast(8)
- aulastlog(8)
- aureport(8)
- ausearch(8)
- ausyscall(8)
- autrace(8)
- auvirt(8)

Chapter 8. Federal Standards and Regulations

8.1. Introduction

In order to maintain security levels, it is possible for your organization to make efforts to comply with federal and industry security specifications, standards and regulations. This chapter describes some of these standards and regulations.

8.2. Federal Information Processing Standard (FIPS)

The Federal Information Processing Standard (FIPS) Publication 140-2, is a computer security standard, developed by a U.S. Government and industry working group to validate the quality of cryptographic modules. FIPS publications (including 140-2) can be found at the following URL: http://csrc.nist.gov/publications/PubsFIPS.html. Note that at the time of writing, Publication 140-3 is at Draft status, and may not represent the completed standard. The FIPS standard provides four (4) security *levels*, to ensure adequate coverage of different industries, implementations of cryptographic modules and organizational sizes and requirements. These levels are described below:

- Level 1 – Security Level 1 provides the lowest level of security. Basic security requirements are specified for a cryptographic module (e.g., at least one Approved algorithm or Approved security function shall be used). No specific physical security mechanisms are required in a Security Level 1 cryptographic module beyond the basic requirement for production-grade components. An example of a Security Level 1 cryptographic module is a personal computer (PC) encryption board.
- Level 2 – Security Level 2 enhances the physical security mechanisms of a Security Level 1 cryptographic module by adding the requirement for tamper-evidence, which includes the use of tamper-evident coatings or seals or for pick-resistant locks on removable covers or doors of the module. Tamper-evident coatings or seals are placed on a cryptographic module so that the coating or seal must be broken to attain physical access to the plain text cryptographic keys and critical security parameters (CSPs) within the module. Tamper-evident seals or pick-resistant locks are placed on covers or doors to protect against unauthorized physical access.
- Level 3 – In addition to the tamper-evident physical security mechanisms required at Security Level 2, Security Level 3 attempts to prevent the intruder from gaining access to CSPs held within the cryptographic module. Physical security mechanisms required at Security Level 3 are intended to have a high probability of detecting and responding to attempts at physical access, use or modification of the cryptographic module. The physical security mechanisms may include the use of strong enclosures and tamper detection/response circuitry that zeroes all plain text CSPs when the removable covers/doors of the cryptographic module are opened.
- Level 4 – Security Level 4 provides the highest level of security defined in this standard. At this security level, the physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access. Penetration of the cryptographic module enclosure from any direction has a very high probability of being detected, resulting in the immediate zeroization of all plain text CSPs. Security Level 4 cryptographic modules are useful for operation in physically unprotected environments.

Refer to the full FIPS 140-2 standard at http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf for further details on these levels and the other specifications of the FIPS standard.

8.2.1. Enabling FIPS Mode

To make Red Hat Enterprise Linux 6 compliant with the Federal Information Processing Standard (FIPS) Publication 140-2 you need to make several changes to ensure that certified cryptographic modules are used. To turn your system (kernel and user space) into FIPS mode, follow these steps:

1. For proper operation of the in-module integrity verification, the prelink has to be disabled. This can be done by setting **`PRELINKING=no`** in the **`/etc/sysconfig/prelink`** configuration file. Existing prelinking, if any, should be undone on all system files using the **`prelink -u -a`** command.
2. Next, install the *dracut-fips* package:

```
~]# yum install dracut-fips
```

Note

FIPS integrity verification is performed when the *dracut-fips* package is present on the system, regardless of whether the system operates in FIPS mode or not. However, the integrity verification results are ignored (or only logged) if the system or a shared library is not in FIPS mode, even when *dracut-fips* is present.

3. Recreate the **`initramfs`** file:

```
~]# dracut -f
```

Warning

This operation will overwrite the existing **`initramfs`** file.

4. Modify the kernel command line of the current kernel in the **`/boot/grub/grub.conf`** file by adding the following option:

```
fips=1
```

Note

If **`/boot`** or **`/boot/efi`** reside on separate partitions, the kernel parameter ***`boot=<partition of /boot or /boot/efi>`*** must be added to the kernel command line. Partitions can be identified with the **`df /boot`** or **`df /boot/efi`** command respectively. For example:

```
~]$ df /boot
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1               495844     53780    416464  12% /boot
```

In the example above, the /boot partition is located on **`/dev/sda1`**. Therefore, the following string needs to be appended to the kernel command line:

```
boot=/dev/sda1
```

5. Reboot your system.

Note

Ciphers and Message Authentication Codes (MACs) are set in FIPS mode by default in the **`/etc/ssh/sshd_config`** file. If your **`sshd_config`** contains any other ciphers and MACs, modify it to only use algorithms supported in FIPS mode, that is the following configuration or a subset thereof:

```
Protocol 2
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc,aes192-
cbc,aes256-cbc
Macs hmac-sha1,hmac-sha2-256,hmac-sha2-512
```

Should you require strict FIPS compliance, the **`fips=1`** kernel option needs to be added to the kernel command line during system installation so that key generation is done with FIPS approved algorithms and continuous monitoring tests in place. Users should also ensure that the system has plenty of entropy during the installation process by moving the mouse around, or if no mouse is available, ensuring that many keystrokes are typed. The recommended amount of keystrokes is 256 and more. Less than 256 keystrokes may generate a non-unique key.

Warning

The described procedure to enable FIPS mode on Red Hat Enterprise Linux systems does not affect a FIPS state of Network Security Services (NSS), and thus does not affect applications using NSS. When required, the user can switch any NSS application to FIPS mode using the following command:

```
~]# modutil -fips true -dbdir dir
```

where ***`dir`*** is a directory specifying an NSS database used for the application. If more than one NSS application uses this database, all these applications will be switched into FIPS mode. The applications have to be restarted for the NSS FIPS mode to take effect.

8.3. National Industrial Security Program Operating Manual (NISPOM)

The NISPOM (also called DoD 5220.22-M), as a component of the National Industrial Security Program (NISP), establishes a series of procedures and requirements for all government contractors with regard to classified information. The current NISPOM is dated February 28, 2006 with incorporated major changes from March 28, 2013. The NISPOM document can be downloaded from the following URL: http://www.nispom.org/NISPOM-download.html.

8.4. Payment Card Industry Data Security Standard (PCI DSS)

From https://www.pcisecuritystandards.org/about/index.shtml: *The PCI Security Standards Council is an open global forum, launched in 2006, that is responsible for the development, management, education, and awareness of the PCI Security Standards, including the Data Security Standard (DSS).*

You can download the PCI DSS standard from https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml.

8.5. Security Technical Implementation Guide

A Security Technical Implementation Guide or STIG is a methodology for standardized secure installation and maintenance of computer software and hardware.

Refer to the following URL for more information on STIG: http://iase.disa.mil/stigs/index.html .

Chapter 9. References

The following references are pointers to additional information that is relevant to SELinux and Red Hat Enterprise Linux but beyond the scope of this guide. Note that due to the rapid development of SELinux, some of this material may only apply to specific releases of Red Hat Enterprise Linux.

Books

SELinux by Example

Mayer, MacMillan, and Caplan

Prentice Hall, 2007

Tutorials and Help

Tutorials and talks from Russell Coker

http://www.coker.com.au/selinux/talks/ibmtu-2004/

Generic Writing SELinux policy HOWTO

http://www.lurking-grue.org/writingselinuxpolicyHOWTO.html

Red Hat Knowledgebase

http://kbase.redhat.com/

General Information

NSA SELinux main website

http://www.nsa.gov/selinux/

NSA SELinux FAQ

http://www.nsa.gov/selinux/info/faq.cfm

SELinux NSA's Open Source Security Enhanced Linux

http://www.oreilly.com/catalog/selinux/

Technology

An Overview of Object Classes and Permissions

http://www.tresys.com/selinux/obj_perms_help.html

Integrating Flexible Support for Security Policies into the Linux Operating System (a history of Flask implementation in Linux)

http://www.nsa.gov/research/_files/selinux/papers/selsymp2005.pdf

Implementing SELinux as a Linux Security Module

http://www.nsa.gov/research/_files/publications/implementing_selinux.pdf

A Security Policy Configuration for the Security-Enhanced Linux

http://www.nsa.gov/research/_files/selinux/papers/policy/policy.shtml

Community

SELinux community page

http://selinuxproject.org/

IRC

irc.freenode.net, #selinux, #fedora-selinux, #security

History

Quick history of Flask

http://www.cs.utah.edu/flux/fluke/html/flask.html

Full background on Fluke

http://www.cs.utah.edu/flux/fluke/html/index.html

Encryption Standards

A.1. Synchronous Encryption

A.1.1. Advanced Encryption Standard - AES

In cryptography, the Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. Government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor, the Data Encryption Standard (DES). [5]

A.1.1.1. AES History

AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process. Fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable. It became effective as a standard May 26, 2002. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information.

The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process. Rijndael is a portmanteau of the names of the two inventors. [6]

A.1.2. Data Encryption Standard - DES

The Data Encryption Standard (DES) is a block cipher (a form of shared secret encryption) that was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally. It is based on a symmetric-key algorithm that uses a 56-bit key. The algorithm was initially controversial with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis. [7]

A.1.2.1. DES History

DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes (see chronology). There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are unfeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES). [8]

In some documentation, a distinction is made between DES as a standard and DES the algorithm which is referred to as the DEA (the Data Encryption Algorithm). When spoken, "DES" is either spelled out as an abbreviation (/ˌdiːˌiːˈɛs/), or pronounced as a one-syllable acronym (/ˈdɛz/). [9]

A.2. Public-key Encryption

Public-key cryptography is a cryptographic approach, employed by many cryptographic algorithms and cryptosystems, whose distinguishing characteristic is the use of asymmetric key algorithms instead of or in addition to symmetric key algorithms. Using the techniques of public key-private key cryptography, many methods of protecting communications or authenticating messages formerly unknown have become practical. They do not require a secure initial exchange of one or more secret keys as is required when using symmetric key algorithms. It can also be used to create digital signatures. [10]

Public key cryptography is a fundamental and widely used technology around the world, and is the approach which underlies such Internet standards as Transport Layer Security (TLS) (successor to SSL), PGP and GPG. [11]

The distinguishing technique used in public key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same as the key used to decrypt it. Each user has a pair of cryptographic keys — a public key and a private key. The private key is kept secret, whilst the public key may be widely distributed. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key. The keys are related mathematically, but the private key cannot be feasibly (ie, in actual or projected practice) derived from the public key. It was the discovery of such algorithms which revolutionized the practice of cryptography beginning in the middle 1970s. [12]

In contrast, Symmetric-key algorithms, variations of which have been used for some thousands of years, use a single secret key shared by sender and receiver (which must also be kept private, thus accounting for the ambiguity of the common terminology) for both encryption and decryption. To use a symmetric encryption scheme, the sender and receiver must securely share a key in advance. [13]

Because symmetric key algorithms are nearly always much less computationally intensive, it is common to exchange a key using a key-exchange algorithm and transmit data using that key and a symmetric key algorithm. PGP, and the SSL/TLS family of schemes do this, for instance, and are called hybrid cryptosystems in consequence. [14]

A.2.1. Diffie-Hellman

Diffie–Hellman key exchange (D–H) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. [15]

A.2.1.1. Diffie-Hellman History

The scheme was first published by Whitfield Diffie and Martin Hellman in 1976, although it later emerged that it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, by Malcolm J. Williamson but was kept classified. In 2002, Hellman suggested the algorithm be called Diffie–Hellman–Merkle key exchange in recognition of Ralph Merkle's contribution to the invention of public-key cryptography (Hellman, 2002). [16]

Although Diffie–Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite). [17]

U.S. Patent 4,200,770, now expired, describes the algorithm and credits Hellman, Diffie, and Merkle as inventors. [18]

A.2.2. RSA

In cryptography, RSA (which stands for Rivest, Shamir and Adleman who first publicly described it) is an algorithm for public-key cryptography. It is the first algorithm known to be suitable for signing as well as encryption, and was one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations.

A.2.3. DSA

DSA (Digital Signature Algorithm) is a standard for digital signatures, a United States federal government standard for digital signatures. DSA is for signatures only and is not an encryption algorithm. [19]

A.2.4. SSL/TLS

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide security for communications over networks such as the Internet. TLS and SSL encrypt the segments of network connections at the Transport Layer end-to-end.

Several versions of the protocols are in widespread use in applications like web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). [20]

A.2.5. Cramer-Shoup Cryptosystem

The Cramer–Shoup system is an asymmetric key encryption algorithm, and was the first efficient scheme proven to be secure against adaptive chosen ciphertext attack using standard cryptographic assumptions. Its security is based on the computational intractability (widely assumed, but not proved) of the decisional Diffie–Hellman assumption. Developed by Ronald Cramer and Victor Shoup in 1998, it is an extension of the ElGamal cryptosystem. In contrast to ElGamal, which is extremely malleable, Cramer–Shoup adds additional elements to ensure non-malleability even against a resourceful attacker. This non-malleability is achieved through the use of a collision-resistant hash function and additional computations, resulting in a ciphertext which is twice as large as in ElGamal. [21]

A.2.6. ElGamal Encryption

In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie-Hellman key agreement. It was described by Taher Elgamal in 1985. ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. [22]

[5] "Advanced Encryption Standard." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[6] "Advanced Encryption Standard." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[7] "Data Encryption Standard." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Data_Encryption_Standard

[8] "Data Encryption Standard." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Data_Encryption_Standard

[9] "Data Encryption Standard." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Data_Encryption_Standard

[10] "Public-key Encryption." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[11] "Public-key Encryption." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[12] "Public-key Encryption." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[13] "Public-key Encryption." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[14] "Public-key Encryption." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[15] "Diffie-Hellman." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Diffie-Hellman

[16] "Diffie-Hellman." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Diffie-Hellman

[17] "Diffie-Hellman." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Diffie-Hellman

[18] "Diffie-Hellman." *Wikipedia.* 14 November 2009 http://en.wikipedia.org/wiki/Diffie-Hellman

[19] "DSA." *Wikipedia.* 24 February 2010 http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

[20] "TLS/SSL." *Wikipedia.* 24 February 2010 http://en.wikipedia.org/wiki/Transport_Layer_Security

[21] "Cramer-Shoup cryptosystem." *Wikipedia.* 24 February 2010 http://en.wikipedia.org/wiki/Cramer–Shoup_cryptosystem

[22] "ElGamal encryption" *Wikipedia.* 24 February 2010 http://en.wikipedia.org/wiki/ElGamal_encryption

Audit System Reference

B.1. Audit Event Fields

Table B.1, “Event Fields” lists all currently-supported Audit event fields. An event field is the value preceding the equal sign in the Audit log files.

Table B.1. Event Fields

Event Field	Explanation
a0, **a1**, **a2**, **a3**	Records the first four arguments of the system call, encoded in hexadecimal notation.
acct	Records a user's account name.
addr	Records the IPv4 or IPv6 address. This field usually follows a **hostname** field and contains the address the host name resolves to.
arch	Records information about the CPU architecture of the system, encoded in hexadecimal notation.
auid	Records the Audit user ID. This ID is assigned to a user upon login and is inherited by every process even when the user's identity changes (for example, by switching user accounts with **su - john**).
capability	Records the number of bits that were used to set a particular Linux capability. For more information on Linux capabilities, refer to the capabilities(7) man page.
cap_fi	Records data related to the setting of an inherited file system-based capability.
cap_fp	Records data related to the setting of a permitted file system-based capability.
cap_pe	Records data related to the setting of an effective process-based capability.
cap_pi	Records data related to the setting of an inherited process-based capability.
cap_pp	Records data related to the setting of a permitted process-based capability.
cgroup	Records the path to the **cgroup** that contains the process at the time the Audit event was generated.
cmd	Records the entire command line that is executed. This is useful in case of shell interpreters where the **exe** field records, for example, **/bin/bash** as the shell interpreter and the **cmd** field records the rest of the command line that is executed, for example **helloworld.sh --help**.
comm	Records the command that is executed. This is useful in case of shell interpreters where the **exe** field records, for example, **/bin/bash** as the shell interpreter and the **comm** field records the name of the script that is executed, for example **helloworld.sh**.
cwd	Records the path to the directory in which a system call was invoked.
data	Records data associated with TTY records.
dev	Records the minor and major ID of the device that contains the file or directory recorded in an event.
devmajor	Records the major device ID.
devminor	Records the minor device ID.
egid	Records the effective group ID of the user who started the analyzed process.
euid	Records the effective user ID of the user who started the analyzed process.

`exe`	Records the path to the executable that was used to invoke the analyzed process.
`exit`	Records the exit code returned by a system call. This value varies by system call. You can interpret the value to its human-readable equivalent with the following command: **`ausearch --interpret --exit`** ***`exit_code`***
`family`	Records the type of address protocol that was used, either IPv4 or IPv6.
`filetype`	Records the type of the file.
`flags`	Records the file system name flags.
`fsgid`	Records the file system group ID of the user who started the analyzed process.
`fsuid`	Records the file system user ID of the user who started the analyzed process.
`gid`	Records the group ID.
`hostname`	Records the host name.
`icmptype`	Records the type of a Internet Control Message Protocol (ICMP) package that is received. Audit messages containing this field are usually generated by **iptables**.
`id`	Records the user ID of an account that was changed.
`inode`	Records the inode number associated with the file or directory recorded in an Audit event.
`inode_gid`	Records the group ID of the inode's owner.
`inode_uid`	Records the user ID of the inode's owner.
`items`	Records the number of path records that are attached to this record.
`key`	Records the user defined string associated with a rule that generated a particular event in the Audit log.
`list`	Records the Audit rule list ID. The following is a list of known IDs: » 0 — **`user`** » 1 — **`task`** » 4 — **`exit`** » 5 — **`exclude`**
`mode`	Records the file or directory permissions, encoded in numerical notation.
`msg`	Records a time stamp and a unique ID of a record, or various event-specific **`<`*`name`*`>=<`*`value`*`>`** pairs provided by the kernel or user space applications.
`msgtype`	Records the message type that is returned in case of a user-based AVC denial. The message type is determined by D-Bus.
`name`	Records the full path of the file or directory that was passed to the system call as an argument.
`new-disk`	Records the name of a new disk resource that is assigned to a virtual machine.
`new-mem`	Records the amount of a new memory resource that is assigned to a virtual machine.
`new-vcpu`	Records the number of a new virtual CPU resource that is assigned to a virtual machine.

new-net	Records the MAC address of a new network interface resource that is assigned to a virtual machine.
new_gid	Records a group ID that is assigned to a user.
oauid	Records the user ID of the user that has logged in to access the system (as opposed to, for example, using **su**) and has started the target process. This field is exclusive to the record of type **OBJ_PID**.
ocomm	Records the command that was used to start the target process.This field is exclusive to the record of type **OBJ_PID**.
opid	Records the process ID of the target process. This field is exclusive to the record of type **OBJ_PID**.
oses	Records the session ID of the target process. This field is exclusive to the record of type **OBJ_PID**.
ouid	Records the real user ID of the target process
obj	Records the SELinux context of an object. An object can be a file, a directory, a socket, or anything that is receiving the action of a subject.
obj_gid	Records the group ID of an object.
obj_lev_high	Records the high SELinux level of an object.
obj_lev_low	Records the low SELinux level of an object.
obj_role	Records the SELinux role of an object.
obj_uid	Records the UID of an object
obj_user	Records the user that is associated with an object.
ogid	Records the object owner's group ID.
old-disk	Records the name of an old disk resource when a new disk resource is assigned to a virtual machine.
old-mem	Records the amount of an old memory resource when a new amount of memory is assigned to a virtual machine.
old-vcpu	Records the number of an old virtual CPU resource when a new virtual CPU is assigned to a virtual machine.
old-net	Records the MAC address of an old network interface resource when a new network interface is assigned to a virtual machine.
old_prom	Records the previous value of the network promiscuity flag.
ouid	Records the real user ID of the user who started the target process.
path	Records the full path of the file or directory that was passed to the system call as an argument in case of AVC-related Audit events
perm	Records the file permission that was used to generate an event (that is, read, write, execute, or attribute change)
pid	The **pid** field semantics depend on the origin of the value in this field. In fields generated from user-space, this field holds a process ID. In fields generated by the kernel, this field holds a thread ID. The thread ID is equal to process ID for single-threaded processes. Note that the value of this thread ID is different from the values of pthread_t IDs used in user-space. For more information, refer to the gettid(2) man page.
ppid	Records the Parent Process ID (PID).
prom	Records the network promiscuity flag.
proto	Records the networking protocol that was used. This field is specific to

	Audit events generated by **iptables**.
res	Records the result of the operation that triggered the Audit event.
result	Records the result of the operation that triggered the Audit event.
saddr	Records the socket address.
sauid	Records the sender Audit login user ID. This ID is provided by D-Bus as the kernel is unable to see which user is sending the original **auid**.
ses	Records the session ID of the session from which the analyzed process was invoked.
sgid	Records the set group ID of the user who started the analyzed process.
sig	Records the number of a signal that causes a program to end abnormally. Usually, this is a sign of a system intrusion.
subj	Records the SELinux context of a subject. A subject can be a process, a user, or anything that is acting upon an object.
subj_clr	Records the SELinux clearance of a subject.
subj_role	Records the SELinux role of a subject.
subj_sen	Records the SELinux sensitivity of a subject.
subj_user	Records the user that is associated with a subject.
success	Records whether a system call was successful or failed.
suid	Records the set user ID of the user who started the analyzed process.
syscall	Records the type of the system call that was sent to the kernel.
terminal	Records the terminal name (without **/dev/**).
tty	Records the name of the controlling terminal. The value **(none)** is used if the process has no controlling terminal.
uid	Records the real user ID of the user who started the analyzed process.
vm	Records the name of a virtual machine from which the Audit event originated.

B.2. Audit Record Types

Table B.2, “Record Types” lists all currently-supported types of Audit records. The event type is specified in the **type=** field at the beginning of every Audit record.

Table B.2. Record Types

Event Type	Explanation
`ADD_GROUP`	Triggered when a user-space group is added.
`ADD_USER`	Triggered when a user-space user account is added.
`ANOM_ABEND` [a]	Triggered when a processes ends abnormally (with a signal that could cause a core dump, if enabled).
`ANOM_ACCESS_FS`[a]	Triggered when a file or a directory access ends abnormally.
`ANOM_ADD_ACCT`[a]	Triggered when a user-space account addition ends abnormally.
`ANOM_AMTU_FAIL`[a]	Triggered when a failure of the Abstract Machine Test Utility (AMTU) is detected.
`ANOM_CRYPTO_FAIL`[a]	Triggered when a failure in the cryptographic system is detected.
`ANOM_DEL_ACCT`[a]	Triggered when a user-space account deletion ends abnormally.
`ANOM_EXEC`[a]	Triggered when an execution of a file ends abnormally.
`ANOM_LOGIN_ACCT`[a]	Triggered when an account login attempt ends abnormally.
`ANOM_LOGIN_FAILURES`[a]	Triggered when the limit of failed login attempts is reached.
`ANOM_LOGIN_LOCATION`[a]	Triggered when a login attempt is made from a forbidden location.
`ANOM_LOGIN_SESSIONS`[a]	Triggered when a login attempt reaches the maximum amount of concurrent sessions.
`ANOM_LOGIN_TIME`[a]	Triggered when a login attempt is made at a time when it is prevented by, for example, **`pam_time`**.
`ANOM_MAX_DAC`[a]	Triggered when the maximum amount of Discretionary Access Control (DAC) failures is reached.
`ANOM_MAX_MAC`[a]	Triggered when the maximum amount of Mandatory Access Control (MAC) failures is reached.
`ANOM_MK_EXEC`[a]	Triggered when a file is made executable.
`ANOM_MOD_ACCT`[a]	Triggered when a user-space account modification ends abnormally.
`ANOM_PROMISCUOUS`[a]	Triggered when a device enables or disables promiscuous mode.
`ANOM_RBAC_FAIL`[a]	Triggered when a Role-Based Access Control (RBAC) self-test failure is detected.
`ANOM_RBAC_INTEGRITY_FAIL`[a]	Triggered when a Role-Based Access Control (RBAC) file integrity test failure is detected.
`ANOM_ROOT_TRANS`[a]	Triggered when a user becomes root.
`AVC`	Triggered to record an SELinux permission check..
`AVC_PATH`	Triggered to record the **`dentry`** and **`vfsmount`** pair when an SELinux permission check occurs.
`BPRM_FCAPS`	Triggered when a user executes a program with a file system capability.
`CAPSET`	Triggered to record the capabilities being set for process-based capabilities, for example, running as root to drop capabilities.
`CHGRP_ID`	Triggered when a user-space group ID is changed.
`CHUSER_ID`	Triggered when a user-space user ID is changed.
`CONFIG_CHANGE`	Triggered when the Audit system configuration is modified.

CRED_ACQ	Triggered when a user acquires user-space credentials.
CRED_DISP	Triggered when a user disposes of user-space credentials.
CRED_REFR	Triggered when a user refreshes their user-space credentials.
CRYPTO_FAILURE_USER	Triggered when a decrypt, encrypt, or randomize cryptographic operation fails.
CRYPTO_KEY_USER	Triggered to record the cryptographic key identifier used for cryptographic purposes.
CRYPTO_LOGIN	Triggered when a cryptographic officer login attempt is detected.
CRYPTO_LOGOUT	Triggered when a crypto officer logout attempt is detected.
CRYPTO_PARAM_CHANGE_USER	Triggered when a change in a cryptographic parameter is detected.
CRYPTO_REPLAY_USER	Triggered when a replay attack is detected.
CRYPTO_SESSION	Triggered to record parameters set during a TLS session establishment.
CRYPTO_TEST_USER	Triggered to record cryptographic test results as required by the FIPS-140 standard.
CWD	Triggered to record the current working directory.
DAC_CHECK	Triggered to record DAC check results.
DAEMON_ABORT	Triggered when a daemon is stopped due to an error.
DAEMON_ACCEPT	Triggered when the **auditd** daemon accepts a remote connection.
DAEMON_CLOSE	Triggered when the **auditd** daemon closes a remote connection.
DAEMON_CONFIG	Triggered when a daemon configuration change is detected.
DAEMON_END	Triggered when a daemon is successfully stopped.
DAEMON_RESUME	Triggered when the **auditd** daemon resumes logging.
DAEMON_ROTATE	Triggered when the **auditd** daemon rotates the Audit log files.
DAEMON_START	Triggered when the **auditd** daemon is started.
DEL_GROUP	Triggered when a user-space group is deleted
DEL_USER	Triggered when a user-space user is deleted
DEV_ALLOC	Triggered when a device is allocated.
DEV_DEALLOC	Triggered when a device is deallocated.
EOE	Triggered to record the end of a multi-record event.
EXECVE	Triggered to record arguments of the **execve(2)** system call.
FD_PAIR	Triggered to record the use of the **pipe** and **socketpair** system calls.
FS_RELABEL	Triggered when a file system relabel operation is detected.
GRP_AUTH	Triggered when a group password is used to authenticate against a user-space group.
INTEGRITY_DATA [b]	Triggered to record a data integrity verification event run by the kernel.
INTEGRITY_HASH[b]	Triggered to record a hash type integrity verification event run by the kernel.
INTEGRITY_METADATA[b]	Triggered to record a metadata integrity verification event run by the kernel.
INTEGRITY_PCR[b]	Triggered to record Platform Configuration Register (PCR) invalidation messages.

INTEGRITY_RULE[b]	Triggered to record a policy rule.
INTEGRITY_STATUS[b]	Triggered to record the status of integrity verification.
IPC	Triggered to record information about a Inter-Process Communication object referenced by a system call.
IPC_SET_PERM	Triggered to record information about new values set by an **IPC_SET** control operation on an IPC object
KERNEL	Triggered to record the initialization of the Audit system.
KERNEL_OTHER	Triggered to record information from third-party kernel modules.
LABEL_LEVEL_CHANGE	Triggered when an object's level label is modified.
LABEL_OVERRIDE	Triggered when an administrator overrides an object's level label.
LOGIN	Triggered to record relevant login information when a user log in to access the system.
MAC_CIPSOV4_ADD	Triggered when a Commercial Internet Protocol Security Option (CIPSO) user adds a new Domain of Interpretation (DOI). Adding DOIs is a part of the packet labeling capabilities of the kernel provided by NetLabel.
MAC_CIPSOV4_DEL	Triggered when a CIPSO user deletes an existing DOI. Adding DOIs is a part of the packet labeling capabilities of the kernel provided by NetLabel.
MAC_CONFIG_CHANGE	Triggered when an SELinux Boolean value is changed.
MAC_IPSEC_EVENT	Triggered to record information about an IPSec event, when one is detected, or when the IPSec configuration changes.
MAC_MAP_ADD	Triggered when a new Linux Security Module (LSM) domain mapping is added. LSM domain mapping is a part of the packet labeling capabilities of the kernel provided by NetLabel.
MAC_MAP_DEL	Triggered when an existing LSM domain mapping is added. LSM domain mapping is a part of the packet labeling capabilities of the kernel provided by NetLabel.
MAC_POLICY_LOAD	Triggered when a SELinux policy file is loaded.
MAC_STATUS	Triggered when the SELinux mode (enforcing, permissive, off) is changed.
MAC_UNLBL_ALLOW	Triggered when unlabeled traffic is allowed when using the packet labeling capabilities of the kernel provided by NetLabel
MAC_UNLBL_STCADD	Triggered when a static label is added when using the packet labeling capabilities of the kernel provided by NetLabel
MAC_UNLBL_STCDEL	Triggered when a static label is deleted when using the packet labeling capabilities of the kernel provided by NetLabel
MMAP	Triggered to record a file descriptor and flags of the **mmap(2)** system call.
MQ_GETSETATTR	Triggered to record the **mq_getattr(3)** and **mq_setattr(3)** message queue attributes.
MQ_NOTIFY	Triggered to record arguments of the **mq_notify(3)** system call.
MQ_OPEN	Triggered to record arguments of the **mq_open(3)** system call.
MQ_SENDRECV	Triggered to record arguments of the **mq_send(3)** and **mq_receive(3)** system calls.
NETFILTER_CFG	Triggered when Netfilter chain modifications are detected.

NETFILTER_PKT	Triggered to record packets traversing Netfilter chains.
OBJ_PID	Triggered to record information about a process to which a signal is sent.
PATH	Triggered to record file name path information.
RESP_ACCT_LOCK [c]	Triggered when a user account is locked.
RESP_ACCT_LOCK_TIMED[c]	Triggered when a user account is locked for a specified period of time.
RESP_ACCT_REMOTE[c]	Triggered when a user account is locked from a remote session.
RESP_ACCT_UNLOCK_TIMED[c]	Triggered when a user account is unlocked after a configured period of time.
RESP_ALERT[c]	Triggered when an alert email is sent.
RESP_ANOMALY[c]	Triggered when an anomaly was not acted upon.
RESP_EXEC[c]	Triggered when an intrusion detection program responds to a threat originating from the execution of a program.
RESP_HALT[c]	Triggered when the system is shut down.
RESP_KILL_PROC[c]	Triggered when a process is terminated.
RESP_SEBOOL[c]	Triggered when an SELinux Boolean value is set.
RESP_SINGLE[c]	Triggered when the system is put into single-user mode.
RESP_TERM_ACCESS[c]	Triggered when a session is terminated.
RESP_TERM_LOCK[c]	Triggered when a terminal is locked.
ROLE_ASSIGN	Triggered when an administrator assigns a user to an SELinux role.
ROLE_MODIFY	Triggered when an administrator modifies an SELinux role.
ROLE_REMOVE	Triggered when an administrator removes a user from an SELinux role.
SELINUX_ERR	Triggered when an internal SELinux error is detected.
SERVICE_START	Triggered when a service is started.
SERVICE_STOP	Triggered when a service is stopped.
SOCKADDR	Triggered to record a socket address user or returned by a system call.
SOCKETCALL	Triggered to record arguments of the **sys_socketcall** system call (used to multiplex many socket-related system calls).
SYSCALL	Triggered to record a system call to the kernel.
SYSTEM_BOOT	Triggered when the system is booted up.
SYSTEM_RUNLEVEL	Triggered when the system's run level is changed.
SYSTEM_SHUTDOWN	Triggered when the system is shut down.
TEST	Triggered to record the success value of a test message.
TRUSTED_APP	The record of this type can be used by third party application that require auditing.
TTY	Triggered when TTY input was sent to an administrative process.
USER_ACCT	Triggered when a user-space user account is modified.
USER_AUTH	Triggered when a user-space authentication attempt is detected.
USER_AVC	Triggered when a user-space AVC message is generated.
USER_CHAUTHTOK	Triggered when a user account attribute is modified.

USER_CMD	Triggered when a user-space shell command is executed.
USER_END	Triggered when a user-space session is terminated.
USER_ERR	Triggered when a user account state error is detected.
USER_LABELED_EXPORT	Triggered when an object is exported with an SELinux label.
USER_LOGIN	Triggered when a user logs in.
USER_LOGOUT	Triggered when a user logs out.
USER_MAC_POLICY_LOAD	Triggered when a user-space daemon loads an SELinux policy.
USER_MGMT	Triggered to record user-space management data.
USER_ROLE_CHANGE	Triggered when a user's SELinux role is changed.
USER_SELINUX_ERR	Triggered when a user-space SELinux error is detected.
USER_START	Triggered when a user-space session is started.
USER_TTY	Triggered when an explanatory message about TTY input to an administrative process is sent from user-space.
USER_UNLABELED_EXPORT	Triggered when an object is exported without SELinux label.
USYS_CONFIG	Triggered when a user-space system configuration change is detected.
VIRT_CONTROL	Triggered when a virtual machine is started, paused, or stopped.
VIRT_MACHINE_ID	Triggered to record the binding of a label to a virtual machine.
VIRT_RESOURCE	Triggered to record resource assignment of a virtual machine.

[a] All Audit event types prepended with **ANOM** are intended to be processed by an intrusion detection program.

[b] This event type is related to the Integrity Measurement Architecture (IMA), which functions best with a Trusted Platform Module (TPM) chip.

[c] All Audit event types prepended with **RESP** are intended responses of an intrusion detection system in case it detects malicious activity on the system.

Revision History

Revision 1-10.406	**Thu Nov 25 2013**	**Tomáš Čapek**
Release of the Security Guide for Red Hat Enterprise Linux 6.5		
Revision 1-9.9	**Feb 21 2013**	**Martin Prpič**
Release of the Security Guide for Red Hat Enterprise Linux 6.4		
Revision 1-8.25	**Jun 20 2012**	**Martin Prpič**
Release of the Security Guide for Red Hat Enterprise Linux 6.3		
Revision 1-7	**Dec 6 2011**	**Martin Prpič**
Release of the Security Guide for Red Hat Enterprise Linux 6.2		
Revision 1-6	**Aug 05 2011**	**Paul Kennedy**
Resolves BZ#702249, broken link.		
Revision 1-5	**Apr 19 2010**	**Scott Radvan**
Minor fixes, final build for Beta		
Revision 1-4	**Mar 5 2010**	**Scott Radvan**
QE Review and Updates		
Revision 1-3	**Feb 19 2010**	**Scott Radvan**
Push to testing area ready for review.		