Fedora 19

Security Guide

A Guide to Securing Fedora Linux

Johnray Fuller

John Ha

David O'Brien

Scott Radvan

Eric Christensen

Adam Ligas

Murray McAllister

Scott Radvan

Daniel Walsh

Dominick Grift

Eric Paris

James Morris

Fedora 19 Security Guide
A Guide to Securing Fedora Linux
Edition 19.1

Author	Johnray Fuller	jrfuller@redhat.com
Author	John Ha	jha@redhat.com
Author	David O'Brien	daobrien@redhat.com
Author	Scott Radvan	sradvan@redhat.com
Author	Eric Christensen	sparks@fedoraproject.org
Author	Adam Ligas	agent86@fedoraproject.org
Author	Murray McAllister	mmcallis@redhat.com
Author	Scott Radvan	sradvan@redhat.com
Author	Daniel Walsh	dwalsh@redhat.com
Author	Dominick Grift	domg472@gmail.com
Author	Eric Paris	eparis@parisplace.org
Author	James Morris	jmorris@redhat.com

The Fedora Security Guide is designed to assist users of Fedora in learning the processes and practices of securing workstations and servers against local and remote intrusion, exploitation, and malicious activity. Focused on Fedora Linux but detailing concepts and techniques valid for all Linux systems, the Fedora Security Guide details the planning and the tools involved in creating a secured computing environment for the data center, workplace, and home. With proper administrative

knowledge, vigilance, and tools, systems running Linux can be both fully functional and secured from most common intrusion and exploit methods.

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*[1] 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 includes 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 keycaps 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 keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

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

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

Proportional Bold

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

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **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:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;
```

```
public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

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/bugzilla/* against the product **Fedora.**

When submitting a bug report, be sure to mention the manual's identifier: *security-guide*

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.

Security Overview

Because of 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 the organization. Because most organizations are increasingly dynamic in nature, with workers accessing company IT resources locally and remotely, the need for secure computing environments has become more pronounced.

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

1.1. 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 crucial information regard their data as an important part of their overall assets. Several terms and metrics have entered our daily business vocabulary, such as total cost of ownership (TCO) and quality of service (QoS). Using these metrics, industries can calculate aspects such as data integrity and high-availability as part of their planning and process management costs. In some industries, such as electronic commerce, the availability and trustworthiness of data can be the difference between success and failure.

1.1.1.1. 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, as well as 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 transaction, the Internet has been regarded as one of the most important developments of the 20th century.

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

[1] http://law.jrank.org/pages/3791/Kevin-Mitnick-Case-1999.html
[2] http://www.livinginternet.com/i/ia_hackers_levin.htm

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 assailants using specially created, widely available programs that scanned vulnerable network servers, installed client applications called *trojans* on the servers, and timed an attack with every infected server flooding the victim sites and rendering them unavailable. Many blame the attack on fundamental flaws in the way routers and the protocols used are structured to accept all incoming data, no matter where or for what purpose the packets are sent.

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.[3]

In a separate incident, the billing records of over 2.2 million patients stored on a backup tape were stolen from the front seat of a courier's car.[4]

Currently, an estimated 1.8 billion people use or have used the Internet worldwide.[5] At the same time:

- On any given day, there are approximately 225 major incidences of security breach reported to the CERT Coordination Center at Carnegie Mellon University.[6]
- In 2003, the number of CERT reported incidences jumped to 137,529 from 82,094 in 2002 and from 52,658 in 2001.[7]
- The worldwide economic impact of the three most dangerous Internet Viruses of the last three years was estimated at US$13.2 Billion.[8]

From a 2008 global survey of business and technology executives "The Global State of Information Security"[9], undertaken by *CIO Magazine*, some points are:

- Just 43% of respondents audit or monitor user compliance with security policies
- Only 22% keep an inventory of the outside companies that use their data
- The source of nearly half of security incidents was marked as "Unknown"
- 44% of respondents plan to increase security spending in the next year
- 59% have an information security strategy

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

[3] http://www.theregister.co.uk/2007/05/04/txj_nonfeasance/
[4] http://www.healthcareitnews.com/story.cms?id=9408
[5] http://www.internetworldstats.com/stats.htm
[6] http://www.cert.org
[7] http://www.cert.org/stats/fullstats.html
[8] http://www.newsfactor.com/perl/story/16407.html
[9] http://www.csoonline.com/article/454939/The_Global_State_of_Information_Security_

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

1.1.1.3. 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

Fedora 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. A detailed discussion of SELinux can be found in *Section 9.1, “Introduction”*.

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 Fedora. 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. 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.

1.2.1. A Quick History of Hackers

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

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

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

1.2.1.1. Shades of Gray

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.

1.2.2. Threats to Network Security

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

1.2.2.1. Insecure Architectures

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

1.2.2.1.1. Broadcast Networks

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

1.2.2.1.2. Centralized Servers

Another potential networking pitfall is the use of centralized computing. A common cost-cutting measure for many businesses is to consolidate all services to a single powerful machine. This can be convenient as it is easier to manage and costs considerably less than multiple-server configurations. However, a centralized server introduces a single point of failure on the network. If the central server is compromised, it may render the network completely useless or worse, prone to data manipulation or theft. In these situations, a central server becomes an open door which allows access to the entire network.

1.2.3. Threats to Server Security

Server security is as important as network security because servers often hold a great deal of an organization's vital information. If a server is compromised, all of its contents may become available for the cracker to steal or manipulate at will. The following sections detail some of the main issues.

1.2.3.1. Unused Services and Open Ports

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

A common occurrence among system administrators is to install the operating system without paying attention to what programs are actually being installed. This can be problematic because unneeded services may be installed, configured with the default settings, and possibly turned on. This can cause unwanted services, such as Telnet, DHCP, or DNS, to run on a server or workstation without the administrator realizing it, which in turn can cause unwanted traffic to the server, or even, a potential pathway into the system for crackers. Refer To *Section 3.2, “Server Security”* for information on closing ports and disabling unused services.

1.2.3.2. Unpatched Services

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

However, there is no such thing as perfect software and there is always room for further refinement. Moreover, newer software is often not as rigorously tested as one might expect, because of its recent arrival to production environments or because it may not be as popular as other server software.

Developers and system administrators often find exploitable bugs in server applications and publish the information on bug tracking and security-related websites such as the Bugtraq mailing list (*http://www.securityfocus.com*) or the Computer Emergency Response Team (CERT) website (*http://www.cert.org*). Although these mechanisms are an effective way of alerting the community to security vulnerabilities, it is up to system administrators to patch their systems promptly. This is particularly true because crackers have access to these same vulnerability tracking services and will use the information to crack unpatched systems whenever they can. Good system administration requires vigilance, constant bug tracking, and proper system maintenance to ensure a more secure computing environment.

Refer to *Section 1.5, “Security Updates”* for more information about keeping a system up-to-date.

1.2.3.3. 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."[10] This applies as much to inexperienced administrators as it does to overconfident or amotivated administrators.

Some administrators fail to patch their servers and workstations, while others fail to watch log messages from the system kernel or network traffic. Another common error is when default passwords or keys to services are left unchanged. For example, some databases have default administration passwords because the database developers assume that the system administrator changes these passwords immediately after installation. If a database administrator fails to change this password, even an inexperienced cracker can use a widely-known default password to gain administrative privileges to the database. These are only a few examples of how inattentive administration can lead to compromised servers.

1.2.3.4. Inherently Insecure Services

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

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

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

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

By default, Fedora 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 3.2, “Server Security”* for more information about setting up services in a safe manner.

[10] http://www.sans.org/resources/errors.php

1.2.4. Threats to Workstation and Home PC Security

Workstations and home PCs may not be as prone to attack as networks or servers, but since they often contain sensitive data, such as credit card information, they are targeted by system crackers. Workstations can also be co-opted without the user's knowledge and used by attackers as "slave" machines in coordinated attacks. For these reasons, knowing the vulnerabilities of a workstation can save users the headache of reinstalling the operating system, or worse, recovering from data theft.

1.2.4.1. Bad Passwords

Bad passwords are one of the easiest ways for an attacker to gain access to a system. For more on how to avoid common pitfalls when creating a password, refer to *Section 3.1.3, “Password Security”*.

1.2.4.2. Vulnerable Client Applications

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

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

Section 3.1, “Workstation Security” discusses in more detail what steps administrators and home users should take to limit the vulnerability of computer workstations.

1.3. Vulnerability Assessment

Given time, resources, and motivation, a cracker can break into nearly any system. At the end of the day, all of the security procedures and technologies currently available cannot guarantee that any systems are 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.

1.3.1. Thinking Like the Enemy

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

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

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

A vulnerability assessment is an internal audit of your network and system security; the results of which indicate the confidentiality, integrity, and availability of your network (as explained in *Section 1.1.1.3, “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.

1.3.2. Defining Assessment and Testing

Vulnerability assessments may be broken down into one of two types: *Outside looking in* and *inside looking around*.

When performing an outside looking in vulnerability assessment, you are attempting to compromise your systems from the outside. Being external to your company provides you with the cracker's viewpoint. You see what a cracker sees — publicly-routable IP addresses, systems on your *DMZ*, external interfaces of your firewall, and more. DMZ stands for "demilitarized zone", which corresponds to a computer or small subnetwork that sits between a trusted internal network, such as a corporate private LAN, and an untrusted external network, such as the public Internet. Typically, the DMZ contains devices accessible to Internet traffic, such as Web (HTTP) servers, FTP servers, SMTP (e-mail) servers and DNS servers.

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

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

Consider the difference between vulnerability assessments and *penetration tests*. Think of a vulnerability assessment as the first step to a penetration test. The information gleaned from the assessment is used for testing. Whereas the assessment is 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 shows an overview, which can turn up false positives and false negatives.

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

Now that the difference between a vulnerability assessment and a penetration test is defined, take the findings of the assessment and review them carefully before conducting a penetration test as part of your new best practices approach.

Warning

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

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

- Creates proactive focus on information security
- Finds potential exploits before crackers find them
- Results in systems being kept up to date and patched
- Promotes growth and aids in developing staff expertise
- Abates financial loss and negative publicity

1.3.2.1. Establishing a Methodology

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

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

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

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

1.3.3. Evaluating the Tools

An assessment can start by using some form of an information gathering tool. When assessing the entire network, map the layout first to find the hosts that are running. Once located, examine each host individually. Focusing on these hosts requires another set of tools. Knowing which tools to use may be the most crucial step in finding vulnerabilities.

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

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

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

1.3.3.1. Scanning Hosts with Nmap

Nmap is a popular port scanning tool included in Fedora that can be used to determine the layout of a network. Nmap has been available for many years and is probably the most often used tool when gathering information. An excellent man page is included that provides a detailed description of its options and usage. Administrators can use Nmap on a network to find host systems and open ports on those systems.

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

1.3.3.1.1. Using Nmap

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

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

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

```
Starting Nmap 4.68 ( http://nmap.org )
Interesting ports on foo.example.com:
Not shown: 1710 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
53/tcp  open   domain
70/tcp  closed gopher
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/

1.3.3.2. 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 is included in Fedora repositories but 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/

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

1.3.3.4. VLAD the Scanner

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

Note

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

More information about VLAD can be found on the RAZOR team website at the following URL:

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

1.3.3.5. Anticipating Your Future Needs

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

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, though some services that run on Linux can contain default administrator passwords.	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 crackers in performing such a vulnerability. Depends on target system running services (such as **`rsh`**, **`telnet`**, FTP and others) that use *source-based* authentication techniques, which are not recommended when compared to PKI or other forms of encrypted

Exploit	Description	Notes
		authentication used in **ssh** or SSL/ TLS.
Eavesdropping	Collecting data that passes between two active nodes on a network by eavesdropping on the connection between the two nodes.	This type of attack works mostly with plain text transmission protocols such as Telnet, FTP, and HTTP transfers. Remote attacker must have access to a compromised system on a LAN in order to perform such an attack; usually the cracker has used an active attack (such as IP spoofing or man-in-the-middle) to compromise a system on the LAN. 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	Workstations and desktops are more prone to exploitation as workers do not have the expertise or experience to prevent or detect a compromise; it

Exploit	Description	Notes
	compromise, or crash systems. Further exploitation can occur if the compromised workstation has administrative privileges on the rest of the network.	is imperative to inform individuals of the risks they are taking when they install unauthorized software or open unsolicited email attachments. Safeguards can be implemented such that email client software does not automatically open or execute attachments. Additionally, the automatic update of workstation software via Red Hat Network or other system management services can alleviate the burdens of multi-seat security deployments.
Denial of Service (DoS) Attacks	Attacker or group of attackers coordinate against an organization's network or server resources by sending unauthorized packets to the target host (either server, router, or workstation). This forces the resource to become unavailable to legitimate users.	The most reported DoS case in the US occurred in 2000. Several highly-trafficked commercial and government sites were rendered unavailable by a coordinated ping flood attack using several compromised systems with high bandwidth connections acting as *zombies*, or redirected broadcast nodes. Source packets are usually forged (as well as rebroadcasted), making investigation as to the true source of the attack difficult. Advances in ingress filtering (IETF rfc2267) using **`iptables`** and Network 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 Fedora distribution that is currently supported, Fedora 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 Fedora 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 Fedora and to check the signature of the package to verify its integrity.

Note

Fedora includes a convenient panel icon that displays visible alerts when there is an update for a Fedora system.

1.5.2. Verifying Signed Packages

All Fedora packages are signed with the Fedora *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 Fedora does not match the private key during RPM verification, the package may have been altered and therefore cannot be trusted.

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

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

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

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

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

The output will look similar to the following:

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

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

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

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

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

For each package, if the GPG key verifies successfully, the command returns **`gpg OK`**. If it doesn't, make sure you are using the correct Fedora public key, as well as verifying the source of the content. Packages that do not pass GPG verifications 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.

1.5.3. Installing Signed Packages

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

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

For kernel packages use the following command:

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

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

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

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

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

Note

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

Important

Before installing any security errata, be sure to read any special instructions contained in the errata report and execute them accordingly. Refer to *Section 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 Fedora operating system. It manages access to memory, the processor, and peripherals as well as schedules all tasks.

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

Shared Libraries

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

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

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

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

SysV Services

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

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

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

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

xinetd Services

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

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

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

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

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

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

```
kill <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 **ps** command) for an IMAP session.

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

```
killall imapd
```

Basic Hardening Guide

Every computer system should be hardened against threats found both over the network as well as those found physically at the computer. The system changes are necessary based on default settings usually being set to allow software to work over the software being secure. As with any change to a system these changes could cause unintended results. Changes should be evaluated for appropriateness on your system before implementing.

2.1. General Principles

Encrypt all data transmitted over the network. Encrypting authentication information (such as passwords) is particularly important.
Minimize the amount of software installed and running in order to minimize vulnerability.
Use security-enhancing software and tools whenever available (e.g. SELinux and IPTables).
Run each network service on a separate server whenever possible. This minimizes the risk that a compromise of one service could lead to a compromise of others.
Maintain user accounts. Create a good password policy and enforce its use. Delete unused user accounts.
Review system and application logs on a routine basis. Send logs to a dedicated log server. This prevents intruders from easily avoiding detection by modifying the local logs.
Never log in directly as root, unless absolutely necessary. Administrators should use **`sudo`** to execute commands as root when required. The accounts capable of using sudo are specified in **`/etc/sudoers`**, which is edited with the visudo utility. By default, relevant logs are written to **`/var/log/secure`**.

2.2. Physical Security

Physical security of the system is of utmost importance. Many of the suggestions given here won't protect your system if the attacker has physical access to the system. Physical access doesn't necessarily mean that the battle is lost, however. Strengthening your BIOS and boot software can help defend your system against certain types of attacks.

Configuring the BIOS to disable booting from CDs/DVDs, floppies, and external devices, can prevent bypassing the boot partition and the boot loader where other protections are in place. It is important to password-protect your BIOS settings so that an attacker cannot just change these, and other, settings. Next, set a password for the GRUB bootloader. Use the **grub2-mkpasswd-pbkdf2** to create your password hash. This prevents users from entering single user mode or changing settings at boot time.

2.3. Why this is important

An attacker could take complete control of your system by booting from an external source. By booting from an external source (e.g. a live Linux CD) many of the security settings are bypassed. If the attacker can modify the GRUB settings they can boot into single user mode which allows admin access to the system.

Additional explaination and hardening can be found in *Section 3.1.2, “BIOS and Boot Loader Security”* of this guide.

2.4. Networking

The computer's network connection is the gateway to your system. Your files and processor time could be available to anyone who successfully connects to your system via this network connection if other safeguards have not been implemented. One of the primary ways to keep you in control of your system is to prevent the attackers from gaining access to your system in the first place.

2.4.1. iptables

iptables is the most widely used firewall software on Linux systems today. This program intercepts packets coming into your computer via the network connection and filters them according to rules you have specified.

2.4.2. IPv6

IPv6 is the latest Internet protocol which aims to solve the address quantity shortfall inherent to IPv4. And while there are no security risks directly associated with the new protocol there are a few things to understand before utilizing this new technology.

Most system administrators are familiar with IPv4 and the work-arounds that were put in place to make IPv4 work. One of these work-arounds is network address translation, or *NAT*. NAT is traditionally used to keep the number of needed public IP addresses to a minimum when setting up a local area network. Systems on these networks do not all require public IP addresses and valuable address space can be saved by implementing this technology. There are some security features that were side effects to NAT; the biggest being that outside traffic cannot make it inside the network unless a port is forwarded across the router. Because IPv6 solves the addressing problem there is no longer a need to use NAT. Everything can have a public IP address and, by extension, everything is not publically routable across the Internet when physical and logical connections are made.

Another thing to worry about is how security software deals with this new protocol. **iptables** does not know or understand IPv6 and so it ignores those packets altogether. That means if your network is utilizing IPv6 and you have not activated **ip6tables** then you have just left the door to your system open to the world.

Using IPv6 is not dangerous as long as you know and understand the changes that your system's software went through to make it possible to use this new network protocol.

2.5. Keeping software up to date

Software gets patched everyday. Some of these updates fix security problems that were identified by the developers. When these patches become available it is important that they are applied to your system as soon as possible. One of the easier ways to manage updates for your system is using **yum**. A special plugin is available to allow only security updates to be installed while ignoring bugfixes and enhancements. This plugin is explained better at *Section 8.1, “YUM Plugin”*.

2.6. Services

Services in Linux are programs that run as daemons in the background. It is important to audit these programs regularly to determine if they need to be running. Many daemons open network ports in order to listen for calls. Having unnecessary ports open can harm the overall security of the system. An unknown security flaw in a piece of software can allow a hacker into a system for no good reason.

2.7. NTP

Network Time Protocol, or *NTP*, keeps the time on your systems accurate. Time is a very important piece of the security puzzle and should be maintained as precisely as possible. Time is used in log files, timestamps, and in encryption. If someone is able to control the time settings on one of your systems then they are able to make the recreation of a break-in that much more difficult.

Securing Your Network

3.1. 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.

3.1.1. Evaluating Workstation Security

When evaluating the security of a Fedora 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?

3.1.2. 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 secure location where only trusted people have access and the computer 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 a trade show, it could lead to a major security breach with ramifications for the entire company.

3.1.2.1. BIOS Passwords

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

1. *Preventing Changes to BIOS Settings* — If an intruder has access to the BIOS, they can set it to boot from a diskette or CD-ROM. This makes it possible for them to enter rescue mode or single user mode, which in turn allows them to start arbitrary processes on the system or copy sensitive data.

[1] 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.

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.

3.1.2.1.1. 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.

3.1.2.2. 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.

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.

Fedora ships with the GRUB boot loader on the x86 platform. For a detailed look at GRUB, refer to the Fedora Installation Guide.

3.1.2.2.1. Password Protecting GRUB

You can configure GRUB to address the first two issues listed in *Section 3.1.2.2, “Boot Loader Passwords”* by adding a password directive to its configuration file.

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.

3.1.3. Password Security

Passwords are the primary method that Fedora 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 *Message-Digest Algorithm* (*MD5*) and shadow passwords. It is highly recommended that you do not alter these settings.

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

If shadow passwords are deselected during installation, all passwords are stored as a one-way hash in the world-readable **`/etc/passwd`** file, which makes the system vulnerable to offline password cracking attacks. If an intruder can gain access to the machine as a regular user, he can copy the **`/etc/passwd`** file to his own machine and run any number of password cracking programs against it. If there is an insecure password in the file, it is only a matter of time before the password cracker discovers it.

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

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

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.

3.1.3.1. Creating Strong Passwords

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

- *Do Not Use Only Letters or Numbers* — Using only letters or numbers does not make for a complex password.

  Some insecure examples include the following:

  - 8675309
  - juan
  - hackme

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

  Some insecure examples include the following:

  - john1
  - DS-9
  - mentat123

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

  Some insecure examples include the following:

  - cheguevara
  - bienvenido1

  - 1dumbKopf

- *Do Not Use Hacker Terminology* — If you think you are elite because you use hacker terminology — also called l337 (LEET) speak — in your password, think again. Many word lists include LEET speak.

  Some insecure examples include the following:

  - H4X0R
  - 1337

- *Do Not Use Personal Information* — 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 or anniversary 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 Twelve Characters Long* — The longer the password, the better. If using MD5 passwords, it should be 15 characters or longer.

- *Mix Upper and Lower Case Letters* — Fedora uses case sensitive passwords, 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.
- *Use a Password Generator* — Using a random password generator, along with secure password storage software, can make it very difficult for an attacker to discover your password.

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.

3.1.3.1.1. 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.

3.1.3.2. 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.

3.1.3.2.1. 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 Manager* (*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. 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://www.kernel.org/pub/linux/libs/pam/modules.html*. For more information about PAM, refer to *Section 3.5, “Pluggable Authentication Modules (PAM)”*.

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 Fedora, 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**. It can be found online at *http://www.crypticide.com/alecm/security/crack/c50-faq.html*.
- ***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.

3.1.3.2.2. 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.

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

The **`-M`** option of the **`chage`** command specifies the maximum number of days the password is valid. 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).

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:

```
[root@myServer ~]# chage davido
Changing the aging information for davido
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]:
[root@myServer ~]#
```

Refer to the man page for chage for more information on the available options.

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.

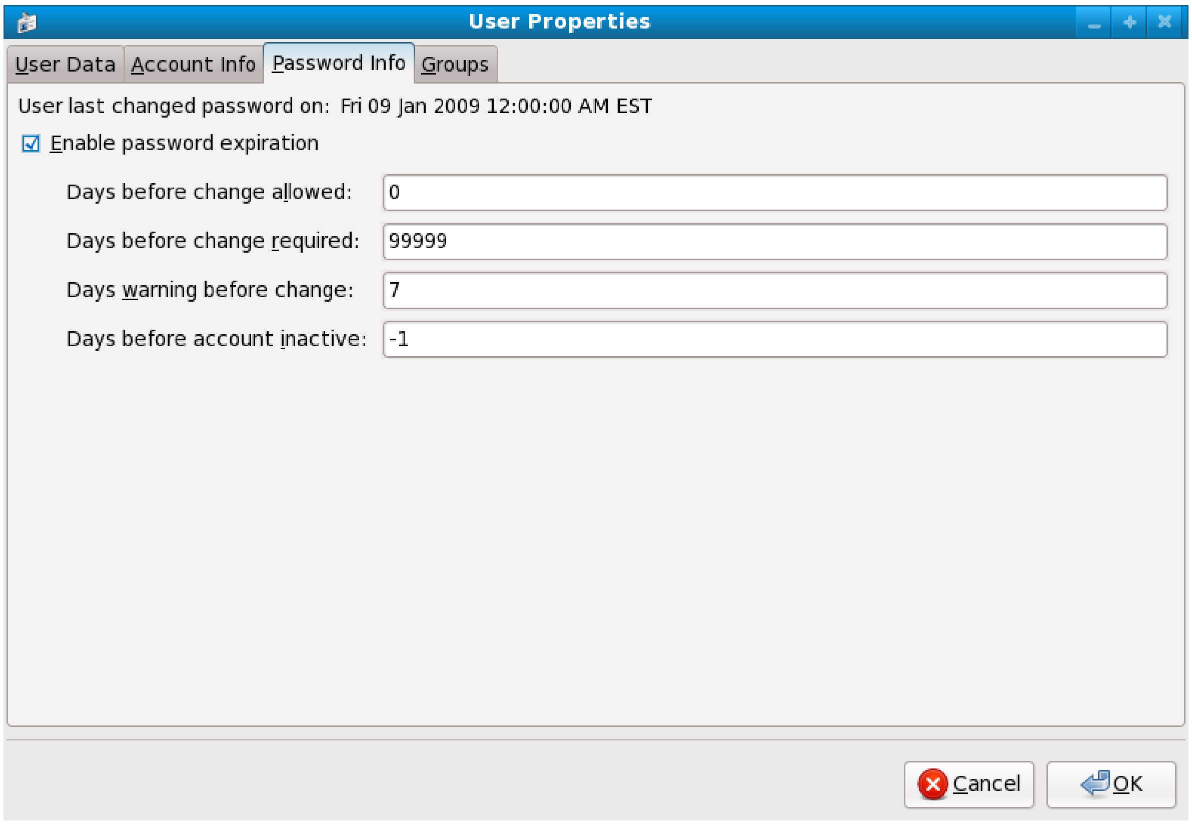
1. Click the **System** menu on the Panel, point to **Administration** and then click **Users and Groups** to display the User Manager. Alternatively, type the command **`system-config-users`** at a shell prompt.

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 3.1. Specifying password aging options

3.1.4. Administrative Controls

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

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

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 *Section 3.5, "Pluggable Authentication Modules (PAM)"* 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.

3.1.4.1. 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.

3.1.4.2. Disallowing Root Access

If an administrator is uncomfortable allowing users to log in as root for these or other reasons, the root password should be kept secret, and access to runlevel one or single user mode should be disallowed through boot loader password protection (refer to *Section 3.1.2.2, “Boot Loader Passwords”* for more information on this topic.)

Table 3.1, “Methods of Disabling the Root Account” describes ways that an administrator can further ensure that root logins are disallowed:

Table 3.1. Methods of Disabling the Root Account

Method	Description	Effects	Does Not Affect
Changing the root shell.	Edit the **`/etc/passwd`** file and change the shell from **`/bin/bash`** to **`/sbin/nologin`**.	Prevents access to the root shell and logs 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).	An empty **`/etc/securetty`** file prevents root login on any devices attached to the computer.	Prevents access to the root account via the console or the network. The following programs are prevented from accessing the root account: · **`login`** · **`gdm`** · **`kdm`** · **`xdm`**	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`**

Method	Description	Effects	Does Not Affect
		· Other network services that open a tty	· **ssh** · **scp** · **sftp**
Disabling root SSH logins.	Edit the **/etc/ssh/sshd_config** file and set the **PermitRootLogin** parameter to **no**.	Prevents root access via the OpenSSH suite of tools. The following programs are prevented from accessing the root account: · **ssh** · **scp** · **sftp**	This only prevents root access to the OpenSSH suite of tools.
Use PAM to limit root access to services.	Edit the file for the target service in the **/etc/pam.d/** directory. Make sure the **pam_listfile.so** is required for authentication.[1]	Prevents root access to network services that are PAM aware. The following services are prevented from accessing the root account: · FTP clients · Email clients · **login** · **gdm** · **kdm** · **xdm** · **ssh** · **scp** · **sftp** · Any PAM aware services	Programs and services that are not PAM aware.

[1] Refer to *Section 3.1.4.2.4, "Disabling Root Using PAM"* for details.

3.1.4.2.1. Disabling 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. This prevents access to the root account through commands that require a shell, such as the **su** and the **ssh** commands.

Important

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

3.1.4.2.2. Disabling Root Logins

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 his machine as root via Telnet, which transmits the password in plain text over the network. By default, Fedora's **/etc/securetty** file only allows the root user to log in at the console physically attached to the machine. To prevent root from logging in, remove the contents of this file by typing the following command:

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

Warning

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

3.1.4.2.3. Disabling Root SSH Logins

Root logins via the SSH protocol are disabled by default in Fedora; however, if this option has been enabled, it can be disabled again by editing the SSH daemon's configuration file (**/etc/ssh/sshd_config**). Change the line that reads:

```
PermitRootLogin yes
```

to read as follows:

```
PermitRootLogin no
```

For these changes to take effect, the SSH daemon must be restarted. This can be done via the following command:

```
kill -HUP `cat /var/run/sshd.pid`
```

3.1.4.2.4. Disabling Root Using PAM

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. Below 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 in the following example is *not* necessary if the directive is on one 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 *Section 3.5, “Pluggable Authentication Modules (PAM)”*.

3.1.4.3. 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**.

3.1.4.3.1. The su Command

When a user executes the **su** command, they are prompted for the root password and, after authentication, is given a root shell prompt.

Once logged in via the **su** command, the user *is* the root user and has absolute administrative access to the system[2]. In addition, once a user has become root, it is possible for them to use the **su** command to change to any other user on the system without being prompted for a password.

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

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

```
usermod -G wheel <username>
```

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

You can also use the **User Manager** to modify group memberships, 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).

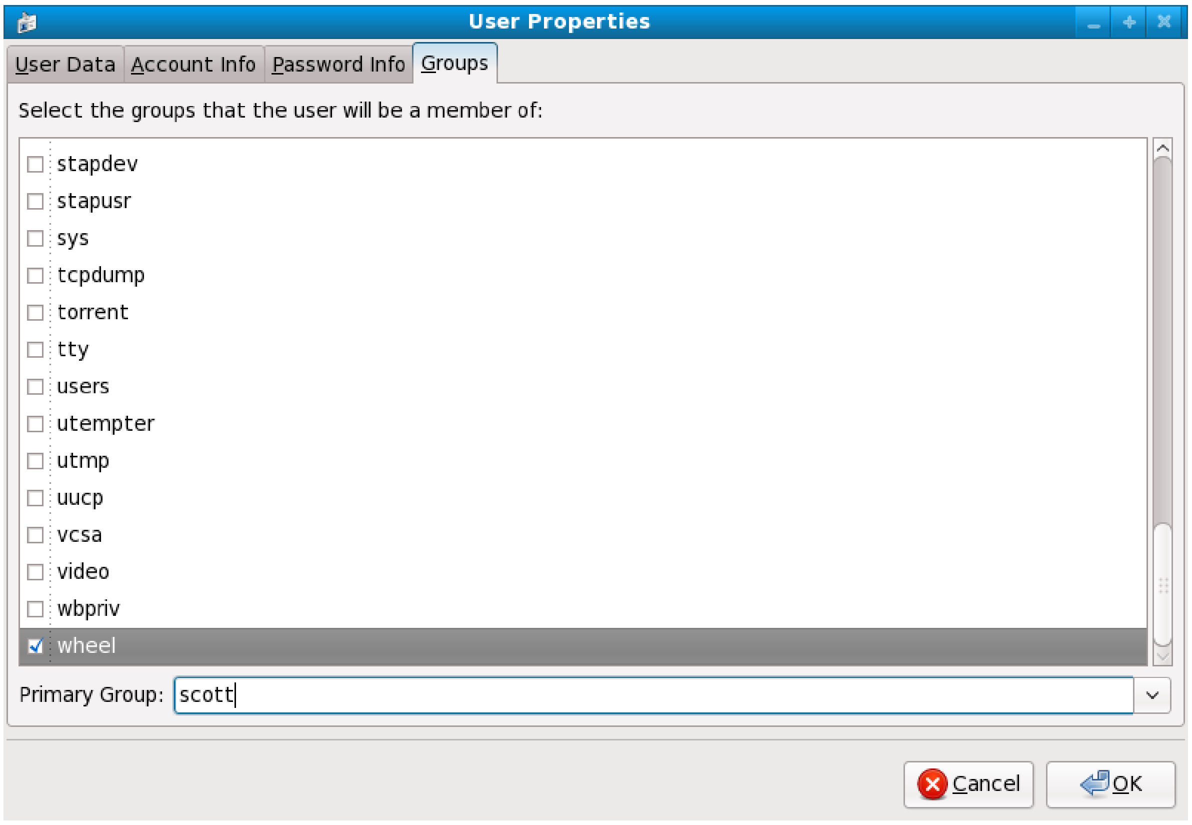
4. Click the **Groups** tab, select the check box for the wheel group, and then click **OK**. Refer to *Figure 3.2, “Adding users to the "wheel" group.”*.

5. Open the PAM configuration file for **su** (**/etc/pam.d/su**) in a text editor and remove the comment **#** from the following line:

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

   This change means that only members of the administrative group **wheel** can use this program.

[2] This access is still subject to the restrictions imposed by SELinux, if it is enabled.


User Properties

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

- [ ] stapdev
- [ ] stapusr
- [ ] sys
- [ ] tcpdump
- [ ] torrent
- [ ] tty
- [ ] users
- [ ] utempter
- [ ] utmp
- [ ] uucp
- [ ] vcsa
- [ ] video
- [ ] wbpriv
- [✓] wheel

Primary Group: scott

Figure 3.2. Adding users to the "wheel" group.

Note

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

3.1.4.3.2. The sudo Command

The **sudo** command offers another approach to giving users administrative access. When trusted users precede an administrative command with **sudo**, they are prompted for *their own* password. Then, when they have been authenticated and assuming that the command is permitted, the administrative command is executed as if they were the root user.

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

```
sudo <command>
```

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

Important

Users of the **sudo** command should take extra care to log out before walking away from their machines since sudoers can use the command again without being asked for a password within a five minute period. This setting can be altered via the configuration file, **`/etc/sudoers`**.

The **sudo** command allows for a high degree of flexibility. For instance, only users listed in the **`/etc/sudoers`** configuration file are allowed to use the **sudo** command and the command is executed in *the user's* shell, not a root shell. This means the root shell can be completely disabled, as shown in *Section 3.1.4.2.1, "Disabling the Root Shell"*.

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

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

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

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

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

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

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

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

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

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

3.1.5. 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 Fedora 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.

3.1.5.1. Risks To Services

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

- *Denial of Service Attacks (DoS)* — By flooding a service with requests, a denial of service attack can 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 co-ordinated 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 Fedora 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 stealthy buffer overflow exploits.

Important

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

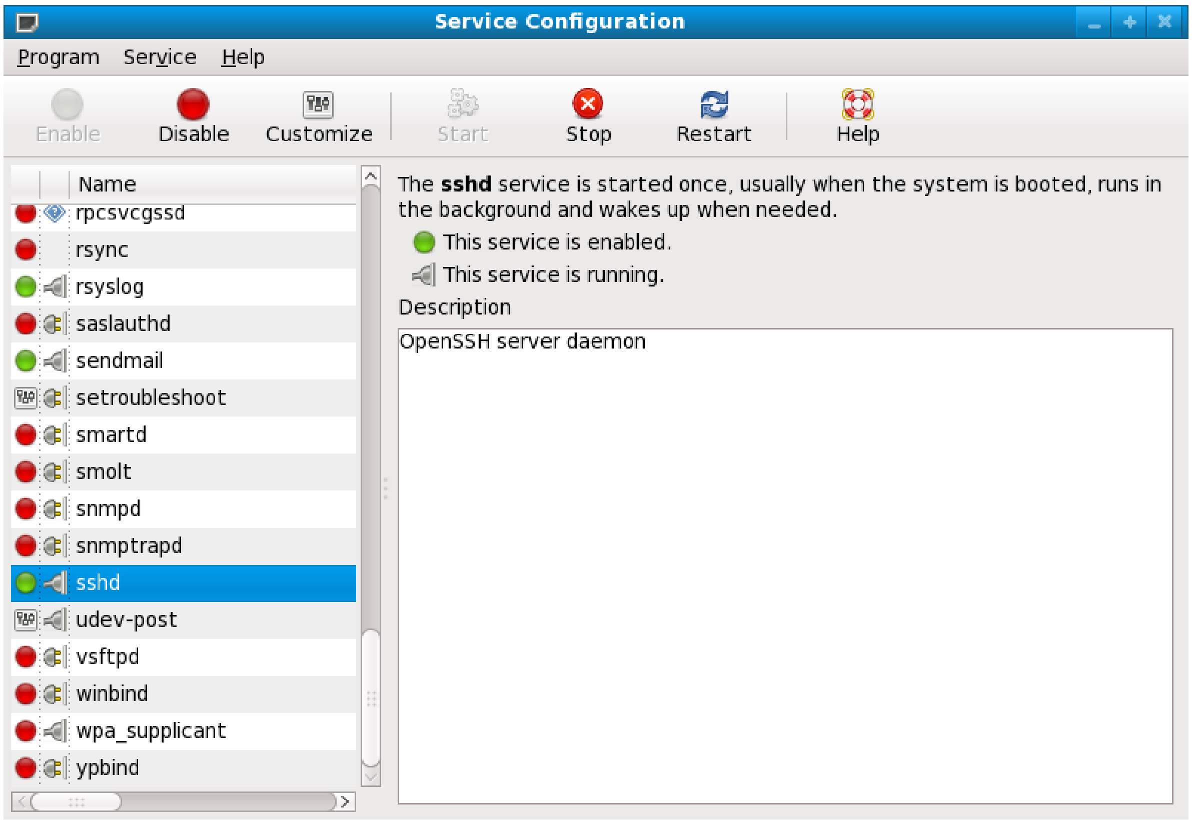
3.1.5.2. Identifying and Configuring Services

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

- **`cupsd`** — The default print server for Fedora.
- **`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 err on the side of caution. For example, if a printer is not available, do not leave **`cupsd`** running. The same is true for **`portmap`**. If you do not mount NFSv3 volumes or use NIS (the **`ypbind`** service), then **`portmap`** should be disabled.



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 3.3. **Services Configuration Tool**

If unsure of the purpose for a particular service, the **Services Configuration Tool** has a description field, illustrated in *Figure 3.3, “**Services Configuration Tool**”*, that provides additional information.

Checking which network services are available to start at boot time is only part of the story. You should also check which ports are open and listening. Refer to *Section 3.2.8, “Verifying Which Ports Are Listening”* for more information.

3.1.5.3. 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 3.1.7, “Security Enhanced Communication Tools”* for more information about **`sshd`**.

FTP is not as inherently dangerous to the security of the system as remote shells, but FTP servers must be carefully configured and monitored to avoid problems. Refer to *Section 3.2.6, “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 Fedora releases.)
- **`netdump`**
- **`netdump-server`**
- **`nfs`**
- **`rwhod`**
- **`sendmail`**
- **`smb`** (Samba)
- **`yppasswdd`**
- **`ypserv`**
- **`ypxfrd`**

More information on securing network services is available in *Section 3.2, “Server Security”*.

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

3.1.6. Personal Firewalls

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

Important

You should 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, care should be taken when configuring a firewall to block access to ports not in use, while not blocking access to ports used by configured services.

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

For advanced users and server administrators, manually configuring a firewall with **`iptables`** is probably a better option. Refer to *Section 3.8, “Using Firewalls”* for more information.

3.1.7. Security Enhanced Communication Tools

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

Fedora 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 4.2.2, “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.

3.2. 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.

3.2.1. 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 3.8, "Using 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.

3.2.1.1. 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.

3.2.1.1.1. 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 line:

```
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/
```

3.2.1.1.2. 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/ '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.

3.2.1.1.3. 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.

3.2.1.2. 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.

3.2.1.2.1. 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.

3.2.1.2.2. 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.

3.2.2. 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.

3.2.2.1. 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.

3.2.2.2. 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 3.8, “Using Firewalls”* for more information about implementing firewalls with iptables commands.

3.2.3. 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.

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

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

NIS is 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 an NIS server first secure the **`portmap`** service as outlined in *Section 3.2.2, “Securing Portmap”*, then address the following issues, such as network planning.

3.2.3.1. 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.

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

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

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

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

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

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

Note

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

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

3.2.3.3. 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 an NIS server for the first time without creating the **`/var/yp/securenets`** file.

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

3.2.3.4. 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 3.8, “Using Firewalls”* for more information about implementing firewalls with iptables commands.

3.2.3.5. 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 an 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 *Section 3.7, "Kerberos"* for more information about Kerberos.

3.2.4. Securing NFS

Important

The version of NFS included in Fedora, NFSv4, no longer requires the **`portmap`** service as outlined in *Section 3.2.2, "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 Fedora supports NFSv2 and NFSv3, both of which utilize **`portmap`**.

3.2.4.1. Carefully Plan the Network

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

3.2.4.2. 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>
```

3.2.4.3. Do Not Use the `no_root_squash` Option

By default, NFS shares change the root user to the **`nfsnobody`** user, an unprivileged user account. 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.

3.2.4.4. 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.

3.2.5. Securing the Apache HTTP Server

The Apache HTTP Server is one of the most stable and secure services that ships with Fedora. 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:

1.
```
chown root <directory_name>
```

2.
```
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.

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.

3.2.6. 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.

Fedora 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.

3.2.6.1. 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 3.2.1.1.1, “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 3.2.1.1.1, “TCP Wrappers and Connection Banners”*.

3.2.6.2. 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.

3.2.6.2.1. Anonymous Upload

To allow anonymous users to upload files, it is recommended that a write-only directory be created within **/var/ftp/pub/**.

To do this, type the following command:

```
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:

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

Warning

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

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

```
anon_upload_enable=YES
```

3.2.6.3. 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
```

3.2.6.3.1. 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 3.1.4.2.4, “Disabling Root Using PAM”*. 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`**

3.2.6.4. Use TCP Wrappers To Control Access

Use TCP Wrappers to control access to either FTP daemon as outlined in *Section 3.2.1.1, “Enhancing Security With TCP Wrappers”*.

3.2.7. 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.

3.2.7.1. 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. 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.

3.2.7.2. NFS and Sendmail

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

Because NFSv2 and NFSv3 do not maintain control over user and group IDs, two or more users can have the same UID, and receive and read each other's mail.

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.

3.2.7.3. Mail-only Users

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

3.2.8. Verifying Which Ports Are Listening

After configuring network services, it is important to pay attention to which ports are actually listening on the system's network interfaces. Any open ports can be evidence of an intrusion.

There are two basic approaches for listing the ports that are listening on the network. The less reliable approach is to query the network stack using commands such as **`netstat -an`** or **`lsof -i`**. This method is less reliable since these programs do not connect to the machine from the network, but rather check to see what is running on the system. For this reason, these applications are frequent targets for replacement by attackers. Crackers attempt to cover their tracks if they open unauthorized network ports by replacing **`netstat`** and **`lsof`** with their own, modified versions.

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

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

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

The output of this command appears as follows:

```
Starting Nmap 4.68 ( http://nmap.org ) at 2009-03-06 12:08 EST
```

```
Interesting ports on localhost.localdomain (127.0.0.1):
Not shown: 1711 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
111/tcp   open  rpcbind
113/tcp   open  auth
631/tcp   open  ipp
834/tcp   open  unknown
2601/tcp  open  zebra
32774/tcp open  sometimes-rpc11
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.24
Uptime: 4.122 days (since Mon Mar  2 09:12:31 2009)
Network Distance: 0 hops
OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.420 seconds
```

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

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

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

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

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

The command returns the following output:

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

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

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

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

The relevant portion of the output from this command follows:

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

These tools reveal a great deal about the status of the services running on a machine. These tools are flexible and can provide a wealth of information about network services and configuration. Refer to the man pages for **lsof**, **netstat**, **nmap**, and **services** for more information.

3.3. Single Sign-on (SSO)

3.3.1. Introduction

The Fedora SSO functionality reduces the number of times Fedora 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 Fedora from the log-in screen, and then not need to re-enter their passwords. These applications are detailed below.

In addition, users can log in to their machines even when there is no network (*offline mode*) or where network connectivity is unreliable, for example, wireless access. In the latter case, services will degrade gracefully.

3.3.1.1. Supported Applications

The following applications are currently supported by the unified log-in scheme in Fedora:

- Login
- Screensaver
- Firefox and Thunderbird

3.3.1.2. Supported Authentication Mechanisms

Fedora currently supports the following authentication mechanisms:

- Kerberos name/password login
- Smart card/PIN login

3.3.1.3. Supported Smart Cards

Fedora has been tested with the Cyberflex e-gate card and reader, but any card that complies with both Java card 2.1.1 and Global Platform 2.0.1 specifications should operate correctly, as should any reader that is supported by PCSC-lite.

Fedora has also been tested with Common Access Cards (CAC). The supported reader for CAC is the SCM SCR 331 USB Reader.

Gemalto smart cards (Cyberflex Access 64k v2, standard with DER SHA1 value configured as in PKCSI v2.1) are now supported. These smart cards now use readers compliant with Chip/Smart Card Interface Devices (CCID).

3.3.1.4. Advantages of Fedora Single Sign-on

Numerous security mechanisms currently exist that utilize a large number of protocols and credential stores. Examples include SSL, SSH, IPsec, and Kerberos. Fedora SSO aims to unify these schemes to support the requirements listed above. This does not mean replacing Kerberos with X.509v3 certificates, but rather uniting them to reduce the burden on both system users and the administrators who manage them.

To achieve this goal, Fedora:

- Provides a single, shared instance of the NSS crypto libraries on each operating system.

- Ships the Certificate System's Enterprise Security Client (ESC) with the base operating system. The ESC application monitors smart card insertion events. If it detects that the user has inserted a smart card that was designed to be used with the Certificate System server product, it displays a user interface instructing the user how to enroll that smart card.
- Unifies Kerberos and NSS so that users who log in to the operating system using a smart card also obtain a Kerberos credential (which allows them to log in to file servers, etc.)

3.3.2. Getting Started with your new Smart Card

Before you can use your smart card to log in to your system and take advantage of the increased security options this technology provides, you need to perform some basic installation and configuration steps. These are described below.

Note

This section provides a high-level view of getting started with your smart card. More detailed information is available in the Red Hat Certificate System Enterprise Security Client Guide.

1. Log in with your Kerberos name and password
2. Make sure you have the **`nss-tools`** package loaded.
3. Download and install your corporate-specific root certificates. Use the following command to install the root CA certificate:

```
certutil -A -d /etc/pki/nssdb -n "root ca cert" -t "CT,C,C" -i ./
ca_cert_in_base64_format.crt
```

4. Verify that you have the following RPMs installed on your system: esc, pam_pkcs11, coolkey, ifd-egate, ccid, gdm, authconfig, and authconfig-gtk.
5. Enable Smart Card Login Support
   a. On the Gnome Title Bar, select System->Administration->Authentication.
   b. Type your machine's root password if necessary.
   c. In the Authentication Configuration dialog, click the **Authentication** tab.
   d. Select the **Enable Smart Card Support** check box.
   e. Click the **Configure Smart Card...** button to display the Smartcard Settings dialog, and specify the required settings:
      - **Require smart card for login** — Clear this check box. After you have successfully logged in with the smart card you can select this option to prevent users from logging in without a smart card.
      - **Card Removal Action** — This controls what happens when you remove the smart card after you have logged in. The available options are:
        - **Lock** — Removing the smart card locks the X screen.

- **Ignore** — Removing the smart card has no effect.

6. If you need to enable the Online Certificate Status Protocol (OCSP), open the **`/etc/pam_pkcs11/pam_pkcs11.conf`** file, and locate the following line:

   `enable_ocsp = false;`

   Change this value to true, as follows:

   `enable_ocsp = true;`

7. Enroll your smart card

8. If you are using a CAC card, you also need to perform the following steps:

   a. Change to the root account and create a file called **`/etc/pam_pkcs11/cn_map`**.

   b. Add the following entry to the **`cn_map`** file:

      `MY.CAC_CN.123454` -> *`myloginid`*

      where *`MY.CAC_CN.123454`* is the Common Name on your CAC and *`myloginid`* is your UNIX login ID.

9. Logout

3.3.2.1. Troubleshooting

If you have trouble getting your smart card to work, try using the following command to locate the source of the problem:

```
pklogin_finder debug
```

If you run the **`pklogin_finder`** tool in debug mode while an enrolled smart card is plugged in, it attempts to output information about the validity of certificates, and if it is successful in attempting to map a login ID from the certificates that are on the card.

3.3.3. How Smart Card Enrollment Works

Smart cards are said to be *enrolled* when they have received an appropriate certificate signed by a valid Certificate Authority (CA). This involves several steps, described below:

1. The user inserts their smart card into the smart card reader on their workstation. This event is recognized by the Enterprise Security Client (ESC).

2. The enrollment page is displayed on the user's desktop. The user completes the required details and the user's system then connects to the Token Processing System (TPS) and the CA.

3. The TPS enrolls the smart card using a certificate signed by the CA.

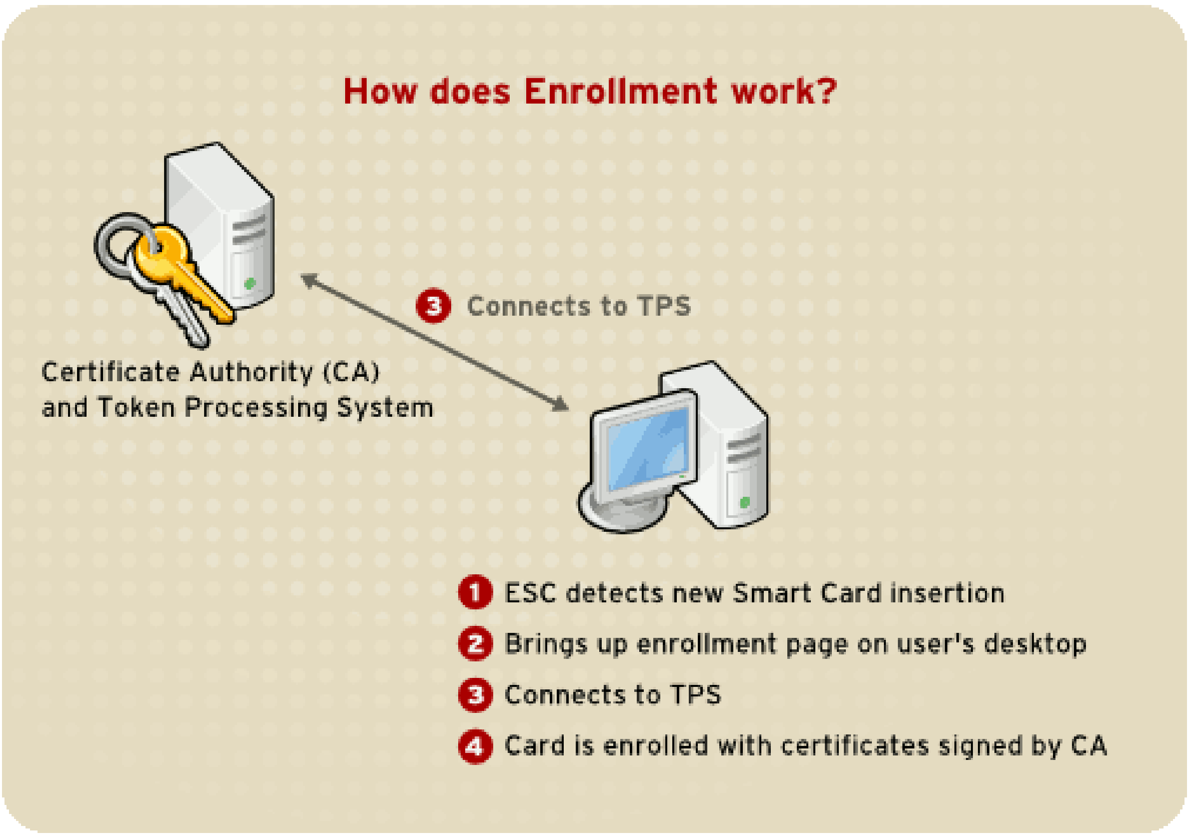

How does Enrollment work?
Certificate Authority (CA) and Token Processing System
3 Connects to TPS
1 ESC detects new Smart Card insertion
2 Brings up enrollment page on user's desktop
3 Connects to TPS
4 Card is enrolled with certificates signed by CA

Figure 3.4. How Smart Card Enrollment Works

3.3.4. How Smart Card Login Works

This section provides a brief overview of the process of logging in using a smart card.

1. When the user inserts their smart card into the smart card reader, this event is recognized by the PAM facility, which prompts for the user's PIN.
2. The system then looks up the user's current certificates and verifies their validity. The certificate is then mapped to the user's UID.
3. This is validated against the KDC and login granted.

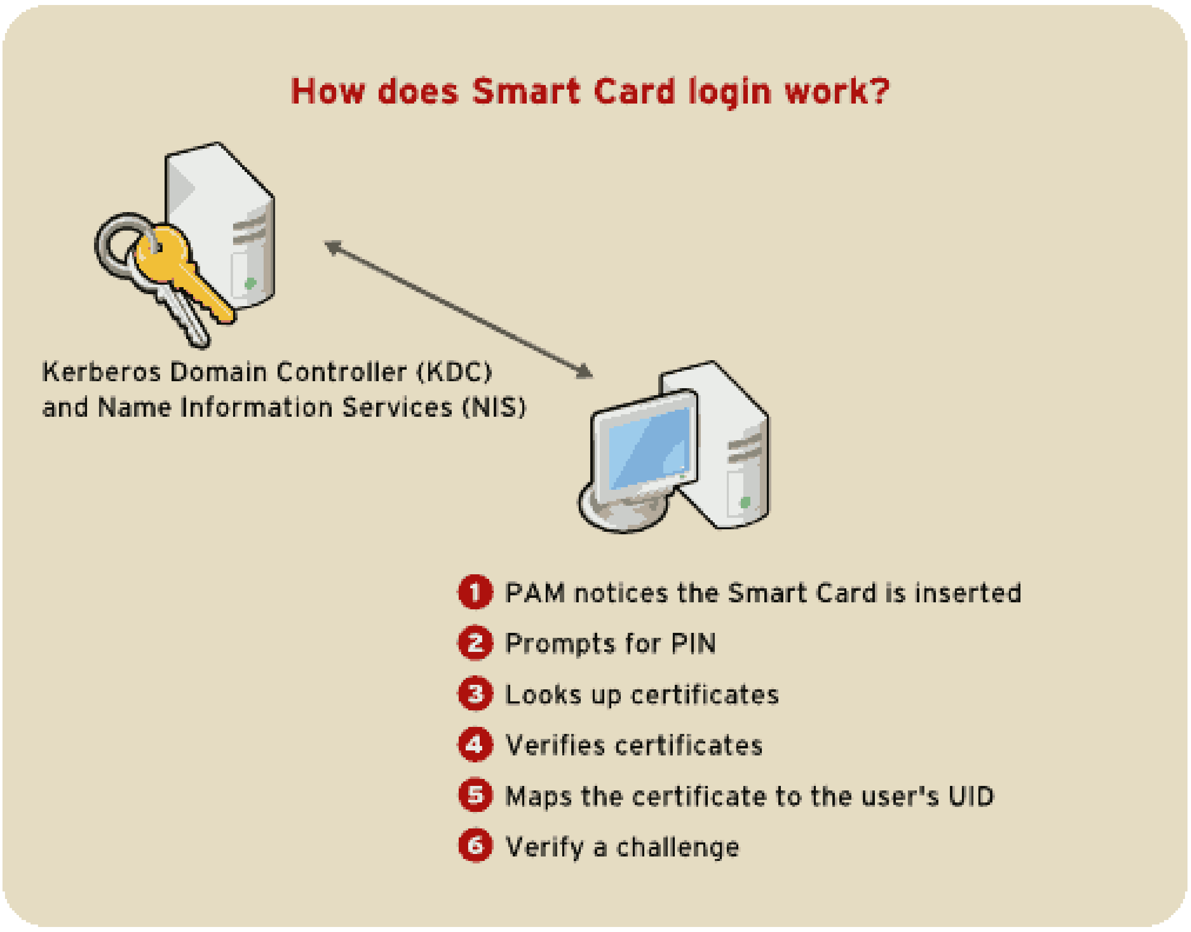

How does Smart Card login work?
Kerberos Domain Controller (KDC)
and Name Information Services (NIS)
1 PAM notices the Smart Card is inserted
2 Prompts for PIN
3 Looks up certificates
4 Verifies certificates
5 Maps the certificate to the user's UID
6 Verify a challenge

Figure 3.5. How Smart Card Login Works

Note

You cannot log in with a card that has not been enrolled, even if it has been formatted. You need to log in with a formatted, enrolled card, or not using a smart card, before you can enroll a new card.

Refer to *Section 3.7, “Kerberos”* and *Section 3.5, “Pluggable Authentication Modules (PAM)”* for more information on Kerberos and PAM.

3.3.5. Configuring Firefox to use Kerberos for SSO

You can configure Firefox to use Kerberos for Single Sign-on. In order for this functionality to work correctly, you need to configure your web browser to send your Kerberos credentials to the appropriate KDC.The following section describes the configuration changes and other requirements to achieve this.

1. In the address bar of Firefox, type **`about:config`** to display the list of current configuration options.

2. In the **Filter** field, type **`negotiate`** to restrict the list of options.

3. Double-click the *network.negotiate-auth.trusted-uris* entry to display the *Enter string value* dialog box.

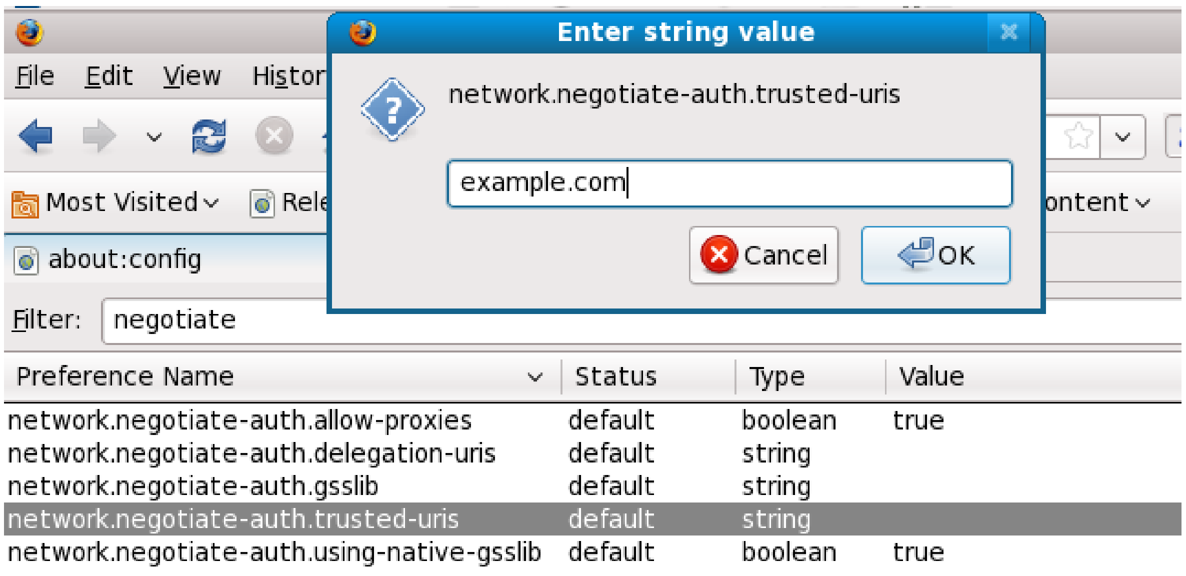
4. Enter the name of the domain against which you want to authenticate, for example, *`.example.com`*.

5. Repeat the above procedure for the *network.negotiate-auth.delegation-uris* entry, using the same domain.

Note

You can leave this value blank, as it allows Kerberos ticket passing, which is not required.

If you do not see these two configuration options listed, your version of Firefox may be too old to support Negotiate authentication, and you should consider upgrading.



network.negotiate-auth.trusted-uris

example.com

Filter: negotiate

Preference Name	Status	Type	Value
network.negotiate-auth.allow-proxies	default	boolean	true
network.negotiate-auth.delegation-uris	default	string	
network.negotiate-auth.gsslib	default	string	
network.negotiate-auth.trusted-uris	default	string	
network.negotiate-auth.using-native-gsslib	default	boolean	true

Figure 3.6. Configuring Firefox for SSO with Kerberos

You now need to ensure that you have Kerberos tickets. In a command shell, type **`kinit`** to retrieve Kerberos tickets. To display the list of available tickets, type **`klist`**. The following shows an example output from these commands:

```
[user@host ~] $ kinit
Password for user@EXAMPLE.COM:

[user@host ~] $ klist
Ticket cache: FILE:/tmp/krb5cc_10920
Default principal: user@EXAMPLE.COM

Valid starting     Expires            Service principal
10/26/06 23:47:54  10/27/06 09:47:54  krbtgt/USER.COM@USER.COM
        renew until 10/26/06 23:47:54

Kerberos 4 ticket cache: /tmp/tkt10920
klist: You have no tickets cached
```

3.3.5.1. Troubleshooting

If you have followed the configuration steps above and Negotiate authentication is not working, you can turn on verbose logging of the authentication process. This could help you find the cause of the problem. To enable verbose logging, use the following procedure:

1. Close all instances of Firefox.

2. Open a command shell, and enter the following commands:

   ```
   export NSPR_LOG_MODULES=negotiateauth:5
   export NSPR_LOG_FILE=/tmp/moz.log
   ```

3. Restart Firefox *from that shell*, and visit the website you were unable to authenticate to earlier. Information will be logged to **`/tmp/moz.log`**, and may give a clue to the problem. For example:

   ```
   -1208550944[90039d0]: entering nsNegotiateAuth::GetNextToken()
   -1208550944[90039d0]: gss_init_sec_context() failed: Miscellaneous failure
   No credentials cache found
   ```

   This indicates that you do not have Kerberos tickets, and need to run **`kinit`**.

If you are able to run **`kinit`** successfully from your machine but you are unable to authenticate, you might see something like this in the log file:

```
-1208994096[8d683d8]: entering nsAuthGSSAPI::GetNextToken()
-1208994096[8d683d8]: gss_init_sec_context() failed: Miscellaneous failure
Server not found in Kerberos database
```

This generally indicates a Kerberos configuration problem. Make sure that you have the correct entries in the [domain_realm] section of the **`/etc/krb5.conf`** file. For example:

```
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

If nothing appears in the log it is possible that you are behind a proxy, and that proxy is stripping off the HTTP headers required for Negotiate authentication. As a workaround, you can try to connect to the server using HTTPS instead, which allows the request to pass through unmodified. Then proceed to debug using the log file, as described above.

3.4. Multifactor Authentication Solutions

3.4.1. Yubikey

Yubikey is a hardware authentication token that utilizes open source software to operate. This token is a simple USB device that appears as a keyboard to your computer. The single touch button on the token provides a one time password (OTP) with each push that can be used to authenticate a user. Currently there are several different implementations of this solution of which we'll cover here.

3.4.1.1. Using Yubikey with a centralized server

A PAM module already exists in the Fedora repositories that allow authentication of computers that can contact an authentication server. The server can either be setup at the domain level or the Yubico's servers can be utilized. This method of authentication is a great enterprise solution where multiple users may need access to multiple computers on the domain. The steps below describe this setup.

1. Install *pam_yubico*

2. For two factor authentication open **`/etc/pam.d/gdm-password`** and locate the following line:

   ```
   auth substack password-auth
   ```

   In a new line after this add:

   ```
   auth sufficient pam_yubico.so id=16
   ```

3. To simple use the yubikey token without your password remove the first line from the step above and replace it with the second.

4. Locate the yubikey token for the first yubikey you will be adding. This can be done by looking at the first 12 characters of any OTP or visit *http://radius.yubico.com/demo/Modhex_Calculator.php* and copy the Modhex encoded string after you enter an OTP into the textbox on the page.

5. Add user's yubikeys to the config file. This can be done either globally in **`/etc/yubikey_mapping`** or by individual user in **`~/.yubico/authorized_yubikeys`**. The following is the syntax:

   ```
   username:yubikey_token:another_yubikey_token
   ```

6. Logout, when you attempt to log back in you should either be prompted to enter both your password and your yubikey OTP or both depending on how you configured your system.

Note

A connection to the authentication server is required or proper authentication will not occur. This can be detrimental to systems that do not have constant network connectivity.

3.4.1.2. Authenticating to websites with your Yubikey

While outside the scope of this guide Yubikey allows you to authenticate to websites supporting this authentication method. These websites typically support Yubico's authentication servers but some can be setup similar to the above centralized authentication. Yubico also provides OpenID services that can be utilized with certain websites.

3.5. Pluggable Authentication Modules (PAM)

Programs that grant users access to a system use *authentication* to verify each other's identity (that is, to establish that a user is who they say they are).

Historically, each program had its own way of authenticating users. In Fedora, many programs are configured to use a centralized authentication mechanism called *Pluggable Authentication Modules* (PAM).

PAM uses a pluggable, modular architecture, which affords the system administrator a great deal of flexibility in setting authentication policies for the system.

In most situations, the default PAM configuration file for a PAM-aware application is sufficient. Sometimes, however, it is necessary to edit a PAM configuration file. Because misconfiguration of PAM can compromise system security, it is important to understand the structure of these files before making any modifications. Refer to *Section 3.5.3, “PAM Configuration File Format”* for more information.

3.5.1. Advantages of PAM

PAM offers the following advantages:

- a common authentication scheme that can be used with a wide variety of applications.
- significant flexibility and control over authentication for both system administrators and application developers.
- a single, fully-documented library which allows developers to write programs without having to create their own authentication schemes.

3.5.2. PAM Configuration Files

The **`/etc/pam.d/`** directory contains the PAM configuration files for each PAM-aware application. In earlier versions of PAM, the **`/etc/pam.conf`** file was used, but this file is now deprecated and is only used if the **`/etc/pam.d/`** directory does not exist.

3.5.2.1. PAM Service Files

Each PAM-aware application or *service* has a file in the **`/etc/pam.d/`** directory. Each file in this directory has the same name as the service to which it controls access.

The PAM-aware program is responsible for defining its service name and installing its own PAM configuration file in the **`/etc/pam.d/`** directory. For example, the **`login`** program defines its service name as **`login`** and installs the **`/etc/pam.d/login`** PAM configuration file.

3.5.3. PAM Configuration File Format

Each PAM configuration file contains a group of directives formatted as follows:

```
<module interface>  <control flag>   <module name>   <module arguments>
```

Each of these elements is explained in the following sections.

3.5.3.1. Module Interface

Four types of PAM module interface are currently available. Each of these corresponds to a different aspect of the authorization process:

- **`auth`** — This module interface authenticates use. For example, it requests and verifies the validity of a password. Modules with this interface can also set credentials, such as group memberships or Kerberos tickets.
- **`account`** — This module interface verifies that access is allowed. For example, it may check if a user account has expired or if a user is allowed to log in at a particular time of day.
- **`password`** — This module interface is used for changing user passwords.
- **`session`** — This module interface configures and manages user sessions. Modules with this interface can also perform additional tasks that are needed to allow access, like mounting a user's home directory and making the user's mailbox available.

Note

An individual module can provide any or all module interfaces. For instance, **`pam_unix.so`** provides all four module interfaces.

In a PAM configuration file, the module interface is the first field defined. For example, a typical line in a configuration may look like this:

```
auth required pam_unix.so
```

This instructs PAM to use the **`pam_unix.so`** module's **`auth`** interface.

3.5.3.1.1. Stacking Module Interfaces

Module interface directives can be *stacked*, or placed upon one another, so that multiple modules are used together for one purpose. If a module's control flag uses the "sufficient" or "requisite" value (refer to *Section 3.5.3.2, “Control Flag”* for more information on these flags), then the order in which the modules are listed is important to the authentication process.

Stacking makes it easy for an administrator to require specific conditions to exist before allowing the user to authenticate. For example, the **`reboot`** command normally uses several stacked modules, as seen in its PAM configuration file:

```
[root@MyServer ~]# cat /etc/pam.d/reboot
#%PAM-1.0
auth sufficient pam_rootok.so
auth required pam_console.so
#auth include  system-auth
account required pam_permit.so
```

- The first line is a comment and is not processed.
- **`auth sufficient pam_rootok.so`** — This line uses the **`pam_rootok.so`** module to check whether the current user is root, by verifying that their UID is 0. If this test succeeds, no other modules are consulted and the command is executed. If this test fails, the next module is consulted.
- **`auth required pam_console.so`** — This line uses the **`pam_console.so`** module to attempt to authenticate the user. If this user is already logged in at the console, **`pam_console.so`** checks whether there is a file in the **`/etc/security/console.apps/`** directory with the same name as the service name (reboot). If such a file exists, authentication succeeds and control is passed to the next module.
- **`#auth include system-auth`** — This line is commented and is not processed.
- **`account required pam_permit.so`** — This line uses the **`pam_permit.so`** module to allow the root user or anyone logged in at the console to reboot the system.

3.5.3.2. Control Flag

All PAM modules generate a success or failure result when called. Control flags tell PAM what do with the result. Modules can be stacked in a particular order, and the control flags determine how important the success or failure of a particular module is to the overall goal of authenticating the user to the service.

There are four predefined control flags:

- **`required`** — The module result must be successful for authentication to continue. If the test fails at this point, the user is not notified until the results of all module tests that reference that interface are complete.
- **`requisite`** — The module result must be successful for authentication to continue. However, if a test fails at this point, the user is notified immediately with a message reflecting the first failed **`required`** *or* **`requisite`** module test.
- **`sufficient`** — The module result is ignored if it fails. However, if the result of a module flagged **`sufficient`** is successful *and* no previous modules flagged **`required`** have failed, then no other results are required and the user is authenticated to the service.
- **`optional`** — The module result is ignored. A module flagged as **`optional`** only becomes necessary for successful authentication when no other modules reference the interface.

Important

The order in which **`required`** modules are called is not critical. Only the **`sufficient`** and **`requisite`** control flags cause order to become important.

A newer control flag syntax that allows for more precise control is now available for PAM.

The **`pam.d`** man page, and the PAM documentation, located in the **`/usr/share/doc/pam-<version-number>/`** directory, where *`<version-number>`* is the version number for PAM on your system, describe this newer syntax in detail.

3.5.3.3. Module Name

The module name provides PAM with the name of the pluggable module containing the specified module interface. In older versions of Fedora, the full path to the module was provided in the PAM configuration file. However, since the advent of *multilib* systems, which store 64-bit PAM modules in the **`/lib64/security/`** directory, the directory name is omitted because the application is linked to the appropriate version of **`libpam`**, which can locate the correct version of the module.

3.5.3.4. Module Arguments

PAM uses *arguments* to pass information to a pluggable module during authentication for some modules.

For example, the **`pam_userdb.so`** module uses information stored in a Berkeley DB file to authenticate the user. Berkeley DB is an open source database system embedded in many applications. The module takes a **`db`** argument so that Berkeley DB knows which database to use for the requested service.

The following is a typical **`pam_userdb.so`** line in a PAM configuration. The *`<path-to-file>`* is the full path to the Berkeley DB database file:

```
auth required pam_userdb.so db=<path-to-file>
```

Invalid arguments are *generally* ignored and do not otherwise affect the success or failure of the PAM module. Some modules, however, may fail on invalid arguments. Most modules report errors to the **`/var/log/secure`** file.

3.5.4. Sample PAM Configuration Files

The following is a sample PAM application configuration file:

```
#%PAM-1.0
auth  required  pam_securetty.so
auth  required  pam_unix.so nullok
auth  required  pam_nologin.so
account  required  pam_unix.so
password required  pam_cracklib.so retry=3
password required  pam_unix.so shadow nullok use_authtok
session required  pam_unix.so
```

- The first line is a comment, indicated by the hash mark (#) at the beginning of the line.
- Lines two through four stack three modules for login authentication.

  `auth required pam_securetty.so` — This module ensures that *if* the user is trying to log in as root, the tty on which the user is logging in is listed in the **`/etc/securetty`** file, *if* that file exists.

  If the tty is not listed in the file, any attempt to log in as root fails with a **`Login incorrect`** message.

  `auth required pam_unix.so nullok` — This module prompts the user for a password and then checks the password using the information stored in **`/etc/passwd`** and, if it exists, **`/etc/shadow`**.

  - The argument **`nullok`** instructs the **`pam_unix.so`** module to allow a blank password.

- **`auth required pam_nologin.so`** — This is the final authentication step. It checks whether the **`/etc/nologin`** file exists. If it exists and the user is not root, authentication fails.

  Note

  In this example, all three **`auth`** modules are checked, even if the first **`auth`** module fails. This prevents the user from knowing at what stage their authentication failed. Such knowledge in the hands of an attacker could allow them to more easily deduce how to crack the system.

- **`account required pam_unix.so`** — This module performs any necessary account verification. For example, if shadow passwords have been enabled, the account interface of the **`pam_unix.so`** module checks to see if the account has expired or if the user has not changed the password within the allowed grace period.
- **`password required pam_cracklib.so retry=3`** — If a password has expired, the password component of the **`pam_cracklib.so`** module prompts for a new password. It then tests the newly created password to see whether it can easily be determined by a dictionary-based password cracking program.

  - The argument **`retry=3`** specifies that if the test fails the first time, the user has two more chances to create a strong password.

- **`password required pam_unix.so shadow nullok use_authtok`** — This line specifies that if the program changes the user's password, it should use the **`password`** interface of the **`pam_unix.so`** module to do so.

  - The argument **`shadow`** instructs the module to create shadow passwords when updating a user's password.

  - The argument **`nullok`** instructs the module to allow the user to change their password *from* a blank password, otherwise a null password is treated as an account lock.

  - The final argument on this line, **`use_authtok`**, provides a good example of the importance of order when stacking PAM modules. This argument instructs the module not to prompt the user for a new password. Instead, it accepts any password that was recorded by a previous password module. In this way, all new passwords must pass the **`pam_cracklib.so`** test for secure passwords before being accepted.

- **`session required pam_unix.so`** — The final line instructs the session interface of the **`pam_unix.so`** module to manage the session. This module logs the user name and the service type to **`/var/log/secure`** at the beginning and end of each session. This module can be supplemented by stacking it with other session modules for additional functionality.

3.5.5. Creating PAM Modules

You can create or add new PAM modules at any time for use by PAM-aware applications.

For example, a developer might create a one-time-password creation method and write a PAM module to support it. PAM-aware programs can immediately use the new module and password method without being recompiled or otherwise modified.

This allows developers and system administrators to mix-and-match, as well as test, authentication methods for different programs without recompiling them.

Documentation on writing modules is included in the **`/usr/share/doc/pam-<`*`version-number`*`>/`** directory, where `<`*`version-number`*`>` is the version number for PAM on your system.

3.5.6. PAM and Administrative Credential Caching

A number of graphical administrative tools in Fedora provide users with elevated privileges for up to five minutes using the **`pam_timestamp.so`** module. It is important to understand how this mechanism works, because a user who walks away from a terminal while **`pam_timestamp.so`** is in effect leaves the machine open to manipulation by anyone with physical access to the console.

In the PAM timestamp scheme, the graphical administrative application prompts the user for the root password when it is launched. When the user has been authenticated, the **`pam_timestamp.so`** module creates a timestamp file. By default, this is created in the **`/var/run/sudo/`** directory. If the timestamp file already exists, graphical administrative programs do not prompt for a password. Instead, the **`pam_timestamp.so`** module freshens the timestamp file, reserving an extra five minutes of unchallenged administrative access for the user.

You can verify the actual state of the timestamp file by inspecting the **`/var/run/sudo/<user>`** file. For the desktop, the relevant file is **`unknown:root`**. If it is present and its timestamp is less than five minutes old, the credentials are valid.

The existence of the timestamp file is indicated by an authentication icon, which appears in the notification area of the panel.

Figure 3.7. The Authentication Icon

3.5.6.1. Removing the Timestamp File

Before abandoning a console where a PAM timestamp is active, it is recommended that the timestamp file be destroyed. To do this from a graphical environment, click the authentication icon on the panel. This causes a dialog box to appear. Click the **Forget Authorization** button to destroy the active timestamp file.

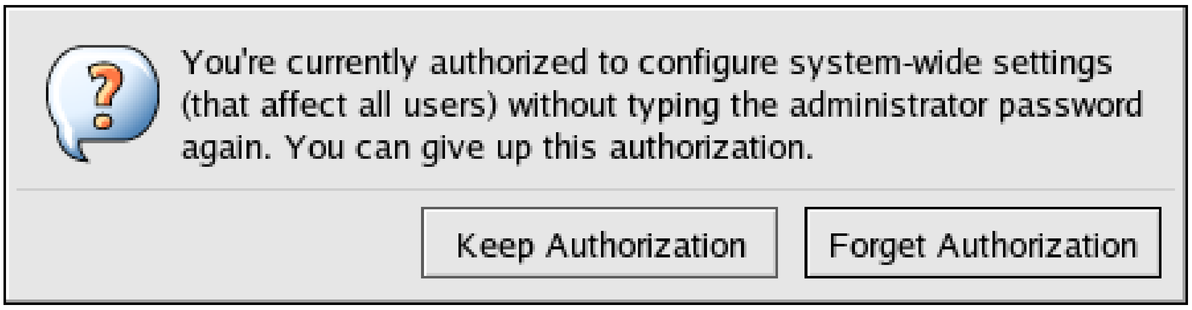

You're currently authorized to configure system-wide settings (that affect all users) without typing the administrator password again. You can give up this authorization.

Figure 3.8. Dismiss Authentication Dialog

You should be aware of the following with respect to the PAM timestamp file:

- If logged in to the system remotely using **`ssh`**, use the **`/sbin/pam_timestamp_check -k root`** command to destroy the timestamp file.
- You need to run the **`/sbin/pam_timestamp_check -k root`** command from the same terminal window from which you launched the privileged application.
- You must be logged in as the user who originally invoked the **`pam_timestamp.so`** module in order to use the **`/sbin/pam_timestamp_check -k`** command. Do not log in as root to use this command.
- If you want to kill the credentials on the desktop (without using the **Forget Authorization** action on the icon), use the following command:

  ```
  /sbin/pam_timestamp_check -k root </dev/null >/dev/null 2>/dev/null
  ```

  Failure to use this command will only remove the credentials (if any) from the pty where you run the command.

Refer to the **`pam_timestamp_check`** man page for more information about destroying the timestamp file using **`pam_timestamp_check`**.

3.5.6.2. Common pam_timestamp Directives

The **`pam_timestamp.so`** module accepts several directives. The following are the two most commonly used options:

- **`timestamp_timeout`** — Specifies the period (in seconds) for which the timestamp file is valid. The default value is 300 (five minutes).

- **`timestampdir`** — Specifies the directory in which the timestamp file is stored. The default value is **`/var/run/sudo/`**.

3.5.7. PAM and Device Ownership

In Fedora, the first user who logs in at the physical console of the machine can manipulate certain devices and perform certain tasks normally reserved for the root user. This is controlled by a PAM module called **`pam_console.so`**.

3.5.7.1. Device Ownership

When a user logs in to a Fedora system, the **`pam_console.so`** module is called by **`login`** or the graphical login programs, **gdm**, **kdm**, and **xdm**. If this user is the first user to log in at the physical console — referred to as the *console user* — the module grants the user ownership of a variety of devices normally owned by root. The console user owns these devices until the last local session for that user ends. After this user has logged out, ownership of the devices reverts back to the root user.

The devices affected include, but are not limited to, sound cards, diskette drives, and CD-ROM drives.

This facility allows a local user to manipulate these devices without obtaining root access, thus simplifying common tasks for the console user.

You can modify the list of devices controlled by **`pam_console.so`** by editing the following files:

- **`/etc/security/console.perms`**
- **`/etc/security/console.perms.d/50-default.perms`**

You can change the permissions of different devices than those listed in the above files, or override the specified defaults. Rather than modify the **`50-default.perms`** file, you should create a new file (for example, ***`xx`*`-name.perms`**) and enter the required modifications. The name of the new default file must begin with a number higher than 50 (for example, **`51-default.perms`**). This will override the defaults in the **`50-default.perms`** file.

Warning

If the **gdm**, **kdm**, or **xdm** display manager configuration file has been altered to allow remote users to log in *and* the host is configured to run at runlevel 5, it is advisable to change the **`<console>`** and **`<xconsole>`** directives in the **`/etc/security/console.perms`** to the following values:

```
<console>=tty[0-9][0-9]* vc/[0-9][0-9]* :0\.[0-9] :0
<xconsole>=:0\.[0-9] :0
```

This prevents remote users from gaining access to devices and restricted applications on the machine.

If the **gdm**, **kdm**, or **xdm** display manager configuration file has been altered to allow remote users to log in *and* the host is configured to run at any multiple user runlevel other than 5, it is advisable to remove the **`<xconsole>`** directive entirely and change the **`<console>`** directive to the following value:

```
<console>=tty[0-9][0-9]* vc/[0-9][0-9]*
```

3.5.7.2. Application Access

The console user also has access to certain programs configured for use in the **`/etc/security/console.apps/`** directory.

This directory contains configuration files which enable the console user to run certain applications in **`/sbin`** and **`/usr/sbin`**.

These configuration files have the same name as the applications that they set up.

One notable group of applications that the console user has access to are three programs that shut down or reboot the system:

- **`/sbin/halt`**
- **`/sbin/reboot`**
- **`/sbin/poweroff`**

Because these are PAM-aware applications, they call the **`pam_console.so`** module as a requirement for use.

3.5.8. Additional Resources

The following resources further explain methods to use and configure PAM. In addition to these resources, read the PAM configuration files on the system to better understand how they are structured.

3.5.8.1. Installed PAM Documentation

- PAM-related man pages — Several man pages exist for the various applications and configuration files involved with PAM. The following is a list of some of the more important man pages.

  Configuration Files

  - **`pam`** — Good introductory information on PAM, including the structure and purpose of the PAM configuration files.

    Note that this man page discusses both **`/etc/pam.conf`** and individual configuration files in the **`/etc/pam.d/`** directory. By default, Fedora uses the individual configuration files in the **`/etc/pam.d/`** directory, ignoring **`/etc/pam.conf`** even if it exists.

  - **`pam_console`** — Describes the purpose of the **`pam_console.so`** module. It also describes the appropriate syntax for an entry within a PAM configuration file.

  - **`console.apps`** — Describes the format and options available in the **`/etc/security/console.apps`** configuration file, which defines which applications are accessible by the console user assigned by PAM.

  - **`console.perms`** — Describes the format and options available in the **`/etc/security/console.perms`** configuration file, which specifies the console user permissions assigned by PAM.

  - **`pam_timestamp`** — Describes the **`pam_timestamp.so`** module.

- **`/usr/share/doc/pam-<version-number>`** — Contains a *System Administrators' Guide*, a *Module Writers' Manual*, and the *Application Developers' Manual*, as well as a copy of the PAM standard, DCE-RFC 86.0, where *`<version-number>`* is the version number of PAM.

- ***/usr/share/doc/pam-<version-number>/txts/README.pam_timestamp*** — Contains information about the **pam_timestamp.so** PAM module, where <*version-number*> is the version number of PAM.

3.5.8.2. Useful PAM Websites

- *http://www.kernel.org/pub/linux/libs/pam/* — The primary distribution website for the Linux-PAM project, containing information on various PAM modules, a FAQ, and additional PAM documentation.

Note

The documentation in the above website is for the last released upstream version of PAM and might not be completely accurate for the PAM version included in Fedora.

3.6. TCP Wrappers and xinetd

Controlling access to network services is one of the most important security tasks facing a server administrator. Fedora 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 3.9, “Access Control to Network Services” is a basic illustration of how these tools work together to protect network services.

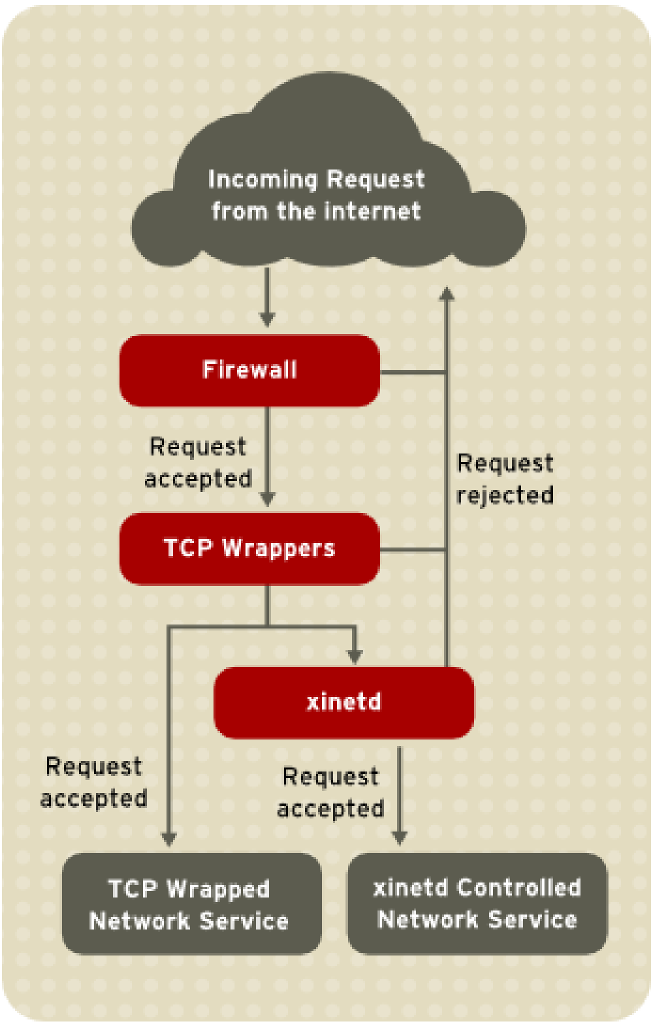

Incoming Request from the internet
Firewall
Request accepted
Request rejected
TCP Wrappers
xinetd
Request accepted
Request accepted
TCP Wrapped Network Service
xinetd Controlled Network Service

Figure 3.9. Access Control to Network Services

This chapter focuses on the role of TCP Wrappers and `xinetd` in controlling access to network services and reviews how these tools can be used to enhance both logging and utilization management. Refer to *Section 3.8, “Using Firewalls”* for information about using firewalls with **`iptables`**.

3.6.1. TCP Wrappers

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

The following example indicates that `/usr/sbin/sshd` is linked to **`libwrap.a`**:

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

3.6.1.1. 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.

3.6.2. 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
```

3.6.2.1. 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 3.6.2.1.4, “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 3.6.2.1.4, “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 the specialist terms above can be found elsewhere in this Guide:

- *Section 3.6.2.1.1, "Wildcards"*
- *Section 3.6.2.1.2, "Patterns"*
- *Section 3.6.2.2.4, "Expansions"*
- *Section 3.6.2.2, "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 3.6.2.2, "Option Fields"* for a more detailed look at available options.

3.6.2.1.1. 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`** — Matches any host where the hostname does not match the host address.

Important

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

3.6.2.1.2. 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, lesser 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.

3.6.2.1.3. 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.

3.6.2.1.4. 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.

3.6.2.2. Option Fields

In addition to basic rules that allow and deny access, the Fedora 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.

3.6.2.2.1. 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.

3.6.2.2.2. 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.

3.6.2.2.3. 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.

3.6.2.2.4. 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 3.6.5, "Additional Resources"* for more information about TCP Wrappers.

3.6.3. 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 can have more resources allotted 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.

3.6.4. 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.

3.6.4.1. 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 3.6.4.2, "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 3.6.4.3.1, "Logging Options"* for further information.

3.6.4.2. 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 correlate 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.

3.6.4.3. Altering xinetd Configuration Files

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

3.6.4.3.1. 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.

3.6.4.3.2. 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 3.6.2, "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.a`** 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.

3.6.4.3.3. 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.

3.6.4.3.4. 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.

3.6.5. Additional Resources

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

3.6.5.1. 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.

3.6.5.2. Useful TCP Wrappers Websites

- *http://www.xinetd.org/*[3] — The home of `xinetd`, containing sample configuration files, a full listing of features, and an informative FAQ.

- *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.

[3] http://www.xinetd.org

3.6.5.3. 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`.

3.7. Kerberos

System security and integrity within a network can be unwieldy. It can occupy the time of several administrators just to keep track of what services are being run on a network and the manner in which these services are used.

Further, authenticating users to network services can prove dangerous when the method used by the protocol is inherently insecure, as evidenced by the transfer of unencrypted passwords over a network using the traditional FTP and Telnet protocols.

Kerberos is a way to eliminate the need for protocols that allow unsafe methods of authentication, thereby enhancing overall network security.

3.7.1. What is Kerberos?

Kerberos is a network authentication protocol created by MIT, and uses symmetric-key cryptography[4] to authenticate users to network services, which means passwords are never actually sent over the network.

Consequently, when users authenticate to network services using Kerberos, unauthorized users attempting to gather passwords by monitoring network traffic are effectively thwarted.

3.7.1.1. Advantages of Kerberos

Most conventional network services use password-based authentication schemes. Such schemes require a user to authenticate to a given network server by supplying their username and password. Unfortunately, the transmission of authentication information for many services is unencrypted. For such a scheme to be secure, the network has to be inaccessible to outsiders, and all computers and users on the network must be trusted and trustworthy.

Even if this is the case, a network that is connected to the Internet can no longer be assumed to be secure. Any attacker who gains access to the network can use a simple packet analyzer, also known as a packet sniffer, to intercept usernames and passwords, compromising user accounts and the integrity of the entire security infrastructure.

The primary design goal of Kerberos is to eliminate the transmission of unencrypted passwords across the network. If used properly, Kerberos effectively eliminates the threat that packet sniffers would otherwise pose on a network.

3.7.1.2. Disadvantages of Kerberos

Although Kerberos removes a common and severe security threat, it may be difficult to implement for a variety of reasons:

- Migrating user passwords from a standard UNIX password database, such as **`/etc/passwd`** or **`/etc/shadow`**, to a Kerberos password database can be tedious, as there is no automated mechanism to perform this task. Refer to Question 2.23 in the online Kerberos FAQ:

[4] A system where both the client and the server share a common key that is used to encrypt and decrypt network communication.

http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html[5]

- Kerberos has only partial compatibility with the Pluggable Authentication Modules (PAM) system used by most Fedora servers. Refer to *Section 3.7.4, “Kerberos and PAM”* for more information about this issue.

- Kerberos assumes that each user is trusted but is using an untrusted host on an untrusted network. Its primary goal is to prevent unencrypted passwords from being transmitted across that network. However, if anyone other than the proper user has access to the one host that issues tickets used for authentication — called the *key distribution center* (*KDC*) — the entire Kerberos authentication system is at risk.

- For an application to use Kerberos, its source must be modified to make the appropriate calls into the Kerberos libraries. Applications modified in this way are considered to be *Kerberos-aware*, or *kerberized*. For some applications, this can be quite problematic due to the size of the application or its design. For other incompatible applications, changes must be made to the way in which the server and client communicate. Again, this may require extensive programming. Closed-source applications that do not have Kerberos support by default are often the most problematic.

- Kerberos is an all-or-nothing solution. If Kerberos is used on the network, any unencrypted passwords transferred to a non-Kerberos aware service is at risk. Thus, the network gains no benefit from the use of Kerberos. To secure a network with Kerberos, one must either use Kerberos-aware versions of *all* client/server applications that transmit passwords unencrypted, or not use *any* such client/server applications at all.

3.7.2. Kerberos Terminology

Kerberos has its own terminology to define various aspects of the service. Before learning how Kerberos works, it is important to learn the following terms.

authentication server (AS)
: A server that issues tickets for a desired service which are in turn given to users for access to the service. The AS responds to requests from clients who do not have or do not send credentials with a request. It is usually used to gain access to the ticket-granting server (TGS) service by issuing a ticket-granting ticket (TGT). The AS usually runs on the same host as the key distribution center (KDC).

ciphertext
: Encrypted data.

client
: An entity on the network (a user, a host, or an application) that can receive a ticket from Kerberos.

credentials
: A temporary set of electronic credentials that verify the identity of a client for a particular service. Also called a ticket.

credential cache or ticket file
: A file which contains the keys for encrypting communications between a user and various network services. Kerberos 5 supports a framework for using other cache types, such as shared memory, but files are more thoroughly supported.

[5] http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html#pwconvert

crypt hash
: A one-way hash used to authenticate users. These are more secure than using unencrypted data, but they are still relatively easy to decrypt for an experienced cracker.

GSS-API
: The Generic Security Service Application Program Interface (defined in RFC-2743 published by The Internet Engineering Task Force) is a set of functions which provide security services. This API is used by clients and services to authenticate to each other without either program having specific knowledge of the underlying mechanism. If a network service (such as cyrus-IMAP) uses GSS-API, it can authenticate using Kerberos.

hash
: Also known as a *hash value*. A value generated by passing a string through a *hash function*. These values are typically used to ensure that transmitted data has not been tampered with.

hash function
: A way of generating a digital "fingerprint" from input data. These functions rearrange, transpose or otherwise alter data to produce a *hash value*.

key
: Data used when encrypting or decrypting other data. Encrypted data cannot be decrypted without the proper key or extremely good fortune on the part of the cracker.

key distribution center (KDC)
: A service that issues Kerberos tickets, and which usually run on the same host as the ticket-granting server (TGS).

keytab (or key table)
: A file that includes an unencrypted list of principals and their keys. Servers retrieve the keys they need from keytab files instead of using **`kinit`**. The default keytab file is **`/etc/krb5.keytab`**. The KDC administration server, **`/usr/kerberos/sbin/kadmind`**, is the only service that uses any other file (it uses **`/var/kerberos/krb5kdc/kadm5.keytab`**).

kinit
: The **`kinit`** command allows a principal who has already logged in to obtain and cache the initial ticket-granting ticket (TGT). Refer to the **`kinit`** man page for more information.

principal (or principal name)
: The principal is the unique name of a user or service allowed to authenticate using Kerberos. A principal follows the form **`root[/instance]@REALM`**. For a typical user, the root is the same as their login ID. The **`instance`** is optional. If the principal has an instance, it is separated from the root with a forward slash ("/"). An empty string ("") is considered a valid instance (which differs from the default **`NULL`** instance), but using it can be confusing. All principals in a realm have their own key, which for users is derived from a password or is randomly set for services.

realm
: A network that uses Kerberos, composed of one or more servers called KDCs and a potentially large number of clients.

service
: A program accessed over the network.

ticket
: A temporary set of electronic credentials that verify the identity of a client for a particular service. Also called credentials.

ticket-granting server (TGS)
: A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.

ticket-granting ticket (TGT)
: A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

unencrypted password
: A plain text, human-readable password.

3.7.3. How Kerberos Works

Kerberos differs from username/password authentication methods. Instead of authenticating each user to each network service, Kerberos uses symmetric encryption and a trusted third party (a KDC), to authenticate users to a suite of network services. When a user authenticates to the KDC, the KDC sends a ticket specific to that session back to the user's machine, and any Kerberos-aware services look for the ticket on the user's machine rather than requiring the user to authenticate using a password.

When a user on a Kerberos-aware network logs in to their workstation, their principal is sent to the KDC as part of a request for a TGT from the Authentication Server. This request can be sent by the log-in program so that it is transparent to the user, or can be sent by the **`kinit`** program after the user logs in.

The KDC then checks for the principal in its database. If the principal is found, the KDC creates a TGT, which is encrypted using the user's key and returned to that user.

The login or **`kinit`** program on the client then decrypts the TGT using the user's key, which it computes from the user's password. The user's key is used only on the client machine and is *not* transmitted over the network.

The TGT is set to expire after a certain period of time (usually ten to twenty-four hours) and is stored in the client machine's credentials cache. An expiration time is set so that a compromised TGT is of use to an attacker for only a short period of time. After the TGT has been issued, the user does not have to re-enter their password until the TGT expires or until they log out and log in again.

Whenever the user needs access to a network service, the client software uses the TGT to request a new ticket for that specific service from the TGS. The service ticket is then used to authenticate the user to that service transparently.

Warning

The Kerberos system can be compromised if a user on the network authenticates against a non-Kerberos aware service by transmitting a password in plain text. The use of non-Kerberos aware services is highly discouraged. Such services include Telnet and FTP. The use of other encrypted protocols, such as SSH or SSL-secured services, however, is preferred, although not ideal.

This is only a broad overview of how Kerberos authentication works. Refer to *Section 3.7.10, “Additional Resources”* for links to more in-depth information.

Note

Kerberos depends on the following network services to function correctly.

- Approximate clock synchronization between the machines on the network.

  A clock synchronization program should be set up for the network, such as **`ntpd`**. Refer to **`/usr/share/doc/ntp-<`*`version-number`*`>/index.html`** for details on setting up Network Time Protocol servers (where <*`version-number`*> is the version number of the **`ntp`** package installed on your system).

- Domain Name Service (DNS).

  You should ensure that the DNS entries and hosts on the network are all properly configured. Refer to the *Kerberos V5 System Administrator's Guide* in **`/usr/share/doc/krb5-server-<`*`version-number`*`>`** for more information (where <*`version-number`*> is the version number of the **`krb5-server`** package installed on your system).

3.7.4. Kerberos and PAM

Kerberos-aware services do not currently make use of Pluggable Authentication Modules (PAM) — these services bypass PAM completely. However, applications that use PAM can make use of Kerberos for authentication if the **`pam_krb5`** module (provided in the **`pam_krb5`** package) is installed. The **`pam_krb5`** package contains sample configuration files that allow services such as **`login`** and **`gdm`** to authenticate users as well as obtain initial credentials using their passwords. If access to network servers is always performed using Kerberos-aware services or services that use GSS-API, such as IMAP, the network can be considered reasonably safe.

Important

Administrators should be careful not to allow users to authenticate to most network services using Kerberos passwords. Many protocols used by these services do not encrypt the password before sending it over the network, destroying the benefits of the Kerberos system. For example, users should not be allowed to authenticate to Telnet services with the same password they use for Kerberos authentication.

3.7.5. Configuring a Kerberos 5 Server

When setting up Kerberos, install the KDC first. If it is necessary to set up slave servers, install the master first.

To configure the first Kerberos KDC, follow these steps:

1. Ensure that time synchronization and DNS are functioning correctly on all client and server machines before configuring Kerberos. Pay particular attention to time synchronization between the Kerberos server and its clients. If the time difference between the server and client is greater than five minutes (this is configurable in Kerberos 5), Kerberos clients can not authenticate to the server. This time synchronization is necessary to prevent an attacker from using an old Kerberos ticket to masquerade as a valid user.

It is advisable to set up a Network Time Protocol (NTP) compatible client/server network even if Kerberos is not being used. Fedora includes the **`ntp`** package for this purpose. Refer to **`/usr/share/doc/ntp-`*`version-number`*`/index.html`** (where *`version-number`* is the version number of the **`ntp`** package installed on your system) for details about how to set up Network Time Protocol servers, and *http://www.ntp.org* for more information about NTP.

2. Install the **`krb5-libs`**, **`krb5-server`**, and **`krb5-workstation`** packages on the dedicated machine which runs the KDC. This machine needs to be very secure — if possible, it should not run any services other than the KDC.

3. Edit the **`/etc/krb5.conf`** and **`/var/kerberos/krb5kdc/kdc.conf`** configuration files to reflect the realm name and domain-to-realm mappings. A simple realm can be constructed by replacing instances of *`EXAMPLE.COM`* and *`example.com`* with the correct domain name — being certain to keep uppercase and lowercase names in the correct format — and by changing the KDC from *`kerberos.example.com`* to the name of the Kerberos server. By convention, all realm names are uppercase and all DNS hostnames and domain names are lowercase. For full details about the formats of these configuration files, refer to their respective man pages.

4. Create the database using the **`kdb5_util`** utility from a shell prompt:

```
/usr/sbin/kdb5_util create -s
```

The **`create`** command creates the database that stores keys for the Kerberos realm. The **`-s`** switch forces creation of a *stash* file in which the master server key is stored. If no stash file is present from which to read the key, the Kerberos server (**`krb5kdc`**) prompts the user for the master server password (which can be used to regenerate the key) every time it starts.

5. Edit the **`/var/kerberos/krb5kdc/kadm5.acl`** file. This file is used by **`kadmind`** to determine which principals have administrative access to the Kerberos database and their level of access. Most organizations can get by with a single line:

```
*/admin@EXAMPLE.COM  *
```

Most users are represented in the database by a single principal (with a *NULL*, or empty, instance, such as *joe@EXAMPLE.COM*). In this configuration, users with a second principal with an instance of *admin* (for example, *joe/admin@EXAMPLE.COM*) are able to wield full power over the realm's Kerberos database.

After **`kadmind`** has been started on the server, any user can access its services by running **`kadmin`** on any of the clients or servers in the realm. However, only users listed in the **`kadm5.acl`** file can modify the database in any way, except for changing their own passwords.

Note

The **`kadmin`** utility communicates with the **`kadmind`** server over the network, and uses Kerberos to handle authentication. Consequently, the first principal must already exist before connecting to the server over the network to administer it. Create the first principal with the **`kadmin.local`** command, which is specifically designed to be used on the same host as the KDC and does not use Kerberos for authentication.

Type the following **`kadmin.local`** command at the KDC terminal to create the first principal:

```
/usr/kerberos/sbin/kadmin.local -q "addprinc username/admin"
```

6. Start Kerberos using the following commands:

```
/sbin/service krb5kdc start
/sbin/service kadmin start
```

7. Add principals for the users using the **`addprinc`** command within **`kadmin`**. **`kadmin`** and **`kadmin.local`** are command line interfaces to the KDC. As such, many commands — such as **`addprinc`** — are available after launching the **`kadmin`** program. Refer to the **`kadmin`** man page for more information.

8. Verify that the KDC is issuing tickets. First, run **`kinit`** to obtain a ticket and store it in a credential cache file. Next, use **`klist`** to view the list of credentials in the cache and use **`kdestroy`** to destroy the cache and the credentials it contains.

Note

By default, **`kinit`** attempts to authenticate using the same system login username. If that username does not correspond to a principal in the Kerberos database, **`kinit`** issues an error message. If that happens, supply **`kinit`** with the name of the correct principal as an argument on the command line (**`kinit <principal>`**).

Once these steps are completed, the Kerberos server should be up and running.

3.7.6. Configuring a Kerberos 5 Client

Setting up a Kerberos 5 client is less involved than setting up a server. At a minimum, install the client packages and provide each client with a valid **`krb5.conf`** configuration file. While **`ssh`** and **`slogin`** are the preferred method of remotely logging in to client systems, Kerberized versions of **`rsh`** and **`rlogin`** are still available, though deploying them requires that a few more configuration changes be made.

1. Be sure that time synchronization is in place between the Kerberos client and the KDC. Refer to *Section 3.7.5, "Configuring a Kerberos 5 Server"* for more information. In addition, verify that DNS is working properly on the Kerberos client before configuring the Kerberos client programs.

2. Install the **`krb5-libs`** and **`krb5-workstation`** packages on all of the client machines. Supply a valid **`/etc/krb5.conf`** file for each client (usually this can be the same **`krb5.conf`** file used by the KDC).

3. Before a workstation in the realm can use Kerberos to authenticate users who connect using **`ssh`** or Kerberized **`rsh`** or **`rlogin`**, it must have its own host principal in the Kerberos database. The **`sshd`**, **`kshd`**, and **`klogind`** server programs all need access to the keys for the *host* service's principal. Additionally, in order to use the kerberized **`rsh`** and **`rlogin`** services, that workstation must have the **`xinetd`** package installed.

Using **kadmin**, add a host principal for the workstation on the KDC. The instance in this case is the hostname of the workstation. Use the **-randkey** option for the **kadmin**'s **addprinc** command to create the principal and assign it a random key:

```
addprinc -randkey host/blah.example.com
```

Now that the principal has been created, keys can be extracted for the workstation by running **kadmin** *on the workstation itself*, and using the **ktadd** command within **kadmin**:

```
ktadd -k /etc/krb5.keytab host/blah.example.com
```

4. To use other kerberized network services, they must first be started. Below is a list of some common kerberized services and instructions about enabling them:

   - **ssh** — OpenSSH uses GSS-API to authenticate users to servers if the client's and server's configuration both have **GSSAPIAuthentication** enabled. If the client also has **GSSAPIDelegateCredentials** enabled, the user's credentials are made available on the remote system.

   - **rsh** and **rlogin** — To use the kerberized versions of **rsh** and **rlogin**, enable **klogin**, **eklogin**, and **kshell**.

   - Telnet — To use kerberized Telnet, **krb5-telnet** must be enabled.

   - FTP — To provide FTP access, create and extract a key for the principal with a root of **ftp**. Be certain to set the instance to the fully qualified hostname of the FTP server, then enable **gssftp**.

   - IMAP — To use a kerberized IMAP server, the **cyrus-imap** package uses Kerberos 5 if it also has the **cyrus-sasl-gssapi** package installed. The **cyrus-sasl-gssapi** package contains the Cyrus SASL plugins which support GSS-API authentication. Cyrus IMAP should function properly with Kerberos as long as the **cyrus** user is able to find the proper key in **/etc/krb5.keytab**, and the root for the principal is set to **imap** (created with **kadmin**).

     An alternative to **cyrus-imap** can be found in the **dovecot** package, which is also included in Fedora. This package contains an IMAP server but does not, to date, support GSS-API and Kerberos.

   - CVS — To use a kerberized CVS server, **gserver** uses a principal with a root of **cvs** and is otherwise identical to the CVS **pserver**.

3.7.7. Domain-to-Realm Mapping

When a client attempts to access a service running on a particular server, it knows the name of the service (*host*) and the name of the server (*foo.example.com*), but because more than one realm may be deployed on your network, it must guess at the name of the realm in which the service resides.

By default, the name of the realm is taken to be the DNS domain name of the server, upper-cased.

```
foo.example.org → EXAMPLE.ORG
  foo.example.com → EXAMPLE.COM
  foo.hq.example.com → HQ.EXAMPLE.COM
```

In some configurations, this will be sufficient, but in others, the realm name which is derived will be the name of a non-existant realm. In these cases, the mapping from the server's DNS domain name to the name of its realm must be specified in the *domain_realm* section of the client system's **`krb5.conf`**. For example:

```
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

The above configuration specifies two mappings. The first mapping specifies that any system in the "example.com" DNS domain belongs to the *EXAMPLE.COM* realm. The second specifies that a system with the exact name "example.com" is also in the realm. (The distinction between a domain and a specific host is marked by the presence or lack of an initial ".".) The mapping can also be stored directly in DNS.

3.7.8. Setting Up Secondary KDCs

For a number of reasons, you may choose to run multiple KDCs for a given realm. In this scenario, one KDC (the *master KDC*) keeps a writable copy of the realm database and runs **`kadmind`** (it is also your realm's *admin server*), and one or more KDCs (*slave KDCs*) keep read-only copies of the database and run **`kpropd`**.

The master-slave propagation procedure entails the master KDC dumping its database to a temporary dump file and then transmitting that file to each of its slaves, which then overwrite their previously-received read-only copies of the database with the contents of the dump file.

To set up a slave KDC, first ensure that the master KDC's **`krb5.conf`** and **`kdc.conf`** files are copied to the slave KDC.

Start **`kadmin.local`** from a root shell on the master KDC and use its **`add_principal`** command to create a new entry for the master KDC's *host* service, and then use its **`ktadd`** command to simultaneously set a random key for the service and store the random key in the master's default keytab file. This key will be used by the **`kprop`** command to authenticate to the slave servers. You will only need to do this once, regardless of how many slave servers you install.

```
# kadmin.local -r EXAMPLE.COM

Authenticating as principal root/admin@EXAMPLE.COM with password.

kadmin: add_principal -randkey host/masterkdc.example.com

Principal "host/host/masterkdc.example.com@EXAMPLE.COM" created.

kadmin: ktadd host/masterkdc.example.com

Entry for principal host/masterkdc.example.com with kvno 3, encryption type Triple DES cbc
 mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/masterkdc.example.com with kvno 3, encryption type ArcFour with
 HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/masterkdc.example.com with kvno 3, encryption type DES with HMAC/
sha1 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/masterkdc.example.com with kvno 3, encryption type DES cbc mode with
 RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.

kadmin: quit
```

Start **`kadmin`** from a root shell on the slave KDC and use its **`add_principal`** command to create a new entry for the slave KDC's *host* service, and then use **`kadmin`**'s **`ktadd`** command to simultaneously set a random key for the service and store the random key in the slave's default keytab file. This key is used by the **kpropd** service when authenticating clients.

```
# kadmin -p jimbo/admin@EXAMPLE.COM -r EXAMPLE.COM

Authenticating as principal jimbo/admin@EXAMPLE.COM with password.

Password for jimbo/admin@EXAMPLE.COM:

kadmin: add_principal -randkey host/slavekdc.example.com

Principal "host/slavekdc.example.com@EXAMPLE.COM" created.

kadmin: ktadd host/slavekdc.example.com@EXAMPLE.COM

Entry for principal host/slavekdc.example.com with kvno 3, encryption type Triple DES cbc
 mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/slavekdc.example.com with kvno 3, encryption type ArcFour with HMAC/
md5 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/slavekdc.example.com with kvno 3, encryption type DES with HMAC/sha1
 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/slavekdc.example.com with kvno 3, encryption type DES cbc mode with
 RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.

kadmin: quit
```

With its service key, the slave KDC could authenticate any client which would connect to it. Obviously, not all of them should be allowed to provide the slave's **kprop** service with a new realm database. To restrict access, the **kprop** service on the slave KDC will only accept updates from clients whose principal names are listed in **/var/kerberos/krb5kdc/kpropd.acl**. Add the master KDC's host service's name to that file.

```
 # echo host/masterkdc.example.com@EXAMPLE.COM > /var/kerberos/krb5kdc/kpropd.acl
```

Once the slave KDC has obtained a copy of the database, it will also need the master key which was used to encrypt it. If your KDC database's master key is stored in a *stash* file on the master KDC (typically named **/var/kerberos/krb5kdc/.k5.REALM**, either copy it to the slave KDC using any available secure method, or create a dummy database and identical stash file on the slave KDC by running **kdb5_util create -s** (the dummy database will be overwritten by the first successful database propagation) and supplying the same password.

Ensure that the slave KDC's firewall allows the master KDC to contact it using TCP on port 754 (*krb5_prop*), and start the **kprop** service. Then, double-check that the **kadmin** service is *disabled*.

Now perform a manual database propagation test by dumping the realm database, on the master KDC, to the default data file which the **kprop** command will read (**/var/kerberos/krb5kdc/slave_datatrans**), and then use the **kprop** command to transmit its contents to the slave KDC.

```
 # /usr/sbin/kdb5_util dump /var/kerberos/krb5kdc/slave_datatrans# kprop slavekdc.example.com
```

Using **kinit**, verify that a client system whose **krb5.conf** lists only the slave KDC in its list of KDCs for your realm is now correctly able to obtain initial credentials from the slave KDC.

That done, simply create a script which dumps the realm database and runs the **kprop** command to transmit the database to each slave KDC in turn, and configure the **cron** service to run the script periodically.

3.7.9. Setting Up Cross Realm Authentication

Cross-realm authentication is the term which is used to describe situations in which clients (typically users) of one realm use Kerberos to authenticate to services (typically server processes running on a particular server system) which belong to a realm other than their own.

For the simplest case, in order for a client of a realm named **`A.EXAMPLE.COM`** to access a service in the **`B.EXAMPLE.COM`** realm, both realms must share a key for a principal named **`krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM`**, and both keys must have the same key version number associated with them.

To accomplish this, select a very strong password or passphrase, and create an entry for the principal in both realms using kadmin.

```
 # kadmin -r A.EXAMPLE.COM kadmin: add_principal krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM
 Enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM": Re-enter
 password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM": Principal
 "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM" created. quit # kadmin -r B.EXAMPLE.COM
 kadmin: add_principal krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM Enter password for principal
 "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM": Re-enter password for principal "krbtgt/
B.EXAMPLE.COM@A.EXAMPLE.COM": Principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM" created. quit
```

Use the **`get_principal`** command to verify that both entries have matching key version numbers (**kvno** values) and encryption types.

Dumping the Database Doesn't Do It

Security-conscious administrators may attempt to use the **`add_principal`** command's **`-randkey`** option to assign a random key instead of a password, dump the new entry from the database of the first realm, and import it into the second. This will not work unless the master keys for the realm databases are identical, as the keys contained in a database dump are themselves encrypted using the master key.

Clients in the **`A.EXAMPLE.COM`** realm are now able to authenticate to services in the **`B.EXAMPLE.COM`** realm. Put another way, the **`B.EXAMPLE.COM`** realm now *trusts* the **`A.EXAMPLE.COM`** realm, or phrased even more simply, **`B.EXAMPLE.COM`** now *trusts* **`A.EXAMPLE.COM`**.

This brings us to an important point: cross-realm trust is unidirectional by default. The KDC for the **`B.EXAMPLE.COM`** realm may trust clients from the **`A.EXAMPLE.COM`** to authenticate to services in the **`B.EXAMPLE.COM`** realm, but the fact that it does has no effect on whether or not clients in the **`B.EXAMPLE.COM`** realm are trusted to authenticate to services in the **`A.EXAMPLE.COM`** realm. To establish trust in the other direction, both realms would need to share keys for the **`krbtgt/A.EXAMPLE.COM@B.EXAMPLE.COM`** service (take note of the reversed in order of the two realms compared to the example above).

If direct trust relationships were the only method for providing trust between realms, networks which contain multiple realms would be very difficult to set up. Luckily, cross-realm trust is transitive. If clients from **`A.EXAMPLE.COM`** can authenticate to services in **`B.EXAMPLE.COM`**, and clients from **`B.EXAMPLE.COM`** can authenticate to services in **`C.EXAMPLE.COM`**, then clients in **`A.EXAMPLE.COM`** can also authenticate to services in **`C.EXAMPLE.COM`**, *even if **`C.EXAMPLE.COM`** doesn't directly trust **`A.EXAMPLE.COM`***. This means that, on a network with multiple realms which all need to trust each other, making good choices about which trust relationships to set up can greatly reduce the amount of effort required.

Now you face the more conventional problems: the client's system must be configured so that it can properly deduce the realm to which a particular service belongs, and it must be able to determine how to obtain credentials for services in that realm.

First things first: the principal name for a service provided from a specific server system in a given realm typically looks like this:

```
service/server.example.com@EXAMPLE.COM
```

In this example, *service* is typically either the name of the protocol in use (other common values include *ldap*, *imap*, *cvs*, and *HTTP*) or *host*, *server.example.com* is the fully-qualified domain name of the system which runs the service, and **`EXAMPLE.COM`** is the name of the realm.

To deduce the realm to which the service belongs, clients will most often consult DNS or the **`domain_realm`** section of **`/etc/krb5.conf`** to map either a hostname (*server.example.com*) or a DNS domain name (*.example.com*) to the name of a realm (*EXAMPLE.COM*).

Having determined which to which realm a service belongs, a client then has to determine the set of realms which it needs to contact, and in which order it must contact them, to obtain credentials for use in authenticating to the service.

This can be done in one of two ways.

The default method, which requires no explicit configuration, is to give the realms names within a shared hierarchy. For an example, assume realms named **`A.EXAMPLE.COM`**, **`B.EXAMPLE.COM`**, and **`EXAMPLE.COM`**. When a client in the **`A.EXAMPLE.COM`** realm attempts to authenticate to a service in **`B.EXAMPLE.COM`**, it will, by default, first attempt to get credentials for the **`EXAMPLE.COM`** realm, and then to use those credentials to obtain credentials for use in the **`B.EXAMPLE.COM`** realm.

The client in this scenario treats the realm name as one might treat a DNS name. It repeatedly strips off the components of its own realm's name to generate the names of realms which are "above" it in the hierarchy until it reaches a point which is also "above" the service's realm. At that point it begins prepending components of the service's realm name until it reaches the service's realm. Each realm which is involved in the process is another "hop".

For example, using credentials in **`A.EXAMPLE.COM`**, authenticating to a service in **`B.EXAMPLE.COMA.EXAMPLE.COM → EXAMPLE.COM → B.EXAMPLE.COM`**

- **`A.EXAMPLE.COM`** and **`EXAMPLE.COM`** share a key for **`krbtgt/EXAMPLE.COM@A.EXAMPLE.COM`**
- **`EXAMPLE.COM`** and **`B.EXAMPLE.COM`** share a key for **`krbtgt/B.EXAMPLE.COM@EXAMPLE.COM`**

Another example, using credentials in **`SITE1.SALES.EXAMPLE.COM`**, authenticating to a service in **`EVERYWHERE.EXAMPLE.COMSITE1.SALES.EXAMPLE.COM → SALES.EXAMPLE.COM → EXAMPLE.COM → EVERYWHERE.EXAMPLE.COM`**

- **`SITE1.SALES.EXAMPLE.COM`** and **`SALES.EXAMPLE.COM`** share a key for **`krbtgt/SALES.EXAMPLE.COM@SITE1.SALES.EXAMPLE.COM`**
- **`SALES.EXAMPLE.COM`** and **`EXAMPLE.COM`** share a key for **`krbtgt/EXAMPLE.COM@SALES.EXAMPLE.COM`**
- **`EXAMPLE.COM`** and **`EVERYWHERE.EXAMPLE.COM`** share a key for **`krbtgt/EVERYWHERE.EXAMPLE.COM@EXAMPLE.COM`**

Another example, this time using realm names whose names share no common suffix (**`DEVEL.EXAMPLE.COM`** and **`PROD.EXAMPLE.ORG DEVEL.EXAMPLE.COM → EXAMPLE.COM → COM → ORG → EXAMPLE.ORG → PROD.EXAMPLE.ORG`**

- **`DEVEL.EXAMPLE.COM`** and **`EXAMPLE.COM`** share a key for **`krbtgt/EXAMPLE.COM@DEVEL.EXAMPLE.COM`**
- **`EXAMPLE.COM`** and **`COM`** share a key for **`krbtgt/COM@EXAMPLE.COM`**
- **`COM`** and **`ORG`** share a key for **`krbtgt/ORG@COM`**
- **`ORG`** and **`EXAMPLE.ORG`** share a key for **`krbtgt/EXAMPLE.ORG@ORG`**
- **`EXAMPLE.ORG`** and **`PROD.EXAMPLE.ORG`** share a key for **`krbtgt/PROD.EXAMPLE.ORG@EXAMPLE.ORG`**

The more complicated, but also more flexible, method involves configuring the **`capaths`** section of **`/etc/krb5.conf`**, so that clients which have credentials for one realm will be able to look up which realm is next in the chain which will eventually lead to the being able to authenticate to servers.

The format of the **`capaths`** section is relatively straightforward: each entry in the section is named after a realm in which a client might exist. Inside of that subsection, the set of intermediate realms from which the client must obtain credentials is listed as values of the key which corresponds to the realm in which a service might reside. If there are no intermediate realms, the value "." is used.

Here's an example:

```
[capaths]
A.EXAMPLE.COM = {
B.EXAMPLE.COM = .
C.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = C.EXAMPLE.COM
}
```

In this example, clients in the **`A.EXAMPLE.COM`** realm can obtain cross-realm credentials for **`B.EXAMPLE.COM`** directly from the **`A.EXAMPLE.COM`** KDC.

If those clients wish to contact a service in the**`C.EXAMPLE.COM`** realm, they will first need to obtain necessary credentials from the **`B.EXAMPLE.COM`** realm (this requires that **`krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM`** exist), and then use **`those`** credentials to obtain credentials for use in the **`C.EXAMPLE.COM`** realm (using **`krbtgt/C.EXAMPLE.COM@B.EXAMPLE.COM`**).

If those clients wish to contact a service in the **`D.EXAMPLE.COM`** realm, they will first need to obtain necessary credentials from the **`B.EXAMPLE.COM`** realm, and then credentials from the **`C.EXAMPLE.COM`** realm, before finally obtaining credentials for use with the **`D.EXAMPLE.COM`** realm.

Note

Without a capath entry indicating otherwise, Kerberos assumes that cross-realm trust relationships form a hierarchy.

Clients in the **`A.EXAMPLE.COM`** realm can obtain cross-realm credentials from **`B.EXAMPLE.COM`** realm directly. Without the "." indicating this, the client would instead attempt to use a hierarchical path, in this case:

```
A.EXAMPLE.COM → EXAMPLE.COM → B.EXAMPLE.COM
```

3.7.10. Additional Resources

For more information about Kerberos, refer to the following resources.

3.7.10.1. Installed Kerberos Documentation

- The *Kerberos V5 Installation Guide* and the *Kerberos V5 System Administrator's Guide* in PostScript and HTML formats. These can be found in the **`/usr/share/doc/krb5-server-<version-number>/`** directory (where `<version-number>` is the version number of the **`krb5-server`** package installed on your system).

- The *Kerberos V5 UNIX User's Guide* in PostScript and HTML formats. These can be found in the **`/usr/share/doc/krb5-workstation-<version-number>/`** directory (where `<version-number>` is the version number of the **`krb5-workstation`** package installed on your system).

- Kerberos man pages — There are a number of man pages for the various applications and configuration files involved with a Kerberos implementation. The following is a list of some of the more important man pages.

  Client Applications

  - **`man kerberos`** — An introduction to the Kerberos system which describes how credentials work and provides recommendations for obtaining and destroying Kerberos tickets. The bottom of the man page references a number of related man pages.

  - **`man kinit`** — Describes how to use this command to obtain and cache a ticket-granting ticket.

  - **`man kdestroy`** — Describes how to use this command to destroy Kerberos credentials.

  - **`man klist`** — Describes how to use this command to list cached Kerberos credentials.

  Administrative Applications

  - **`man kadmin`** — Describes how to use this command to administer the Kerberos V5 database.

  - **`man kdb5_util`** — Describes how to use this command to create and perform low-level administrative functions on the Kerberos V5 database.

  Server Applications

  - **`man krb5kdc`** — Describes available command line options for the Kerberos V5 KDC.

- **`man kadmind`** — Describes available command line options for the Kerberos V5 administration server.

Configuration Files

- **`man krb5.conf`** — Describes the format and options available within the configuration file for the Kerberos V5 library.
- **`man kdc.conf`** — Describes the format and options available within the configuration file for the Kerberos V5 AS and KDC.

3.7.10.2. Useful Kerberos Websites

- *http://web.mit.edu/kerberos/www/* — *Kerberos: The Network Authentication Protocol* webpage from MIT.
- *http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html* — The Kerberos Frequently Asked Questions (FAQ).
- *ftp://athena-dist.mit.edu/pub/kerberos/doc/usenix.PS* — The PostScript version of *Kerberos: An Authentication Service for Open Network Systems* by Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. This document is the original paper describing Kerberos.
- *http://web.mit.edu/kerberos/www/dialogue.html* — *Designing an Authentication System: a Dialogue in Four Scenes* originally by Bill Bryant in 1988, modified by Theodore Ts'o in 1997. This document is a conversation between two developers who are thinking through the creation of a Kerberos-style authentication system. The conversational style of the discussion make this a good starting place for people who are completely unfamiliar with Kerberos.
- *http://www.ornl.gov/~jar/HowToKerb.html* — *How to Kerberize your site* is a good reference for kerberizing a network.
- *http://www.networkcomputing.com/netdesign/kerb1.html* — *Kerberos Network Design Manual* is a thorough overview of the Kerberos system.

3.8. Using Firewalls

3.8.1. Introduction to firewalld

The dynamic firewall daemon `firewalld` provides a dynamically managed firewall with support for network zones to assign a level of trust to a network and its associated connections and interfaces. It has support for `IPv4` and `IPv6` firewall settings. It supports Ethernet bridges and has a separation of runtime and permanent configuration options. It also has an interface for services or applications to add firewall rules directly.

3.8.2. Understanding firewalld

A graphical configuration tool, **firewall-config**, is used to configure `firewalld`, which in turn uses **iptables tool** to communicate with **Netfilter** in the kernel which implements packet filtering.

To use the graphical **firewall-config** tool, press the super key and start typing **`firewall`**. The firewall icon will appear. Press enter once it is highlighted. The **firewall-config** tool appears. You will be prompted for your user password.

The **firewall-config** tool has drop a down selection menu labeled **Current View**. This enables selecting between **Runtime Configuration** and **Permanent Configuration** mode. Notice that if you select **Permanent Configuration**, an **Edit Services** button appears on the right hand side of the **Services** tab and an **Edit ICMP Types** button appears on the right hand side of the **ICMP Filter** tab. The reason these buttons only appear in permanent configuration mode is that runtime changes are limited to enabling or disabling a service. You cannot change a service's parameters in run time mode.

The firewall service provided by `firewalld` is dynamic rather than static because changes to the configuration can be made at anytime and are immediately implemented, there is no need to save or apply the changes. No unintended disruption of existing network connections occurs as no part of the firewall has to be reloaded.

There is also an applet, **firewall-applet**, which can be used to quickly launch the **NetworkManager** configuration tab for the network connection in use. From the **General** tab changes to the assigned firewall zone can be made. This applet is not installed by default in Fedora.

A command line client, **firewall-cmd**, is provided. It can be used to make permanent and non-permanent run-time changes as explained in **`man firewall-cmd(1)`**. Permanent changes need to be made as explained in **`man firewalld(1)`**.

The configuration for `firewalld` is stored in various XML files in **`/usr/lib/firewalld/`** and **`/etc/firewalld/`**. This allows a great deal of flexibility as the files can be edited, written to, backed up, used as templates for other installations and so on.

Other applications can communicate with `firewalld` using D-bus.

3.8.3. Comparison of Firewalld to system-config-firewall and iptables

The essential differences between `firewalld` and the **iptables service** are:

- The **iptables service** stores configuration in **`/etc/sysconfig/iptables`** while `firewalld` stores it in various XML files in **`/usr/lib/firewalld/`** and **`/etc/firewalld/`**. Note that the **`/etc/sysconfig/iptables`** file does not exist as `firewalld` is installed be default on Fedora.

- With the **iptables service**, every single change means flushing all the old rules and reading all the new rules from **`/etc/sysconfig/iptables`** while with `firewalld` there is no re-creating of all the rules; only the differences are applied. Consequently, `firewalld` can change the settings during run time without existing connections being lost.

Both use **iptables tool** to talk to the kernel packet filter.

3.8.4. Understanding Network Zones

Firewalls can be used to separate networks into different zones based on the level of trust the user has decided to place on the devices and traffic within that network. **NetworkManager** informs `firewalld` to which zone an interface belongs. An interface's assigned zone can be changed by **NetworkManager** or via the **firewall-config** tool which can open the relevant **NetworkManager** window for you.

The zone settings in **`/etc/firewalld/`** are a range of preset settings which can be quickly applied to a network interface. They are listed here with a brief explanation:

`drop (immutable)`
: Any incoming network packets are dropped, there is no reply. Only outgoing network connections are possible.

`block (immutable)`
: Any incoming network connections are rejected with an icmp-host-prohibited message for `IPv4` and icmp6-adm-prohibited for `IPv6`. Only network connections initiated from within the system are possible.

`public`
: For use in public areas. You do not trust the other computers on the network to not harm your computer. Only selected incoming connections are accepted.

`external`
: For use on external networks with masquerading enabled especially for routers. You do not trust the other computers on the network to not harm your computer. Only selected incoming connections are accepted.

`dmz`
: For computers in your demilitarized zone that are publicly-accessible with limited access to your internal network. Only selected incoming connections are accepted.

`work`
: For use in work areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.

`home`
: For use in home areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.

`internal`
: For use on internal networks. You mostly trust the other computers on the networks to not harm your computer. Only selected incoming connections are accepted.

`trusted (immutable)`
: All network connections are accepted.

It is possible to designate one of these zones to be the default zone. When interface connections are added to **NetworkManager**, they are assigned to the default zone. On installation, the default zone in `firewalld` is set to be the public zone.

3.8.5. Choosing a Network Zone

The network zone names have been chosen to be self-explanatory and to allow users to quickly make a reasonable decision. However, a review of the default configuration settings should be made and unnecessary services disabled according to your needs and risk assessments.

3.8.6. Understanding Predefined Services

A service can be a list of local ports and destinations as well as a list of firewall helper modules automatically loaded if a service is enabled. The use of predefined services makes it easier for the user to enable and disable access to a service. Using the predefined services, or custom defined services, as opposed to opening ports or ranges or ports may make administration easier. Service configuration options and generic file information are described in the **`firewalld.service(5)`** man page. The services are specified by means of individual XML configuration files which are named in the following format: ***`service-name`*** **`.xml`**.

To view the list of services using the graphical **firewall-config** tool, press the super key and start typing **`firewall`**. The firewall icon will appear. Press enter once it is highlighted. The **firewall-config** tool appears. You will be prompted for your user password. You can now view the list of services under the **Services** tab.

To list the default predefined services available using the command line, issue the following command as root:

```
~]# ls /usr/lib/firewalld/services/
```

Files in **`/usr/lib/firewalld/services/`** must not be edited. Only the files in **`/etc/firewalld/services/`** should be edited.

To list the system or user created services, issue the following command as root:

```
~]# ls /etc/firewalld/services/
```

Services can be added and removed using the graphical **firewall-config** tool and by editing the XML files in **`/etc/firewalld/services/`**. If a service has not be added or changed by the user, then no corresponding XML file will be found in **`/etc/firewalld/services/`**. The files **`/usr/lib/firewalld/services/`** can be used as templates if you wish to add or change a service. As root, issue a command in the following format:

```
~]# cp /usr/lib/firewalld/services/[service].xml /etc/firewalld/services/[service].xml
```

You may then edit the newly created file. `firewalld` will prefer files in **`/etc/firewalld/services/`** but will fall back to **`/usr/lib/firewalld/services/`** should a file be deleted, but only after a reload.

3.8.7. Understanding The Direct Interface

`firewalld` has a so called direct interface, which enables directly passing rules to **iptables**, **ip6tables** and **ebtables**. It is intended for use by applications and not users. It is dangerous to use the direct interface if you are not very familiar with **iptables** as you could inadvertently cause a breach in the firewall. `firewalld` still tracks what has been added, so it is still possible to query `firewalld` and see the changes made by an application using the direct interface mode. The direct interface is used by adding the **`--direct`** option to **`firewall-cmd`**.

The direct interface mode is intended for services or applications to add specific firewall rules during run time. The rules are not permanent and need to be applied every time after receiving the start, restart or reload message from `firewalld` using D-BUS.

3.8.8. Check if firewalld is installed

In Fedora `firewalld` and the graphical user interface configuration tool **firewall-config** are installed by default but **firewall-applet** is not. This can be checked by running the following command as root:

```
~]# yum install firewalld firewall-config
```

3.8.9. Disabling firewalld

To disable `firewalld`, run the following commands as root:

```
~]# systemctl disable firewalld # systemctl stop firewalld
```

3.8.9.1. Using the iptables service

To use the **iptables service** instead of `firewalld`, first disable `firewalld` by running the following command as root:

```
~]# systemctl disable firewalld # systemctl stop firewalld
```

Then install the *iptables-services* package by entering the following command as root:

```
~]# yum install iptables-services
```

Then, to start **iptables service**, run the following commands as root:

```

  # touch /etc/sysconfig/iptables
  # touch /etc/sysconfig/ip6tables
  # systemctl start iptables
  # systemctl start ip6tables
  # systemctl enable iptables
  # systemctl enable ip6tables
```

3.8.10. Start firewalld

To start `firewalld`, enter the following command as root:

```
~]# systemctl start firewalld
```

3.8.11. Check if firewalld is running

To check if `firewalld` is running, enter the following command:

```
~]$ systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
   Active: active (running) since Sat 2013-04-06 22:56:59 CEST; 2 days ago
 Main PID: 688 (firewalld)
   CGroup: name=systemd:/system/firewalld.service
```

In addition, check if **firewall-cmd** can connect to the daemon by entering the following command:

```
~]$ firewall-cmd --state
running
```

3.8.12. Installing firewalld

To install `firewalld`, run the following command as root:

```
~]# yum install firewalld
```

To install the graphical user interface tool **firewall-config**, run the following command as root:

```
~]# yum install firewall-config
```

To install the optional **firewall-applet**, run the following command as root:

```
~]# yum install firewall-applet
```

3.8.13. Configuring the Firewall

The firewall can be configured using the graphical user interface tool **firewall-config**, using the command line interface tool **firewall-cmd** and by editing XML configuration files. These methods will be described in order.

3.8.13.1. Configuring the Firewall using the graphical user interface

3.8.13.1.1. Start the graphical firewall configuration tool

To start the graphical **firewall-config** tool, press the super key and start typing `firewall`. The firewall icon will appear. Press enter once it is highlighted. The **firewall-config** tool appears. You will be prompted for your user password.

To start the graphical firewall configuration tool using the command line, enter the following command as root user:

```
~]# firewall-config
```

The **Firewall Configuration** window opens. Note, this command can be run as normal user but you will then be prompted for the root password from time to time.

Look for the word Connected in the lower left corner. This indicates that the **firewall-config** tool is connected to the user space daemon, `firewalld`.

3.8.13.1.2. Change the firewall settings

To immediately change the current firewall settings, ensure the current view is set to **Runtime Configuration**. Alternatively, to edit the settings to be applied at the next system start, or firewall reload, select **Permanent Configuration** from the drop down list.

Note

When making changes to the firewall settings in **Runtime Configuration** mode, your selection takes immediate effect when you set or clear the check box associated with the service. You should keep this in mind when working on a system that may be in use by other users.

When making changes to the firewall settings in **Permanent Configuration** mode, your selection will only take effect when you reload the firewall or the system restarts. You can use the reload icon below the **File** menu, or click the **Options** menu and select **Reload Firewall**.

You can select zones in the left hand side column. You will notice the zones have some services enabled, you may need to resize the window or scroll to see the full list. You can customize the settings by selecting and deselecting a service except for the zones **block**, **drop**, and **trusted** as those zone settings are classified as immutable, they cannot be changed.

3.8.13.1.3. Add an Interface to a zone

To add or reassign an interface of a connection to zone, start **firewall-config**, select **Options** from the menu bar, select **Change Zones of Connections** from the drop down menu. The **Network Connections** window appears. Select the connection you wish to add or reassign and select **Edit**. The **Editing** a connection window appears. Select the **General** tab. Select the new firewall zone from the drop down menu and click **Save**.

3.8.13.1.4. Set the Default Zone

To set the default zone that new interfaces will be assigned to, start **firewall-config**, select **Options** from the menu bar, select **Change Default Zone** from the drop down menu. The **System Default Zone** window appears. Select the zone form the list that you want to be used as the default zone and click **OK**.

3.8.13.1.5. Configuring Services

To enable or disable a predefined or custom service, start the **firewall-config** tool and select the network zone whose services are to be configured. Select the **Services** tab and select the check box for each type of service you want to trust. Clear the check box to block a service.

To edit a service, start the **firewall-config** tool and then select **Permanent Configuration** mode from the drop-down selection menu labeled **Current View**. An **Edit Services** button appears on the right hand side of the **ICMP Filter** tab. Click **Edit Services**, the **Service Settings** window appears. Select the service you wish to configure. The **Ports and Protocols** tab enables adding, changing, and removing of ports and protocols for the selected service. The modules tab is for configuring **Netfilter** helper modules. The **Destination** tab enables limiting traffic to a particular destination address and Internet Protocol (`IPv4` or `IPv6`.

3.8.13.1.6. Open Ports in the firewall

To permit traffic through the firewall to a certain port, start the **firewall-config** tool and select the network zone whose settings you want to change. Select the **Ports** tab and the click the **Add** button on the right hand side. The **Port and Protocol** window opens.

Enter the port number or range of ports to permit. Select **tcp** or **udp** from the drop down list.

3.8.13.1.7. Enable IP Address Masquerading

To translate `IPv4` addresses to a single external address, start the **firewall-config** tool and select the network zone whose addresses are to be translated. Select the **Masquerading** tab and select the check box to enable the translation of `IPv4` addresses to a single address.

3.8.13.1.8. Configure Port Forwarding

To forward inbound network traffic, or packets, for a specific port to an internal address or alternative port, first enable IP address masquerading, then select the **Port Forwarding** tab.

Select the protocol of the incoming traffic and the port or range of ports on the upper section of the window. The lower section is for setting details about the destination.

To forward traffic to a local port, that is to say to a port on the same system, select the **Local forwarding** check box. Enter the local port or range of ports for the traffic to be sent to.

To forward traffic to another `IPv4` address, select the **Forward to another port** check box. Enter the destination IP address and port or port range. The default is to send to the same port if the port field is left empty. Click **OK** to apply the changes.

3.8.13.1.9. Configuring the ICMP Filter

To enable or disable an `ICMP` filter, start the **firewall-config** tool and select the network zone whose messages are to be filtered. Select the **ICMP Filter** tab and select the check box for each type of `ICMP` message you want to filter. Clear the check box to disable a filter. This setting is per direction and the default allows everything.

To edit an `ICMP` filter, start the **firewall-config** tool and then select **Permanent Configuration** mode from the drop-down selection menu labeled **Current View**. An **Edit ICMP Types** button appears on the right hand side of the **ICMP Filter** tab.

3.8.13.2. Configuring the Firewall using the command line tool, firewall-cmd

The command line tool **firewall-cmd** is part of the `firewalld` application which is installed by default. You can verify that it is installed by checking the version or displaying the help output. Enter the following command to check the version:

```
~]$  firewall-cmd -V, --version
```

Enter the following command to view the help output:

```
~]$  firewall-cmd -h, --help
```

We list a selection of commands below, for a full list please see the man page, **`man firewall-cmd(1)`**.

Note

In order to make a command permanent or persistent, add the **`--permanent`** option to all commands apart from the **`--direct`** commands (which are by their nature temporary). Note that this not only means the change will be permanent but that the change will only take effect after firewall reload, service restart, or after system reboot. Settings made with **firewall-cmd** without the **`--permanent`** option take effect immediately, but are only valid till next firewall reload, system boot, or `firewalld` service restart. Reloading the firewall does not in itself break connections, but be aware you are discarding temporary changes by doing so.

3.8.13.3. View the firewall settings using the CLI

To get a text display of the state of `firewalld`, enter the following command:

```
~]$  firewall-cmd --state
```

To view the list of active zones, with a list of the interfaces currently assigned to them, enter the following command:

```
~]$  firewall-cmd --get-active-zones
    public: em1 wlan0
```

To find out the zone that an interface, for example em1, is currently assigned to, enter the following command:

```
~]$  firewall-cmd --get-zone-of-interface=em1
public
```

To find out all the interfaces assigned to a zone, for example the public zone, enter the following command as root:

```
~]# firewall-cmd --zone=public --list-interfaces
     em1 wlan0
```

This information is obtained from **NetworkManager** and only shows interfaces not connections.

To find out all the settings of a zone, for example the public zone, enter the following command as root:

```
~]# firewall-cmd --zone=public --list-all
public
  interfaces:
  services: mdns dhcpv6-client ssh
  ports:
  forward-ports:
  icmp-blocks: source-quench
```

To view the network zones currently active, enter the following command as root:

```
~]# firewall-cmd --get-service
    cluster-suite pop3s bacula-client smtp ipp radius bacula ftp mdns samba dhcpv6-client
 dns openvpn imaps samba-client http https ntp vnc-server telnet libvirt ssh ipsec ipp-client
 amanda-client tftp-client nfs tftp libvirt-tls
```

This will list the names of the services in **`/usr/lib/firewalld/services/`**. Note that the configuration files themselves are named ***`service-name`*****`.xml`**.

To view the network zones that will be active after the next firewall reload, enter the following command as root:

```
~]# firewall-cmd --get-service --permanent
```

3.8.13.4. View the firewall settings using nmcli

To get a list of all the interfaces and actions assigned to a zone, enter the following command:

```
~]$  nmcli -f NAME,DEVICES,ZONE con status
NAME                      DEVICES    ZONE
my-little-wifi            wlan0      home
VPN connection 1          wlan0      work
System em1                em1        --
```

-- means the interface is assigned to the default zone.

3.8.13.5. Change the firewall settings using the Command Line Interface (CLI)

3.8.13.5.1. Drop All Packets (Panic Mode)

To start dropping all incoming and outgoing packets, enter the following command as root:

```
~]# firewall-cmd --panic-on
```

All incoming and outgoing packets will be dropped. Active connections will be terminated after a period of inactivity; the time taken depends on the individual session time out values.

To start passing incoming and outgoing packets again, enter the following command as root:

```
~]# firewall-cmd --panic-off
```

After disabling panic mode, established connections might work again if panic mode was enabled for a short period of time.

To get a text indication if panic mode is enabled or disabled, enter the following command:

```
~]$  firewall-cmd --query-panic && echo "enabled" || echo "Not enabled"
```

3.8.13.5.2. Reload the firewall using the CLI

To reload the firewall with out interrupting user connections, that is to say, with out losing state information, enter the following command as root:

```
~]# firewall-cmd --reload
```

To reload the firewall and interrupt user connections, that is to say, to discard state information, enter the following command as root:

```
~]# firewall-cmd --complete-reload
```

This command should normally only be used in case of severe firewall problems. For example, if there are state information problems and no connection can be established but the firewall rules are correct.

3.8.13.5.3. Add an Interface to a Zone using the CLI

To add an interface to a zone, for example to add em1 to the public zone, enter the following command as root:

```
~]# firewall-cmd --zone=public --add-interface=em1
```

To make this setting permanent, add the **`--permanent`** option and reload the firewall.

3.8.13.5.4. Add an Interface to a Zone by Editing the Interface Configuration File

To add an interface to a zone by editing the **`ifcfg-em1`** configuration file, for example to add em1 to the work zone, as root use an editor to add the following line to **`ifcfg-em1`**:

```
ZONE=work
```

Note that if you omit the **`ZONE`** option, or use **`ZONE=`**, or **`ZONE=''`**, then the default zone will be used.

NetworkManager will automatically reconnect and the zone will be set accordingly.

3.8.13.5.5. Configure the default zone by Editing the firewalld Configuration File

As root, open **`/etc/firewalld/firewalld.conf`** and edit the file as follows:

```
# default zone
# The default zone used if an empty zone string is used.
# Default: public
DefaultZone=home
```

Reload the firewall, by entering the following command as root:

```
~]# firewall-cmd --reload
```

This will reload the firewall without losing state information (TCP sessions will not be interrupted).

3.8.13.5.6. Set the default zone by using the CLI

To set the default zone, for example to public, enter the following command as root:

```
~]# firewall-cmd --set-default-zone=public
```

This change will take immediate effect and in this case it is not necessary to reload the firewall.

3.8.13.5.7. Open Ports in the Firewall using the CLI

List all open ports for a zone, for example dmz, by entering the following command as root:

```
~]# firewall-cmd --zone=dmz --list-ports
```

To add a port to a zone, for example to allow `TCP` traffic to port `8080` to the dmz zone, enter the following command as root:

```
~]# firewall-cmd --zone=dmz --add-port=8080/tcp
```

To make this setting permanent, add the **`--permanent`** option and reload the firewall.

To add a range of ports to a zone, for example to allow the ports from 5060 to 5061 to the public zone, enter the following command as root:

```
~]# firewall-cmd --zone=public --add-port=5060-5061/udp
```

To make this setting permanent, add the **`--permanent`** option and reload the firewall.

3.8.13.5.8. Add a Service to a Zone using the CLI

To add a service to a zone, for example to allow `SMTP` to the work zone, enter the following command as root:

```
~]# firewall-cmd --zone=work --add-service=smtp
```

To make this setting permanent, add the **`--permanent`** option and reload the firewall.

3.8.13.5.9. Remove a Service from a Zone using the CLI

To remove a service from a zone, for example to remove `SMTP` from the work zone, enter the following command as root:

```
~]# firewall-cmd --zone=work --remove-service=smtp
```

Add the **`--permanent`** option to make the change persist after system boot. If using this option and you wish to make the change immediate, reload the firewall, by entering the following command as root:

```
~]# firewall-cmd --reload
```

Note, this will not break established connections. If that is your intention, you could use the **`--complete-reload`** option but this will break all established connections not just for the service you have removed.

3.8.13.5.10. Add a Service to a Zone by Editing XML files

To view the default zone files, enter the following command as root:

```
~]# ls /usr/lib/firewalld/zones/
block.xml  drop.xml      home.xml      public.xml   work.xml
dmz.xml    external.xml  internal.xml  trusted.xml
```

These files must not be edited. They are used by default if no equivalent file exists in the **`/etc/firewalld/zones/`** directory.

To view the zone files that have been changed from the default, enter the following command as root:

```
~]# ls /etc/firewalld/zones/
external.xml  public.xml  public.xml.old
```

In the example shown above, the work zone file does not exist. To add the work zone file, enter the following command as root:

```
~]# cp /usr/lib/firewalld/zones/work.xml /etc/firewalld/zones/
```

You can now edit the file in the **`/etc/firewalld/zones/`** directory. If you delete the file, `firewalld` will fall back to using the default file in **`/usr/lib/firewalld/zones/`**.

To add a service to a zone, for example to allow `SMTP` to the work zone, use an editor with root privileges to edit the **`/etc/firewalld/zones/work.xml`** file to include the following line:

```
<service name="smtp"/>
```

3.8.13.5.11. Remove a Service from a Zone by Editing XML files

An editor running with root privileges is required to edit the XML zone files. To view the files for previously configured zones, enter the following command as root:

```
~]# ls /etc/firewalld/zones/
external.xml  public.xml  work.xml
```

To remove a service from a zone, for example to remove `SMTP` from the work zone, use an editor with root privileges to edit the **`/etc/firewalld/zones/work.xml`** file to remove the following line:

```
<service name="smtp"/>
```

If no other changes have been made to the **`work.xml`** file, it can be removed and `firewalld` will use the default **`/usr/lib/firewalld/zones/work.xml`** configuration file after the next reload or system boot.

3.8.13.5.12. Configure IP Address Masquerading

To check if IP masquerading is enabled, for example for the external zone, enter the following command as root:

```
~]# firewall-cmd --zone=external --query-masquerade && echo "enabled" || echo "Not enabled"
```

If **`zone`** is omitted, the default zone will be used.

To enable IP masquerading, enter the following command as root:

```
~]# firewall-cmd --zone=external --add-masquerade
```

To make this setting permanent, add the **`--permanent`** option and reload the firewall.

To disable IP masquerading, enter the following command as root:

```
~]# firewall-cmd --zone=external --remove-masquerade
```

To make this setting permanent, add the **`--permanent`** option and reload the firewall.

3.8.13.5.13. Configure Port Forwarding using the CLI

To forward inbound network packets from one port to an alternative port or address, first enable IP address masquerading for a zone, for example external, by entering the following command as root:

```
~]# firewall-cmd --zone=external --add-masquerade
```

To forward packets to a local port, that is to say to a port on the same system, enter the following command as root:

```
~]# firewall-cmd --zone=external --add-forward-port=port=22:proto=tcp:toport=3753
```

In this example, the packets intended for port 22 are now forwarded to port 3753. The original destination port is specified with the **`port`** option. This option can be a port, or port range, together with a protocol. The protocol, if specified, must be one of either `tcp` or `udp`. The new local port, the port or range of ports to which the traffic is being forwarded to, is specified with the **`toport`** option. To make this setting permanent, add the **`--permanent`** option and reload the firewall.

To forward packets to another `IPv4` address, usually an internal address, without changing the destination port, enter the following command as root:

```
~]# firewall-cmd --zone=external --add-forward-port=port=22:proto=tcp:toaddr=192.0.2.55
```

In this example, the packets intended for port 22 are now forwarded to the same port at the address given with the **`toaddr`**. The original destination port is specified with the **`port`**. This option can be a port, or port range, together with a protocol. The protocol, if specified, must be one of either `tcp` or `udp`. The new destination port, the port or range of ports to which the traffic is being forwarded to, is specified with the **`toport`**. To make this setting permanent, add the **`--permanent`** option and reload the firewall.

To forward packets to another port at another `IPv4` address, usually an internal address, enter the following command as root:

```
~]# firewall-cmd --zone=external --add-forward-
port=port=22:proto=tcp:toport=2055:toaddr=192.0.2.55
```

In this example, the packets intended for port 22 are now forwarded to port 2055 at the address given with the **`toaddr`**. The original destination port is specified with the **`port`**. This option can be a port, or port range, together with a protocol. The protocol, if specified, must be one of either `tcp` or `udp`. The new destination port, the port or range of ports to which the traffic is being forwarded to, is specified with the **`toport`**. To make this setting permanent, add the **`--permanent`** option and reload the firewall.

3.8.13.6. Configuring The Firewall Using XML Files

The configuration settings for **firewalld** are stored in XML files in the **`/etc/firewalld/`** directory. Do not edit the files in the **`/usr/lib/firewalld/`** directory, they are for the default settings. You will need root user permissions to view and edit the XML files. The XML files are explained in three man pages:

- **`firewalld.icmptype(5)`** man page — Describes XML configuration files for `ICMP` filtering.
- **`firewalld.service(5)`** man page — Describes XML configuration files for **firewalld service**.
- **`firewalld.zone(5)`** man page — Describes XML configuration files for `firewalld` zone configuration.

The XML files can be created and edited directly or created indirectly using the graphical and command line tools. Organizations can distribute them in RPM files which can make management and version control easier. Tools such as **Puppet** can distribute such configuration files.

3.8.13.7. Using the direct interface

It is possible to add and remove chains during runtime by using the **`--direct`** option with the **firewall-cmd** tool. A few examples are presented here, please see the **`firewall-cmd(1)`** man page for more information.

It is dangerous to use the direct interface if you are not very familiar with **iptables** as you could inadvertently cause a breach in the firewall.

The direct interface mode is intended for services or applications to add specific firewall rules during run time. The rules are not permanent and need to be applied every time after receiving the start, restart or reload message from `firewalld` using D-BUS.

3.8.13.7.1. Adding a custom rule using the direct interface

To add a custom rule to the chain IN_ZONE_public_allow, issuing a command as root in the following format:

```
~]# firewall-cmd --direct --add-rule ipv4 filter IN_ZONE_public_allow 0 -m tcp -p tcp --dport
 666 -j ACCEPT
```

3.8.13.7.2. Removing a custom rule using the direct interface

To remove a custom rule from the chain IN_ZONE_public_allow, issuing a command as root in the following format:

```
~]# firewall-cmd --direct --remove-rule ipv4 filter IN_ZONE_public_allow -m tcp -p tcp --
dport 666 -j ACCEPT
```

3.8.13.7.3. Listing custom rules using the direct interface

To list the rules in the chain IN_ZONE_public_allow, issuing a command as root in the following format:

```
~]# firewall-cmd --direct --get-rules ipv4 filter IN_ZONE_public_allow
```

3.8.14. Additional Resources

The following sources of information provide additional resources regarding `firewalld`.

3.8.14.1. Installed Documentation

- **`firewalld(1)`** man page — Describes command options for `firewalld`.
- **`firewalld.conf(5)`** man page — Contains information to configure `firewalld`.
- **`firewall-cmd(1)`** man page — Describes command options for the `firewalld` command line client.
- **`firewalld.icmptype(5)`** man page — Describes XML configuration files for `ICMP` filtering.
- **`firewalld.service(5)`** man page — Describes XML configuration files for **firewalld serivice**.
- **`firewalld.zone(5)`** man page — Describes XML configuration files for `firewalld` zone configuration.

3.8.14.2. Useful Websites

https://fedoraproject.org/wiki/FirewallD

The website of the upstream project.

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.

4.1. 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 was encrypted on the media then you wouldn't have to worry as much about the data being compromised.

4.1.1. 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 don't have to worry about selecting what you want to protect and possibly missing a file.

Fedora natively supports LUKS Encryption. LUKS will bulk encrypt 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.

4.1.2. 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 intended to protect a file after it has left your computer, such as when you send a CD through the mail. Some file based encryption solutions will 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 those files from attackers who may have access to your computer, use file based encryption combined with another solution such as full disk encryption.

4.2. 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 user name 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. Other private information such as bank account information should also be protected when transmitted across a network. If the network session was encrypted then you would not have to worry as much about the data being compromised while it is being transmitted.

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.

4.2.1. 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 (SMBs) that want to expand without paying the high costs associated with enterprise-level, dedicated digital circuits.

To address this need, *Virtual Private Networks* (VPNs) were developed. Following the same functional principles as dedicated circuits, VPNs allow for secured digital communication between two parties (or networks), creating a *Wide Area Network* (WAN) from existing *Local Area Networks* (LANs). Where it differs from frame relay or ATM is in its transport medium. VPNs transmit over IP using datagrams as the transport layer, making it a secure conduit through the Internet to an intended destination. Most free software VPN implementations incorporate open standard encryption methods to further mask data in transit.

Some organizations employ hardware VPN solutions to augment security, while others use software or protocol-based implementations. Several vendors provide hardware VPN solutions, such as Cisco, Nortel, IBM, and Checkpoint. There is a free software-based VPN solution for Linux called FreeS/Wan that utilizes a standardized *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.

4.2.1.1. How Does a VPN Work?

When a packet is transmitted from a client, it sends it through the VPN router or gateway, which adds an *Authentication Header* (AH) for routing and authentication. The data is then encrypted and, finally, enclosed with an *Encapsulating Security Payload* (ESP). This latter constitutes the decryption and handling instructions.

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/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 intercept a packet, but decrypt the packet as well. Intruders who employ a man-in-the-middle attack between a server and client must also have access to at least one of the private keys for authenticating sessions. Because they employ several layers of authentication and encryption, VPNs are a secure and effective means of connecting multiple remote nodes to act as a unified intranet.

4.2.1.2. VPNs and Fedora

Fedora provides various options in terms of implementing a software solution to securely connect to a WAN. *Internet Protocol Security* (IPsec) is the supported VPN implementation for Fedora, and sufficiently addresses the usability needs of organizations with branch offices or remote users.

4.2.1.3. IPsec

Fedora supports IPsec for connecting remote hosts and networks to each other using a secure tunnel on a common carrier network such as the Internet. IPsec can be implemented using a host-to-host (one computer workstation to another) or network-to-network (one LAN/WAN to another) configuration.

The IPsec implementation in Fedora uses *Internet Key Exchange* (*IKE*), a protocol implemented by the Internet Engineering Task Force (IETF), used for mutual authentication and secure associations between connecting systems.

4.2.1.4. Creating an IPsec Connection

An IPsec connection is split into two logical phases. In phase 1, an IPsec node initializes the connection with the remote node or network. The remote node or network checks the requesting node's credentials and both parties negotiate the authentication method for the connection.

On Fedora systems, an IPsec connection uses the *pre-shared key* method of IPsec node authentication. In a pre-shared key IPsec connection, both hosts must use the same key in order to move to Phase 2 of the IPsec connection.

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

The Fedora implementation of IPsec uses IKE for sharing keys between hosts across the Internet. The **`racoon`** keying daemon handles the IKE key distribution and exchange. Refer to the **`racoon`** man page for more information about this daemon.

4.2.1.5. IPsec Installation

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

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

To configure IPsec on Fedora, you can use the **Network Administration Tool**, or manually edit the networking and IPsec configuration files.

- To connect two network-connected hosts via IPsec, refer to *Section 4.2.1.6, “IPsec Host-to-Host Configuration”*.
- To connect one LAN/WAN to another via IPsec, refer to *Section 4.2.1.7, “IPsec Network-to-Network Configuration”*.

4.2.1.6. IPsec Host-to-Host Configuration

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

4.2.1.6.1. Host-to-Host Connection

A host-to-host IPsec connection is an encrypted connection between two systems, both running IPsec with the same authentication key. With the IPsec connection active, any network traffic between the two hosts is encrypted.

To configure a host-to-host IPsec connection, use the following steps for each host:

Note

You should perform the following procedures on the actual machine that you are configuring. Avoid attempting to configure and establish IPsec connections remotely.

1. In a command shell, type **`system-config-network`** to start the **Network Administration Tool**.

2. On the **IPsec** tab, click **New** to start the IPsec configuration wizard.

3. Click **Forward** to start configuring a host-to-host IPsec connection.

4. Enter a unique name for the connection, for example, **`ipsec0`**. If required, select the check box to automatically activate the connection when the computer starts. Click **Forward** to continue.

5. Select **Host to Host encryption** as the connection type, and then click **Forward**.

6. Select the type of encryption to use: manual or automatic.

   If you select manual encryption, an encryption key must be provided later in the process. If you select automatic encryption, the **`racoon`** daemon manages the encryption key. The **`ipsec-tools`** package must be installed if you want to use automatic encryption.

   Click **Forward** to continue.

7. Enter the IP address of the remote host.

   To determine the IP address of the remote host, use the following command *on the remote host*:

   ```
   [root@myServer ~] # /sbin/ifconfig <device>
   ```

   where *<device>* is the Ethernet device that you want to use for the VPN connection.

   If only one Ethernet card exists in the system, the device name is typically eth0. The following example shows the relevant information from this command (note that this is an example output only):

   ```
   eth0      Link encap:Ethernet  HWaddr 00:0C:6E:E8:98:1D
   ```

```
          inet addr:172.16.44.192  Bcast:172.16.45.255  Mask:255.255.254.0
```

The IP address is the number following the **`inet addr:`** label.

Note

For host-to-host connections, both hosts should have a public, routable address. Alternatively, both hosts can have a private, non-routable address (for example, from the 10.x.x.x or 192.168.x.x ranges) as long as they are on the same LAN.

If the hosts are on different LANs, or one has a public address while the other has a private address, refer to *Section 4.2.1.7, "IPsec Network-to-Network Configuration"*.

Click **Forward** to continue.

8. If manual encryption was selected in step *6*, specify the encryption key to use, or click **Generate** to create one.

   a. Specify an authentication key or click **Generate** to generate one. It can be any combination of numbers and letters.

   b. Click **Forward** to continue.

9. Verify the information on the **IPsec — Summary** page, and then click **Apply**.

10. Click **File** => **Save** to save the configuration.

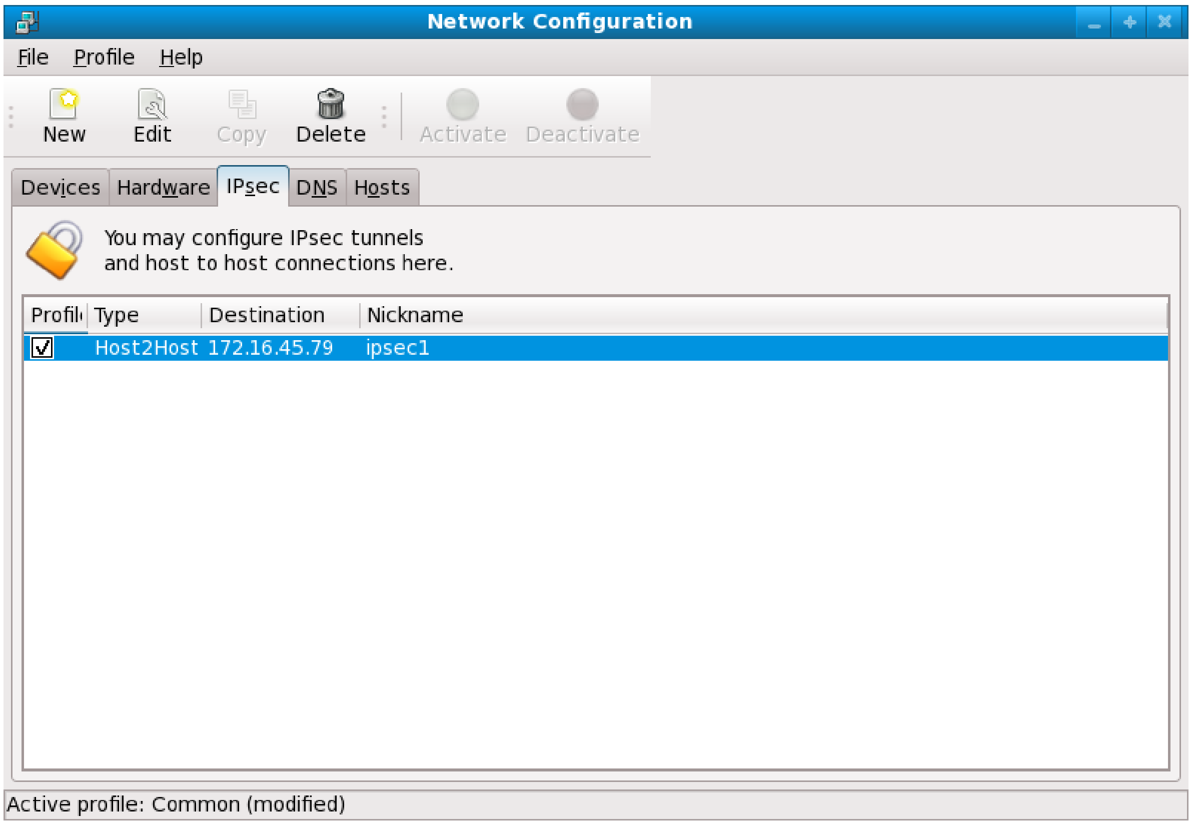
    You may need to restart the network for the changes to take effect. To restart the network, use the following command:

```
[root@myServer ~]# service network restart
```

11. Select the IPsec connection from the list and click the **Activate** button.

12. Repeat the entire procedure for the other host. It is essential that the same keys from step *8* be used on the other hosts. Otherwise, IPsec will not work.

After configuring the IPsec connection, it appears in the IPsec list as shown in *Figure 4.1, "IPsec Connection"*.


Network Configuration
File Profile Help
New Edit Copy Delete Activate Deactivate
Devices Hardware IPsec DNS Hosts

You may configure IPsec tunnels
and host to host connections here.

Profile	Type	Destination	Nickname
✓	Host2Host	172.16.45.79	ipsec1

Active profile: Common (modified)

Figure 4.1. IPsec Connection

The following files are created when the IPsec connection is configured:

- **`/etc/sysconfig/network-scripts/ifcfg-<nickname>`**
- **`/etc/sysconfig/network-scripts/keys-<nickname>`**
- **`/etc/racoon/<remote-ip>.conf`**
- **`/etc/racoon/psk.txt`**

If automatic encryption is selected, **`/etc/racoon/racoon.conf`** is also created.

When the interface is up, **`/etc/racoon/racoon.conf`** is modified to include **`<remote-ip>.conf`**.

4.2.1.6.2. Manual IPsec Host-to-Host Configuration

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

- The IP address of each host
- A unique name, for example, **`ipsec1`**. This is used to identify the IPsec connection and to distinguish it from other devices or connections.
- A fixed encryption key or one automatically generated by **`racoon`**.
- A pre-shared authentication key that is used during the initial stage of the connection and to exchange encryption keys during the session.

For example, suppose Workstation A and Workstation B want to connect to each other through an IPsec tunnel. They want to connect using a pre-shared key with the value of **`Key_Value01`**, and the users agree to let **racoon** automatically generate and share an authentication key between each host. Both host users decide to name their connections **ipsec1**.

Note

You should choose a PSK that uses a mixture of upper- and lower-case characters, numbers and punctuation. An easily-guessable PSK constitutes a security risk.

It is not necessary to use the same connection name for each host. You should choose a name that is convenient and meaningful for your installation.

The following is the IPsec configuration file for Workstation A for a host-to-host IPsec connection with Workstation B. The unique name to identify the connection in this example is *ipsec1*, so the resulting file is called **/etc/sysconfig/network-scripts/ifcfg-ipsec1**.

```
DST=X.X.X.XTYPE=IPSEC
ONBOOT=no
IKE_METHOD=PSK
```

For Workstation A, *X.X.X.X* is the IP address of Workstation B. For Workstation B, *X.X.X.X* is the IP address of Workstation A. This connection is not set to initiate on boot-up (**ONBOOT=no**) and it uses the pre-shared key method of authentication (**IKE_METHOD=PSK**).

The following is the content of the pre-shared key file (called **/etc/sysconfig/network-scripts/keys-ipsec1**) that both workstations need to authenticate each other. The contents of this file should be identical on both workstations, and only the root user should be able to read or write this file.

```
IKE_PSK=Key_Value01
```

Important

To change the **keys-ipsec1** file so that only the root user can read or edit the file, use the following command after creating the file:

```
[root@myServer ~] # chmod 600 /etc/sysconfig/network-scripts/keys-ipsec1
```

To change the authentication key at any time, edit the **keys-ipsec1** file on both workstations. *Both authentication keys must be identical for proper connectivity*.

The next example shows the specific configuration for the phase 1 connection to the remote host. The file is called ***X.X.X.X*.conf**, where *X.X.X.X* is the IP address of the remote IPsec host. Note that this file is automatically generated when the IPsec tunnel is activated and should not be edited directly.

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

The default phase 1 configuration file that is created when an IPsec connection is initialized contains the following statements used by the Fedora implementation of IPsec:

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

exchange_mode aggressive
: The default configuration for IPsec on Fedora uses an aggressive authentication mode, which lowers the connection overhead while allowing configuration of several IPsec connections with multiple hosts.

my_identifier address
: Specifies the identification method to use when authenticating nodes. Fedora uses IP addresses to identify nodes.

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

hash_algorithm sha1;
: Specifies the hash algorithm used during phase 1 negotiation between nodes. By default, Secure Hash Algorithm version 1 is used.

authentication_method pre_shared_key
: Specifies the authentication method used during node negotiation. By default, Fedora uses pre-shared keys for authentication.

dh_group 2
: Specifies the Diffie-Hellman group number for establishing dynamically-generated session keys. By default, modp1024 (group 2) is used.

4.2.1.6.2.1. The Racoon Configuration File

The **/etc/racoon/racoon.conf** files should be identical on all IPsec nodes *except* for the **include "/etc/racoon/*X.X.X.X*.conf"** statement. This statement (and the file it references) is generated when the IPsec tunnel is activated. For Workstation A, the *X.X.X.X* in the **include** statement is Workstation B's IP address. The opposite is true of Workstation B. The following shows a typical **racoon.conf** file when the IPsec connection is activated.

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

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

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

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

sainfo anonymous
: Denotes that SA can anonymously initialize with any peer provided that the IPsec credentials match.

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

lifetime time 1 hour
: This parameter specifies the lifetime of an SA and can be quantified either by time or by bytes of data. The default Fedora implementation of IPsec specifies a one hour lifetime.

encryption_algorithm 3des, blowfish 448, rijndael
: Specifies the supported encryption ciphers for phase 2. Fedora supports 3DES, 448-bit Blowfish, and Rijndael (the cipher used in the *Advanced Encryption Standard*, or AES).

authentication_algorithm hmac_sha1, hmac_md5
: Lists the supported hash algorithms for authentication. Supported modes are sha1 and md5 hashed message authentication codes (HMAC).

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

To start the connection, use the following command on each host:

```
[root@myServer ~]# /sbin/ifup <nickname>
```

where <nickname> is the name you specified for the IPsec connection.

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

```
[root@myServer ~]# tcpdump -n -i eth0 host <targetSystem>

IP 172.16.45.107 > 172.16.44.192: AH(spi=0x0954ccb6,seq=0xbb): ESP(spi=0x0c9f2164,seq=0xbb)
```

4.2.1.7. IPsec Network-to-Network Configuration

IPsec can also be configured to connect an entire network (such as a LAN or WAN) to a remote network using a network-to-network connection. A network-to-network connection requires the setup of IPsec routers on each side of the connecting networks to transparently process and route information from one node on a LAN to a node on a remote LAN. *Figure 4.2, “A network-to-network IPsec tunneled connection”* shows a network-to-network IPsec tunneled connection.

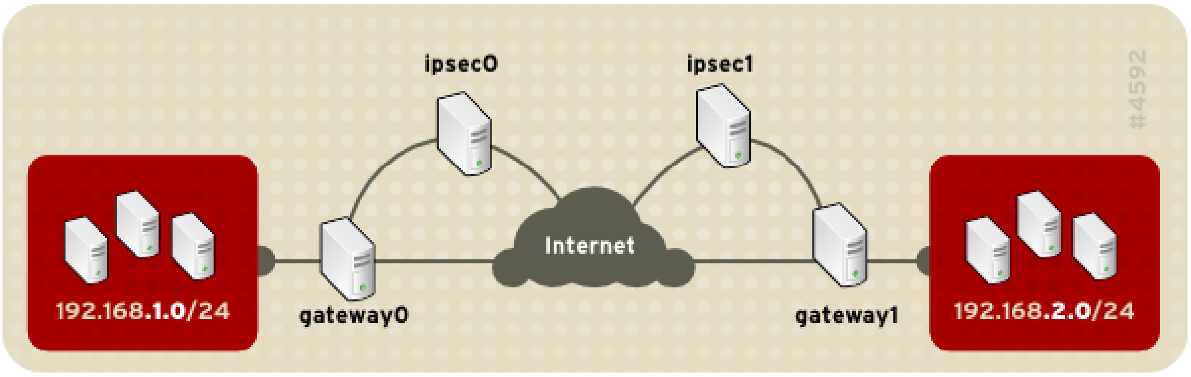


Figure 4.2. A network-to-network IPsec tunneled connection

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

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

- The externally-accessible IP addresses of the dedicated IPsec routers
- The network address ranges of the LAN/WAN served by the IPsec routers (such as 192.168.1.0/24 or 10.0.1.0/24)
- The IP addresses of the gateway devices that route the data from the network nodes to the Internet
- A unique name, for example, **`ipsec1`**. This is used to identify the IPsec connection and to distinguish it from other devices or connections.
- A fixed encryption key or one automatically generated by **`racoon`**
- A pre-shared authentication key that is used during the initial stage of the connection and to exchange encryption keys during the session.

4.2.1.7.1. Network-to-Network (VPN) Connection

A network-to-network IPsec connection uses two IPsec routers, one for each network, through which the network traffic for the private subnets is routed.

For example, as shown in *Figure 4.3, “Network-to-Network IPsec”*, if the 192.168.1.0/24 private network sends network traffic to the 192.168.2.0/24 private network, the packets go through gateway0, to ipsec0, through the Internet, to ipsec1, to gateway1, and to the 192.168.2.0/24 subnet.

IPsec routers require publicly addressable IP addresses and a second Ethernet device connected to their respective private networks. Traffic only travels through an IPsec router if it is intended for another IPsec router with which it has an encrypted connection.

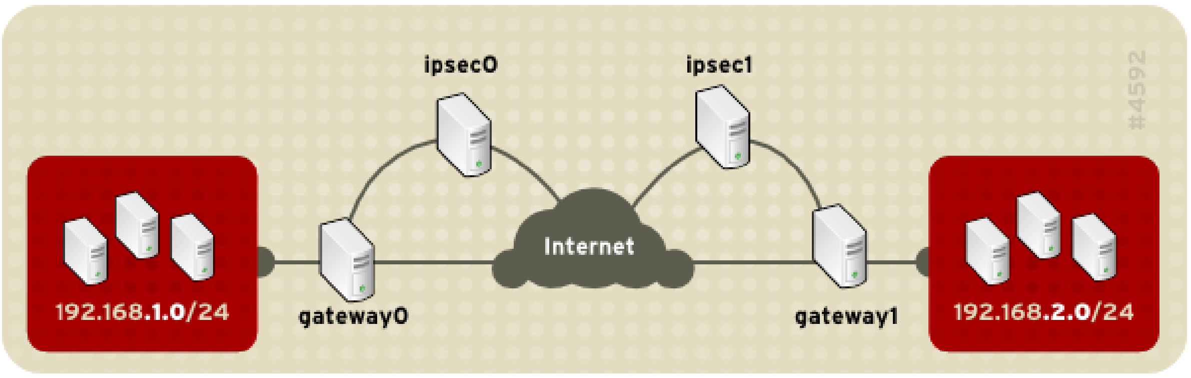


Figure 4.3. Network-to-Network IPsec

Alternate network configuration options include a firewall between each IP router and the Internet, and an intranet firewall between each IPsec router and subnet gateway. The IPsec router and the gateway for the subnet can be one system with two Ethernet devices: one with a public IP address that acts as the IPsec router; and one with a private IP address that acts as the gateway for the private subnet. Each IPsec router can use the gateway for its private network or a public gateway to send the packets to the other IPsec router.

Use the following procedure to configure a network-to-network IPsec connection:

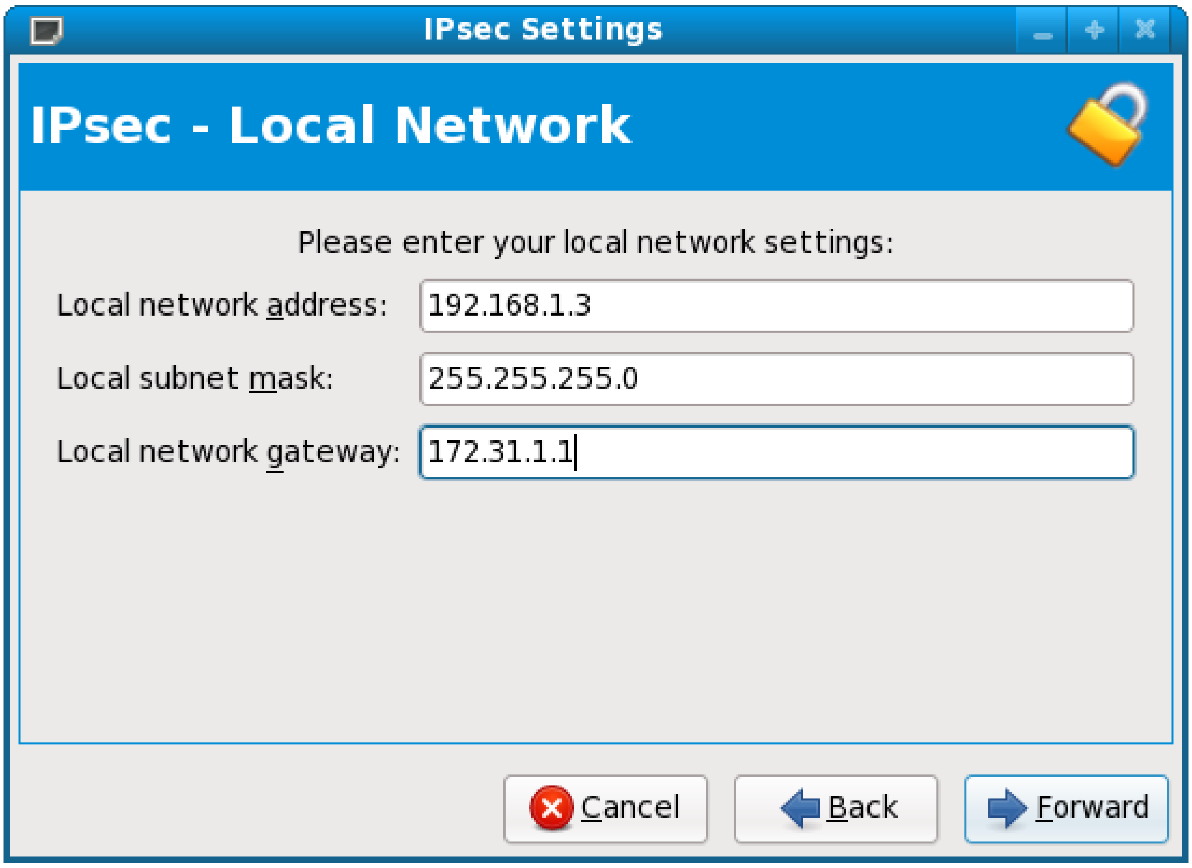
1. In a command shell, type **`system-config-network`** to start the **Network Administration Tool**.
2. On the **IPsec** tab, click **New** to start the IPsec configuration wizard.
3. Click **Forward** to start configuring a network-to-network IPsec connection.
4. Enter a unique nickname for the connection, for example, **`ipsec0`**. If required, select the check box to automatically activate the connection when the computer starts. Click **Forward** to continue.
5. Select **Network to Network encryption (VPN)** as the connection type, and then click **Forward**.
6. Select the type of encryption to use: manual or automatic.

   If you select manual encryption, an encryption key must be provided later in the process. If you select automatic encryption, the **`racoon`** daemon manages the encryption key. The **`ipsec-tools`** package must be installed if you want to use automatic encryption.

   Click **Forward** to continue.
7. On the **Local Network** page, enter the following information:

   - **Local Network Address** — The IP address of the device on the IPsec router connected to the private network.
   - **Local Subnet Mask** — The subnet mask of the local network IP address.
   - **Local Network Gateway** — The gateway for the private subnet.

   Click **Forward** to continue.


IPsec Settings

IPsec - Local Network

Please enter your local network settings:

Local network address: 192.168.1.3

Local subnet mask: 255.255.255.0

Local network gateway: 172.31.1.1

Figure 4.4. Local Network Information

8. On the **Remote Network** page, enter the following information:

   - **Remote IP Address** — The publicly addressable IP address of the IPsec router for the *other* private network. In our example, for ipsec0, enter the publicly addressable IP address of ipsec1, and vice versa.

   - **Remote Network Address** — The network address of the private subnet behind the *other* IPsec router. In our example, enter **`192.168.1.0`** if configuring ipsec1, and enter **`192.168.2.0`** if configuring ipsec0.

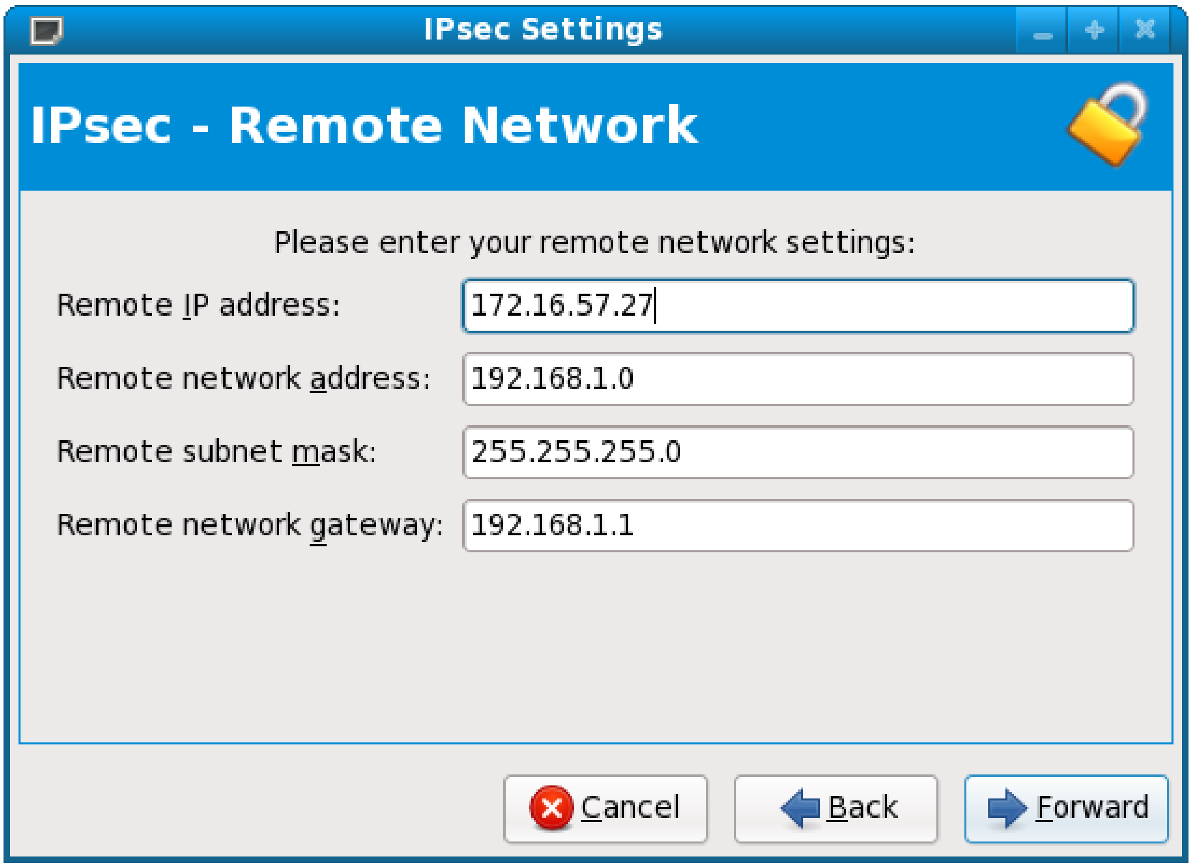
   - **Remote Subnet Mask** — The subnet mask of the remote IP address.

   - **Remote Network Gateway** — The IP address of the gateway for the remote network address.

   - If manual encryption was selected in step 6, specify the encryption key to use or click **Generate** to create one.

     Specify an authentication key or click **Generate** to generate one. This key can be any combination of numbers and letters.

   Click **Forward** to continue.



Please enter your remote network settings:

Remote IP address: 172.16.57.27

Remote network address: 192.168.1.0

Remote subnet mask: 255.255.255.0

Remote network gateway: 192.168.1.1

Figure 4.5. Remote Network Information

9. Verify the information on the **IPsec — Summary** page, and then click **Apply**.

10. Select **File** => **Save** to save the configuration.

11. Select the IPsec connection from the list, and then click **Activate** to activate the connection.

12. Enable IP forwarding:

    a. Edit **`/etc/sysctl.conf`** and set **`net.ipv4.ip_forward`** to **`1`**.

    b. Use the following command to enable the change:

    ```
    [root@myServer ~]# /sbin/sysctl -p /etc/sysctl.conf
    ```

The network script to activate the IPsec connection automatically creates network routes to send packets through the IPsec router if necessary.

4.2.1.7.2. Manual IPsec Network-to-Network Configuration

Suppose LAN A (lana.example.com) and LAN B (lanb.example.com) want to connect to each other through an IPsec tunnel. The network address for LAN A is in the 192.168.1.0/24 range, while LAN B uses the 192.168.2.0/24 range. The gateway IP address is 192.168.1.254 for LAN A and 192.168.2.254 for LAN B. The IPsec routers are separate from each LAN gateway and use two network devices: eth0 is assigned to an externally-accessible static IP address which accesses the Internet, while eth1 acts as a routing point to process and transmit LAN packets from one network node to the remote network nodes.

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

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

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

The following list describes the contents of this file:

TYPE=IPSEC
: Specifies the type of connection.

ONBOOT=yes
: Specifies that the connection should initiate on boot-up.

IKE_METHOD=PSK
: Specifies that the connection uses the pre-shared key method of authentication.

SRCGW=192.168.1.254
: The IP address of the source gateway. For LAN A, this is the LAN A gateway, and for LAN B, the LAN B gateway.

DSTGW=192.168.2.254
: The IP address of the destination gateway. For LAN A, this is the LAN B gateway, and for LAN B, the LAN A gateway.

SRCNET=192.168.1.0/24
: Specifies the source network for the IPsec connection, which in this example is the network range for LAN A.

DSTNET=192.168.2.0/24
: Specifies the destination network for the IPsec connection, which in this example is the network range for LAN B.

DST=X.X.X.X
: The externally-accessible IP address of LAN B.

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

```
IKE_PSK=r3dh4tl1nux
```

Important

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

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

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

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

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and entries.
path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

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

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

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

Prior to starting the IPsec connection, IP forwarding should be enabled in the kernel. To enable IP forwarding:

1. Edit **`/etc/sysctl.conf`** and set **`net.ipv4.ip_forward`** to **`1`**.

2. Use the following command to enable the change:

   ```
   [root@myServer ~] # sysctl -p /etc/sysctl.conf
   ```

To start the IPsec connection, use the following command on each router:

```
[root@myServer ~] # /sbin/ifup ipsec0
```

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

```
[root@myServer ~] # /sbin/ip route list
```

To test the IPsec connection, run the **`tcpdump`** utility on the externally-routable device (eth0 in this example) to view the network packets being transfered between the hosts (or networks), and verify that they are encrypted via IPsec. For example, to check the IPsec connectivity of LAN A, use the following command:

```
[root@myServer ~] # tcpdump -n -i eth0 host lana.example.com
```

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

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

4.2.1.8. Starting and Stopping an IPsec Connection

If the IPsec connection was not configured to activate on boot, you can control it from the command line.

To start the connection, use the following command on each host for host-to-host IPsec, or each IPsec router for network-to-network IPsec:

```
[root@myServer ~] # /sbin/ifup <nickname>
```

where <*nickname*> is the nickname configured earlier, such as **`ipsec0`**.

To stop the connection, use the following command:

```
[root@myServer ~] # /sbin/ifdown <nickname>
```

4.2.2. 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 simply starting the sshd service, the system will begin 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*[1] and traffic will be encrypted as it passes over that tunnel but using port forwarding is not as fluid as a VPN.

4.2.2.1. Cryptographic Logon

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 that 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. 2048-bits is probably okay but you can go up to, and possibly beyond, 8192-bit keys.

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 transfered 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 key from the authorized_key file.

4.2.3. 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.

4.2.3.1. LUKS Implementation in Fedora

Fedora 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 you hard drive, you will be prompted for a passphrase that will be asked every time you boot the computer. This passphrase

[1] http://www.redhatmagazine.com/2007/11/27/advanced-ssh-configuration-and-tunneling-we-dont-need-no-stinking-vpn-software

"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

Fedora's default implementation of LUKS is AES 128 with a SHA256 hashing. Ciphers that are available are:

- AES - Advanced Encryption Standard - *FIPS PUB 197*[2]
- Twofish (A 128-bit Block Cipher)
- Serpent
- cast5 - *RFC 2144*[3]
- cast6 - *RFC 2612*[4]

4.2.3.2. 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!

Note

This procedure uses *scrub* to destroy the existing data on the partition and provide a random base for LUKS to use. This random base is important to prevent certain attacks against the cryptography. *Scrub* is not installed by default and you will have to install it before use. Alternatively you may use another random number generator to accomplish the same thing.

If you want to manually encrypt a partition the following directions are for you. The below example demonstrates encrypting your /home partition but any partition can be used.

The following procedure will wipe all your existing data, so be sure to have a tested backup before you start. This also requires you to have a separate partition for /home (in my case that is /dev/VG00/LV_home). All the following must be done as root. Any of these steps failing means you must not continue until the step succeeded.

4.2.3.3. Step-by-Step Instructions

1. enter runlevel 1: **`telinit 1`**

[2] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[3] http://www.ietf.org/rfc/rfc2144.txt
[4] http://www.ietf.org/rfc/rfc2612.txt

2. Fill your partition with random data: **`scrub -p random /home`**
3. unmount your existing /home: **`umount /home`**
4. if it fails use **`fuser`** to find and kill processes hogging /home: **`fuser -mvk /home`**
5. verify /home is not mounted any longer: **`cat /proc/mounts | grep home`**
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. check it's there: **`ls -l /dev/mapper | grep home`**
9. create a filesystem: **`mkfs.ext3 /dev/mapper/home`**
10. mount it: **`mount /dev/mapper/home /home`**
11. check it's visible: **`df -h | grep home`**
12. add the following to /etc/crypttab: **`home /dev/VG00/LV_home none`**
13. edit your /etc/fstab, removing the old entry for /home and adding **`/dev/mapper/home /home ext3 defaults 1 2`**
14. verify your fstab entry: **`mount /home`**
15. restore default SELinux security contexts: **`/sbin/restorecon -v -R /home`**
16. reboot: **`shutdown -r now`**
17. The entry in /etc/crypttab makes your computer ask your **`luks`** passphrase on boot
18. Login as root and restore your backup

4.2.3.4. What you have just accomplished.

Congratulations, you now have an encrypted partition for all of your data to safely rest while the computer is off.

4.2.3.5. Links of Interest

For additional information on LUKS or encrypting hard drives please visit one of the following links:

- *LUKS - Linux Unified Key Setup*[5]
- *HOWTO: Creating an encrypted Physical Volume (PV) using a second hard drive, pvmove, and a Fedora LiveCD*[6]

4.2.4. 7-Zip Encrypted Archives

7-Zip[7] is a cross-platform, next generation, file compression tool that can also use strong encryption (AES-256) to protect the contents of the archive. This is extremely useful when you need to move data

[5] https://code.google.com/p/cryptsetup/
[6] https://bugzilla.redhat.com/attachment.cgi?id=161912
[7] http://www.7-zip.org/

between multiple computers that use varying operating systems (i.e. Linux at home, Windows at work) and you want a portable encryption solution.

4.2.4.1. 7-Zip Installation

7-Zip is not a base package in Fedora, but it is available in the software repository. Once installed, the package will update alongside the rest of the software on the computer with no special attention necessary.

4.2.4.2. Step-by-Step Installation Instructions

- Open a Terminal: **`Click Applications -> System Tools -> Terminal`** or in GNOME 3: **`Activities -> Applications -> Terminal`**
- Install 7-Zip with sudo access: **`sudo yum install p7zip`**
- Close the Terminal: **`exit`**

4.2.4.3. Step-by-Step Usage Instructions

By following these instructions you are going to compress and encrypt your "Documents" directory. Your original "Documents" directory will remain unaltered. This technique can be applied to any directory or file you have access to on the filesystem.

- Open a Terminal:**`Click Applications -> System Tools -> Terminal`**
- Compress and Encrypt: (enter a password when prompted) **`7za a -mhe=on -ms=on -p Documents.7z Documents/`**

The "Documents" directory is now compressed and encrypted. The following instructions will move the encrypted archive somewhere new and then extract it.

- Create a new directory: **`mkdir newplace`**
- Move the encrypted file: **`mv Documents.7z newplace`**
- Go to the new directory: **`cd newplace`**
- Extract the file: (enter the password when prompted) **`7za x Documents.7z`**

The archive is now extracted into the new location. The following instructions will clean up all the prior steps and restore your computer to its previous state.

- Go up a directory: **`cd ..`**
- Delete the test archive and test extraction: **`rm -r newplace`**
- Close the Terminal: **`exit`**

4.2.4.4. Creating a Secure 7-Zip Archive via the GUI

7-Zip archives can be extracted just like any other archive via the GUI, but creating a secure 7-Zip archive requires a few additional steps.

By following these instructions you are going to compress and encrypt your "Documents" directory. Your original "Documents" directory will remain unaltered. This technique can be applied to any directory or file you have access to on the filesystem.

- Open the file browser: Click Activities -> Files
- Right-Click on the "Documents" folder
- Select the "Compress" option
- Select ".7z" as the file extension
- Expand "Other Options"
- Check "Encrypt the file list too"
- Enter a password into the password field
- Click the "Create" button

You will now see a "Documents.7z" file appear in your home directory. If you try to open the file, you will be asked for the archive password before being shown the contents of the archive. The file will open once the correct password is supplied, and the archive can then be manipulated as usual. Deleting the "Documents.7z" file will conclude this exercise and return your computer to its previous state.

4.2.4.5. Things of note

7-Zip is not shipped by default with Microsoft Windows or Mac OS X. If you need to use your 7-Zip files on those platforms you will need to install the appropriate version of 7-Zip on those computers. See the 7-Zip *download page*[8].

4.2.5. Using GNU Privacy Guard (GnuPG)

GnuPG (GPG) is used to identify yourself and authenticate your communications, including those with people you don't 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.

GPG can also be used to sign and/or encrypt files kept on your computer or on a network drive. This can add additional protection in preventing a file from being altered or read by unauthorized people.

To utilize GPG for authentication or encryption of email you must first generate your public and private keys. After generating the keys you will have to setup your email client to utilize them.

4.2.5.1. Generating GPG Keys in GNOME

The Seahorse utility makes GPG key management easier. You can install *Seahorse* at the command line with the command **`su -c "yum install seahorse"`** or in the GUI using **Add/Remove Software**.

To create a key select **Passwords and Keys**, which starts the application **Seahorse**. From the **`File`** menu select **`New`** then **`PGP Key`** then select **`Continue`**. Type your full name, email address, and an optional comment describing who are you (e.g.: John C. Smith, jsmith@example.com, The Man). Select **`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.

[8] http://www.7-zip.org/download.html

Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

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, you should prepend "0x" to the key ID, as in "0x6789ABCD". You should make a backup of your private key and store it somewhere secure.

4.2.5.2. Generating GPG Keys in KDE

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. 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. The next dialog box prompts you for your passphrase. At this point, your key appears in the main **`KGpg`** window.

Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

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, you should prepend "0x" to the key ID, as in "0x6789ABCD". You should make a backup of your private key and store it somewhere secure.

4.2.5.3. Generating GPG Keys Using the Command Line

Use the following shell command: **`gpg --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.

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. A RSA key allows you not only to sign communications, but also to encrypt files.

Next, choose the key size:

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

Again, the default is sufficient for almost all users, and represents a strong level of security.

Next, 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.)

Before the **gpg**code> program asks for signature information, the following prompt appears: **`Is this correct (y/n)?`** Enter **y**code> to finish the process.

Next, enter your name and email address. Remember this process is about authenticating you as a real individual. For this reason, include your real name. Do not use aliases or handles, since these disguise or obfuscate your identity.

Enter your real email address for your GPG key. 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 [[DocsProject/SelfIntroduction| 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.")

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 **gpg** program asks you to enter your passphrase twice to ensure you made no typing errors.

Finally, **gpg** 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]
```

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: **`gpg --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, you should prepend "0x" to the key ID, as in "0x1B2AFA1C".

Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

4.2.5.4. Using GPG with Alpine

If you are using the email client *Alpine* or *Pine* then you will also need to download and install *ez-pine-gpg*. This software is currently available from *http://business-php.com/opensource/ez-pine-gpg/*. Once you have installed ez-pine-gpg you will need to modify your **`~/.pinerc`** file. You need to:

1. /home/username/bin should be replaced with the installation path that you specified.

2. In two places, the gpg-identifier after _RECIPIENTS_ should be replaced with your GPG public key's identifier. The reason you include your own GPG identifier here is so that if you send an encrypted message to "Alice", that message is also encrypted with your public key -- if you don't do this, then you will not be able to open that message in your sent-mail folder and remind yourself of what you wrote.

It should look something like this:

```
# This variable takes a list of programs that message text is piped into
# after MIME decoding, prior to display.
display-filters=_LEADING("-----BEGIN PGP")_ /home/max/bin/ez-pine-gpg-incoming

# This defines a program that message text is piped into before MIME
# encoding, prior to sending
sending-filters=/home/max/bin/ez-pine-gpg-sign _INCLUDEALLHDRS_,
    /home/username/bin/ez-pine-gpg-encrypt _RECIPIENTS_ gpg-identifier,
    /home/username/bin/ez-pine-gpg-sign-and-encrypt _INCLUDEALLHDRS_ _RECIPIENTS_ gpg-
identifier
```

4.2.5.5. Using GPG with Evolution

4.2.5.5.1. Configuring GPG for use with Evolution

To configure GPG for use in **Evolution** select from the **Evolution** Main Menu, select Tools, Settings... In the left pane, select Mail Accounts. In the right pane, select the email account you use for correspondence. Then select the Edit button. The **Evolution** Account Editor dialog appears. Select the Security tab.

In the PGP/GPG Key ID field, enter the GPG key ID matching this account's email address. If you are not sure what your key ID is, use this command: **`gpg --fingerprint EMAIL_ADDRESS`**. The key ID is the same as the last eight characters (4 bytes) of the key fingerprint. It is a good idea to click the option Always encrypt to myself when sending encrypted mail. You may also want to select Always sign outgoing messages when using this account.

Notice

If you do not mark public keys as trusted in your keyring, you will not be able to encrypt email to their owners unless you select the option Always trust keys in my keyring when encrypting. You will instead receive a dialog indicating that a trust check has failed.

4.2.5.5.2. Verifying email with Evolution

Evolution will automatically check any incoming GPG-signed messages for validity. If Evolution cannot GPG verify a message due to a missing public key (or tampering), it will end with a red banner. If the message is verified but you have not signed the key either locally or globally, the banner will be yellow. If the message is verified and you have signed the key, the banner will be green. When you click the seal icon, Evolution displays a dialog with more security information about the signature. To add a public key to your keyring, use the search function along with the key owner's email address: **`gpg --keyserver pgp.mit.edu --search email address`**. To import the correct key, you may need to match the key ID with the information provided by Evolution.

4.2.5.5.3. Signing and Encrypting email with Evolution

Signing email allows the recipients to verify that the email actually came from you.

While composing your email, choose the Security menu, and then select PGP Sign to sign your message. To encrypt your message, select PGP Encrypt. You may sign an encrypted message as well, which is good practice. When you send the message, Evolution will ask you to enter your GPG key passphrase. (After three unsuccessful attempts Evolution generates an error.) If you select the option Remember this password for the remainder of this session, you will not need to use your passphrase again to sign or decrypt, unless you quit and restart Evolution.

4.2.5.6. Using GPG with Thunderbird

Fedora includes Mozilla Thunderbird in the thunderbird package, and the mozilla-mail package for the Mozilla Suite email application. Thunderbird is the recommended Mozilla email application. This appears on your desktop as Applications > Internet > Thunderbird Email.

Mozilla products support extensions, plugins that add new features to the main application. The Enigmail extensions provide GPG support to email products from Mozilla. Versions of Enigmail exist for both Mozilla Thunderbird, and the Mozilla Suite (Seamonkey). Netscape software from AOL is based on the Mozilla products, and may also use this extension.

To install Enigmail on Fedora systems, follow the instructions given below.

Enigmail uses the term OpenPGP in menu items and options. GPG is an implementation of OpenPGP, and you may treat the terms as equivalent.

The homepage for Enigmail is: *http://enigmail.mozdev.org/download.html*.

This page provides screenshots of Enigmail and GPG in action: *http://enigmail.mozdev.org/screenshots.html*.

4.2.5.6.1. Installing Enigmail

Enigmail is now available in fedora repository. It can be installed by typing: **`yum install thunderbird-enigmail`** at a command line. Alternatively, you can install *thunderbird-enigmail* using by going to **`System -> Administration -> Add/Remove Software`**.

4.2.5.7. About Public Key Encryption

1. *Wikipedia - Public Key Cryptography*[9]

2. *HowStuffWorks - Encryption*[10]

[9] http://en.wikipedia.org/wiki/Public-key_cryptography
[10] http://computer.howstuffworks.com/encryption.htm

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 (GnuPG) 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`**.

Secure Installation

Security begins with the first time you put that CD or DVD into your disk drive to install Fedora. Configuring your system securely from the beginning makes it easier to implement additional security settings later.

6.1. Disk Partitions

The NSA recommends creating separate partitions for /boot, /, /home, /tmp, and /var/tmp. 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 Fedora 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 Fedora it is a lot easier when you can keep your data in the /home partition as it will not be overwritten during installation. 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.

/tmp and /var/tmp - Both the /tmp and the /var/tmp directories are used to store data that doesn't 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.

6.2. Utilize LUKS Partition Encryption

The implementation of *Linux Unified Key Setup-on-disk-format*[1](LUKS) encryption has become a lot easier in recent years. During the installation process an option to encrypt your partitions will be presented to the user. The user must supply a passphrase that will be the key to unlock the bulk encryption key that will be used to secure the partition's data.

[1] http://fedoraproject.org/wiki/Security_Guide/9/LUKSDiskEncryption

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.

7.1. Install Minimal Software

It is best 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.

7.2. 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 schedule 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.

7.3. Adjusting Automatic Updates

Fedora is configured to apply all updates on a daily schedule. If you want to change the 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`**.

7.4. 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.

Common Vulnerabilities and Exposures

The Common Vulnerabilities and Exposures or CVE system provides a reference method for publicly-known information security vulnerabilities and exposures. ITRE Corporation maintains the system, with funding from the National Cyber Security Division of the United States Department of Homeland Security.

MITRE Corporation assigns a CVE identifier to every vulnerability or exposure. The CVE is used to track the vulnerability through different pieces of software, as a single CVE can affect multiple software packages and multiple vendors.

8.1. YUM Plugin

The *yum-plugin-security* package is a feature of Fedora. If installed, the yum module provided by this package can be used to limit yum to retrieve only security-related updates. It can also be used to provide information about which Red Hat advisory, which bug in Red Hat's Bugzilla database, or which CVE number from MITRE's Common Vulnerabilities and Exposures directory is addressed by a package update.

Enabling these features is as simple as running the **`yum install yum-plugin-security`** command.

Chapter 9.

SELinux

9.1. Introduction

Security-Enhanced Linux (SELinux) is an implementation of a *mandatory access control* mechanism in the Linux kernel, checking for allowed operations after standard *discretionary access controls* are checked. It was created by the National Security Agency and can enforce rules on files and processes in a Linux system, and on their actions, based on defined policy.

When using SELinux, files, including directories and devices, are referred to as objects. Processes, such as a user running a command or the Mozilla® Firefox® application, are referred to as subjects. Most operating systems use a Discretionary Access Control (DAC) system that controls how subjects interact with objects, and how subjects interact with each other. On operating systems using DAC, users control the permissions of files (objects) that they own. For example, on Linux® operating systems, users could make their home directories world-readable, giving users and processes (subjects) access to potentially sensitive information, with no further protection over this unwanted action.

Relying on DAC mechanisms alone is fundamentally inadequate for strong system security. DAC access decisions are only based on user identity and ownership, ignoring other security-relevant information such as the role of the user, the function and trustworthiness of the program, and the sensitivity and integrity of the data. Each user has complete discretion over their files, making it impossible to enforce a system-wide security policy. Furthermore, every program run by a user inherits all of the permissions granted to the user and is free to change access to the user's files, so no protection is provided against malicious software. Many system services and privileged programs must run with coarse-grained privileges that far exceed their requirements, so that a flaw in any one of these programs could be exploited to obtain further system access.[1]

The following is an example of permissions used on Linux operating systems that do not run Security-Enhanced Linux (SELinux). The permissions and output in these examples may differ from your system. Use the **`ls -l`** command to view file permissions:

```
$ ls -l file1
-rw-rw-r--. 1 user1 group1 0 May 11 10:46 file1
```

The first three permission bits, **`rw`**, control the access the Linux **`user1`** user (in this case, the owner) has to **`file1`**. The next three permission bits, **`rw-`**, control the access the Linux **`group1`** group has to **`file1`**. The last three permission bits, **`r--`**, control the access everyone else has to **`file1`**, which includes all users and processes.

Security-Enhanced Linux (SELinux) adds Mandatory Access Control (MAC) to the Linux kernel, and is enabled by default in Fedora. A general purpose MAC architecture needs the ability to enforce an administratively-set security policy over all processes and files in the system, basing decisions on labels containing a variety of security-relevant information. When properly implemented, it enables a system to adequately defend itself and offers critical support for application security by protecting against the tampering with, and bypassing of, secured applications. MAC provides strong separation of applications that permits the safe execution of untrustworthy applications. Its ability to limit the privileges associated with executing processes limits the scope of potential damage that can result

[1] "Integrating Flexible Support for Security Policies into the Linux Operating System", by Peter Loscocco and Stephen Smalley. This paper was originally prepared for the National Security Agency and is, consequently, in the public domain. Refer to the *original paper* [http://www.nsa.gov/research/_files/selinux/papers/freenix01/index.shtml] for details and the document as it was first released. Any edits and changes were done by Murray McAllister.

from the exploitation of vulnerabilities in applications and system services. MAC enables information to be protected from legitimate users with limited authorization as well as from authorized users who have unwittingly executed malicious applications.[2]

The following is an example of the labels containing security-relevant information that are used on processes, Linux users, and files, on Linux operating systems that run SELinux. This information is called the SELinux *context*, and is viewed using the **`ls -Z`** command:

```
$ ls -Z file1
-rw-rw-r--. user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

In this example, SELinux provides a user (**`unconfined_u`**), a role (**`object_r`**), a type (**`user_home_t`**), and a level (**`s0`**). This information is used to make access control decisions. With DAC, access is controlled based only on Linux user and group IDs. It is important to remember that SELinux policy rules are checked *after* DAC rules. SELinux policy rules are not used if DAC rules deny access first.

Linux and SELinux Users

On Linux operating systems that run SELinux, there are Linux users as well as SELinux users. SELinux users are part of SELinux policy. Linux users are mapped to SELinux users. To avoid confusion, this guide uses "Linux user" and "SELinux user" to differentiate between the two.

9.1.1. Benefits of running SELinux

- All processes and files are labeled with a type. A type defines a domain for processes, and a type for files. Processes are separated from each other by running in their own domains, and SELinux policy rules define how processes interact with files, as well as how processes interact with each other. Access is only allowed if an SELinux policy rule exists that specifically allows it.
- Fine-grained access control. Stepping beyond traditional UNIX® permissions that are controlled at user discretion and based on Linux user and group IDs, SELinux access decisions are based on all available information, such as an SELinux user, role, type, and, optionally, a level.
- SELinux policy is administratively-defined, enforced system-wide, and is not set at user discretion.
- Reduced vulnerability to privilege escalation attacks. One example: since processes run in domains, and are therefore separated from each other, and because SELinux policy rules define how processes access files and other processes, if a process is compromised, the attacker only has access to the normal functions of that process, and to files the process has been configured to have access to. For example, if the Apache HTTP Server is compromised, an attacker can not use that process to read files in user home directories, unless a specific SELinux policy rule was added or configured to allow such access.
- Confined services. SELinux ships with the ability to confine services and daemons so that they are more predictable and are only allowed access that is required for their normal operation.
- SELinux can be used to enforce data confidentiality and integrity, as well as protecting processes from untrusted inputs.

[2] "Meeting Critical Security Objectives with Security-Enhanced Linux", by Peter Loscocco and Stephen Smalley. This paper was originally prepared for the National Security Agency and is, consequently, in the public domain. Refer to the *original paper* [http://www.nsa.gov/research/_files/selinux/papers/ottawa01/index.shtml] for details and the document as it was first released. Any edits and changes were done by Murray McAllister.

SELinux is not:

- antivirus software.
- a replacement for passwords, firewalls, or other security systems.
- an all-in-one security solution.

SELinux is designed to enhance existing security solutions, not replace them. Even when running SELinux, it is important to continue to follow good security practices, such as keeping software up-to-date, using hard-to-guess passwords, firewalls, and so on.

9.1.2. Examples

The following examples demonstrate how SELinux increases security:

- The default action is deny. If a specific SELinux policy rule does not exist to allow access, such as for a process opening a file, access is denied.
- SELinux can confine Linux users. A number of confined SELinux users exist in SELinux policy. Linux users can be mapped to confined SELinux users to take advantage of the security rules and mechanisms applied to them. For example, mapping a Linux user to the SELinux user_u user, results in a Linux user that is not able to run (unless configured otherwise) set user ID (setuid) applications, such as **sudo** and **su**, as well as preventing them from executing files and applications in their home directory - if configured, this prevents users from executing malicious files from their home directories.
- Process separation is used. Processes run in their own domains, preventing processes from accessing files used by other processes, as well as preventing processes from accessing other processes. For example, when running SELinux, unless otherwise configured, an attacker can not compromise a Samba server, and then use that Samba server as an attack vector to read and write to files used by other processes, such as databases used by MySQL®.
- SELinux helps limit the damage made by configuration mistakes. *Domain Name System (DNS)*[3] servers often replicate information between each other in what is known as a zone transfer. Attackers can use zone transfers to update DNS servers with false information. When running the *Berkeley Internet Name Daemon (BIND)*[4] as a DNS server in Fedora, even if an administrator forgets to limit which servers can perform a zone transfer, the default SELinux policy prevents zone files [5] from being updated via zone transfers, by the BIND `named` daemon itself, and by other processes.
- Refer to the *Red Hat® Magazine*[6] article, *Risk report: Three years of Red Hat Enterprise Linux 4*[7][8], for exploits that were restricted due to the default SELinux targeted policy in Red Hat® Enterprise Linux® 4.

[3] http://en.wikipedia.org/wiki/Domain_Name_System
[4] https://www.isc.org/software/bind
[5] Text files that include information, such as hostname to IP address mappings, that are used by DNS servers.
[6] http://www.redhatmagazine.com/
[7] http://www.redhatmagazine.com/2008/02/26/risk-report-three-years-of-red-hat-enterprise-linux-4/
[8] Cox, Mark. "Risk report: Three years of Red Hat Enterprise Linux 4". Published 26 February 2008. Accessed 27 August 2009: *http://www.redhatmagazine.com/2008/02/26/risk-report-three-years-of-red-hat-enterprise-linux-4/*.

- Refer to the *LinuxWorld.com*[9] article, *A seatbelt for server software: SELinux blocks real-world exploits*[10][11], for background information about SELinux, and information about various exploits that SELinux has prevented.
- Refer to James Morris's *SELinux mitigates remote root vulnerability in OpenPegasus*[12] blog post for information about an exploit in *OpenPegasus*[13] that was mitigated by SELinux as shipped with Red Hat Enterprise Linux 4 and 5.

The *Tresys Technology*[14] website has an *SELinux Mitigation News*[15] section (on the right-hand side), that lists recent exploits that have been mitigated or prevented by SELinux.

9.1.3. SELinux Architecture

SELinux is a Linux security module that is built into the Linux kernel. SELinux is driven by loadable policy rules. When security-relevant access is taking place, such as when a process attempts to open a file, the operation is intercepted in the kernel by SELinux. If an SELinux policy rule allows the operation, it continues, otherwise, the operation is blocked and the process receives an error.

SELinux decisions, such as allowing or disallowing access, are cached. This cache is known as the Access Vector Cache (AVC). Caching decisions decreases how often SELinux policy rules need to be checked, which increases performance. Remember that SELinux policy rules have no effect if DAC rules deny access first.

9.1.4. SELinux on Other Operating Systems

Refer to the following for information about running SELinux on operating systems:

- Hardened Gentoo: *http://www.gentoo.org/proj/en/hardened/selinux/selinux-handbook.xml*.
- Debian: *http://wiki.debian.org/SELinux*.
- Ubuntu: *https://wiki.ubuntu.com/SELinux* and *https://help.ubuntu.com/community/SELinux*.
- Red Hat Enterprise Linux: *Red Hat Enterprise Linux Deployment Guide*[16] and *Red Hat Enterprise Linux 4 SELinux Guide*[17].
- Fedora: *http://fedoraproject.org/wiki/SELinux* and the *Fedora Core 5 SELinux FAQ*[18].

9.2. SELinux Contexts

Processes and files are labeled with an SELinux context that contains additional information, such as an SELinux user, role, type, and, optionally, a level. When running SELinux, all of this information is used to make access control decisions. In Fedora, SELinux provides a combination of Role-Based Access Control (RBAC), Type Enforcement® (TE), and, optionally, Multi-Level Security (MLS).

[9] http://www.linuxworld.com
[10] http://www.linuxworld.com/news/2008/022408-selinux.html?page=1
[11] Marti, Don. "A seatbelt for server software: SELinux blocks real-world exploits". Published 24 February 2008. Accessed 27 August 2009: *http://www.linuxworld.com/news/2008/022408-selinux.html?page=1*.
[12] http://james-morris.livejournal.com/25421.html
[13] http://www.openpegasus.org/
[14] http://www.tresys.com/
[15] http://www.tresys.com/innovation.php
[16] http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/html/Deployment_Guide/selg-overview.html
[17] http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide/
[18] http://docs.fedoraproject.org/selinux-faq-fc5/

The following is an example showing SELinux context. SELinux contexts are used on processes, Linux users, and files, on Linux operating systems that run SELinux. Use the **`ls -Z`** command to view the SELinux context of files and directories:

```
$ ls -Z file1
-rw-rw-r--. user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

SELinux contexts follow the *SELinux user:role:type:level* syntax:

SELinux user

The SELinux user identity is an identity known to the policy that is authorized for a specific set of roles, and for a specific MLS range. Each Linux user is mapped to an SELinux user via SELinux policy. This allows Linux users to inherit the restrictions placed on SELinux users. The mapped SELinux user identity is used in the SELinux context for processes in that session, in order to define what roles and levels they can enter. Run the **`semanage login -l`** command as the Linux root user to view a list of mappings between SELinux and Linux user accounts:

```

# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

Output may differ slightly from system to system. The **`Login Name`** column lists Linux users, and the **`SELinux User`** column lists which SELinux user the Linux user is mapped to. For processes, the SELinux user limits which roles and levels are accessible. The last column, **`MLS/MCS Range`**, is the level used by Multi-Level Security (MLS) and Multi-Category Security (MCS). Levels are briefly discussed later.

role

Part of SELinux is the Role-Based Access Control (RBAC) security model. The role is an attribute of RBAC. SELinux users are authorized for roles, and roles are authorized for domains. The role serves as an intermediary between domains and SELinux users. The roles that can be entered determine which domains can be entered - ultimately, this controls which object types can be accessed. This helps reduce vulnerability to privilege escalation attacks.

type

The type is an attribute of Type Enforcement. The type defines a domain for processes, and a type for files. SELinux policy rules define how types can access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

level

The level is an attribute of MLS and Multi-Category Security (MCS). An MLS range is a pair of levels, written as *lowlevel-highlevel* if the levels differ, or *lowlevel* if the levels are identical (**`s0-s0`** is the same as **`s0`**). Each level is a sensitivity-category pair, with categories being optional. If there are categories, the level is written as *sensitivity:category-set*. If there are no categories, it is written as *sensitivity*.

If the category set is a contiguous series, it can be abbreviated. For example, **`c0.c3`** is the same as **`c0,c1,c2,c3`**. The **`/etc/selinux/targeted/setrans.conf`** file maps levels (**`s0:c0`**) to human-readable form (ie. **`CompanyConfidential`**). Do not edit **`setrans.conf`** with a text editor: use **`semanage`** to make changes. Refer to the semanage(8) manual page for further information. In Fedora, targeted policy enforces MCS, and in MCS, there is just one sensitivity, **`s0`**. MCS in Fedora supports 1024 different categories: **`c0`** through to **`c1023`**. **`s0-s0:c0.c1023`** is sensitivity **`s0`** and authorized for all categories.

MLS enforces the *Bell-La Padula Mandatory Access Model*[19], and is used in Labeled Security Protection Profile (LSPP) environments. To use MLS restrictions, install the *selinux-policy-mls* package, and configure MLS to be the default SELinux policy via the **`/etc/selinux/config`** file. The MLS policy shipped with Fedora omits many program domains that were not part of the evaluated configuration, and therefore, MLS on a desktop workstation is unusable (no support for the X Window System); however, an MLS policy from the *upstream SELinux Reference Policy*[20] can be built that includes all program domains.

9.2.1. Domain Transitions

A process in one domain transitions to another domain by executing an application that has the **`entrypoint`** type for the new domain. The **`entrypoint`** permission is used in SELinux policy, and controls which applications can be used to enter a domain. The following example demonstrates a domain transition:

1. A user wants to change their password. To do this, they run the **`passwd`** application. The **`/usr/bin/passwd`** executable is labeled with the **`passwd_exec_t`** type:

   ```
   $ ls -Z /usr/bin/passwd
   -rwsr-xr-x  root root system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd
   ```

   The **passwd** application accesses **`/etc/shadow`**, which is labeled with the **`shadow_t`** type:

   ```
   $ ls -Z /etc/shadow
   ----------. root root system_u:object_r:shadow_t:s0    /etc/shadow
   ```

2. An SELinux policy rule states that processes running in the **`passwd_t`** domain are allowed to read and write to files labeled with the **`shadow_t`** type. The **`shadow_t`** type is only applied to files that are required for a password change. This includes **`/etc/gshadow`**, **`/etc/shadow`**, and their backup files.

3. An SELinux policy rule states that the **`passwd_t`** domain has **`entrypoint`** permission to the **`passwd_exec_t`** type.

4. When a user runs the **`/usr/bin/passwd`** application, the user's shell process transitions to the **`passwd_t`** domain. With SELinux, since the default action is to deny, and a rule exists that allows (among other things) applications running in the **`passwd_t`** domain to access files labeled with the **`shadow_t`** type, the **passwd** application is allowed to access **`/etc/shadow`**, and update the user's password.

This example is not exhaustive, and is used as a basic example to explain domain transition. Although there is an actual rule that allows subjects running in the **`passwd_t`** domain to access objects labeled with the **`shadow_t`** file type, other SELinux policy rules must be met before the subject can transition to a new domain. In this example, Type Enforcement ensures:

[19] http://en.wikipedia.org/wiki/Bell-LaPadula_model
[20] http://oss.tresys.com/projects/refpolicy

- the **passwd_t** domain can only be entered by executing an application labeled with the **passwd_exec_t** type; can only execute from authorized shared libraries, such as the **lib_t** type; and can not execute any other applications.
- only authorized domains, such as **passwd_t**, can write to files labeled with the **shadow_t** type. Even if other processes are running with superuser privileges, those processes can not write to files labeled with the **shadow_t** type, as they are not running in the **passwd_t** domain.
- only authorized domains can transition to the **passwd_t** domain. For example, the sendmail process running in the **sendmail_t** domain does not have a legitimate reason to execute **passwd**; therefore, it can never transition to the **passwd_t** domain.
- processes running in the **passwd_t** domain can only read and write to authorized types, such as files labeled with the **etc_t** or **shadow_t** types. This prevents the **passwd** application from being tricked into reading or writing arbitrary files.

9.2.2. SELinux Contexts for Processes

Use the **ps -eZ** command to view the SELinux context for processes. For example:

1. Open a terminal, such as **Applications** → **System Tools** → **Terminal**.
2. Run the **/usr/bin/passwd** command. Do not enter a new password.
3. Open a new tab, or another terminal, and run the **ps -eZ | grep passwd** command. The output is similar to the following:

   ```
   unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 13212 pts/1 00:00:00 passwd
   ```

4. In the first tab/terminal, press **Ctrl+C** to cancel the **passwd** application.

In this example, when the **/usr/bin/passwd** application (labeled with the **passwd_exec_t** type) is executed, the user's shell process transitions to the **passwd_t** domain. Remember: the type defines a domain for processes, and a type for files.

Use the **ps -eZ** command to view the SELinux contexts for running processes. The following is a limited example of the output, and may differ on your system:

```

system_u:system_r:dhcpc_t:s0     1869 ?        00:00:00 dhclient
system_u:system_r:sshd_t:s0-s0:c0.c1023 1882 ? 00:00:00 sshd
system_u:system_r:gpm_t:s0       1964 ?        00:00:00 gpm
system_u:system_r:crond_t:s0-s0:c0.c1023 1973 ? 00:00:00 crond
system_u:system_r:kerneloops_t:s0 1983 ?       00:00:05 kerneloops
system_u:system_r:crond_t:s0-s0:c0.c1023 1991 ? 00:00:00 atd
```

The **system_r** role is used for system processes, such as daemons. Type Enforcement then separates each domain.

9.2.3. SELinux Contexts for Users

Use the **id -Z** command to view the SELinux context associated with your Linux user:

```
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

In Fedora, Linux users run unconfined by default. This SELinux context shows that the Linux user is mapped to the SELinux **unconfined_u** user, running as the **unconfined_r** role, and is running in the **`unconfined_t`** domain. **`s0-s0`** is an MLS range, which in this case, is the same as just **`s0`**. The categories the user has access to is defined by **`c0.c1023`**, which is all categories (**`c0`** through to **`c1023`**).

9.3. Targeted Policy

Targeted policy is the default SELinux policy used in Fedora. When using targeted policy, processes that are targeted run in a confined domain, and processes that are not targeted run in an unconfined domain. For example, by default, logged in users run in the **`unconfined_t`** domain, and system processes started by init run in the **`initrc_t`** domain - both of these domains are unconfined.

Unconfined domains (as well as confined domains) are subject to executable and writeable memory checks. By default, subjects running in an unconfined domain can not allocate writeable memory and execute it. This reduces vulnerability to *buffer overflow attacks*[21]. These memory checks are disabled by setting Booleans, which allow the SELinux policy to be modified at runtime. Boolean configuration is discussed later.

9.3.1. Confined Processes

Almost every service that listens on a network is confined in Fedora. Also, most processes that run as the Linux root user and perform tasks for users, such as the **passwd** application, are confined. When a process is confined, it runs in its own domain, such as the `httpd` process running in the **`httpd_t`** domain. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited.

The following example demonstrates how SELinux prevents the Apache HTTP Server (`httpd`) from reading files that are not correctly labeled, such as files intended for use by Samba. This is an example, and should not be used in production. It assumes that the *httpd*, *wget*, *setroubleshoot-server*, *dbus* and *audit* packages are installed, that the SELinux targeted policy is used, and that SELinux is running in enforcing mode:

1. Run the **`sestatus`** command to confirm that SELinux is enabled, is running in enforcing mode, and that targeted policy is being used:

   ```
   $ /usr/sbin/sestatus
   SELinux status:                 enabled
   SELinuxfs mount:                /selinux
   Current mode:                   enforcing
   Mode from config file:          enforcing
   Policy version:                 24
   Policy from config file:        targeted
   ```

   `SELinux status: enabled` is returned when SELinux is enabled. **`Current mode: enforcing`** is returned when SELinux is running in enforcing mode. **`Policy from config file: targeted`** is returned when the SELinux targeted policy is used.

2. As the Linux root user, run the **`touch /var/www/html/testfile`** command to create a file.

3. Run the **`ls -Z /var/www/html/testfile`** command to view the SELinux context:

   ```
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/testfile
   ```

[21] http://en.wikipedia.org/wiki/Buffer_overflow

By default, Linux users run unconfined in Fedora, which is why the **`testfile`** file is labeled with the SELinux **`unconfined_u`** user. RBAC is used for processes, not files. Roles do not have a meaning for files - the **`object_r`** role is a generic role used for files (on persistent storage and network file systems). Under the **`/proc/`** directory, files related to processes may use the **`system_r`** role.[22] The **`httpd_sys_content_t`** type allows the `httpd` process to access this file.

4. As the Linux root user, run the **`service httpd start`** command to start the `httpd` process. The output is as follows if `httpd` starts successfully:

```
# /sbin/service httpd start
Starting httpd:                                            [  OK  ]
```

5. Change into a directory where your Linux user has write access to, and run the **`wget http://localhost/testfile`** command. Unless there are changes to the default configuration, this command succeeds:

```

$ wget http://localhost/testfile

--2010-05-11 13:19:07--  http://localhost/testfile
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: “testfile”

    [ <=>            ] 0           --.-K/s   in 0s

2010-05-11 13:19:07 (0.00 B/s) - “testfile” saved [0/0]
```

6. The **chcon** command relabels files; however, such label changes do not survive when the file system is relabeled. For permanent changes that survive a file system relabel, use the **semanage** command, which is discussed later. As the Linux root user, run the following command to change the type to a type used by Samba:

   `chcon -t samba_share_t /var/www/html/testfile`

   Run the **`ls -Z /var/www/html/testfile`** command to view the changes:

```
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/testfile
```

7. Note: the current DAC permissions allow the `httpd` process access to **`testfile`**. Change into a directory where your Linux user has write access to, and run the **`wget http://localhost/testfile`** command. Unless there are changes to the default configuration, this command fails:

```

$ wget http://localhost/testfile

--2010-05-11 13:23:49--  http://localhost/testfile
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
```

[22] When using other policies, such as MLS, other roles may be used, for example, **`secadm_r`**.

```
2010-05-11 13:23:49 ERROR 403: Forbidden.
```

8. As the Linux root user, run the **`rm -i /var/www/html/testfile`** command to remove **`testfile`**.

9. If you do not require `httpd` to be running, as the Linux root user, run the **`service httpd stop`** command to stop `httpd`:

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK  ]
```

This example demonstrates the additional security added by SELinux. Although DAC rules allowed the `httpd` process access to **`testfile`** in step 7, because the file was labeled with a type that the `httpd` process does not have access to, SELinux denied access. After step 7, an error similar to the following is logged to **`/var/log/messages`**:

```

May 11 13:23:51 localhost setroubleshoot: SELinux is preventing /usr/sbin/httpd "getattr"
 access to /var/www/html/testfile. For complete SELinux messages. run sealert -l ca2ab0df-
fcb9-46d1-8283-037450d1efcc
```

Previous log files may use a **`/var/log/messages.`*`YYYYMMDD`*** format. When running **syslog-ng**, previous log files may use a **`/var/log/messages.`*`X`*** format. If the `setroubleshootd` and `auditd` processes are running, errors similar to the following are logged to **`/var/log/audit/audit.log`**:

```
type=AVC msg=audit(1220706212.937:70): avc:  denied  { getattr } for  pid=1904 comm="httpd"
 path="/var/www/html/testfile" dev=sda5 ino=247576 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:samba_share_t:s0  tclass=file

type=SYSCALL msg=audit(1220706212.937:70): arch=40000003 syscall=196 success=no exit=-13
 a0=b9e21da0 a1=bf9581dc a2=555ff4 a3=2008171 items=0 ppid=1902 pid=1904 auid=500 uid=48
 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd"
 exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

Also, an error similar to the following is logged to **`/var/log/httpd/error_log`**:

```
[Tue May 11 13:23:49 2010] [error] [client ::1] (13)Permission denied: access to /testfile
 denied
```

9.3.2. Unconfined Processes

Unconfined processes run in unconfined domains, for example, init programs run in the unconfined **`initrc_t`** domain, unconfined kernel processes run in the **`kernel_t`** domain, and unconfined Linux users run in the **`unconfined_t`** domain. For unconfined processes, SELinux policy rules are applied, but policy rules exist that allow processes running in unconfined domains almost all access. Processes running in unconfined domains fall back to using DAC rules exclusively. If an unconfined process is compromised, SELinux does not prevent an attacker from gaining access to system resources and data, but of course, DAC rules are still used. SELinux is a security enhancement on top of DAC rules - it does not replace them.

The following example demonstrates how the Apache HTTP Server (`httpd`) can access data intended for use by Samba, when running unconfined. Note: in Fedora, the `httpd` process runs in the confined **`httpd_t`** domain by default. This is an example, and should not be used in production. It assumes that the *httpd*, *wget*, *setroubleshoot-server*, *dbus* and *audit* packages are installed, that the SELinux targeted policy is used, and that SELinux is running in enforcing mode:

1. Run the **`sestatus`** command to confirm that SELinux is enabled, is running in enforcing mode, and that targeted policy is being used:

   ```
   $ /usr/sbin/sestatus
   SELinux status:                 enabled
   SELinuxfs mount:                /selinux
   Current mode:                   enforcing
   Mode from config file:          enforcing
   Policy version:                 24
   Policy from config file:        targeted
   ```

   `SELinux status: enabled` is returned when SELinux is enabled. **`Current mode: enforcing`** is returned when SELinux is running in enforcing mode. **`Policy from config file: targeted`** is returned when the SELinux targeted policy is used.

2. As the Linux root user, run the **`touch /var/www/html/test2file`** command to create a file.

3. Run the **`ls -Z /var/www/html/test2file`** command to view the SELinux context:

   ```
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
   test2file
   ```

   By default, Linux users run unconfined in Fedora, which is why the **`test2file`** file is labeled with the SELinux **`unconfined_u`** user. RBAC is used for processes, not files. Roles do not have a meaning for files - the **`object_r`** role is a generic role used for files (on persistent storage and network file systems). Under the **`/proc/`** directory, files related to processes may use the **`system_r`** role.[23] The **`httpd_sys_content_t`** type allows the `httpd` process to access this file.

4. The **chcon** command relabels files; however, such label changes do not survive when the file system is relabeled. For permanent changes that survive a file system relabel, use the **semanage** command, which is discussed later. As the Linux root user, run the following command to change the type to a type used by Samba:

   `chcon -t samba_share_t /var/www/html/test2file`

   Run the **`ls -Z /var/www/html/test2file`** command to view the changes:

   ```
   -rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/test2file
   ```

5. Run the **`service httpd status`** command to confirm that the `httpd` process is not running:

   ```
   $ /sbin/service httpd status
   httpd is stopped
   ```

   If the output differs, run the **`service httpd stop`** command as the Linux root user to stop the `httpd` process:

   ```
   # /sbin/service httpd stop
   Stopping httpd:                                            [  OK  ]
   ```

[23] When using other policies, such as MLS, other roles may also be used, for example, **`secadm_r`**.

6. To make the `httpd` process run unconfined, run the following command as the Linux root user to change the type of **`/usr/sbin/httpd`**, to a type that does not transition to a confined domain:

   `chcon -t unconfined_exec_t /usr/sbin/httpd`

7. Run the **`ls -Z /usr/sbin/httpd`** command to confirm that **`/usr/sbin/httpd`** is labeled with the **`unconfined_exec_t`** type:

```
-rwxr-xr-x  root root system_u:object_r:unconfined_exec_t /usr/sbin/httpd
```

8. As the Linux root user, run the **`service httpd start`** command to start the `httpd` process. The output is as follows if `httpd` starts successfully:

```
# /sbin/service httpd start
Starting httpd:                                            [  OK  ]
```

9. Run the **`ps -eZ | grep httpd`** command to view the `httpd` running in the **`unconfined_t`** domain:

```
$ ps -eZ | grep httpd
unconfined_u:system_r:unconfined_t 7721 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7723 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7724 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7725 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7726 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7727 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7728 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7729 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7730 ?      00:00:00 httpd
```

10. Change into a directory where your Linux user has write access to, and run the **`wget http://localhost/test2file`** command. Unless there are changes to the default configuration, this command succeeds:

```
--2009-05-07 01:41:10--  http://localhost/test2file
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `test2file.1'

[ <=>                           ]--.-K/s   in 0s

2009-05-07 01:41:10 (0.00 B/s) - `test2file.1' saved [0/0]
```

    Although the `httpd` process does not have access to files labeled with the **`samba_share_t`** type, `httpd` is running in the unconfined **`unconfined_t`** domain, and falls back to using DAC rules, and as such, the **`wget`** command succeeds. Had `httpd` been running in the confined **`httpd_t`** domain, the **`wget`** command would have failed.

11. The **`restorecon`** command restores the default SELinux context for files. As the Linux root user, run the **`restorecon -v /usr/sbin/httpd`** command to restore the default SELinux context for **`/usr/sbin/httpd`**:

```
# /sbin/restorecon -v /usr/sbin/httpd
```

```
restorecon reset /usr/sbin/httpd context system_u:object_r:unconfined_notrans_exec_t:s0-
>system_u:object_r:httpd_exec_t:s0
```

Run the **`ls -Z /usr/sbin/httpd`** command to confirm that **`/usr/sbin/httpd`** is labeled with the **`httpd_exec_t`** type:

```
$ ls -Z /usr/sbin/httpd
-rwxr-xr-x  root root system_u:object_r:httpd_exec_t   /usr/sbin/httpd
```

12. As the Linux root user, run the **`/sbin/service httpd restart`** command to restart `httpd`. After restarting, run the **`ps -eZ | grep httpd`** to confirm that `httpd` is running in the confined **`httpd_t`** domain:

```
# /sbin/service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
# ps -eZ | grep httpd
unconfined_u:system_r:httpd_t    8880 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8882 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8883 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8884 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8885 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8886 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8887 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8888 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8889 ?        00:00:00 httpd
```

13. As the Linux root user, run the **`rm -i /var/www/html/test2file`** command to remove **`test2file`**.

14. If you do not require `httpd` to be running, as the Linux root user, run the **`service httpd stop`** command to stop `httpd`:

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK  ]
```

The examples in these sections demonstrate how data can be protected from a compromised confined-process (protected by SELinux), as well as how data is more accessible to an attacker from a compromised unconfined-process (not protected by SELinux).

9.3.3. Confined and Unconfined Users

Each Linux user is mapped to an SELinux user via SELinux policy. This allows Linux users to inherit the restrictions on SELinux users. This Linux user mapping is seen by running the **`semanage login -l`** command as the Linux root user:

```
# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

In Fedora 19, Linux users are mapped to the SELinux **`__default__`** login by default (which is mapped to the SELinux **`unconfined_u`** user). The following defines the default-mapping:

```
__default__               unconfined_u              s0-s0:c0.c1023
```

The following example demonstrates adding a new Linux user, and that Linux user being mapped to the SELinux **`unconfined_u`** user. It assumes that the Linux root user is running unconfined, as it does by default in Fedora 19:

1. As the Linux root user, run the **`/usr/sbin/useradd newuser`** command to create a new Linux user named newuser.

2. As the Linux root user, run the **`passwd newuser`** command to assign a password to the Linux newuser user:

   ```
   # passwd newuser
   Changing password for user newuser.
   New UNIX password: Enter a password
   Retype new UNIX password: Enter the same password again
   passwd: all authentication tokens updated successfully.
   ```

3. Log out of your current session, and log in as the Linux newuser user. When you log in, pam_selinux maps the Linux user to an SELinux user (in this case, unconfined_u), and sets up the resulting SELinux context. The Linux user's shell is then launched with this context. Run the **`id -Z`** command to view the context of a Linux user:

   ```
   [newuser@localhost ~]$ id -Z
   unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
   ```

4. Log out of the Linux newuser's session, and log in with your account. If you do not want the Linux newuser user, run the **`/usr/sbin/userdel -r newuser`** command as the Linux root user to remove it, along with the Linux newuser's home directory.

Confined and unconfined Linux users are subject to executable and writeable memory checks, and are also restricted by MCS (and MLS, if the MLS policy is used). If unconfined Linux users execute an application that SELinux policy defines can transition from the **`unconfined_t`** domain to its own confined domain, unconfined Linux users are still subject to the restrictions of that confined domain. The security benefit of this is that, even though a Linux user is running unconfined, the application remains confined, and therefore, the exploitation of a flaw in the application can be limited by policy. Note: this does not protect the system from the user. Instead, the user and the system are being protected from possible damage caused by a flaw in the application.

The following confined SELinux users are available in Fedora 19:

Table 9.1. SELinux User Capabilities

User	Domain	X Window System	su and sudo	Execute in home directory and /tmp/	Networking
guest_u	guest_t	no	no	optional	no
xguest_u	xguest_t	yes	no	optional	only **Firefox**
user_u	user_t	yes	no	optional	yes
staff_u	staff_t	yes	only **sudo**	optional	yes

- Linux users in the **`guest_t`**, **`xguest_t`**, and **`user_t`** domains can only run set user ID (setuid) applications if SELinux policy permits it (such as **`passwd`**). They can not run the **su** and **`/usr/bin/`**

sudo setuid applications, and therefore, can not use these applications to become the Linux root user.

- Linux users in the **`guest_t`** domain have no network access, and can only log in via a terminal (including `ssh`; they can log in via `ssh`, but can not use `ssh` to connect to another system).
- The only network access Linux users in the **`xguest_t`** domain have is **Firefox** connecting to web pages.
- Linux users in the **`xguest_t`**, **`user_t`** and **`staff_t`** domains can log in via the X Window System and a terminal.
- By default, Linux users in the **`staff_t`** domain do not have permissions to execute applications with **`/usr/bin/sudo`**. These permissions must be configured by an administrator.

By default, Linux users in the **`guest_t`** and **`xguest_t`** domains can not execute applications in their home directories or **`/tmp/`**, preventing them from executing applications (which inherit users' permissions) in directories they have write access to. This helps prevent flawed or malicious applications from modifying files users' own.

By default, Linux users in the **`user_t`** and **`staff_t`** domains can execute applications in their home directories and **`/tmp/`**. Refer to *Section 9.5.6, "Booleans for Users Executing Applications"* for information about allowing and preventing users from executing applications in their home directories and **`/tmp/`**.

9.4. Working with SELinux

The following sections give a brief overview of the main SELinux packages in Fedora, installing and updating packages, which log files are used, the main SELinux configuration file, enabling and disabling SELinux, SELinux modes, configuring Booleans, temporarily and persistently changing file and directory labels, overriding file system labels with the **`mount`** command, mounting NFS file systems, and how to preserve SELinux contexts when copying and archiving files and directories.

9.4.1. SELinux Packages

In Fedora, the SELinux packages are installed by default in a full installation, unless they are manually excluded during installation. If performing a minimal installation in text mode, the *policycoreutils-python* package will not be installed by default. Also, by default, SELinux targeted policy is used, and SELinux runs in enforcing mode. The following is a brief description of the main SELinux packages:

policycoreutils-python: provides utilities such as **`semanage`**, **`audit2allow`**, **`audit2why`** and **`chcat`**, for operating and managing SELinux.

policycoreutils: provides utilities such as **`restorecon`**, **`secon`**, **`setfiles`**, **`semodule`**, **`load_policy`**, and **`setsebool`**, for operating and managing SELinux.

policycoreutils-gui: provides **`system-config-selinux`**, a graphical tool for managing SELinux.

selinux-policy: provides the SELinux Reference Policy. The SELinux Reference Policy is a complete SELinux policy, and is used as a basis for other policies, such as the SELinux targeted policy. Refer to the Tresys Technology *SELinux Reference Policy*[24] page for further information. The *selinux-policy-devel* package provides development tools, such as **`/usr/share/selinux/devel/`**

[24] http://oss.tresys.com/projects/refpolicy

policygentool and **/usr/share/selinux/devel/policyhelp**, as well as example policy files. This package was merged into the *selinux-policy* package.

selinux-policy-`policy`: provides SELinux policies. For targeted policy, install *selinux-policy-targeted*. For MLS, install *selinux-policy-mls*.

setroubleshoot-server: translates denial messages, produced when access is denied by SELinux, into detailed descriptions that are viewed with **sealert** (which is provided by this package).

setools, *setools-gui*, and *setools-console*: these packages provide the *Tresys Technology SETools distribution*[25], a number of tools and libraries for analyzing and querying policy, audit log monitoring and reporting, and file context management[26]. The *setools* package is a meta-package for SETools. The *setools-gui* package provides the **apol**, **seaudit**, and **sediffx** tools. The *setools-console* package provides the **seaudit-report**, **sechecker**, **sediff**, **seinfo**, **sesearch**, **findcon**, **replcon**, and **indexcon** command line tools. Refer to the *Tresys Technology SETools*[27] page for information about these tools.

libselinux-utils: provides the **avcstat**, **getenforce**, **getsebool**, **matchpathcon**, **selinuxconlist**, **selinuxdefcon**, **selinuxenabled**, **setenforce**, **togglesebool** tools.

mcstrans: translates levels, such as **s0-s0:c0.c1023**, to an easier to read form, such as **SystemLow-SystemHigh**. This package is not installed by default.

To install packages in Fedora, as the Linux root user, run the **yum install *package-name*** command. For example, to install the *mcstrans* package, run the **yum install mcstrans** command. To upgrade all installed packages in Fedora, run the **yum update** command.

Refer to *Managing Software with yum*[28][29] for further information about using **yum** to manage packages.

Note

In previous versions of Fedora, the *selinux-policy-devel* package is required when making a local policy module with **audit2allow -M**.

9.4.2. Which Log File is Used

In Fedora 19, the *dbus*, *setroubleshoot-server* and *audit* packages are installed if packages are not removed from the default package selection.

SELinux denial messages, such as the following, are written to **/var/log/audit/audit.log** by default:

```
type=AVC msg=audit(1223024155.684:49): avc:  denied  { getattr } for  pid=2000 comm="httpd"
 path="/var/www/html/file1" dev=dm-0 ino=399185 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=system_u:object_r:samba_share_t:s0 tclass=file
```

[25] http://oss.tresys.com/projects/setools
[26] Brindle, Joshua. "Re: blurb for fedora setools packages" Email to Murray McAllister. 1 November 2008. Any edits or changes in this version were done by Murray McAllister.
[27] http://oss.tresys.com/projects/setools
[28] http://docs.fedoraproject.org/yum/en/
[29] Managing Software with yum, written by Stuart Ellis, edited by Paul W. Frields, Rodrigo Menezes, and Hugo Cisneiros.

Also, if `setroubleshootd` is running, denial messages from **`/var/log/audit/audit.log`** are translated to an easier-to-read form and sent to **`/var/log/messages`**:

```
May  7 18:55:56 localhost setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr"
 to /var/www/html/file1 (samba_share_t). For complete SELinux messages. run sealert -l
 de7e30d6-5488-466d-a606-92c9f40d316d
```

In Fedora 19, `setroubleshootd` no longer constantly runs as a service, however it is still used to analyze the AVC messages. Two new programs act as a method to start setroubleshoot when needed: `sedispatch` and `seapplet`. `sedispatch` runs as part of the audit subsystem, and via `dbus`, sends a message when an AVC denial occurs, which will go straight to `setroubleshootd` if it is already running, or it will start `setroubleshootd` if it is not running. `seapplet` is a tool which runs in the system's toolbar, waiting for dbus messages in `setroubleshootd`, and will launch the notification bubble, allowing the user to review the denial.

Denial messages are sent to a different location, depending on which daemons are running:

Daemon	Log Location
auditd on	**`/var/log/audit/audit.log`**
auditd off; rsyslogd on	**`/var/log/messages`**
rsyslogd and auditd on	**`/var/log/audit/audit.log`**. Easier-to-read denial messages also sent to **`/var/log/messages`**

Starting Daemons Automatically

To configure the `auditd`, `rsyslogd`, and `setroubleshootd` daemons to automatically start at boot, run the following commands as the Linux root user:

```
/sbin/chkconfig --levels 2345 auditd on
```

```
/sbin/chkconfig --levels 2345 rsyslog on
```

Use the **`service`** ***`service-name`*** **`status`** command to check if these services are running, for example:

```
$ /sbin/service auditd status
auditd (pid  1318) is running...
```

If the above services are not running (***`service-name`*** **`is stopped`**), use the **`service`** ***`service-name`*** **`start`** command as the Linux root user to start them. For example:

```
# /sbin/service auditd start
Starting auditd:                                  [  OK  ]
```

9.4.3. Main Configuration File

The **`/etc/selinux/config`** file is the main SELinux configuration file. It controls the SELinux mode and the SELinux policy to use:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

`SELINUX=enforcing`
: The **`SELINUX`** option sets the mode SELinux runs in. SELinux has three modes: enforcing, permissive, and disabled. When using enforcing mode, SELinux policy is enforced, and SELinux denies access based on SELinux policy rules. Denial messages are logged. When using permissive mode, SELinux policy is not enforced. SELinux does not deny access, but denials are logged for actions that would have been denied if running SELinux in enforcing mode. When using disabled mode, SELinux is disabled (the SELinux module is not registered with the Linux kernel), and only DAC rules are used.

`SELINUXTYPE=targeted`
: The **`SELINUXTYPE`** option sets the SELinux policy to use. Targeted policy is the default policy. Only change this option if you want to use the MLS policy. To use the MLS policy, install the *selinux-policy-mls* package; configure **`SELINUXTYPE=mls`** in **`/etc/selinux/config`**; and reboot your system.

Important

When systems run with SELinux in permissive or disabled mode, users have permission to label files incorrectly. Also, files created while SELinux is disabled are not labeled. This causes problems when changing to enforcing mode. To prevent incorrectly labeled and unlabeled files from causing problems, file systems are automatically relabeled when changing from disabled mode to permissive or enforcing mode.

9.4.4. Enabling and Disabling SELinux

Use the **`/usr/sbin/getenforce`** or **`/usr/sbin/sestatus`** commands to check the status of SELinux. The **`getenforce`** command returns **`Enforcing`**, **`Permissive`**, or **`Disabled`**. The **`getenforce`** command returns **`Enforcing`** when SELinux is enabled (SELinux policy rules are enforced):

```
$ /usr/sbin/getenforce
Enforcing
```

The **`getenforce`** command returns **`Permissive`** when SELinux is enabled, but SELinux policy rules are not enforced, and only DAC rules are used. The **`getenforce`** command returns **`Disabled`** if SELinux is disabled.

The **`sestatus`** command returns the SELinux status and the SELinux policy being used:

```
$ /usr/sbin/sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 23
Policy from config file:        targeted
```

`SELinux status: enabled` is returned when SELinux is enabled. **`Current mode: enforcing`** is returned when SELinux is running in enforcing mode. **`Policy from config file: targeted`** is returned when the SELinux targeted policy is used.

9.4.4.1. Enabling SELinux

On systems with SELinux disabled, the **`SELINUX=disabled`** option is configured in **`/etc/selinux/config`**:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Also, the **`getenforce`** command returns **`Disabled`**:

```
$ /usr/sbin/getenforce
Disabled
```

To enable SELinux:

1. Use the **`rpm -qa | grep selinux`**, **`rpm -q policycoreutils`**, and **`rpm -qa | grep setroubleshoot`** commands to confirm that the SELinux packages are installed. This guide assumes the following packages are installed: *selinux-policy-targeted*, *selinux-policy*, *libselinux*, *libselinux-python*, *libselinux-utils*, *policycoreutils*, *setroubleshoot*, *setroubleshoot-server*, *setroubleshoot-plugins*. If these packages are not installed, as the Linux root user, install them via the **`yum install`** ***`package-name`*** command. The following packages are optional: *policycoreutils-gui*, *setroubleshoot*, *selinux-policy-devel*, and *mcstrans*.

2. Before SELinux is enabled, each file on the file system must be labeled with an SELinux context. Before this happens, confined domains may be denied access, preventing your system from booting correctly. To prevent this, configure **`SELINUX=permissive`** in **`/etc/selinux/config`**:

   ```
   # This file controls the state of SELinux on the system.
   # SELINUX= can take one of these three values:
   #       enforcing - SELinux security policy is enforced.
   #       permissive - SELinux prints warnings instead of enforcing.
   #       disabled - No SELinux policy is loaded.
   SELINUX=permissive
   # SELINUXTYPE= can take one of these two values:
   #       targeted - Targeted processes are protected,
   #       mls - Multi Level Security protection.
   SELINUXTYPE=targeted
   ```

3. As the Linux root user, run the **`reboot`** command to restart the system. During the next boot, file systems are labeled. The label process labels all files with an SELinux context:

   ```
   *** Warning -- SELinux targeted policy relabel is required.
   *** Relabeling could take a very long time, depending on file
   *** system size and speed of hard drives.
   ****
   ```

Each **`*`** character on the bottom line represents 1000 files that have been labeled. In the above example, four **`*`** characters represent 4000 files have been labeled. The time it takes to label all files depends upon the number of files on the system, and the speed of the hard disk drives. On modern systems, this process can take as little as 10 minutes.

4. In permissive mode, SELinux policy is not enforced, but denials are still logged for actions that would have been denied if running in enforcing mode. Before changing to enforcing mode, as the Linux root user, run the **`grep "SELinux is preventing" /var/log/messages`** command as the Linux root user to confirm that SELinux did not deny actions during the last boot. If SELinux did not deny actions during the last boot, this command does not return any output. Refer to *Section 9.6, "Troubleshooting"* for troubleshooting information if SELinux denied access during boot.

5. If there were no denial messages in **`/var/log/messages`**, configure **`SELINUX=enforcing`** in **`/etc/selinux/config`**:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

6. Reboot your system. After reboot, confirm that the **`getenforce`** command returns **`Enforcing`**:

```
$ /usr/sbin/getenforce
Enforcing
```

7. As the Linux root user, run the **`/usr/sbin/semanage login -l`** command to view the mapping between SELinux and Linux users. The output should be as follows:

```
Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

If this is not the case, run the following commands as the Linux root user to fix the user mappings. It is safe to ignore the **`SELinux-user`** ***`username`*** **`is already defined`** warnings if they occur, where *`username`* can be **`unconfined_u`**, **`guest_u`**, or **`xguest_u`**:

1.
```
/usr/sbin/semanage user -a -S targeted -P user -R "unconfined_r system_r" -r s0-
s0:c0.c1023 unconfined_u
```

2.
```
/usr/sbin/semanage login -m -S targeted -s "unconfined_u" -r s0-s0:c0.c1023 __default__
```

3.
```
/usr/sbin/semanage login -m -S targeted -s "unconfined_u" -r s0-s0:c0.c1023 root
```

4.
```
/usr/sbin/semanage user -a -S targeted -P user -R guest_r guest_u
```

5.
```
/usr/sbin/semanage user -a -S targeted  -P user -R xguest_r xguest_u
```

Important

When systems run with SELinux in permissive or disabled mode, users have permission to label files incorrectly. Also, files created while SELinux is disabled are not labeled. This causes problems when changing to enforcing mode. To prevent incorrectly labeled and unlabeled files from causing problems, file systems are automatically relabeled when changing from disabled mode to permissive or enforcing mode.

9.4.4.2. Disabling SELinux

To disable SELinux, configure **`SELINUX=disabled`** in **`/etc/selinux/config`**:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Reboot your system. After reboot, confirm that the **`getenforce`** command returns **`Disabled`**:

```
$ /usr/sbin/getenforce
Disabled
```

9.4.5. SELinux Modes

SELinux has three modes:

- Enforcing: SELinux policy is enforced. SELinux denies access based on SELinux policy rules.
- Permissive: SELinux policy is not enforced. SELinux does not deny access, but denials are logged for actions that would have been denied if running in enforcing mode.
- Disabled: SELinux is disabled. Only DAC rules are used.

Use the **`/usr/sbin/setenforce`** command to change between enforcing and permissive mode. Changes made with **`/usr/sbin/setenforce`** do not persist across reboots. To change to enforcing mode, as the Linux root user, run the **`/usr/sbin/setenforce 1`** command. To change to permissive mode, run the **`/usr/sbin/setenforce 0`** command. Use the **`/usr/sbin/getenforce`** command to view the current SELinux mode.

Persistent mode changes are covered in *Section 9.4.4, “Enabling and Disabling SELinux”*.

9.4.6. Booleans

Booleans allow parts of SELinux policy to be changed at runtime, without any knowledge of SELinux policy writing. This allows changes, such as allowing services access to NFS file systems, without reloading or recompiling SELinux policy.

9.4.6.1. Listing Booleans

For a list of Booleans, an explanation of what each one is, and whether they are on or off, run the **`semanage boolean -l`** command as the Linux root user. The following example does not list all Booleans:

```
# /usr/sbin/semanage boolean -l
SELinux boolean                          Description

ftp_home_dir                   -> off   Allow ftp to read and write files in the user home
 directories
xen_use_nfs                    -> off   Allow xen to manage nfs files
xguest_connect_network         -> on    Allow xguest to configure Network Manager
```

The **`SELinux boolean`** column lists Boolean names. The **`Description`** column lists whether the Booleans are on or off, and what they do.

In the following example, the **`ftp_home_dir`** Boolean is off, preventing the FTP daemon (`vsftpd`) from reading and writing to files in user home directories:

```
ftp_home_dir                   -> off   Allow ftp to read and write files in the user home
 directories
```

The **`getsebool -a`** command lists Booleans, whether they are on or off, but does not give a description of each one. The following example does not list all Booleans:

```
$ /usr/sbin/getsebool -a
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
```

Run the **`getsebool`** ***`boolean-name`*** command to only list the status of the *`boolean-name`* Boolean:

```
$ /usr/sbin/getsebool allow_console_login
allow_console_login --> off
```

Use a space-separated list to list multiple Booleans:

```
$ getsebool allow_console_login allow_cvs_read_shadow allow_daemons_dump_core
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
```

9.4.6.2. Configuring Booleans

The **`setsebool`** ***`boolean-name x`*** command turns Booleans on or off, where *`boolean-name`* is a Boolean name, and *x* is either **on** to turn the Boolean on, or **`off`** to turn it off.

The following example demonstrates configuring the **`httpd_can_network_connect_db`** Boolean:

1. By default, the **`httpd_can_network_connect_db`** Boolean is off, preventing Apache HTTP Server scripts and modules from connecting to database servers:

   ```
   $ /usr/sbin/getsebool httpd_can_network_connect_db
   httpd_can_network_connect_db --> off
   ```

2. To temporarily enable Apache HTTP Server scripts and modules to connect to database servers, run the **`setsebool httpd_can_network_connect_db on`** command as the Linux root user.

3. Use the **`getsebool httpd_can_network_connect_db`** command to verify the Boolean is turned on:

   ```
   $ /usr/sbin/getsebool httpd_can_network_connect_db
   httpd_can_network_connect_db --> on
   ```

   This allows Apache HTTP Server scripts and modules to connect to database servers.

4. This change is not persistent across reboots. To make changes persistent across reboots, run the **`setsebool -P`** ***`boolean-name`*** **`on`** command as the Linux root user:

   ```
   # /usr/sbin/setsebool -P httpd_can_network_connect_db on
   ```

5. To temporarily revert to the default behavior, as the Linux root user, run the **`setsebool httpd_can_network_connect_db off`** command. For changes that persist across reboots, run the **`setsebool -P httpd_can_network_connect_db off`** command.

9.4.6.3. Booleans for NFS and CIFS

By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **`nfs_t`** type. Also, by default, Samba shares mounted on the client side are labeled with a default context defined by policy. In common policies, this default context uses the **`cifs_t`** type.

Depending on policy configuration, services may not be able to read files labeled with the **`nfs_t`** or **`cifs_t`** types. This may prevent file systems labeled with these types from being mounted and then read or exported by other services. Booleans can be turned on or off to control which services are allowed to access the **`nfs_t`** and **`cifs_t`** types.

The **`setsebool`** and **`semanage`** commands must be run as the Linux root user. The **`setsebool -P`** command makes persistent changes. Do not use the **`-P`** option if you do not want changes to persist across reboots:

Apache HTTP Server

To allow access to NFS file systems (files labeled with the **`nfs_t`** type):

`/usr/sbin/setsebool -P httpd_use_nfs on`

To allow access to Samba file systems (files labeled with the **`cifs_t`** type):

`/usr/sbin/setsebool -P httpd_use_cifs on`

Samba

To export NFS file systems:

`/usr/sbin/setsebool -P samba_share_nfs on`

FTP (`vsftpd`)

To allow access to NFS file systems:

/usr/sbin/setsebool -P allow_ftpd_use_nfs on

To allow access to Samba file systems:

/usr/sbin/setsebool -P allow_ftpd_use_cifs on

Other Services

For a list of NFS related Booleans for other services:

/usr/sbin/semanage boolean -l | grep nfs

For a list of Samba related Booleans for other services:

/usr/sbin/semanage boolean -l | grep cifs

Note

These Booleans exist in SELinux policy as shipped with Fedora 19. They may not exist in policy shipped with other versions of Fedora or other operating systems.

Refer to the SELinux Managing Confined Services Guide at *http://docs.fedoraproject.org* for more information regarding SELinux Booleans.

9.4.7. SELinux Contexts - Labeling Files

On systems running SELinux, all processes and files are labeled in a way that represents security-relevant information. This information is called the SELinux context. For files, this is viewed using the **ls -Z** command:

```
$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

In this example, SELinux provides a user (**unconfined_u**), a role (**object_r**), a type (**user_home_t**), and a level (**s0**). This information is used to make access control decisions. On DAC systems, access is controlled based on Linux user and group IDs. SELinux policy rules are checked after DAC rules. SELinux policy rules are not used if DAC rules deny access first.

There are multiple commands for managing the SELinux context for files, such as **chcon**, **semanage fcontext**, and **restorecon**.

9.4.7.1. Temporary Changes: chcon

The **chcon** command changes the SELinux context for files. However, changes made with the **chcon** command do not survive a file system relabel, or the execution of the **/sbin/restorecon** command. SELinux policy controls whether users are able to modify the SELinux context for any given file. When using **chcon**, users provide all or part of the SELinux context to change. An incorrect file type is a common cause of SELinux denying access.

Quick Reference

- Run the **chcon -t *type* *file-name*** command to change the file type, where *type* is a type, such as **httpd_sys_content_t**, and *file-name* is a file or directory name.

- Run the **chcon -R -t *type directory-name*** command to change the type of the directory and its contents, where *type* is a type, such as **httpd_sys_content_t**, and *directory-name* is a directory name.

Changing a File's or Directory's Type

The following example demonstrates changing the type, and no other attributes of the SELinux context:

1. Run the **cd** command without arguments to change into your home directory.

2. Run the **touch file1** command to create a new file. Use the **ls -Z file1** command to view the SELinux context for **file1**:

   ```
   $ ls -Z file1
   -rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
   ```

   In this example, the SELinux context for **file1** includes the SELinux **unconfined_u** user, **object_r** role, **user_home_t** type, and the **s0** level. For a description of each part of the SELinux context, refer to *Section 9.2, "SELinux Contexts"*.

3. Run the **chcon -t samba_share_t file1** command to change the type to **samba_share_t**. The **-t** option only changes the type. View the change with **ls -Z file1**:

   ```
   $ ls -Z file1
   -rw-rw-r--  user1 group1 unconfined_u:object_r:samba_share_t:s0 file1
   ```

4. Use the **/sbin/restorecon -v file1** command to restore the SELinux context for the **file1** file. Use the **-v** option to view what changes:

   ```
   $ /sbin/restorecon -v file1
   restorecon reset file1 context unconfined_u:object_r:samba_share_t:s0-
   >system_u:object_r:user_home_t:s0
   ```

   In this example, the previous type, **samba_share_t**, is restored to the correct, **user_home_t** type. When using targeted policy (the default SELinux policy in Fedora 19), the **/sbin/restorecon** command reads the files in the **/etc/selinux/targeted/contexts/files/** directory, to see which SELinux context files should have.

The example in this section works the same for directories, for example, if **file1** was a directory.

Changing a Directory and its Contents Types

The following example demonstrates creating a new directory, and changing the directory's file type (along with its contents) to a type used by the Apache HTTP Server. The configuration in this example is used if you want Apache HTTP Server to use a different document root (instead of **/var/www/html/**):

1. As the Linux root user, run the **mkdir /web** command to create a new directory, and then the **touch /web/file{1,2,3}** command to create 3 empty files (**file1**, **file2**, and **file3**). The **/web/** directory and files in it are labeled with the **default_t** type:

   ```
   # ls -dZ /web
   drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
   # ls -lZ /web
   -rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
   ```

```
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
```

2. As the Linux root user, run the **`chcon -R -t httpd_sys_content_t /web/`** command to change the type of the **`/web/`** directory (and its contents) to **`httpd_sys_content_t`**:

```
# chcon -R -t httpd_sys_content_t /web/
# ls -dZ /web/
drwxr-xr-x  root root unconfined_u:object_r:httpd_sys_content_t:s0 /web/
# ls -lZ /web/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

3. As the Linux root user, run the **`/sbin/restorecon -R -v /web/`** command to restore the default SELinux contexts:

```
# /sbin/restorecon -R -v /web/
restorecon reset /web context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file2 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file3 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file1 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
```

Refer to the chcon(1) manual page for further information about **`chcon`**.

Note

Type Enforcement is the main permission control used in SELinux targeted policy. For the most part, SELinux users and roles can be ignored.

9.4.7.2. Persistent Changes: semanage fcontext

The **`/usr/sbin/semanage fcontext`** command changes the SELinux context for files. When using targeted policy, changes made with this command are added to the **`/etc/selinux/targeted/contexts/files/file_contexts`** file if the changes are to files that exists in **`file_contexts`**, or are added to **`file_contexts.local`** for new files and directories, such as creating a **`/web/`** directory. **`setfiles`**, which is used when a file system is relabeled, and **`/sbin/restorecon`**, which restores the default SELinux contexts, read these files. This means that changes made by **`/usr/sbin/semanage fcontext`** are persistent, even if the file system is relabeled. SELinux policy controls whether users are able to modify the SELinux context for any given file.

Quick Reference

To make SELinux context changes that survive a file system relabel:

1. Run the **`/usr/sbin/semanage fcontext -a`** ***`options file-name|directory-name`*** command, remembering to use the full path to the file or directory.

2. Run the **`/sbin/restorecon -v`** ***`file-name|directory-name`*** command to apply the context changes.

Changing a File's Type

The following example demonstrates changing a file's type, and no other attributes of the SELinux context:

1. As the Linux root user, run the **`touch /etc/file1`** command to create a new file. By default, newly-created files in the **`/etc/`** directory are labeled with the **`etc_t`** type:

```
# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0       /etc/file1
```

2. As the Linux root user, run the **`/usr/sbin/semanage fcontext -a -t samba_share_t /etc/file1`** command to change the **`file1`** type to **`samba_share_t`**. The **`-a`** option adds a new record, and the **`-t`** option defines a type (**`samba_share_t`**). Note: running this command does not directly change the type - **`file1`** is still labeled with the **`etc_t`** type:

```
# /usr/sbin/semanage fcontext -a -t samba_share_t /etc/file1
# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0       /etc/file1
```

The **`/usr/sbin/semanage fcontext -a -t samba_share_t /etc/file1`** command adds the following entry to **`/etc/selinux/targeted/contexts/files/file_contexts.local`**:

```
/etc/file1    unconfined_u:object_r:samba_share_t:s0
```

3. As the Linux root user, run the **`/sbin/restorecon -v /etc/file1`** command to change the type. Since the **`semanage`** command added an entry to **`file.contexts.local`** for **`/etc/file1`**, the **`/sbin/restorecon`** command changes the type to **`samba_share_t`**:

```
# /sbin/restorecon -v /etc/file1
restorecon reset /etc/file1 context unconfined_u:object_r:etc_t:s0-
>system_u:object_r:samba_share_t:s0
```

4. As the Linux root user, run the **`rm -i /etc/file1`** command to remove **`file1`**.

5. As the Linux root user, run the **`/usr/sbin/semanage fcontext -d /etc/file1`** command to remove the context added for **`/etc/file1`**. When the context is removed, running **`restorecon`** changes the type to **`etc_t`**, rather than **`samba_share_t`**.

Changing a Directory's Type

The following example demonstrates creating a new directory and changing that directory's file type, to a type used by Apache HTTP Server:

1. As the Linux root user, run the **`mkdir /web`** command to create a new directory. This directory is labeled with the **`default_t`** type:

```
# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
```

The **`ls -d`** option makes **`ls`** list information about a directory, rather than its contents, and the **`-Z`** option makes **`ls`** display the SELinux context (in this example, **`unconfined_u:object_r:default_t:s0`**).

2. As the Linux root user, run the **`/usr/sbin/semanage fcontext -a -t httpd_sys_content_t /web`** command to change the **`/web/`** type to **`httpd_sys_content_t`**. The **`-a`** option adds a new record, and the **`-t`** option defines a type (**`httpd_sys_content_t`**). Note: running this command does not directly change the type - **`/web/`** is still labeled with the **`default_t`** type:

   ```
   # /usr/sbin/semanage fcontext -a -t httpd_sys_content_t /web
   # ls -dZ /web
   drwxr-xr-x  root root unconfined_u:object_r:default_t:s0   /web
   ```

   The **`/usr/sbin/semanage fcontext -a -t httpd_sys_content_t /web`** command adds the following entry to **`/etc/selinux/targeted/contexts/files/file_contexts.local`**:

   ```
   /web    unconfined_u:object_r:httpd_sys_content_t:s0
   ```

3. As the Linux root user, run the **`/sbin/restorecon -v /web`** command to change the type. Since the **`semanage`** command added an entry to **`file.contexts.local`** for **`/web`**, the **`/sbin/restorecon`** command changes the type to **`httpd_sys_content_t`**:

   ```
   # /sbin/restorecon -v /web
   restorecon reset /web context unconfined_u:object_r:default_t:s0-
   >system_u:object_r:httpd_sys_content_t:s0
   ```

   By default, newly-created files and directories inherit the SELinux type of their parent folders. When using this example, and before removing the SELinux context added for **`/web/`**, files and directories created in the **`/web/`** directory are labeled with the **`httpd_sys_content_t`** type.

4. As the Linux root user, run the **`/usr/sbin/semanage fcontext -d /web`** command to remove the context added for **`/web/`**.

5. As the Linux root user, run the **`/sbin/restorecon -v /web`** command to restore the default SELinux context.

Changing a Directory and its Contents Types

The following example demonstrates creating a new directory, and changing the directory's file type (along with its contents) to a type used by Apache HTTP Server. The configuration in this example is used if you want Apache HTTP Server to use a different document root (instead of **`/var/www/html/`**):

1. As the Linux root user, run the **`mkdir /web`** command to create a new directory, and then the **`touch /web/file{1,2,3}`** command to create 3 empty files (**`file1`**, **`file2`**, and **`file3`**). The **`/web/`** directory and files in it are labeled with the **`default_t`** type:

   ```
   # ls -dZ /web
   drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
   # ls -lZ /web
   -rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
   -rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
   -rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
   ```

2. As the Linux root user, run the **`/usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"`** command to change the type of the **`/web/`** directory and the files in it, to **`httpd_sys_content_t`**. The **`-a`** option adds a new record, and the **`-t`**

option defines a type (httpd_sys_content_t). The **"/web(/.*)?"** regular expression causes the **semanage** command to apply changes to the **/web/** directory, as well as the files in it. Note: running this command does not directly change the type - **/web/** and files in it are still labeled with the **default_t** type:

```
# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
# ls -lZ /web
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
```

The **/usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"** command adds the following entry to **/etc/selinux/targeted/contexts/files/file_contexts.local**:

```
/web(/.*)?    system_u:object_r:httpd_sys_content_t:s0
```

3. As the Linux root user, run the **/sbin/restorecon -R -v /web** command to change the type of the **/web/** directory, as well as all files in it. The **-R** is for recursive, which means all files and directories under the **/web/** directory are labeled with the **httpd_sys_content_t** type. Since the **semanage** command added an entry to **file.contexts.local** for **/web(/.*)?**, the **/sbin/restorecon** command changes the types to **httpd_sys_content_t**:

```
# /sbin/restorecon -R -v /web
restorecon reset /web context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file2 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file3 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file1 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

   By default, newly-created files and directories inherit the SELinux type of their parents. In this example, files and directories created in the **/web/** directory will be labeled with the **httpd_sys_content_t** type.

4. As the Linux root user, run the **/usr/sbin/semanage fcontext -d "/web(/.*)?"** command to remove the context added for **"/web(/.*)?"**.

5. As the Linux root user, run the **/sbin/restorecon -R -v /web** command to restore the default SELinux contexts.

Deleting an added Context

The following example demonstrates adding and removing an SELinux context:

1. As the Linux root user, run the **/usr/sbin/semanage fcontext -a -t httpd_sys_content_t /test** command. The **/test/** directory does not have to exist. This command adds the following context to **/etc/selinux/targeted/contexts/files/file_contexts.local**:

```
/test    system_u:object_r:httpd_sys_content_t:s0
```

2. To remove the context, as the Linux root user, run the **`/usr/sbin/semanage fcontext -d`** ***`file-name|directory-name`*** command, where *`file-name|directory-name`* is the first part in **`file_contexts.local`**. The following is an example of a context in **`file_contexts.local`**:

   ```
   /test    system_u:object_r:httpd_sys_content_t:s0
   ```

   With the first part being **`/test`**. To prevent the **`/test/`** directory from being labeled with the **`httpd_sys_content_t`** after running **`/sbin/restorecon`**, or after a file system relabel, run the following command as the Linux root user to delete the context from **`file_contexts.local`**:

   `/usr/sbin/semanage fcontext -d /test`

If the context is part of a regular expression, for example, **`/web(/.*)?`**, use quotation marks around the regular expression:

`/usr/sbin/semanage fcontext -d "/web(/.*)?"`

Refer to the semanage(8) manual page for further information about **`/usr/sbin/semanage`**.

Important

When changing the SELinux context with **`/usr/sbin/semanage fcontext -a`**, use the full path to the file or directory to avoid files being mislabeled after a file system relabel, or after the **`/sbin/restorecon`** command is run.

9.4.8. The file_t and default_t Types

For file systems that support extended attributes, when a file that lacks an SELinux context on disk is accessed, it is treated as if it had a default context as defined by SELinux policy. In common policies, this default context uses the **`file_t`** type. This should be the only use of this type, so that files without a context on disk can be distinguished in policy, and generally kept inaccessible to confined domains. The **`file_t`** type should not exist on correctly-labeled file systems, because all files on a system running SELinux should have an SELinux context, and the **`file_t`** type is never used in file-context configuration[30].

The **`default_t`** type is used on files that do not match any other pattern in file-context configuration, so that such files can be distinguished from files that do not have a context on disk, and generally kept inaccessible to confined domains. If you create a new top-level directory, such as **`/mydirectory/`**, this directory may be labeled with the **`default_t`** type. If services need access to such a directory, update the file-contexts configuration for this location. Refer to *Section 9.4.7.2, “Persistent Changes: semanage fcontext”* for details on adding a context to the file-context configuration.

9.4.9. Mounting File Systems

By default, when a file system that supports extended attributes is mounted, the security context for each file is obtained from the *security.selinux* extended attribute of the file. Files in file systems that

[30] Files in **`/etc/selinux/targeted/contexts/files/`** define contexts for files and directories. Files in this directory are read by **`restorecon`** and **`setfiles`** to restore files and directories to their default contexts.

do not support extended attributes are assigned a single, default security context from the policy configuration, based on file system type.

Use the **mount -o context** command to override existing extended attributes, or to specify a different, default context for file systems that do not support extended attributes. This is useful if you do not trust a file system to supply the correct attributes, for example, removable media used in multiple systems. The **mount -o context** command can also be used to support labeling for file systems that do not support extended attributes, such as File Allocation Table (FAT) or NFS file systems. The context specified with the **context** is not written to disk: the original contexts are preserved, and are seen when mounting without a **context** option (if the file system had extended attributes in the first place).

For further information about file system labeling, refer to James Morris's "Filesystem Labeling in SELinux" article: *http://www.linuxjournal.com/article/7426*.

9.4.9.1. Context Mounts

To mount a file system with the specified context, overriding existing contexts if they exist, or to specify a different, default context for a file system that does not support extended attributes, as the Linux root user, use the **mount -o context=*SELinux_user:role:type:level*** command when mounting the desired file system. Context changes are not written to disk. By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **nfs_t** type. Without additional mount options, this may prevent sharing NFS file systems via other services, such as the Apache HTTP Server. The following example mounts an NFS file system so that it can be shared via the Apache HTTP Server:

```
# mount server:/export /local/mount/point -o\
context="system_u:object_r:httpd_sys_content_t:s0"
```

Newly-created files and directories on this file system appear to have the SELinux context specified with **-o context**; however, since context changes are not written to disk for these situations, the context specified with the **context** option is only retained if the **context** option is used on the next mount, and if the same context is specified.

Type Enforcement is the main permission control used in SELinux targeted policy. For the most part, SELinux users and roles can be ignored, so, when overriding the SELinux context with **-o context**, use the SELinux **system_u** user and **object_r** role, and concentrate on the type. If you are not using the MLS policy or multi-category security, use the **s0** level.

Note

When a file system is mounted with a **context** option, context changes (by users and processes) are prohibited. For example, running **chcon** on a file system mounted with a **context** option results in a **Operation not supported** error.

9.4.9.2. Changing the Default Context

As mentioned in *Section 9.4.8, “The file_t and default_t Types”*, on file systems that support extended attributes, when a file that lacks an SELinux context on disk is accessed, it is treated as if it had a default context as defined by SELinux policy. In common policies, this default context uses the **file_t** type. If it is desirable to use a different default context, mount the file system with the **defcontext** option.

The following example mounts a newly-created file system (on **/dev/sda2**) to the newly-created **/test/** directory. It assumes that there are no rules in **/etc/selinux/targeted/contexts/files/** that define a context for the **/test/** directory:

```
# mount /dev/sda2 /test/ -o defcontext="system_u:object_r:samba_share_t:s0"
```

In this example:

- the **defcontext** option defines that **system_u:object_r:samba_share_t:s0** is "the default security context for unlabeled files"[31].
- when mounted, the root directory (**/test/**) of the file system is treated as if it is labeled with the context specified by **defcontext** (this label is not stored on disk). This affects the labeling for files created under **/test/**: new files inherit the **samba_share_t** type, and these labels are stored on disk.
- files created under **/test/** while the file system was mounted with a **defcontext** option retain their labels.

9.4.9.3. Mounting an NFS File System

By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **nfs_t** type. Depending on policy configuration, services, such as Apache HTTP Server and MySQL, may not be able to read files labeled with the **nfs_t** type. This may prevent file systems labeled with this type from being mounted and then read or exported by other services.

If you would like to mount an NFS file system and read or export that file system with another service, use the **context** option when mounting to override the **nfs_t** type. Use the following context option to mount NFS file systems so that they can be shared via the Apache HTTP Server:

```
mount server:/export /local/mount/point -o\
context="system_u:object_r:httpd_sys_content_t:s0"
```

Since context changes are not written to disk for these situations, the context specified with the **context** option is only retained if the **context** option is used on the next mount, and if the same context is specified.

As an alternative to mounting file systems with **context** options, Booleans can be turned on to allow services access to file systems labeled with the **nfs_t** type. Refer to *Section 9.4.6.3, “Booleans for NFS and CIFS”* for instructions on configuring Booleans to allow services access to the **nfs_t** type.

9.4.9.4. Multiple NFS Mounts

When mounting multiple mounts from the same NFS export, attempting to override the SELinux context of each mount with a different context, results in subsequent mount commands failing. In the following example, the NFS server has a single export, **/export**, which has two subdirectories, **web/** and **database/**. The following commands attempt two mounts from a single NFS export, and try to override the context for each one:

[31] Morris, James. "Filesystem Labeling in SELinux". Published 1 October 2004. Accessed 14 October 2008: *http://www.linuxjournal.com/article/7426*.

```
# mount server:/export/web /local/web -o\
context="system_u:object_r:httpd_sys_content_t:s0"

# mount server:/export/database /local/database -o\
context="system_u:object_r:mysqld_db_t:s0"
```

The second mount command fails, and the following is logged to **`/var/log/messages`**:

```
kernel: SELinux: mount invalid.  Same superblock, different security settings for (dev 0:15,
 type nfs)
```

To mount multiple mounts from a single NFS export, with each mount having a different context, use the **`-o nosharecache,context`** options. The following example mounts multiple mounts from a single NFS export, with a different context for each mount (allowing a single service access to each one):

```
# mount server:/export/web /local/web -o\
nosharecache,context="system_u:object_r:httpd_sys_content_t:s0"

# mount server:/export/database /local/database -o\
nosharecache,context="system_u:object_r:mysqld_db_t:s0"
```

In this example, **`server:/export/web`** is mounted locally to **`/local/web/`**, with all files being labeled with the **`httpd_sys_content_t`** type, allowing Apache HTTP Server access. **`server:/export/database`** is mounted locally to **`/local/database`**, with all files being labeled with the **`mysqld_db_t`** type, allowing MySQL access. These type changes are not written to disk.

Important

The **`nosharecache`** options allows you to mount the same subdirectory of an export multiple times with different contexts (for example, mounting **`/export/web`** multiple times). Do not mount the same subdirectory from an export multiple times with different contexts, as this creates an overlapping mount, where files are accessible under two different contexts.

9.4.9.5. Making Context Mounts Persistent

To make context mounts persistent across remounting and reboots, add entries for the file systems in **`/etc/fstab`** or an automounter map, and use the desired context as a mount option. The following example adds an entry to **`/etc/fstab`** for an NFS context mount:

```
server:/export /local/mount/ nfs context="system_u:object_r:httpd_sys_content_t:s0" 0 0
```

Refer to the *Red Hat Enterprise Linux 5 Deployment Guide, Section 19.2. "NFS Client Configuration"*[32] for information about mounting NFS file systems.

[32] http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/html/Deployment_Guide/s1-nfs-client-config.html

9.4.10. Maintaining SELinux Labels

These sections describe what happens to SELinux contexts when copying, moving, and archiving files and directories. Also, it explains how to preserve contexts when copying and archiving.

9.4.10.1. Copying Files and Directories

When a file or directory is copied, a new file or directory is created if it does not exist. That new file or directory's context is based on default-labeling rules, not the original file or directory's context (unless options were used to preserve the original context). For example, files created in user home directories are labeled with the **`user_home_t`** type:

```
$ touch file1
$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

If such a file is copied to another directory, such as **`/etc/`**, the new file is created in accordance to default-labeling rules for the **`/etc/`** directory. Copying a file (without additional options) may not preserve the original context:

```
$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
# cp file1 /etc/
$ ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

When **`file1`** is copied to **`/etc/`**, if **`/etc/file1`** does not exist, **`/etc/file1`** is created as a new file. As shown in the example above, **`/etc/file1`** is labeled with the **`etc_t`** type, in accordance to default-labeling rules.

When a file is copied over an existing file, the existing file's context is preserved, unless the user specified **`cp`** options to preserve the context of the original file, such as **`--preserve=context`**. SELinux policy may prevent contexts from being preserved during copies.

Copying Without Preserving SELinux Contexts

When copying a file with the **`cp`** command, if no options are given, the type is inherited from the targeted, parent directory:

```
$ touch file1
$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
# cp file1 /var/www/html/
$ ls -Z /var/www/html/file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
```

In this example, **`file1`** is created in a user's home directory, and is labeled with the **`user_home_t`** type. The **`/var/www/html/`** directory is labeled with the **`httpd_sys_content_t`** type, as shown with the **`ls -dZ /var/www/html/`** command. When **`file1`** is copied to **`/var/www/html/`**, it inherits the **`httpd_sys_content_t`** type, as shown with the **`ls -Z /var/www/html/file1`** command.

Preserving SELinux Contexts When Copying

Use the **`cp --preserve=context`** command to preserve contexts when copying:

```
$ touch file1
$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
# cp --preserve=context file1 /var/www/html/
$ ls -Z /var/www/html/file1
-rw-r--r--  root root unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
```

In this example, **`file1`** is created in a user's home directory, and is labeled with the **`user_home_t`** type. The **`/var/www/html/`** directory is labeled with the **`httpd_sys_content_t`** type, as shown with the **`ls -dZ /var/www/html/`** command. Using the **`--preserve=context`** option preserves SELinux contexts during copy operations. As shown with the **`ls -Z /var/www/html/file1`** command, the **`file1`** **`user_home_t`** type was preserved when the file was copied to **`/var/www/html/`**.

Copying and Changing the Context

Use the **`cp -Z`** command to change the destination copy's context. The following example was performed in the user's home directory:

```
$ touch file1
$ cp -Z system_u:object_r:samba_share_t:s0 file1 file2
$ ls -Z file1 file2
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
-rw-rw-r--  user1 group1 system_u:object_r:samba_share_t:s0 file2
$ rm file1 file2
```

In this example, the context is defined with the **`-Z`** option. Without the **`-Z`** option, **`file2`** would be labeled with the **`unconfined_u:object_r:user_home_t`** context.

Copying a File Over an Existing File

When a file is copied over an existing file, the existing file's context is preserved (unless an option is used to preserve contexts). For example:

```
# touch /etc/file1
# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
# touch /tmp/file2
# ls -Z /tmp/file2
-rw-r--r--  root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
# cp /tmp/file2 /etc/file1
# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

In this example, two files are created: **`/etc/file1`**, labeled with the **`etc_t`** type, and **`/tmp/file2`**, labeled with the **`user_tmp_t`** type. The **`cp /tmp/file2 /etc/file1`** command overwrites **`file1`** with **`file2`**. After copying, the **`ls -Z /etc/file1`** command shows **`file1`** labeled with the **`etc_t`** type, not the **`user_tmp_t`** type from **`/tmp/file2`** that replaced **`/etc/file1`**.

Important

Copy files and directories, rather than moving them. This helps ensure they are labeled with the correct SELinux contexts. Incorrect SELinux contexts can prevent processes from accessing such files and directories.

9.4.10.2. Moving Files and Directories

File and directories keep their current SELinux context when they are moved. In many cases, this is incorrect for the location they are being moved to. The following example demonstrates moving a file from a user's home directory to **`/var/www/html/`**, which is used by the Apache HTTP Server. Since the file is moved, it does not inherit the correct SELinux context:

1. Run the **`cd`** command without any arguments to change into your home directory. Once in your home directory, run the **`touch file1`** command to create a file. This file is labeled with the **`user_home_t`** type:

   ```
   $ ls -Z file1
   -rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
   ```

2. Run the **`ls -dZ /var/www/html/`** command to view the SELinux context of the **`/var/www/html/`** directory:

   ```
   $ ls -dZ /var/www/html/
   drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
   ```

   By default, the **`/var/www/html/`** directory is labeled with the **`httpd_sys_content_t`** type. Files and directories created under the **`/var/www/html/`** directory inherit this type, and as such, they are labeled with this type.

3. As the Linux root user, run the **`mv file1 /var/www/html/`** command to move **`file1`** to the **`/var/www/html/`** directory. Since this file is moved, it keeps its current **`user_home_t`** type:

   ```
   # mv file1 /var/www/html/
   # ls -Z /var/www/html/file1
   -rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
   ```

By default, the Apache HTTP Server can not read files that are labeled with the **`user_home_t`** type. If all files comprising a web page are labeled with the **`user_home_t`** type, or another type that the Apache HTTP Server can not read, permission is denied when attempting to access them via Firefox or text-based Web browsers.

Important

Moving files and directories with the **`mv`** command may result in the wrong SELinux context, preventing processes, such as the Apache HTTP Server and Samba, from accessing such files and directories.

9.4.10.3. Checking the Default SELinux Context

Use the **`/usr/sbin/matchpathcon`** command to check if files and directories have the correct SELinux context. From the matchpathcon(8) manual page: "**`matchpathcon`** queries the system policy and outputs the default security context associated with the file path."[33]. The following example demonstrates using the **`/usr/sbin/matchpathcon`** command to verify that files in **`/var/www/html/`** directory are labeled correctly:

1. As the Linux root user, run the **`touch /var/www/html/file{1,2,3}`** command to create three files (**`file1`**, **`file2`**, and **`file3`**). These files inherit the **`httpd_sys_content_t`** type from the **`/var/www/html/`** directory:

```
# touch /var/www/html/file{1,2,3}
# ls -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2. As the Linux root user, run the **`chcon -t samba_share_t /var/www/html/file1`** command to change the **`file1`** type to **`samba_share_t`**. Note: the Apache HTTP Server can not read files or directories labeled with the **`samba_share_t`** type.

3. The **`/usr/sbin/matchpathcon -V`** option compares the current SELinux context to the correct, default context in SELinux policy. Run the **`/usr/sbin/matchpathcon -V /var/www/html/*`** command to check all files in the **`/var/www/html/`** directory:

```
$ /usr/sbin/matchpathcon -V /var/www/html/*
/var/www/html/file1 has context unconfined_u:object_r:samba_share_t:s0, should be
 system_u:object_r:httpd_sys_content_t:s0
/var/www/html/file2 verified.
/var/www/html/file3 verified.
```

The following output from the **`/usr/sbin/matchpathcon`** command explains that **`file1`** is labeled with the **`samba_share_t`** type, but should be labeled with the **`httpd_sys_content_t`** type:

```
/var/www/html/file1 has context unconfined_u:object_r:samba_share_t:s0, should be
 system_u:object_r:httpd_sys_content_t:s0
```

To resolve the label problem and allow the Apache HTTP Server access to **`file1`**, as the Linux root user, run the **`/sbin/restorecon -v /var/www/html/file1`** command:

```
# /sbin/restorecon -v /var/www/html/file1
```

[33] The matchpathcon(8) manual page, as shipped with the *libselinux-utils* package in Fedora, is written by Daniel Walsh. Any edits or changes in this version were done by Murray McAllister.

```
restorecon reset /var/www/html/file1 context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

9.4.10.4. Archiving Files with tar

`tar` does not retain extended attributes by default. Since SELinux contexts are stored in extended attributes, contexts can be lost when archiving files. Use **`tar --selinux`** to create archives that retain contexts. If a Tar archive contains files without extended attributes, or if you want the extended attributes to match the system defaults, run the archive through **`/sbin/restorecon`**:

```
$ tar -xvf archive.tar | /sbin/restorecon -f -
```

Note: depending on the directory, you may need to be the Linux root user to run the **`/sbin/restorecon`** command.

The following example demonstrates creating a Tar archive that retains SELinux contexts:

1. As the Linux root user, run the **`touch /var/www/html/file{1,2,3}`** command to create three files (**`file1`**, **`file2`**, and **`file3`**). These files inherit the **`httpd_sys_content_t`** type from the **`/var/www/html/`** directory:

   ```
   # touch /var/www/html/file{1,2,3}
   # ls -Z /var/www/html/
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
   ```

2. Run the **`cd /var/www/html/`** command to change into the **`/var/www/html/`** directory. Once in this directory, as the Linux root user, run the **`tar --selinux -cf test.tar file{1,2,3}`** command to create a Tar archive named **`test.tar`**.

3. As the Linux root user, run the **`mkdir /test`** command to create a new directory, and then, run the **`chmod 777 /test/`** command to allow all users full-access to the **`/test/`** directory.

4. Run the **`cp /var/www/html/test.tar /test/`** command to copy the **`test.tar`** file in to the **`/test/`** directory.

5. Run the **`cd /test/`** command to change into the **`/test/`** directory. Once in this directory, run the **`tar -xvf test.tar`** command to extract the Tar archive.

6. Run the **`ls -lZ /test/`** command to view the SELinux contexts. The **`httpd_sys_content_t`** type has been retained, rather than being changed to **`default_t`**, which would have happened had the **`--selinux`** not been used:

   ```
   $ ls -lZ /test/
   -rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file1
   -rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file2
   -rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file3
   -rw-r--r--  user1 group1 unconfined_u:object_r:default_t:s0 test.tar
   ```

7. If the **`/test/`** directory is no longer required, as the Linux root user, run the **`rm -ri /test/`** command to remove it, as well as all files in it.

Refer to the tar(1) manual page for further information about **tar**, such as the **--xattrs** option that retains all extended attributes.

9.4.10.5. Archiving Files with star

star does not retain extended attributes by default. Since SELinux contexts are stored in extended attributes, contexts can be lost when archiving files. Use **star -xattr -H=exustar** to create archives that retain contexts. The *star* package is not installed by default. To install **star**, run the **yum install star** command as the Linux root user.

The following example demonstrates creating a Star archive that retains SELinux contexts:

1. As the Linux root user, run the **touch /var/www/html/file{1,2,3}** command to create three files (**file1**, **file2**, and **file3**). These files inherit the **httpd_sys_content_t** type from the **/var/www/html/** directory:

```
# touch /var/www/html/file{1,2,3}
# ls -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2. Run the **cd /var/www/html/** command to change into the **/var/www/html/** directory. Once in this directory, as the Linux root user, run the **star -xattr -H=exustar -c -f=test.star file{1,2,3}** command to create a Star archive named **test.star**:

```
# star -xattr -H=exustar -c -f=test.star file{1,2,3}
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
```

3. As the Linux root user, run the **mkdir /test** command to create a new directory, and then, run the **chmod 777 /test/** command to allow all users full-access to the **/test/** directory.

4. Run the **cp /var/www/html/test.star /test/** command to copy the **test.star** file in to the **/test/** directory.

5. Run the **cd /test/** command to change into the **/test/** directory. Once in this directory, run the **star -x -f=test.star** command to extract the Star archive:

```
$ star -x -f=test.star
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
```

6. Run the **ls -lZ /test/** command to view the SELinux contexts. The **httpd_sys_content_t** type has been retained, rather than being changed to **default_t**, which would have happened had the **--selinux** not been used:

```
$ ls -lZ /test/
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file3
-rw-r--r--  user1 group1 unconfined_u:object_r:default_t:s0 test.star
```

7. If the **`/test/`** directory is no longer required, as the Linux root user, run the **`rm -ri /test/`** command to remove it, as well as all files in it.

8. If **`star`** is no longer required, as the Linux root user, run the **`yum remove star`** command to remove the package.

Refer to the star(1) manual page for further information about **`star`**.

9.5. Confining Users

A number of confined SELinux users are available in Fedora 19. Each Linux user is mapped to an SELinux user via SELinux policy, allowing Linux users to inherit the restrictions placed on SELinux users, for example (depending on the user), not being able to: run the X Window System; use networking; run setuid applications (unless SELinux policy permits it); or run the **`su`** and **`sudo`** commands. This helps protect the system from the user. Refer to *Section 9.3.3, “Confined and Unconfined Users”* for further information about confined users.

9.5.1. Linux and SELinux User Mappings

As the Linux root user, run the **`semanage login -l`** command to view the mapping between Linux users and SELinux users:

```
# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

In Fedora 19, Linux users are mapped to the SELinux **`__default__`** login by default (which is in turn mapped to the SELinux **`unconfined_u`** user). When a Linux user is created with the **`useradd`** command, if no options are specified, they are mapped to the SELinux **`unconfined_u`** user. The following defines the default-mapping:

```

__default__               unconfined_u              s0-s0:c0.c1023
```

9.5.2. Confining New Linux Users: useradd

Linux users mapped to the SELinux **`unconfined_u`** user run in the **`unconfined_t`** domain. This is seen by running the **`id -Z`** command while logged-in as a Linux user mapped to **`unconfined_u`**:

```

$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

When Linux users run in the **`unconfined_t`** domain, SELinux policy rules are applied, but policy rules exist that allow Linux users running in the **`unconfined_t`** domain almost all access. If unconfined Linux users execute an application that SELinux policy defines can transition from the **`unconfined_t`** domain to its own confined domain, unconfined Linux users are still subject to the restrictions of that confined domain. The security benefit of this is that, even though a Linux user is running unconfined, the application remains confined, and therefore, the exploitation of a flaw in the application can be limited by policy. Note: this does not protect the system from the user. Instead, the user and the system are being protected from possible damage caused by a flaw in the application.

When creating Linux users with **`useradd`**, use the **`-Z`** option to specify which SELinux user they are mapped to. The following example creates a new Linux user, useruuser, and maps that user to the SELinux **`user_u`** user. Linux users mapped to the SELinux **`user_u`** user run in the **`user_t`** domain. In this domain, Linux users are unable to run setuid applications unless SELinux policy permits it (such as **`passwd`**), and can not run **`su`** or **`sudo`**, preventing them from becoming the Linux root user with these commands.

1. As the Linux root user, run the **`/usr/sbin/useradd -Z user_u useruuser`** command to create a new Linux user (useruuser) that is mapped to the SELinux **`user_u`** user.

2. As the Linux root user, run the **`semanage login -l`** command to view the mapping between the Linux **`useruuser`** user and **`user_u`**:

```
# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
useruuser                 user_u                    s0
```

3. As the Linux root user, run the **`passwd useruuser`** command to assign a password to the Linux useruuser user:

```
# passwd useruuser
Changing password for user useruuser.
New UNIX password: Enter a password
Retype new UNIX password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

4. Log out of your current session, and log in as the Linux useruuser user. When you log in, pam_selinux maps the Linux user to an SELinux user (in this case, **`user_u`**), and sets up the resulting SELinux context. The Linux user's shell is then launched with this context. Run the **`id -Z`** command to view the context of a Linux user:

```
[useruuser@localhost ~]$ id -Z
user_u:user_r:user_t:s0
```

5. Log out of the Linux useruuser's session, and log back in with your account. If you do not want the Linux useruuser user, run the **`/usr/sbin/userdel -r useruuser`** command as the Linux root user to remove it, along with its home directory.

9.5.3. Confining Existing Linux Users: semanage login

If a Linux user is mapped to the SELinux **`unconfined_u`** user (the default behavior), and you would like to change which SELinux user they are mapped to, use the **`semanage login`** command.
The following example creates a new Linux user named newuser, then maps that Linux user to the SELinux **`user_u`** user:

1. As the Linux root user, run the **`/usr/sbin/useradd newuser`** command to create a new Linux user (newuser). Since this user uses the default mapping, it does not appear in the **`/usr/sbin/semanage login -l`** output:

```
# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

2. To map the Linux newuser user to the SELinux **`user_u`** user, run the following command as the Linux root user:

   `/usr/sbin/semanage login -a -s user_u newuser`

   The **`-a`** option adds a new record, and the **`-s`** option specifies the SELinux user to map a Linux user to. The last argument, **`newuser`**, is the Linux user you want mapped to the specified SELinux user.

3. To view the mapping between the Linux newuser user and **`user_u`**, run the **`semanage login -l`** command as the Linux root user:

```
# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
newuser                   user_u                    s0
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

4. As the Linux root user, run the **`passwd newuser`** command to assign a password to the Linux newuser user:

```
# passwd newuser
Changing password for user newuser.
New UNIX password: Enter a password
Retype new UNIX password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

5. Log out of your current session, and log in as the Linux newuser user. Run the **`id -Z`** command to view the newuser's SELinux context:

```
[newuser@rlocalhost ~]$ id -Z
user_u:user_r:user_t:s0
```

6. Log out of the Linux newuser's session, and log back in with your account. If you do not want the Linux newuser user, run the **`userdel -r newuser`** command as the Linux root user to remove it, along with its home directory. Also, the mapping between the Linux newuser user and **`user_u`** is removed:

```
# /usr/sbin/userdel -r newuser
# /usr/sbin/semanage login -l
```

```
Login Name                SELinux User              MLS/MCS Range

__default__               unconfined_u              s0-s0:c0.c1023
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

9.5.4. Changing the Default Mapping

In Fedora 19, Linux users are mapped to the SELinux **`__default__`** login by default (which is in turn mapped to the SELinux **`unconfined_u`** user). If you would like new Linux users, and Linux users not specifically mapped to an SELinux user to be confined by default, change the default mapping with the **`semanage login`** command.

For example, run the following command as the Linux root user to change the default mapping from **`unconfined_u`** to **`user_u`**:

`/usr/sbin/semanage login -m -S targeted -s "user_u" -r s0 __default__`

Run the **`semanage login -l`** command as the Linux root user to verify the **`__default__`** login is mapped to **`user_u`**:

```

# /usr/sbin/semanage login -l

Login Name                SELinux User              MLS/MCS Range

__default__               user_u                    s0
root                      unconfined_u              s0-s0:c0.c1023
system_u                  system_u                  s0-s0:c0.c1023
```

If a new Linux user is created and an SELinux user is not specified, or if an existing Linux user logs in and does not match a specific entry from the **`semanage login -l`** output, they are mapped to **`user_u`**, as per the **`__default__`** login.

To change back to the default behavior, run the following command as the Linux root user to map the **`__default__`** login to the SELinux **`unconfined_u`** user:

```
/usr/sbin/semanage login -m -S targeted -s "unconfined_u" -r\
s0-s0:c0.c1023 __default__
```

9.5.5. xguest: Kiosk Mode

The *xguest* package provides a kiosk user account. This account is used to secure machines that people walk up to and use, such as those at libraries, banks, airports, information kiosks, and coffee shops. The kiosk user account is very limited: essentially, it only allows users to log in and use **Firefox** to browse Internet websites. Any changes made while logged in with his account, such as creating files or changing settings, are lost when you log out.

To set up the kiosk account:

1. As the Linux root user, run **`yum install xguest`** command to install the *xguest* package. Install dependencies as required.

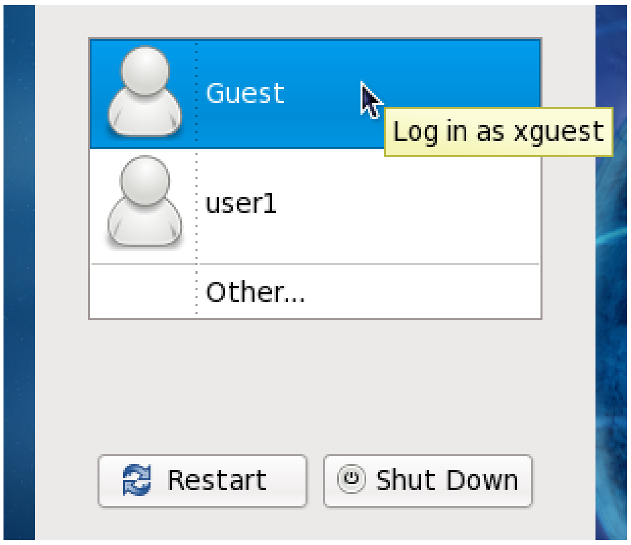
2. In order to allow the kiosk account to be used by a variety of people, the account is not password-protected, and as such, the account can only be protected if SELinux is running in enforcing mode.

Before logging in with this account, use the **getenforce** command to confirm that SELinux is running in enforcing mode:

```
$ /usr/sbin/getenforce
Enforcing
```

If this is not the case, refer to *Section 9.4.5, “SELinux Modes”* for information about changing to enforcing mode. It is not possible to log in with this account if SELinux is in permissive mode or disabled.

3. You can only log in to this account via the GNOME Display Manager (GDM). Once the *xguest* package is installed, a **Guest** account is added to GDM. To log in, click on the **Guest** account:



9.5.6. Booleans for Users Executing Applications

Not allowing Linux users to execute applications (which inherit users' permissions) in their home directories and **/tmp/**, which they have write access to, helps prevent flawed or malicious applications from modifying files that users own. In Fedora 19, by default, Linux users in the **guest_t** and **xguest_t** domains can not execute applications in their home directories or **/tmp/**; however, by default, Linux users in the **user_t** and **staff_t** domains can.

Booleans are available to change this behavior, and are configured with the **setsebool** command. The **setsebool** command must be run as the Linux root user. The **setsebool -P** command makes persistent changes. Do not use the **-P** option if you do not want changes to persist across reboots:

guest_t

To *allow* Linux users in the **guest_t** domain to execute applications in their home directories and **/tmp/**:

/usr/sbin/setsebool -P allow_guest_exec_content on

xguest_t

To *allow* Linux users in the **xguest_t** domain to execute applications in their home directories and **/tmp/**:

/usr/sbin/setsebool -P allow_xguest_exec_content on

user_t

To *prevent* Linux users in the **`user_t`** domain from executing applications in their home directories and **`/tmp/`**:

```
/usr/sbin/setsebool -P allow_user_exec_content off
```

staff_t

To *prevent* Linux users in the **`staff_t`** domain from executing applications in their home directories and **`/tmp/`**:

```
/usr/sbin/setsebool -P allow_staff_exec_content off
```

9.6. Troubleshooting

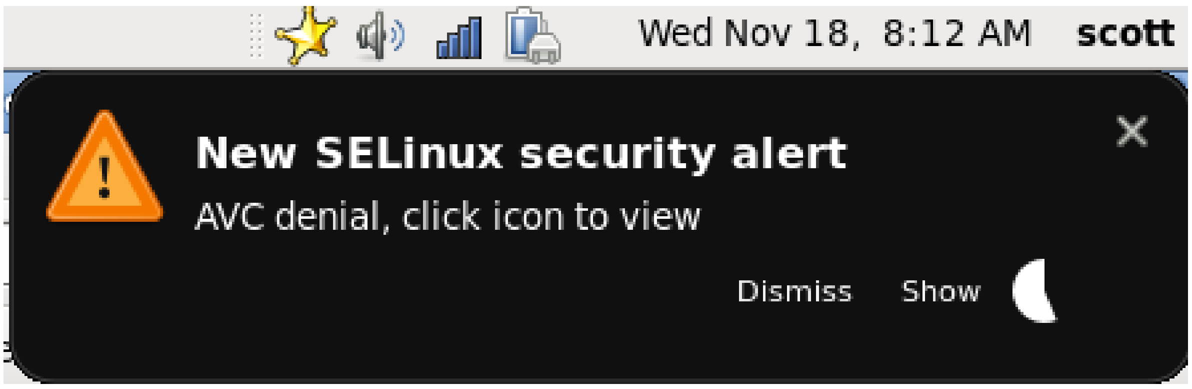
The following chapter describes what happens when SELinux denies access; the top three causes of problems; where to find information about correct labeling; analyzing SELinux denials; and creating custom policy modules with **`audit2allow`**.

9.6.1. What Happens when Access is Denied

SELinux decisions, such as allowing or disallowing access, are cached. This cache is known as the Access Vector Cache (AVC). Denial messages are logged when SELinux denies access. These denials are also known as "AVC denials", and are logged to a different location, depending on which daemons are running:

Daemon	Log Location
auditd on	**`/var/log/audit/audit.log`**
auditd off; rsyslogd on	**`/var/log/messages`**
setroubleshootd, rsyslogd, and auditd on	**`/var/log/audit/audit.log`**. Easier-to-read denial messages also sent to **`/var/log/messages`**

If you are running the X Window System, have the *setroubleshoot* and *setroubleshoot-server* packages installed, and the `setroubleshootd` and `auditd` daemons are running, a warning is displayed when access is denied by SELinux:



New SELinux security alert

AVC denial, click icon to view

Clicking on 'Show' presents a detailed analysis of why SELinux denied access, and a possible solution for allowing access. If you are not running the X Window System, it is less obvious when access is denied by SELinux. For example, users browsing your website may receive an error similar to the following:

```
Forbidden
```

```
You don't have permission to access file name on this server
```

For these situations, if DAC rules (standard Linux permissions) allow access, check **`/var/log/messages`** and **`/var/log/audit/audit.log`** for **`"SELinux is preventing"`** and **`"denied"`** errors respectively. This can be done by running the following commands as the Linux root user:

`grep "SELinux is preventing" /var/log/messages`

`grep "denied" /var/log/audit/audit.log`

9.6.2. Top Three Causes of Problems

The following sections describe the top three causes of problems: labeling problems, configuring Booleans and ports for services, and evolving SELinux rules.

9.6.2.1. Labeling Problems

On systems running SELinux, all processes and files are labeled with a label that contains security-relevant information. This information is called the SELinux context. If these labels are wrong, access may be denied. If an application is labeled incorrectly, the process it transitions to may not have the correct label, possibly causing SELinux to deny access, and the process being able to create mislabeled files.

A common cause of labeling problems is when a non-standard directory is used for a service. For example, instead of using **`/var/www/html/`** for a website, an administrator wants to use **`/srv/myweb/`**. On Fedora 19, the **`/srv/`** directory is labeled with the **`var_t`** type. Files and directories created and **`/srv/`** inherit this type. Also, newly-created top-level directories (such as **`/myserver/`**) may be labeled with the **`default_t`** type. SELinux prevents the Apache HTTP Server (`httpd`) from accessing both of these types. To allow access, SELinux must know that the files in **`/srv/myweb/`** are to be accessible to `httpd`:

```
# /usr/sbin/semanage fcontext -a -t httpd_sys_content_t \
"/srv/myweb(/.*)?"
```

This **`semanage`** command adds the context for the **`/srv/myweb/`** directory (and all files and directories under it) to the SELinux file-context configuration[34]. The **`semanage`** command does not change the context. As the Linux root user, run the **`restorecon`** command to apply the changes:

```
# /sbin/restorecon -R -v /srv/myweb
```

Refer to *Section 9.4.7.2, “Persistent Changes: semanage fcontext”* for further information about adding contexts to the file-context configuration.

9.6.2.1.1. What is the Correct Context?

The **`matchpathcon`** command checks the context of a file path and compares it to the default label for that path. The following example demonstrates using **`matchpathcon`** on a directory that contains incorrectly labeled files:

[34] Files in **`/etc/selinux/targeted/contexts/files/`** define contexts for files and directories. Files in this directory are read by **`restorecon`** and **`setfiles`** to restore files and directories to their default contexts.

```
$ /usr/sbin/matchpathcon -V /var/www/html/*
/var/www/html/index.html has context unconfined_u:object_r:user_home_t:s0, should be
 system_u:object_r:httpd_sys_content_t:s0
/var/www/html/page1.html has context unconfined_u:object_r:user_home_t:s0, should be
 system_u:object_r:httpd_sys_content_t:s0
```

In this example, the **`index.html`** and **`page1.html`** files are labeled with the **`user_home_t`** type. This type is used for files in user home directories. Using the **`mv`** command to move files from your home directory may result in files being labeled with the **`user_home_t`** type. This type should not exist outside of home directories. Use the **`restorecon`** command to restore such files to their correct type:

```
# /sbin/restorecon -v /var/www/html/index.html
restorecon reset /var/www/html/index.html context unconfined_u:object_r:user_home_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

To restore the context for all files under a directory, use the **`-R`** option:

```
# /sbin/restorecon -R -v /var/www/html/
restorecon reset /var/www/html/page1.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /var/www/html/index.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

Refer to *Section 9.4.10.3, “Checking the Default SELinux Context”* for a more detailed example of **`matchpathcon`**.

9.6.2.2. How are Confined Services Running?

Services can be run in a variety of ways. To cater for this, you must tell SELinux how you are running services. This can be achieved via Booleans that allow parts of SELinux policy to be changed at runtime, without any knowledge of SELinux policy writing. This allows changes, such as allowing services access to NFS file systems, without reloading or recompiling SELinux policy. Also, running services on non-default port numbers requires policy configuration to be updated via the **`semanage`** command.

For example, to allow the Apache HTTP Server to communicate with MySQL, turn the **`httpd_can_network_connect_db`** Boolean on:

```
# /usr/sbin/setsebool -P httpd_can_network_connect_db on
```

If access is denied for a particular service, use the **`getsebool`** and **`grep`** commands to see if any Booleans are available to allow access. For example, use the **`getsebool -a | grep ftp`** command to search for FTP related Booleans:

```
$ /usr/sbin/getsebool -a | grep ftp
allow_ftpd_anon_write --> off
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
```

For a list of Booleans and whether they are on or off, run the **`/usr/sbin/getsebool -a`** command. For a list of Booleans, an explanation of what each one is, and whether they are on or off, run the **`/usr/sbin/semanage boolean -l`** command as the Linux root user. Refer to *Section 9.4.6, "Booleans"* for information about listing and configuring Booleans.

Port Numbers

Depending on policy configuration, services may only be allowed to run on certain port numbers. Attempting to change the port a service runs on without changing policy may result in the service failing to start. For example, run the **`semanage port -l | grep http`** command as the Linux root user to list `http` related ports:

```
# /usr/sbin/semanage port -l | grep http
http_cache_port_t              tcp      3128, 8080, 8118
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

The **`http_port_t`** port type defines the ports Apache HTTP Server can listen on, which in this case, are TCP ports 80, 443, 488, 8008, 8009, and 8443. If an administrator configures **`httpd.conf`** so that `httpd` listens on port 9876 (**`Listen 9876`**), but policy is not updated to reflect this, the **`service httpd start`** command fails:

```
# /sbin/service httpd start
Starting httpd: (13)Permission denied: make_sock: could not bind to address [::]:9876
(13)Permission denied: make_sock: could not bind to address 0.0.0.0:9876
no listening sockets available, shutting down
Unable to open logs
                  [FAILED]
```

An SELinux denial similar to the following is logged to **`/var/log/audit/audit.log`**:

```
type=AVC msg=audit(1225948455.061:294): avc:  denied  { name_bind } for
 pid=4997 comm="httpd" src=9876 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=system_u:object_r:port_t:s0 tclass=tcp_socket
```

To allow `httpd` to listen on a port that is not listed for the **`http_port_t`** port type, run the **`semanage port`** command to add a port to policy configuration[35]:

```
# /usr/sbin/semanage port -a -t http_port_t -p tcp 9876
```

The **`-a`** option adds a new record; the **`-t`** option defines a type; and the **`-p`** option defines a protocol. The last argument is the port number to add.

9.6.2.3. Evolving Rules and Broken Applications

Applications may be broken, causing SELinux to deny access. Also, SELinux rules are evolving - SELinux may not have seen an application running in a certain way, possibly causing it to deny

[35] The **`semanage port -a`** command adds an entry to the **`/etc/selinux/targeted/modules/active/ports.local`** file. Note: by default, this file can only be viewed by the Linux root user.

access, even though the application is working as expected. For example, if a new version of PostgreSQL is released, it may perform actions the current policy has not seen before, causing access to be denied, even though access should be allowed.

For these situations, after access is denied, use **`audit2allow`** to create a custom policy module to allow access. Refer to *Section 9.6.3.8, “Allowing Access: audit2allow”* for information about using **`audit2allow`**.

9.6.3. Fixing Problems

The following sections help troubleshoot issues. They go over: checking Linux permissions, which are checked before SELinux rules; possible causes of SELinux denying access, but no denials being logged; manual pages for services, which contain information about labeling and Booleans; permissive domains, for allowing one process to run permissive, rather than the whole system; how to search for and view denial messages; analyzing denials; and creating custom policy modules with **`audit2allow`**.

9.6.3.1. Linux Permissions

When access is denied, check standard Linux permissions. As mentioned in *Section 9.1, “Introduction”*, most operating systems use a Discretionary Access Control (DAC) system to control access, allowing users to control the permissions of files that they own. SELinux policy rules are checked after DAC rules. SELinux policy rules are not used if DAC rules deny access first.

If access is denied and no SELinux denials are logged, use the **`ls -l`** command to view the standard Linux permissions:

```
$ ls -l /var/www/html/index.html
-rw-r----- 1 root root 0 2009-05-07 11:06 index.html
```

In this example, **`index.html`** is owned by the root user and group. The root user has read and write permissions (**`-rw`**), and members of the root group have read permissions (**`-r-`**). Everyone else has no access (**`---`**). By default, such permissions do not allow `httpd` to read this file. To resolve this issue, use the **chown** command to change the owner and group. This command must be run as the Linux root user:

```
# chown apache:apache /var/www/html/index.html
```

This assumes the default configuration, in which `httpd` runs as the Linux apache user. If you run `httpd` with a different user, replace **`apache:apache`** with that user.

9.6.3.2. Possible Causes of Silent Denials

In certain situations, AVC denials may not be logged when SELinux denies access. Applications and system library functions often probe for more access than required to perform their tasks. To maintain least privilege without filling audit logs with AVC denials for harmless application probing, the policy can silence AVC denials without allowing a permission by using **`dontaudit`** rules. These rules are common in standard policy. The downside of **`dontaudit`** is that, although SELinux denies access, denial messages are not logged, making troubleshooting hard.

To temporarily disable **`dontaudit`** rules, allowing all denials to be logged, run the following command as the Linux root user:

`/usr/sbin/semodule -DB`

The **`-D`** option disables **`dontaudit`** rules; the **`-B`** option rebuilds policy. After running **`semodule -DB`**, try exercising the application that was encountering permission problems, and see if SELinux denials — relevant to the application — are now being logged. Take care in deciding which denials should be allowed, as some should be ignored and handled via **`dontaudit`** rules. If in doubt, or in search of guidance, contact other SELinux users and developers on an SELinux list, such as *fedora-selinux-list*[36].

To rebuild policy and enable **`dontaudit`** rules, run the following command as the Linux root user:

`/usr/sbin/semodule -B`

This restores the policy to its original state. For a full list of **`dontaudit`** rules, run the **`sesearch --dontaudit`** command. Narrow down searches using the **`-s`** ***`domain`*** option and the **`grep`** command. For example:

```
$ sesearch --dontaudit -s smbd_t | grep squid
WARNING: This policy contained disabled aliases; they have been removed.
dontaudit smbd_t squid_port_t : tcp_socket name_bind ;
dontaudit smbd_t squid_port_t : udp_socket name_bind ;
```

Refer to *Section 9.6.3.6, “Raw Audit Messages”* and *Section 9.6.3.7, “sealert Messages”* for information about analyzing denials.

9.6.3.3. Manual Pages for Services

Manual pages for services contain valuable information, such as what file type to use for a given situation, and Booleans to change the access a service has (such as `httpd` accessing NFS file systems). This information may be in the standard manual page, or a manual page with **`selinux`** prepended or appended.

For example, the httpd_selinux(8) manual page has information about what file type to use for a given situation, as well as Booleans to allow scripts, sharing files, accessing directories inside user home directories, and so on. Other manual pages with SELinux information for services include:

- Samba: the samba_selinux(8) manual page describes that files and directories to be exported via Samba must be labeled with the **`samba_share_t`** type, as well as Booleans to allow files labeled with types other than **`samba_share_t`** to be exported via Samba.

- NFS: the nfs_selinux(8) manual page describes that, by default, file systems can not be exported via NFS, and that to allow file systems to be exported, Booleans such as **`nfs_export_all_ro`** or **`nfs_export_all_rw`** must be turned on.

- Berkeley Internet Name Domain (BIND): the named(8) manual page describes what file type to use for a given situation (see the **`Red Hat SELinux BIND Security Profile`** section). The named_selinux(8) manual page describes that, by default, `named` can not write to master zone files, and to allow such access, the **`named_write_master_zones`** Boolean must be turned on.

The information in manual pages helps you configure the correct file types and Booleans, helping to prevent SELinux from denying access.

[36] http://www.redhat.com/mailman/listinfo/fedora-selinux-list

9.6.3.4. Permissive Domains

When SELinux is running in permissive mode, SELinux does not deny access, but denials are logged for actions that would have been denied if running in enforcing mode. Previously, it was not possible to make a single domain permissive (remember: processes run in domains). In certain situations, this led to making the whole system permissive to troubleshoot issues.

Fedora 19 includes permissive domains, where an administrator can configure a single process (domain) to run permissive, rather than making the whole system permissive. SELinux checks are still performed for permissive domains; however, the kernel allows access and reports an AVC denial for situations where SELinux would have denied access. Permissive domains are also available in Fedora.

In Red Hat Enterprise Linux 4 and 5, ***domain*_disable_trans** Booleans are available to prevent an application from transitioning to a confined domain, and therefore, the process runs in an unconfined domain, such as **initrc_t**. Turning such Booleans on can cause major problems. For example, if the **httpd_disable_trans** Boolean is turned on:

- `httpd` runs in the unconfined **initrc_t** domain. Files created by processes running in the **initrc_t** domain may not have the same labeling rules applied as files created by a process running in the **httpd_t** domain, potentially allowing processes to create mislabeled files. This causes access problems later on.
- confined domains that are allowed to communicate with **httpd_t** can not communicate with **initrc_t**, possibly causing additional failures.

The ***domain*_disable_trans** Booleans were removed from Fedora, even though there was no replacement. Permissive domains solve the above issues: transition rules apply, and files are created with the correct labels.

Permissive domains can be used for:

- making a single process (domain) run permissive to troubleshoot an issue, rather than putting the entire system at risk by making the entire system permissive.
- creating policies for new applications. Previously, it was recommended that a minimal policy be created, and then the entire machine put into permissive mode, so that the application could run, but SELinux denials still logged. **audit2allow** could then be used to help write the policy. This put the whole system at risk. With permissive domains, only the domain in the new policy can be marked permissive, without putting the whole system at risk.

9.6.3.4.1. Making a Domain Permissive

To make a domain permissive, run the **semanage permissive -a *domain*** command, where *domain* is the domain you want to make permissive. For example, run the following command as the Linux root user to make the **httpd_t** domain (the domain the Apache HTTP Server runs in) permissive:

/usr/sbin/semanage permissive -a httpd_t

To view a list of domains you have made permissive, run the **semodule -l | grep permissive** command as the Linux root user. For example:

```
# /usr/sbin/semodule -l | grep permissive
permissive_httpd_t      1.0
```

If you no longer want a domain to be permissive, run the **`semanage permissive -d`** ***`domain`*** command as the Linux root user. For example:

`/usr/sbin/semanage permissive -d httpd_t`

9.6.3.4.2. Denials for Permissive Domains

The **`SYSCALL`** message is different for permissive domains. The following is an example AVC denial (and the associated system call) from the Apache HTTP Server:

```
type=AVC msg=audit(1226882736.442:86): avc:  denied  { getattr } for  pid=2427 comm="httpd"
 path="/var/www/html/file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

type=SYSCALL msg=audit(1226882736.442:86): arch=40000003 syscall=196 success=no exit=-13
 a0=b9a1e198 a1=bfc2921c a2=54dff4 a3=2008171 items=0 ppid=2425 pid=2427 auid=502 uid=48
 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4 comm="httpd"
 exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

By default, the **`httpd_t`** domain is not permissive, and as such, the action is denied, and the **`SYSCALL`** message contains **`success=no`**. The following is an example AVC denial for the same situation, except the **`semanage permissive -a httpd_t`** command has been run to make the **`httpd_t`** domain permissive:

```
type=AVC msg=audit(1226882925.714:136): avc:  denied  { read } for  pid=2512
 comm="httpd" name="file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

type=SYSCALL msg=audit(1226882925.714:136): arch=40000003 syscall=5 success=yes exit=11
 a0=b962a1e8 a1=8000 a2=0 a3=8000 items=0 ppid=2511 pid=2512 auid=502 uid=48 gid=48 euid=48
 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4 comm="httpd" exe="/usr/sbin/
httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

In this case, although an AVC denial was logged, access was not denied, as shown by **`success=yes`** in the **`SYSCALL`** message.

Refer to Dan Walsh's *"Permissive Domains"*[37] blog entry for further information about permissive domains.

9.6.3.5. Searching For and Viewing Denials

This section assumes the *setroubleshoot*, *setroubleshoot-server*, *dbus* and *audit* packages are installed, and that the `auditd`, `rsyslogd`, and `setroubleshootd` daemons are running. Refer to *Section 9.4.2, “Which Log File is Used”* for information about starting these daemons. A number of tools are available for searching for and viewing SELinux denials, such as **`ausearch`**, **`aureport`**, and **`sealert`**.

ausearch

The *audit* package provides **`ausearch`**. From the ausearch(8) manual page: "**`ausearch`** is a tool that can query the audit daemon logs based for events based on different search criteria"[38]. The

[37] http://danwalsh.livejournal.com/24537.html
[38] From the ausearch(8) manual page, as shipped with the *audit* package in Fedora 19.

ausearch tool accesses **/var/log/audit/audit.log**, and as such, must be run as the Linux root user:

Searching For	Command
all denials	**/sbin/ausearch -m avc**
denials for that today	**/sbin/ausearch -m avc -ts today**
denials from the last 10 minutes	**/sbin/ausearch -m avc -ts recent**

To search for SELinux denials for a particular service, use the **-c *comm-name*** option, where *comm-name* "is the executable's name"[39], for example, `httpd` for the Apache HTTP Server, and `smbd` for Samba:

/sbin/ausearch -m avc -c httpd

/sbin/ausearch -m avc -c smbd

Refer to the ausearch(8) manual page for further **ausearch** options.

aureport

The *audit* package provides **aureport**. From the aureport(8) manual page: "**aureport** is a tool that produces summary reports of the audit system logs"[40]. The **aureport** tool accesses **/var/log/audit/audit.log**, and as such, must be run as the Linux root user. To view a list of SELinux denials and how often each one occurred, run the **aureport -a** command. The following is example output that includes two denials:

```
# /sbin/aureport -a

AVC Report
========================================================
# date time comm subj syscall class permission obj event
========================================================
1. 05/01/2009 21:41:39 httpd unconfined_u:system_r:httpd_t:s0 195 file getattr
 system_u:object_r:samba_share_t:s0 denied 2
2. 05/03/2009 22:00:25 vsftpd unconfined_u:system_r:ftpd_t:s0 5 file read
 unconfined_u:object_r:cifs_t:s0 denied 4
```

Refer to the aureport(8) manual page for further **aureport** options.

sealert

The *setroubleshoot-server* package provides **sealert**, which reads denial messages translated by *setroubleshoot-server*. Denials are assigned IDs, as seen in **/var/log/messages**. The following is an example denial from **messages**:

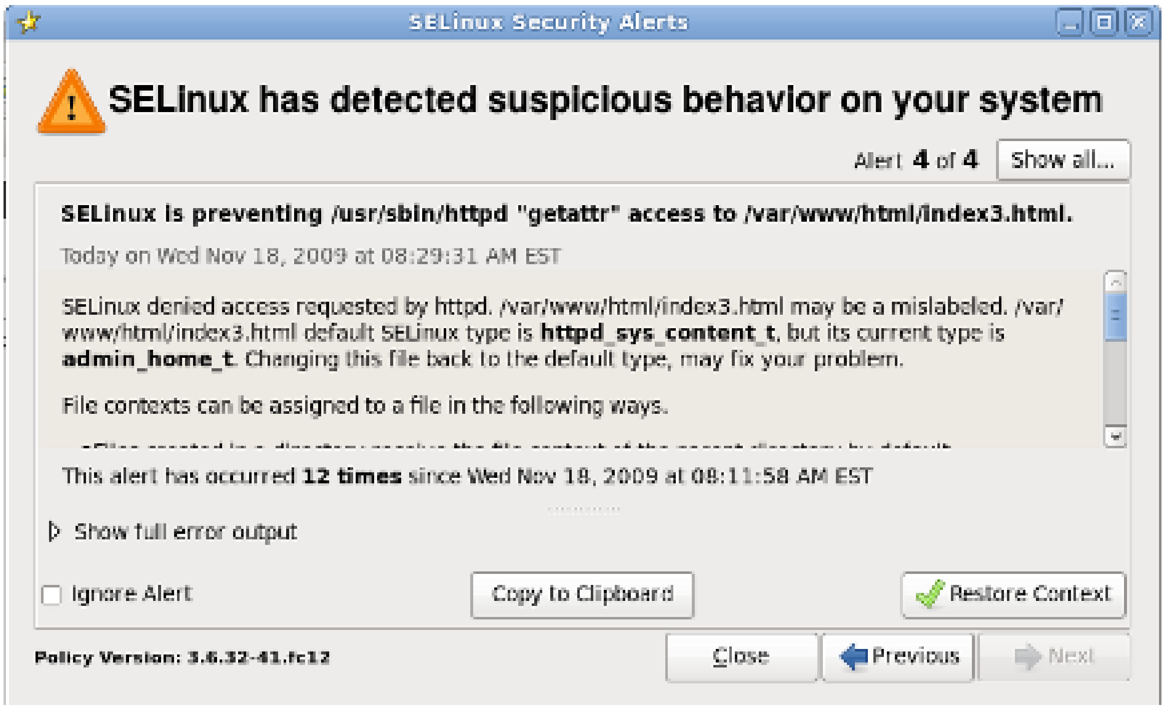
```
setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to /var/www/html/
file1 (samba_share_t). For complete SELinux messages. run sealert -l 84e0b04d-
d0ad-4347-8317-22e74f6cd020
```

In this example, the denial ID is **84e0b04d-d0ad-4347-8317-22e74f6cd020**. The **-l** option takes an ID as an argument. Running the **sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020**

[39] From the ausearch(8) manual page, as shipped with the *audit* package in Fedora 19.
[40] From the aureport(8) manual page, as shipped with the *audit* package in Fedora 19.

command presents a detailed analysis of why SELinux denied access, and a possible solution for allowing access.

If you are running the X Window System, have the *setroubleshoot* and *setroubleshoot-server* packages installed, and the `setroubleshootd`, `dbus` and `auditd` daemons are running, a warning is displayed when access is denied by SELinux. Clicking on 'Show' launches the **`sealert`** GUI, and displays denials in HTML output:


SELinux Security Alerts

SELinux has detected suspicious behavior on your system

Alert 4 of 4 Show all...

SELinux is preventing /usr/sbin/httpd "getattr" access to /var/www/html/index3.html.

Today on Wed Nov 18, 2009 at 08:29:31 AM EST

SELinux denied access requested by httpd. /var/www/html/index3.html may be a mislabeled. /var/www/html/index3.html default SELinux type is **httpd_sys_content_t**, but its current type is **admin_home_t**. Changing this file back to the default type, may fix your problem.

File contexts can be assigned to a file in the following ways.

This alert has occurred **12 times** since Wed Nov 18, 2009 at 08:11:58 AM EST

- Run the **`sealert -b`** command to launch the **`sealert`** GUI.
- Run the **`sealert -l \*`** command to view a detailed analysis of all denials.
- As the Linux root user, run the **`sealert -a /var/log/audit/audit.log -H > audit.html`** command to create a HTML version of the **`sealert`** analysis, as seen with the **`sealert`** GUI.

Refer to the sealert(8) manual page for further **`sealert`** options.

9.6.3.6. Raw Audit Messages

Raw audit messages are logged to **`/var/log/audit/audit.log`**. The following is an example AVC denial (and the associated system call) that occurred when the Apache HTTP Server (running in the **`httpd_t`** domain) attempted to access the **`/var/www/html/file1`** file (labeled with the **`samba_share_t`** type):

```
type=AVC msg=audit(1226874073.147:96): avc:  denied  { getattr } for  pid=2465 comm="httpd"
 path="/var/www/html/file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

type=SYSCALL msg=audit(1226874073.147:96): arch=40000003 syscall=196 success=no exit=-13
 a0=b98df198 a1=bfec85dc a2=54dff4 a3=2008171 items=0 ppid=2463 pid=2465 auid=502 uid=48
 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=6 comm="httpd"
 exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

`{ getattr }`

The item in braces indicates the permission that was denied. **`getattr`** indicates the source process was trying to read the target file's status information. This occurs before reading files. This action is denied due to the file being accessed having the wrong label. Commonly seen permissions include **`getattr`**, **`read`**, and **`write`**.

comm="*`httpd`*"

The executable that launched the process. The full path of the executable is found in the **`exe=`** section of the system call (**`SYSCALL`**) message, which in this case, is **`exe="/usr/sbin/httpd"`**.

path="*`/var/www/html/file1`*"

The path to the object (target) the process attempted to access.

scontext="*`unconfined_u:system_r:httpd_t:s0`*"

The SELinux context of the process that attempted the denied action. In this case, it is the SELinux context of the Apache HTTP Server, which is running in the **`httpd_t`** domain.

tcontext="*`unconfined_u:object_r:samba_share_t:s0`*"

The SELinux context of the object (target) the process attempted to access. In this case, it is the SELinux context of **`file1`**. Note: the **`samba_share_t`** type is not accessible to processes running in the **`httpd_t`** domain.

In certain situations, the **`tcontext`** may match the **`scontext`**, for example, when a process attempts to execute a system service that will change characteristics of that running process, such as the user ID. Also, the **`tcontext`** may match the **`scontext`** when a process tries to use more resources (such as memory) than normal limits allow, resulting in a security check to see if that process is allowed to break those limits.

From the system call (**`SYSCALL`**) message, two items are of interest:

- **`success=`*`no`***: indicates whether the denial (AVC) was enforced or not. **`success=no`** indicates the system call was not successful (SELinux denied access). **`success=yes`** indicates the system call was successful - this can be seen for permissive domains or unconfined domains, such as **`initrc_t`** and **`kernel_t`**.
- **`exe="`*`/usr/sbin/httpd`*`"`**: the full path to the executable that launched the process, which in this case, is **`exe="/usr/sbin/httpd"`**.

An incorrect file type is a common cause for SELinux denying access. To start troubleshooting, compare the source context (**`scontext`**) with the target context (**`tcontext`**). Should the process (**`scontext`**) be accessing such an object (**`tcontext`**)? For example, the Apache HTTP Server (**`httpd_t`**) should only be accessing types specified in the httpd_selinux(8) manual page, such as **`httpd_sys_content_t`**, **`public_content_t`**, and so on, unless configured otherwise.

9.6.3.7. sealert Messages

Denials are assigned IDs, as seen in **`/var/log/messages`**. The following is an example AVC denial (logged to **`messages`**) that occurred when the Apache HTTP Server (running in the **`httpd_t`** domain) attempted to access the **`/var/www/html/file1`** file (labeled with the **`samba_share_t`** type):

```
hostname setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to /var/www/
html/file1 (samba_share_t). For complete SELinux messages. run sealert -l 84e0b04d-
d0ad-4347-8317-22e74f6cd020
```

As suggested, run the **`sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020`** command to view the complete message. This command only works on the local machine, and presents the same information as the **`sealert`** GUI:

```
$ sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020

Summary:

SELinux is preventing httpd (httpd_t) "getattr" to /var/www/html/file1
(samba_share_t).

Detailed Description:

SELinux denied access to /var/www/html/file1 requested by httpd.
/var/www/html/file1 has a context used for sharing by different program. If you
would like to share /var/www/html/file1 from httpd also, you need to change its
file context to public_content_t. If you did not intend to this access, this
could signal a intrusion attempt.

Allowing Access:

You can alter the file context by executing chcon -t public_content_t
'/var/www/html/file1'

Fix Command:

chcon -t public_content_t '/var/www/html/file1'

Additional Information:

Source Context                unconfined_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:samba_share_t:s0
Target Objects                /var/www/html/file1 [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          hostname
Source RPM Packages           httpd-2.2.10-2
Target RPM Packages
Policy RPM                    selinux-policy-3.5.13-11.fc12
Selinux Enabled               True
Policy Type                   targeted
MLS Enabled                   True
Enforcing Mode                Enforcing
Plugin Name                   public_content
Host Name                     hostname
Platform                      Linux hostname 2.6.27.4-68.fc12.i686 #1 SMP Thu Oct
30 00:49:42 EDT 2008 i686 i686
Alert Count                   4
First Seen                    Wed Nov  5 18:53:05 2008
Last Seen                     Wed Nov  5 01:22:58 2008
Local ID                      84e0b04d-d0ad-4347-8317-22e74f6cd020
Line Numbers

Raw Audit Messages

node=hostname type=AVC msg=audit(1225812178.788:101): avc:  denied  { getattr }
 for  pid=2441 comm="httpd" path="/var/www/html/file1" dev=dm-0 ino=284916
 scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:samba_share_t:s0
 tclass=file

node=hostname type=SYSCALL msg=audit(1225812178.788:101): arch=40000003 syscall=196
 success=no exit=-13 a0=b8e97188 a1=bf87aaac a2=54dff4 a3=2008171 items=0 ppid=2439 pid=2441
```

```
auid=502 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=3
comm="httpd" exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

Summary

A brief summary of the denied action. This is the same as the denial in **`/var/log/messages`**. In this example, the `httpd` process was denied access to a file (**`file1`**), which is labeled with the **`samba_share_t`** type.

Detailed Description

A more verbose description. In this example, **`file1`** is labeled with the **`samba_share_t`** type. This type is used for files and directories that you want to export via Samba. The description suggests changing the type to a type that can be accessed by the Apache HTTP Server and Samba, if such access is desired.

Allowing Access

A suggestion for how to allow access. This may be relabeling files, turning a Boolean on, or making a local policy module. In this case, the suggestion is to label the file with a type accessible to both the Apache HTTP Server and Samba.

Fix Command

A suggested command to allow access and resolve the denial. In this example, it gives the command to change the **`file1`** type to **`public_content_t`**, which is accessible to the Apache HTTP Server and Samba.

Additional Information

Information that is useful in bug reports, such as the policy package name and version (**`selinux-policy-3.5.13-11.fc12`**), but may not help towards solving why the denial occurred.

Raw Audit Messages

The raw audit messages from **`/var/log/audit/audit.log`** that are associated with the denial. Refer to *Section 9.6.3.6, “Raw Audit Messages”* for information about each item in the AVC denial.

9.6.3.8. Allowing Access: audit2allow

Do not use the example in this section in production. It is used only to demonstrate the use of **`audit2allow`**.

From the audit2allow(1) manual page: "**`audit2allow`** - generate SELinux policy allow rules from logs of denied operations"[41]. After analyzing denials as per *Section 9.6.3.7, “sealert Messages”*, and if no label changes or Booleans allowed access, use **`audit2allow`** to create a local policy module. After access is denied by SELinux, running the **`audit2allow`** command presents Type Enforcement rules that allow the previously denied access.

The following example demonstrates using **`audit2allow`** to create a policy module:

1. A denial and the associated system call are logged to **`/var/log/audit/audit.log`**:

```
type=AVC msg=audit(1226270358.848:238): avc:  denied  { write }
 for  pid=13349 comm="certwatch" name="cache" dev=dm-0 ino=218171
 scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0 tclass=dir

type=SYSCALL msg=audit(1226270358.848:238): arch=40000003 syscall=39 success=no exit=-13
 a0=39a2bf a1=3ff a2=3a0354 a3=94703c8 items=0 ppid=13344 pid=13349 auid=4294967295
```

[41] From the audit2allow(1) manual page, as shipped with the *policycoreutils* package in Fedora 19.

```
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295
comm="certwatch" exe="/usr/bin/certwatch" subj=system_u:system_r:certwatch_t:s0
key=(null)
```

In this example, **certwatch** (**`comm="certwatch"`**) was denied write access (**`{ write }`**) to a directory labeled with the **`var_t`** type (**`tcontext=system_u:object_r:var_t:s0`**). Analyze the denial as per *Section 9.6.3.7, "sealert Messages"*. If no label changes or Booleans allowed access, use **`audit2allow`** to create a local policy module.

2. With a denial logged, such as the **`certwatch`** denial in step 1, run the **`audit2allow -w -a`** command to produce a human-readable description of why access was denied. The **`-a`** option causes all audit logs to be read. The **`-w`** option produces the human-readable description. The **`audit2allow`** tool accesses **`/var/log/audit/audit.log`**, and as such, must be run as the Linux root user:

```
# audit2allow -w -a
type=AVC msg=audit(1226270358.848:238): avc:  denied  { write }
 for  pid=13349 comm="certwatch" name="cache" dev=dm-0 ino=218171
 scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0 tclass=dir
 Was caused by:
  Missing type enforcement (TE) allow rule.

 You can use audit2allow to generate a loadable module to allow this access.
```

As shown, access was denied due to a missing Type Enforcement rule.

3. Run the **`audit2allow -a`** command to view the Type Enforcement rule that allows the denied access:

```
# audit2allow -a


#============= certwatch_t ==============
allow certwatch_t var_t:dir write;
```

Important

Missing Type Enforcement rules are usually caused by bugs in SELinux policy, and should be reported in *Red Hat Bugzilla*[42]. For Fedora, create bugs against the **`Fedora`** product, and select the **`selinux-policy`** component. Include the output of the **`audit2allow -w -a`** and **`audit2allow -a`** commands in such bug reports.

4. To use the rule displayed by **`audit2allow -a`**, run the **`audit2allow -a -M`** ***`mycertwatch`*** command as the Linux root user to create custom module. The **`-M`** option creates a Type Enforcement file (**`.te`**) with the name specified with **`-M`**, in your current working directory:

[42] https://bugzilla.redhat.com/

```
# audit2allow -a -M mycertwatch

******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i mycertwatch.pp

# ls
mycertwatch.pp  mycertwatch.te
```

Also, **audit2allow** compiles the Type Enforcement rule into a policy package (**.pp**). To install the module, run the **/usr/sbin/semodule -i *mycertwatch.pp*** command as the Linux root user.

Important

Modules created with **audit2allow** may allow more access than required. It is recommended that policy created with **audit2allow** be posted to an SELinux list, such as *fedora-selinux-list*[43], for review. If you believe their is a bug in policy, create a bug in *Red Hat Bugzilla*[44].

If you have multiple denials from multiple processes, but only want to create a custom policy for a single process, use the **grep** command to narrow down the input for **audit2allow**. The following example demonstrates using **grep** to only send denials related to **certwatch** through **audit2allow**:

```
# grep certwatch /var/log/audit/audit.log | audit2allow -M mycertwatch2
******************** IMPORTANT ***********************
To make this policy package active, execute:

# /usr/sbin/semodule -i mycertwatch2.pp
```

Refer to Dan Walsh's *"Using audit2allow to build policy modules. Revisited."*[45] blog entry for further information about using **audit2allow** to build policy modules.

9.7. Further Information

9.7.1. Contributors

- *Geert Warrink*[46] (translation - Dutch)
- *Domingo Becker*[47] (translation - Spanish)
- *Daniel Cabrera*[48] (translation - Spanish)

[43] http://www.redhat.com/mailman/listinfo/fedora-selinux-list
[44] https://bugzilla.redhat.com/
[45] http://danwalsh.livejournal.com/24750.html
[46] http://fedoraproject.org/wiki/GeertWarrink
[47] http://fedoraproject.org/wiki/User:Beckerde
[48] http://fedoraproject.org/wiki/User:Logan

9.7.2. Other Resources

The National Security Agency (NSA)

From the NSA *Contributors to SELinux*[49] page:

Researchers in NSA's National Information Assurance Research Laboratory (NIARL) designed and implemented flexible mandatory access controls in the major subsystems of the Linux kernel and implemented the new operating system components provided by the Flask architecture, namely the security server and the access vector cache. The NSA researchers reworked the LSM-based SELinux for inclusion in Linux 2.6. NSA has also led the development of similar controls for the X Window System (XACE/XSELinux) and for Xen (XSM/Flask).

- Main SELinux website: *http://www.nsa.gov/research/selinux/index.shtml*.
- SELinux documentation: *http://www.nsa.gov/research/selinux/docs.shtml*.
- SELinux background: *http://www.nsa.gov/research/selinux/background.shtml*.

Tresys Technology

Tresys Technology[50] are the upstream for:

- *SELinux userland libraries and tools*[51].
- *SELinux Reference Policy*[52].

SELinux News

- News: *http://selinuxnews.org/wp/*.
- Planet SELinux (blogs): *http://selinuxnews.org/planet/*.

SELinux Project Wiki

- Main page: *http://selinuxproject.org/page/Main_Page*.
- User resources, including links to documentation, mailing lists, websites, and tools: *http://selinuxproject.org/page/User_Resources*.

Red Hat Enterprise Linux

- The *Red Hat Enterprise Linux Deployment Guide*[53] contains an SELinux *References*[54] section, that has links to SELinux tutorials, general information, and the technology behind SELinux.
- The *Red Hat Enterprise Linux 4 SELinux Guide*[55].

[49] http://www.nsa.gov/research/selinux/contrib.shtml
[50] http://www.tresys.com/
[51] http://userspace.selinuxproject.org/trac/
[52] http://oss.tresys.com/projects/refpolicy
[53] http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/html/Deployment_Guide/index.html
[54] http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/html/Deployment_Guide/selg-chapter-0054.html
[55] http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide/index.html

Fedora

- Main page: *http://fedoraproject.org/wiki/SELinux*.
- Troubleshooting: *http://fedoraproject.org/wiki/SELinux/Troubleshooting*.
- Fedora SELinux FAQ: *http://docs.fedoraproject.org/selinux-faq/*.
- SELinux Managing Confined Services Guide: *http://docs.fedoraproject.org/selinux-managing-confined-services-guide/*

The UnOfficial SELinux FAQ

http://www.crypt.gen.nz/selinux/faq.html

IRC

On *Freenode*[56]:

- #selinux
- #fedora-selinux
- #security

[56] http://freenode.net/

Managing Confined Services

10.1. Introduction

Security-Enhanced Linux (SELinux) refers to files, such as directories and devices, as objects. Processes, such as a user running a command or the Mozilla® Firefox® application, are referred to as subjects. Most operating systems use a Discretionary Access Control (DAC) system that controls how subjects interact with objects, and how subjects interact with each other. On operating systems using DAC, users control the permissions of files (objects) that they own. For example, on Linux® operating systems, users could make their home directories world-readable, inadvertently giving users and processes (subjects) access to potentially sensitive information.

DAC mechanisms are fundamentally inadequate for strong system security. DAC access decisions are only based on user identity and ownership, ignoring other security-relevant information such as the role of the user, the function and trustworthiness of the program, and the sensitivity and integrity of the data. Each user has complete discretion over their files, making it impossible to enforce a system-wide security policy. Furthermore, every program run by a user inherits all of the permissions granted to the user and is free to change access to the user's files, so no protection is provided against malicious software. Many system services and privileged programs must run with coarse-grained privileges that far exceed their requirements, so that a flaw in any one of these programs can be exploited to obtain complete system access.[1]

The following is an example of permissions used on Linux operating systems that do not run Security-Enhanced Linux (SELinux). The permissions in these examples may differ from your system. Use the **`ls -l`** command to view file permissions:

```
$ ls -l file1
-rwxrw-r-- 1 user1 group1 0 2010-02-28 07:12 file1
```

The first three permission bits, **`rwx`**, control the access the Linux **`user1`** user (in this case, the owner) has to **`file1`**. The next three permission bits, **`rw-`**, control the access the Linux **`group1`** group has to **`file1`**. The last three permission bits, **`r--`**, control the access everyone else has to **`file1`**, which includes all users and processes.

Security-Enhanced Linux (SELinux) adds Mandatory Access Control (MAC) to the Linux kernel, and is enabled by default in Fedora. A general purpose MAC architecture needs the ability to enforce an administratively-set security policy over all processes and files in the system, basing decisions on labels containing a variety of security-relevant information. When properly implemented, it enables a system to adequately defend itself and offers critical support for application security by protecting against the tampering with, and bypassing of, secured applications. MAC provides strong separation of applications that permits the safe execution of untrustworthy applications. Its ability to limit the privileges associated with executing processes limits the scope of potential damage that can result from the exploitation of vulnerabilities in applications and system services. MAC enables information

[1] "Integrating Flexible Support for Security Policies into the Linux Operating System", by Peter Loscocco and Stephen Smalley. This paper was originally prepared for the National Security Agency and is, consequently, in the public domain. Refer to the *original paper* [http://www.nsa.gov/research/_files/selinux/papers/freenix01/index.shtml] for details and the document as it was first released. Any edits and changes were done by Murray McAllister.

to be protected from legitimate users with limited authorization as well as from authorized users who have unwittingly executed malicious applications.[2]

The following is an example of the labels containing security-relevant information that are used on processes, Linux users, and files, on Linux operating systems that run SELinux. This information is called the SELinux context, and is viewed using the **ls -Z** command:

```
$ ls -Z file1
-rwxrw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0      file1
```

In this example, SELinux provides a user (**unconfined_u**), a role (**object_r**), a type (**user_home_t**), and a level (**s0**). This information is used to make access control decisions. This example also displays the DAC rules, which are shown in the SELinux context via the **ls -Z** command. SELinux policy rules are checked after DAC rules. SELinux policy rules are not used if DAC rules deny access first.

10.2. Targeted policy

Targeted policy is the default SELinux policy used in Fedora. When using targeted policy, processes that are targeted run in a confined domain, and processes that are not targeted run in an unconfined domain. For example, by default, logged in users run in the **unconfined_t** domain, and system processes started by init run in the **initrc_t** domain - both of these domains are unconfined.

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. This can be achieved via Booleans that allow parts of SELinux policy to be changed at runtime, without any knowledge of SELinux policy writing. This allows changes, such as allowing services access to NFS file systems, without reloading or recompiling SELinux policy. Boolean configuration is discussed later.

Other changes, such as using non-default directories to store files for services, and changing services to run on non-default port numbers, require policy configuration to be updated via tools such as **semanage**. This is discussed later using detailed configuration examples.

10.2.1. Type Enforcement

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

10.2.2. Confined processes

Almost every service that listens on a network is confined in Fedora. Also, most processes that run as the root user and perform tasks for users, such as the **passwd** application, are confined. When a process is confined, it runs in its own domain, such as the `httpd` process running in the **httpd_t** domain. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited.

[2] "Meeting Critical Security Objectives with Security-Enhanced Linux", by Peter Loscocco and Stephen Smalley. This paper was originally prepared for the National Security Agency and is, consequently, in the public domain. Refer to the *original paper* [http://www.nsa.gov/research/_files/selinux/papers/ottawa01/index.shtml] for details and the document as it was first released. Any edits and changes were done by Murray McAllister.

The following example demonstrates how SELinux prevents the Apache HTTP Server (`httpd`) from reading files that are not correctly labeled, such as files intended for use by Samba. This is an example, and should not be used in production. It assumes that the *httpd*, *wget*, *setroubleshoot-server*, and *audit* packages are installed, that the SELinux targeted policy is used, and that SELinux is running in enforcing mode:

1. Run the **`sestatus`** command to confirm that SELinux is enabled, is running in enforcing mode, and that targeted policy is being used:

   ```
   $ /usr/sbin/sestatus
   SELinux status:                 enabled
   SELinuxfs mount:                /selinux
   Current mode:                   enforcing
   Mode from config file:          enforcing
   Policy version:                 24
   Policy from config file:        targeted
   ```

   `SELinux status: enabled` is returned when SELinux is enabled. **`Current mode: enforcing`** is returned when SELinux is running in enforcing mode. **`Policy from config file: targeted`** is returned when the SELinux targeted policy is used.

2. As the root user, run the **`touch /var/www/html/testfile`** command to create a file.

3. Run the **`ls -Z /var/www/html/testfile`** command to view the SELinux context:

   ```
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/testfile
   ```

   The **`testfile`** file is labeled with the SELinux **`unconfined_u`** user because a Linux user that is mapped to the **`unconfined_u`** SELinux user created the file. Role-Based Access Control (RBAC) is used for processes, not files. Roles do not have a meaning for files - the **`object_r`** role is a generic role used for files (on persistent storage and network file systems). Under the **`/proc/`** directory, files related to processes may use the **`system_r`** role.[3] The **`httpd_sys_content_t`** type allows the `httpd` process to access this file.

4. As the root user, run the **`service httpd start`** command to start the `httpd` process. The output is as follows if `httpd` starts successfully:

   ```
   # /sbin/service httpd start
   Starting httpd:                                            [  OK  ]
   ```

5. Change into a directory where your Linux user has write access to, and run the **`wget http://localhost/testfile`** command. Unless there are changes to the default configuration, this command succeeds:

   ```
   --2010-02-28 08:44:36--  http://localhost/testfile
   Resolving localhost... 127.0.0.1
   Connecting to localhost|127.0.0.1|:80... connected.
   HTTP request sent, awaiting response... 200 OK
   Length: 0 [text/plain]
   Saving to: `testfile'

   [ <=>                              ] 0     --.-K/s   in 0s
   ```

[3] When using other policies, such as MLS, other roles may be used, for example, **`secadm_r`**.

```
2010-02-28 08:44:36 (0.00 B/s) - `testfile' saved [0/0]
```

6. The **chcon** command relabels files; however, such label changes do not survive when the file system is relabeled. For permanent changes that survive a file system relabel, use the **`semanage`** command, which is discussed later. As the root user, run the following command to change the type to a type used by Samba:

   `chcon -t samba_share_t /var/www/html/testfile`

   Run the **`ls -Z /var/www/html/testfile`** command to view the changes:

```
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/testfile
```

7. Note: the current DAC permissions allow the `httpd` process access to **`testfile`**. Change into a directory where your Linux user has write access to, and run the **`wget http://localhost/testfile`** command. Unless there are changes to the default configuration, this command fails:

```
--2010-02-28 08:45:07--  http://localhost/testfile
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2010-02-28 08:45:08 ERROR 403: Forbidden.
```

8. As the root user, run the **`rm -i /var/www/html/testfile`** command to remove **`testfile`**.

9. If you do not require `httpd` to be running, as the root user, run the **`service httpd stop`** command to stop `httpd`:

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK  ]
```

This example demonstrates the additional security added by SELinux. DAC rules allowed the `httpd` process access to **`testfile`** in step 7, but because the file was labeled with a type that the `httpd` process does not have access to, SELinux denied access. After step 7, an error similar to the following is logged to **`/var/log/messages`**:

```
Apr  6 23:00:54 localhost setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr"
to /var/www/html/testfile (samba_share_t). For complete SELinux messages.
run sealert -l c05911d3-e680-4e42-8e36-fe2ab9f8e654
```

Previous log files may use a **`/var/log/messages.`*`YYYYMMDD`*** format. When running **syslog-ng**, previous log files may use a **`/var/log/messages.`*`X`*** format. If the `setroubleshootd` and `auditd` processes are running, errors similar to the following are logged to **`/var/log/audit/audit.log`**:

```
type=AVC msg=audit(1220706212.937:70): avc:  denied  { getattr } for  pid=1904 comm="httpd"
 path="/var/www/html/testfile" dev=sda5 ino=247576 scontext=unconfined_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:samba_share_t:s0  tclass=file

type=SYSCALL msg=audit(1220706212.937:70): arch=40000003 syscall=196 success=no exit=-13
 a0=b9e21da0 a1=bf9581dc a2=555ff4 a3=2008171 items=0 ppid=1902 pid=1904 auid=500 uid=48
 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd"
 exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

Also, an error similar to the following is logged to **`/var/log/httpd/error_log`**:

```
[Sat Apr 06 23:00:54 2009] [error] [client 127.0.0.1] (13)Permission denied: access to /
testfile denied
```

10.2.3. Unconfined processes

Unconfined processes run in unconfined domains. For example, init programs run in the unconfined **`initrc_t`** domain, unconfined kernel processes run in the **`kernel_t`** domain, and unconfined Linux users run in the **`unconfined_t`** domain. For unconfined processes, SELinux policy rules are applied, but policy rules exist that allow processes running in unconfined domains almost all access. Processes running in unconfined domains fall back to using DAC rules exclusively. If an unconfined process is compromised, SELinux does not prevent an attacker from gaining access to system resources and data, but of course, DAC rules are still used. SELinux is a security enhancement on top of DAC rules - it does not replace them.

The following example demonstrates how the Apache HTTP Server (`httpd`) can access data intended for use by Samba, when running unconfined. Note: in Fedora, the `httpd` process runs in the confined **`httpd_t`** domain by default. This is an example, and should not be used in production. It assumes that the *httpd*, *wget*, *setroubleshoot-server*, and *audit* packages are installed, that the SELinux targeted policy is used, and that SELinux is running in enforcing mode:

1. Run the **`sestatus`** command to confirm that SELinux is enabled, is running in enforcing mode, and that targeted policy is being used:

   ```
   $ /usr/sbin/sestatus
   SELinux status:                 enabled
   SELinuxfs mount:                /selinux
   Current mode:                   enforcing
   Mode from config file:          enforcing
   Policy version:                 24
   Policy from config file:        targeted
   ```

   `SELinux status: enabled` is returned when SELinux is enabled. **`Current mode: enforcing`** is returned when SELinux is running in enforcing mode. **`Policy from config file: targeted`** is returned when the SELinux targeted policy is used.

2. As the root user, run the **`touch /var/www/html/test2file`** command to create a file.

3. Run the **`ls -Z /var/www/html/test2file`** command to view the SELinux context:

   ```
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
   test2file
   ```

   `test2file` is labeled with the SELinux **`unconfined_u`** user because a Linux user that is mapped to the **`unconfined_u`** SELinux user created the file. RBAC is used for processes, not files. Roles do not have a meaning for files - the **`object_r`** role is a generic role used for files (on persistent storage and network file systems). Under the **`/proc/`** directory, files related to processes may use the **`system_r`** role.[4] The **`httpd_sys_content_t`** type allows the `httpd` process to access this file.

4. The **chcon** command relabels files; however, such label changes do not survive when the file system is relabeled. For permanent changes that survive a file system relabel, use the **semanage**

[4] When using other policies, such as MLS, other roles may also be used, for example, **`secadm_r`**.

command, which is discussed later. As the root user, run the following command to change the type to a type used by Samba:

`chcon -t samba_share_t /var/www/html/test2file`

Run the **`ls -Z /var/www/html/test2file`** command to view the changes:

```
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/test2file
```

5. Run the **`service httpd status`** command to confirm that the `httpd` process is not running:

```
$ /sbin/service httpd status
httpd is stopped
```

If the output differs, run the **`service httpd stop`** command as the root user to stop the `httpd` process:

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK  ]
```

6. To make the `httpd` process run unconfined, run the following command as the root user to change the type of **`/usr/sbin/httpd`**, to a type that does not transition to a confined domain:

`chcon -t unconfined_exec_t /usr/sbin/httpd`

7. Run the **`ls -Z /usr/sbin/httpd`** command to confirm that **`/usr/sbin/httpd`** is labeled with the **`unconfined_exec_t`** type:

```
-rwxr-xr-x  root root system_u:object_r:unconfined_exec_t /usr/sbin/httpd
```

8. As the root user, run the **`service httpd start`** command to start the `httpd` process. The output is as follows if `httpd` starts successfully:

```
# /sbin/service httpd start
Starting httpd:                                            [  OK  ]
```

9. Run the **`ps -eZ | grep httpd`** command to view the `httpd` processes running in the **`unconfined_t`** domain:

```
$ ps -eZ | grep httpd
unconfined_u:system_r:unconfined_t 7721 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7723 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7724 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7725 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7726 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7727 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7728 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7729 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7730 ?      00:00:00 httpd
```

10. Change into a directory where your Linux user has write access to, and run the **`wget http://localhost/test2file`** command. Unless there are changes to the default configuration, this command succeeds:

```
--2008-09-07 01:41:10--  http://localhost/test2file
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `test2file.1'

[ <=>                           ]--.-K/s   in 0s

2008-09-07 01:41:10 (0.00 B/s) - `test2file.1' saved [0/0]
```

Although the `httpd` process does not have access to files labeled with the **`samba_share_t`** type, `httpd` is running in the unconfined **`unconfined_t`** domain, and falls back to using DAC rules, and as such, the **`wget`** command succeeds. Had `httpd` been running in the confined **`httpd_t`** domain, the **`wget`** command would have failed.

11. The **`restorecon`** command restores the default SELinux context for files. As the root user, run the **`restorecon -v /usr/sbin/httpd`** command to restore the default SELinux context for **`/usr/sbin/httpd`**:

```
# /sbin/restorecon -v /usr/sbin/httpd
restorecon reset /usr/sbin/httpd context system_u:object_r:unconfined_notrans_exec_t:s0-
>system_u:object_r:httpd_exec_t:s0
```

Run the **`ls -Z /usr/sbin/httpd`** command to confirm that **`/usr/sbin/httpd`** is labeled with the **`httpd_exec_t`** type:

```
$ ls -Z /usr/sbin/httpd
-rwxr-xr-x  root root system_u:object_r:httpd_exec_t   /usr/sbin/httpd
```

12. As the root user, run the **`/sbin/service httpd restart`** command to restart `httpd`. After restarting, run the **`ps -eZ | grep httpd`** to confirm that `httpd` is running in the confined **`httpd_t`** domain:

```
# /sbin/service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
# ps -eZ | grep httpd
unconfined_u:system_r:httpd_t    8880 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8882 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8883 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8884 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8885 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8886 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8887 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8888 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8889 ?        00:00:00 httpd
```

13. As the root user, run the **`rm -i /var/www/html/test2file`** command to remove **`test2file`**.

14. If you do not require `httpd` to be running, as the root user, run the **`service httpd stop`** command to stop `httpd`:

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK  ]
```

The examples in these sections demonstrate how data can be protected from a compromised confined process (protected by SELinux), as well as how data is more accessible to an attacker from a compromised unconfined process (not protected by SELinux).

10.3. The Apache HTTP Server

From the *Apache HTTP Server Project*[5] page:

"The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards".[6]

In Fedora, the *httpd* package provides the Apache HTTP Server. Run **`rpm -q httpd`** to see if the *httpd* package is installed. If it is not installed and you want to use the Apache HTTP Server, run the following command as the root user to install it:

```
yum install httpd
```

10.3.1. The Apache HTTP Server and SELinux

When SELinux is enabled, the Apache HTTP Server (`httpd`) runs confined by default. Confined processes run in their own domains, and are separated from other confined processes. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited. The following example demonstrates the `httpd` processes running in their own domain. This example assumes the *httpd* package is installed:

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run **`service httpd start`** as the root user to start `httpd`:

   ```
   # service httpd start
   Starting httpd:                                             [  OK  ]
   ```

3. Run **`ps -eZ | grep httpd`** to view the `httpd` processes:

   ```
   $ ps -eZ | grep httpd
   unconfined_u:system_r:httpd_t:s0 2850 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2852 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2853 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2854 ?        00:00:00 httpd
   ```

[5] http://httpd.apache.org/
[6] From the "The Number One HTTP Server On The Internet" section of the Apache HTTP Server Project page: *http://httpd.apache.org/*. Copyright © 2010 The Apache Software Foundation. Accessed 1 March 2010.

```
unconfined_u:system_r:httpd_t:s0 2855 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 2856 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 2857 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 2858 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 2859 ?        00:00:00 httpd
```

The SELinux context associated with the `httpd` processes is **`unconfined_u:system_r:httpd_t:s0`**. The second last part of the context, **`httpd_t`**, is the type. A type defines a domain for processes and a type for files. In this case, the `httpd` processes are running in the **`httpd_t`** domain.

SELinux policy defines how processes running in confined domains, such as **`httpd_t`**, interact with files, other processes, and the system in general. Files must be labeled correctly to allow `httpd` access to them. For example, `httpd` can read files labeled with the **`httpd_sys_content_t`** type, but can not write to them, even if Linux permissions allow write access. Booleans must be turned on to allow certain behavior, such as allowing scripts network access, allowing `httpd` access to NFS and CIFS file systems, and `httpd` being allowed to execute Common Gateway Interface (CGI) scripts.

When **`/etc/httpd/conf/httpd.conf`** is configured so `httpd` listens on a port other than TCP ports 80, 443, 488, 8008, 8009, or 8443, the **`semanage port`** command must be used to add the new port number to SELinux policy configuration. The following example demonstrates configuring `httpd` to listen on a port that is not defined in SELinux policy configuration for `httpd`, and, as a consequence, `httpd` failing to start. This example also demonstrates how to then configure the SELinux system to allow `httpd` to successfully listen on a non-standard port that is not already defined in the policy. This example assumes the *httpd* package is installed. Run each command in the example as the root user:

1. Run **`service httpd status`** to confirm `httpd` is not running:

```
# service httpd status
httpd is stopped
```

   If the output differs, run **`service httpd stop`** to stop the process:

```
# service httpd stop
Stopping httpd:                                            [  OK  ]
```

2. Run **`semanage port -l | grep -w http_port_t`** to view the ports SELinux allows `httpd` to listen on:

```
# semanage port -l | grep -w http_port_t
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
```

3. Edit **`/etc/httpd/conf/httpd.conf`** as the root user. Configure the **`Listen`** option so it lists a port that is not configured in SELinux policy configuration for `httpd`. In this example, `httpd` is configured to listen on port 12345:

```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 127.0.0.1:12345
```

4. Run **`service httpd start`** to start `httpd`:

   ```
   # service httpd start
   Starting httpd: (13)Permission denied: make_sock: could not bind to address
    127.0.0.1:12345
   no listening sockets available, shutting down
   Unable to open logs        [FAILED]
   ```

   An SELinux denial similar to the following is logged to **`/var/log/messages`**:

   ```
   setroubleshoot: SELinux is preventing the httpd (httpd_t) from binding to port 12345. For
    complete SELinux messages. run sealert -l f18bca99-db64-4c16-9719-1db89f0d8c77
   ```

5. For SELinux to allow `httpd` to listen on port 12345, as used in this example, the following command is required:

   ```
   # semanage port -a -t http_port_t -p tcp 12345
   ```

6. Run **`service httpd start`** again to start `httpd` and have it listen on the new port:

   ```
   # service httpd start
   Starting httpd:         [  OK  ]
   ```

7. Now that SELinux has been configured to allow `httpd` to listen on a non-standard port (TCP 12345 in this example), `httpd` starts successfully on this port.

8. To prove that `httpd` is listening and communicating on TCP port 12345, open a telnet connection to the specified port and issue a HTTP GET command, as follows:

   ```
   # telnet localhost 12345
   Trying 127.0.0.1...
   Connected to localhost.
   Escape character is '^]'.
   GET / HTTP/1.0

   HTTP/1.1 200 OK
   Date: Tue, 31 Mar 2009 13:12:10 GMT
   Server: Apache/2.2.11 (Fedora)
   Accept-Ranges: bytes
   Content-Length: 3918
   Content-Type: text/html; charset=UTF-8
   [...continues...]
   ```

10.3.2. Types

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

The following example creates a new file in the **`/var/www/html/`** directory, and shows the file inheriting the **`httpd_sys_content_t`** type from its parent directory (**`/var/www/html/`**):

1. Run **`ls -dZ /var/www/html`** to view the SELinux context of **`/var/www/html/`**:

   ```
   $ ls -dZ /var/www/html
   drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html
   ```

   This shows **`/var/www/html/`** is labeled with the **`httpd_sys_content_t`** type.

2. Run **`touch /var/www/html/file1`** as the root user to create a new file.

3. Run **`ls -Z /var/www/html/file1`** to view the SELinux context:

   ```
   $ ls -Z /var/www/html/file1
   -rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
   ```

The **`ls -Z`** command shows **`file1`** labeled with the **`httpd_sys_content_t`** type. SELinux allows `httpd` to read files labeled with this type, but not write to them, even if Linux permissions allow write access. SELinux policy defines what types a process running in the **`httpd_t`** domain (where `httpd` runs) can read and write to. This helps prevent processes from accessing files intended for use by another process.

For example, `httpd` can access files labeled with the **`httpd_sys_content_t`** type (intended for the Apache HTTP Server), but by default, can not access files labeled with the **`samba_share_t`** type (intended for Samba). Also, files in user home directories are labeled with the **`user_home_t`** type: by default, this prevents `httpd` from reading or writing to files in user home directories.

The following types are used with `httpd`. Different types allow you to configure flexible access:

`httpd_sys_content_t`
: Use this type for static web content, such as **`.html`** files used by a static website. Files labeled with this type are accessible (read only) to `httpd` and scripts executed by `httpd`. By default, files and directories labeled with this type can not be written to or modified by `httpd` or other processes. Note: by default, files created in or copied into **`/var/www/html/`** are labeled with the **`httpd_sys_content_t`** type.

`httpd_sys_script_exec_t`
: Use this type for scripts you want `httpd` to execute. This type is commonly used for Common Gateway Interface (CGI) scripts in **`/var/www/cgi-bin/`**. By default, SELinux policy prevents `httpd` from executing CGI scripts. To allow this, label the scripts with the **`httpd_sys_script_exec_t`** type and turn the **`httpd_enable_cgi`** Boolean on. Scripts labeled with **`httpd_sys_script_exec_t`** run in the **`httpd_sys_script_t`** domain when executed by `httpd`. The **`httpd_sys_script_t`** domain has access to other system domains, such as **`postgresql_t`** and **`mysqld_t`**.

`httpd_sys_content_rw_t`
: Files labeled with this type can be written to by scripts labeled with the **`httpd_sys_script_exec_t`** type, but can not be modified by scripts labeled with any other type. You must use the **`httpd_sys_content_rw_t`** type to label files that will be read from and written to by scripts labeled with the **`httpd_sys_script_exec_t`** type.

`httpd_sys_content_ra_t`
: Files labeled with this type can be appended to by scripts labeled with the **`httpd_sys_script_exec_t`** type, but can not be modified by scripts labeled with any other type. You must use the **`httpd_sys_content_ra_t`** type to label files that will be read from and appended to by scripts labeled with the **`httpd_sys_script_exec_t`** type.

`httpd_unconfined_script_exec_t`
Scripts labeled with this type run without SELinux protection. Only use this type for complex scripts, after exhausting all other options. It is better to use this type instead of turning SELinux protection off for `httpd`, or for the entire system.

Changing the SELinux Context

The type for files and directories can be changed with the **`chcon`** command. Changes made with **`chcon`** do not survive a file system relabel or the **`restorecon`** command. SELinux policy controls whether users are able to modify the SELinux context for any given file. The following example demonstrates creating a new directory and an **`index.html`** file for use by `httpd`, and labeling that file and directory to allow `httpd` access to them:

1. Run **`mkdir -p /my/website`** as the root user to create a top-level directory structure to store files to be used by `httpd`.

2. Files and directories that do not match a pattern in file-context configuration may be labeled with the **`default_t`** type. This type is inaccessible to confined services:

```
$ ls -dZ /my
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /my
```

3. Run **`chcon -R -t httpd_sys_content_t /my/`** as the root user to change the type of the **`/my/`** directory and subdirectories, to a type accessible to `httpd`. Now, files created under **`/my/website/`** inherit the **`httpd_sys_content_t`** type, rather than the **`default_t`** type, and are therefore accessible to httpd:

```
# chcon -R -t httpd_sys_content_t /my/
# touch /my/website/index.html
# ls -Z /my/website/index.html
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /my/website/index.html
```

Use the **`semanage fcontext`** command to make label changes that survive a relabel and the **`restorecon`** command. This command adds changes to file-context configuration. Then, run the **`restorecon`** command, which reads file-context configuration, to apply the label change. The following example demonstrates creating a new directory and an **`index.html`** file for use by `httpd`, and persistently changing the label of that directory and file to allow `httpd` access to them:

1. Run **`mkdir -p /my/website`** as the root user to create a top-level directory structure to store files to be used by `httpd`.

2. Run the following command as the root user to add the label change to file-context configuration:

```
semanage fcontext -a -t httpd_sys_content_t "/my(/.*)?"
```

   The **`"/my(/.*)?"`** expression means the label change applies to the **`/my/`** directory and all files and directories under it.

3. Run **`touch /my/website/index.html`** as the root user to create a new file.

4. Run **`restorecon -R -v /my/`** as the root user to apply the label changes (**`restorecon`** reads file-context configuration, which was modified by the **`semanage`** command in step 2):

```
# restorecon -R -v /my/
restorecon reset /my context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /my/website context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /my/website/index.html context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

10.3.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. This can be achieved via Booleans that allow parts of SELinux policy to be changed at runtime, without any knowledge of SELinux policy writing. This allows changes, such as allowing services access to NFS file systems, without reloading or recompiling SELinux policy.

To modify the state of a Boolean, use the **`setsebool`** command. For example, to turn the **`allow_httpd_anon_write`** Boolean on, run the following command as the root user:

```
# setsebool -P allow_httpd_anon_write on
```

To turn a Boolean off, using the same example, simply change **`on`** to **`off`** in the command, as shown below:

```
# setsebool -P allow_httpd_anon_write off
```

Note

Do not use the **`-P`** option if you do not want **`setsebool`** changes to persist across reboots.

Below is a description of common Booleans available that cater for the way `httpd` is running:

`allow_httpd_anon_write`
: When disabled, this Boolean allows `httpd` only read access to files labeled with the **`public_content_rw_t`** type. Enabling this Boolean will allow `httpd` to write to files labeled with the **`public_content_rw_t`** type, such as a public directory containing files for a public file transfer service.

`allow_httpd_mod_auth_ntlm_winbind`
: Enabling this Boolean allows access to NTLM and Winbind authentication mechanisms via the **`mod_auth_ntlm_winbind`** module in `httpd`.

`allow_httpd_mod_auth_pam`
: Enabling this Boolean allows access to PAM authentication mechanisms via the **`mod_auth_pam`** module in `httpd`.

`allow_httpd_sys_script_anon_write`
: This Boolean defines whether or not HTTP scripts are allowed write access to files labeled with the **`public_content_rw_t`** type, as used in a public file transfer service.

`httpd_builtin_scripting`
: This Boolean defines access to `httpd` scripting. Having this Boolean enabled is often required for PHP content.

`httpd_can_network_connect`
: When disabled, this Boolean prevents HTTP scripts and modules from initiating a connection to a network or remote port. Turn this Boolean on to allow this access.

`httpd_can_network_connect_db`
: When disabled, this Boolean prevents HTTP scripts and modules from initiating a connection to database servers. Turn this Boolean on to allow this access.

`httpd_can_network_relay`
: Turn this Boolean on when `httpd` is being used as a forward or reverse proxy.

`httpd_can_sendmail`
: When disabled, this Boolean prevents HTTP modules from sending mail. This can prevent spam attacks should a vulnerability be found in `httpd`. Turn this Boolean on to allow HTTP modules to send mail.

`httpd_dbus_avahi`
: When off, this Boolean denies `httpd` access to the **`avahi`** service via **`D-Bus`**. Turn this Boolean on to allow this access.

`httpd_enable_cgi`
: When disabled, this Boolean prevents `httpd` from executing CGI scripts. Turn this Boolean on to allow `httpd` to execute CGI scripts (CGI scripts must be labeled with the **`httpd_sys_script_exec_t`** type).

`httpd_enable_ftp_server`
: Turning this Boolean on will allow `httpd` to listen on the FTP port and act as an FTP server.

`httpd_enable_homedirs`
: When disabled, this Boolean prevents `httpd` from accessing user home directories. Turn this Boolean on to allow `httpd` access to user home directories; for example, content in **`/home/*/`**.

`httpd_execmem`
: When enabled, this Boolean allows `httpd` to execute programs that require memory addresses that are both executable and writeable. Enabling this Boolean is not recommended from a security standpoint as it reduces protection against buffer overflows, however certain modules and applications (such as Java and Mono applications) require this privilege.

`httpd_ssi_exec`
: This Boolean defines whether or not server side include (SSI) elements in a web page can be executed.

`httpd_tmp_exec`
: Enabling this Boolean allows `httpd` to execute files in temporary directories.

`httpd_tty_comm`
: This Boolean defines whether or not `httpd` is allowed access to the controlling terminal. Usually this access is not required, however in cases such as configuring an SSL certificate file, terminal access is required to display and process a password prompt.

`httpd_unified`

When enabled, this Boolean allows **`httpd_t`** complete access to all of the `httpd` types (i.e. to execute, read, or write sys_content_t). When disabled, there is separation in place between web content that is read-only, writeable or executable. Disabling this Boolean ensures an extra level of security but adds the administrative overhead of having to individually label scripts and other web content based on the file access that each should have.

`httpd_use_cifs`

Turn this Boolean on to allow `httpd` access to files on CIFS file systems that are labeled with the **`cifs_t`** type, such as file systems mounted via Samba.

`httpd_use_gpg`

Enabling this Boolean allows `httpd` to make use of GPG encryption.

`httpd_use_nfs`

Turn this Boolean on to allow `httpd` access to files on NFS file systems that are labeled with the **`nfs_t`** type, such as file systems mounted via NFS.

10.3.4. Configuration examples

The following examples provide real-world demonstrations of how SELinux complements the Apache HTTP Server and how full function of the Apache HTTP Server can be maintained.

10.3.4.1. Running a static site

To create a static website, label the **`.html`** files for that website with the **`httpd_sys_content_t`** type. By default, the Apache HTTP Server can not write to files that are labeled with the **`httpd_sys_content_t`** type. The following example creates a new directory to store files for a read-only website:

1. Run **`mkdir /mywebsite`** as the root user to create a top-level directory.

2. As the root user, create a **`/mywebsite/index.html`** file. Copy and paste the following content into **`/mywebsite/index.html`**:

```
<html>
<h2>index.html from /mywebsite/</h2>
</html>
```

3. To allow the Apache HTTP Server read only access to **`/mywebsite/`**, as well as files and subdirectories under it, label **`/mywebsite/`** with the **`httpd_sys_content_t`** type. Run the following command as the root user to add the label change to file-context configuration:

```
# semanage fcontext -a -t httpd_sys_content_t "/mywebsite(/.*)?"
```

4. Run **`restorecon -R -v /mywebsite`** as the root user to make the label changes:

```
# restorecon -R -v /mywebsite
restorecon reset /mywebsite context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /mywebsite/index.html context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

5. For this example, edit **/etc/httpd/conf/httpd.conf** as the root user. Comment out the existing **DocumentRoot** option. Add a **DocumentRoot "/mywebsite"** option. After editing, these options should look as follows:

```
#DocumentRoot "/var/www/html"
DocumentRoot "/mywebsite"
```

6. Run **service httpd status** as the root user to see the status of the Apache HTTP Server. If the server is stopped, run **service httpd start** as the root user to start it. If the server is running, run **service httpd restart** as the root user to restart the service (this also applies any changes made to **httpd.conf**).

7. Use a web browser to navigate to **http://localhost/index.html**. The following is displayed:

```
index.html from /mywebsite/
```

10.3.4.2. Sharing NFS and CIFS file systems

By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **nfs_t** type. Also, by default, Samba shares mounted on the client side are labeled with a default context defined by policy. In common policies, this default context uses the **cifs_t** type.

Depending on policy configuration, services may not be able to read files labeled with the **nfs_t** or **cifs_t** types. This may prevent file systems labeled with these types from being mounted and then read or exported by other services. Booleans can be turned on or off to control which services are allowed to access the **nfs_t** and **cifs_t** types.

Turn the **httpd_use_nfs** Boolean on to allow `httpd` to access and share NFS file systems (labeled with the **nfs_t** type. Run the **setsebool** command as the root user to turn the Boolean on:

```
setsebool -P httpd_use_nfs on
```

Turn the **httpd_use_cifs** Boolean on to allow `httpd` to access and share CIFS file systems (labeled with the **cifs_t** type. Run the **setsebool** command as the root user to turn the Boolean on:

```
setsebool -P httpd_use_cifs on
```

Note

Do not use the **-P** option if you do not want **setsebool** changes to persist across reboots.

10.3.4.3. Sharing files between services

Type Enforcement helps prevent processes from accessing files intended for use by another process. For example, by default, Samba can not read files labeled with the **httpd_sys_content_t** type, which are intended for use by the Apache HTTP Server. Files can be shared between the Apache

HTTP Server, FTP, rsync, and Samba, if the desired files are labeled with the **`public_content_t`** or **`public_content_rw_t`** type.

The following example creates a directory and files, and allows that directory and files to be shared (read only) through the Apache HTTP Server, FTP, rsync, and Samba:

1. Run **`mkdir /shares`** as the root user to create a new top-level directory to share files between multiple services.

2. Files and directories that do not match a pattern in file-context configuration may be labeled with the **`default_t`** type. This type is inaccessible to confined services:

   ```
   $ ls -dZ /shares
   drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /shares
   ```

3. As the root user, create a **`/shares/index.html`** file. Copy and paste the following content into **`/shares/index.html`**:

   ```
   <html>
   <body>
   <p>Hello</p>
   </body>
   </html>
   ```

4. Labeling **`/shares/`** with the **`public_content_t`** type allows read-only access by the Apache HTTP Server, FTP, rsync, and Samba. Run the following command as the root user to add the label change to file-context configuration:

   ```
   semanage fcontext -a -t public_content_t "/shares(/.*)?"
   ```

5. Run **`restorecon -R -v /shares/`** as the root user to apply the label changes:

   ```
   # restorecon -R -v /shares/
   restorecon reset /shares context unconfined_u:object_r:default_t:s0-
   >system_u:object_r:public_content_t:s0
   restorecon reset /shares/index.html context unconfined_u:object_r:default_t:s0-
   >system_u:object_r:public_content_t:s0
   ```

To share **`/shares/`** through Samba:

1. Run **`rpm -q samba samba-common samba-client`** to confirm the *samba*, *samba-common*, and *samba-client* packages are installed (version numbers may differ):

   ```
   $ rpm -q samba samba-common samba-client
   samba-3.5.2-59.fc13.i386
   samba-common-3.5.2-59.fc13.i386
   samba-client-3.5.2-59.fc13.i386
   ```

   If any of these packages are not installed, install them by running **`yum install`** ***`package-name`*** as the root user.

2. Edit **/etc/samba/smb.conf** as the root user. Add the following entry to the bottom of this file to share the **/shares/** directory through Samba:

```
[shares]
comment = Documents for Apache HTTP Server, FTP, rsync, and Samba
path = /shares
public = yes
writeable = no
```

3. A Samba account is required to mount a Samba file system. Run **smbpasswd -a *username*** as the root user to create a Samba account, where *username* is an existing Linux user. For example, **smbpasswd -a testuser** creates a Samba account for the Linux testuser user:

```
# smbpasswd -a testuser
New SMB password: Enter a password
Retype new SMB password: Enter the same password again
Added user testuser.
```

   Running **smbpasswd -a *username***, where *username* is the username of a Linux account that does not exist on the system, causes a **Cannot locate Unix account for '*username*'!** error.

4. Run **service smb start** as the root user to start the Samba service:

```
service smb start
Starting SMB services:                                     [  OK  ]
```

5. Run **smbclient -U *username* -L localhost** to list the available shares, where *username* is the Samba account added in step 3. When prompted for a password, enter the password assigned to the Samba account in step 3 (version numbers may differ):

```
$ smbclient -U username -L localhost
Enter username's password:
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.5.2-59.fc13]

Sharename       Type      Comment
---------       ----      -------
shares          Disk      Documents for Apache HTTP Server, FTP, rsync, and Samba
IPC$            IPC       IPC Service (Samba Server Version 3.5.2-59)
username        Disk      Home Directories
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.5.2-59.fc13]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

6. Run **mkdir /test/** as the root user to create a new directory. This directory will be used to mount the **shares** Samba share.

7. Run the following command as the root user to mount the **shares** Samba share to **/test/**, replacing *username* with the username from step 3:

```
mount //localhost/shares /test/ -o user=username
```

Enter the password for *`username`*, which was configured in step 3.

8. Run **`cat /test/index.html`** to view the file, which is being shared through Samba:

```
$ cat /test/index.html
<html>
<body>
<p>Hello</p>
</body>
</html>
```

To share **`/shares/`** through the Apache HTTP Server:

1. Run **`rpm -q httpd`** to confirm the *httpd* package is installed (version number may differ):

   ```
   $ rpm -q httpd
   httpd-2.2.11-6.i386
   ```

   If this package is not installed, run **`yum install httpd`** as the root user to install it.

2. Change into the **`/var/www/html/`** directory. Run the following command as the root user to create a link (named **`shares`**) to the **`/shares/`** directory:

   ```
   ln -s /shares/ shares
   ```

3. Run **`service httpd start`** as the root user to start the Apache HTTP Server:

   ```
   service httpd start
   Starting httpd:                                            [  OK  ]
   ```

4. Use a web browser to navigate to **`http://localhost/shares`**. The **`/shares/index.html`** file is displayed.

By default, the Apache HTTP Server reads an **`index.html`** file if it exists. If **`/shares/`** did not have **`index.html`**, and instead had **`file1`**, **`file2`**, and **`file3`**, a directory listing would occur when accessing **`http://localhost/shares`**:

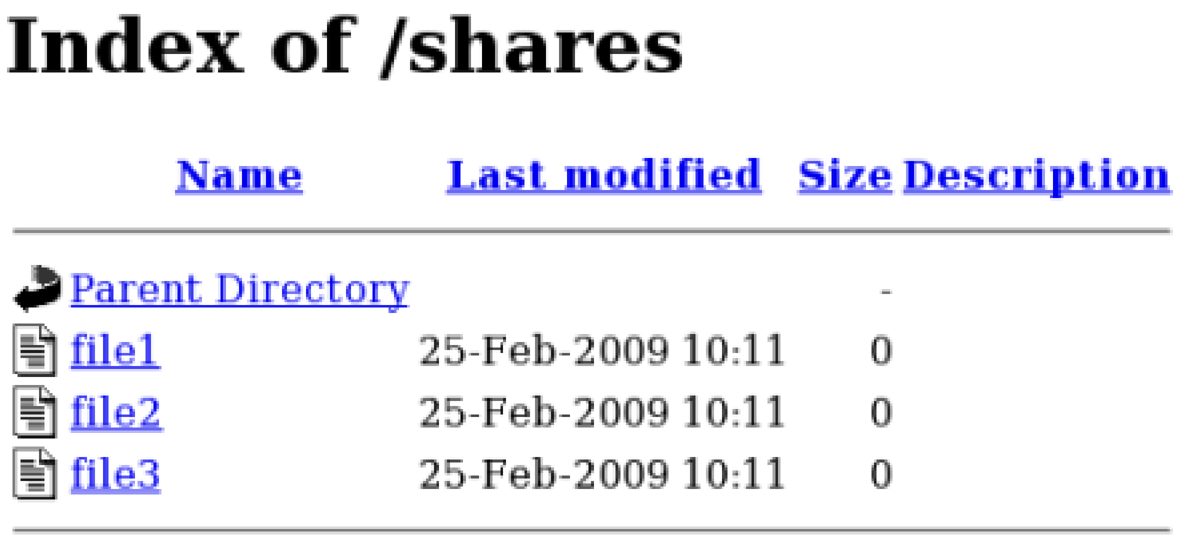
1. Run **`rm -i /shares/index.html`** as the root user to remove the **`index.html`** file.

2. Run **`touch /shares/file{1,2,3}`** as the root user to create three files in **`/shares/`**:

   ```
   # touch /shares/file{1,2,3}
   # ls -Z /shares/
   -rw-r--r--  root root system_u:object_r:public_content_t:s0 file1
   -rw-r--r--  root root unconfined_u:object_r:public_content_t:s0 file2
   -rw-r--r--  root root unconfined_u:object_r:public_content_t:s0 file3
   ```

3. Run **`service httpd status`** as the root user to see the status of the Apache HTTP Server. If the server is stopped, run **`service httpd start`** as the root user to start it.

4. Use a web browser to navigate to **`http://localhost/shares`**. A directory listing is displayed:



Index of /shares

Name	Last modified	Size	Description
Parent Directory		-	
file1	25-Feb-2009 10:11	0	
file2	25-Feb-2009 10:11	0	
file3	25-Feb-2009 10:11	0	

10.3.4.4. Changing port numbers

Depending on policy configuration, services may only be allowed to run on certain port numbers. Attempting to change the port a service runs on without changing policy may result in the service failing to start. Run **`semanage port -l | grep -w "http_port_t"`** as the root user to list the ports SELinux allows `httpd` to listen on:

```
# semanage port -l | grep -w http_port_t
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
```

By default, SELinux allows `http` to listen on TCP ports 80, 443, 488, 8008, 8009, or 8443. If **`/etc/httpd/conf/httpd.conf`** is configured so that `httpd` listens on any port not listed for **`http_port_t`**, `httpd` fails to start.

To configure `httpd` to run on a port other than TCP ports 80, 443, 488, 8008, 8009, or 8443:

1. Edit **`/etc/httpd/conf/httpd.conf`** as the root user so the **`Listen`** option lists a port that is not configured in SELinux policy for `httpd`. The following example configures `httpd` to listen on the 10.0.0.1 IP address, and on port 12345:

```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 10.0.0.1:12345
```

2. Run **`semanage port -a -t http_port_t -p tcp 12345`** as the root user to add the port to SELinux policy configuration.

3. Run **`semanage port -l | grep -w http_port_t`** as the root user to confirm the port is added:

```
# semanage port -l | grep -w http_port_t
http_port_t                    tcp      12345, 80, 443, 488, 8008, 8009, 8443
```

If you no longer run `httpd` on port 12345, run **`semanage port -d -t http_port_t -p tcp 12345`** as the root user to remove the port from policy configuration.

10.4. Samba

From the *Samba*[7] website:

"Samba is an *Open Source*[8]/*Free Software*[9] suite that has, *since 1992*[10], provided file and print services to all manner of SMB/CIFS clients, including the numerous versions of Microsoft Windows operating systems. Samba is freely available under the *GNU General Public License*[11].".[12]

In Fedora, the *samba* package provides the Samba server. Run **`rpm -q samba`** to see if the *samba* package is installed. If it is not installed and you want to use Samba, run the following command as the root user to install it:

```
yum install samba
```

10.4.1. Samba and SELinux

When SELinux is enabled, the Samba server (`smbd`) runs confined by default. Confined services run in their own domains, and are separated from other confined services. The following example demonstrates the `smbd` process running in its own domain. This example assumes the *samba* package is installed:

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run **`service smbd start`** as the root user to start `smbd`:

   ```
   service smb start
   Starting SMB services:                                    [  OK  ]
   ```

3. Run **`ps -eZ | grep smb`** to view the `smbd` processes:

[7] http://samba.org/
[8] http://www.opensource.org/
[9] http://www.gnu.org/philosophy/free-sw.html
[10] http://us1.samba.org/samba/docs/10years.html
[11] http://us1.samba.org/samba/docs/GPL.html
[12] From the opening paragraph on the Samba website: *http://samba.org*. Accessed 20 January 2009.

```
$ ps -eZ | grep smb
unconfined_u:system_r:smbd_t:s0 16420 ?        00:00:00 smbd
unconfined_u:system_r:smbd_t:s0 16422 ?        00:00:00 smbd
```

The SELinux context associated with the smbd processes is **unconfined_u:system_r:smbd_t:s0**. The second last part of the context, **smbd_t**, is the type. A type defines a domain for processes and a type for files. In this case, the smbd processes are running in the smbd_t domain.

Files must be labeled correctly to allow smbd to access and share them. For example, smbd can read and write to files labeled with the **samba_share_t** type, but by default, can not access files labeled with the **httpd_sys_content_t** type, which is intended for use by the Apache HTTP Server. Booleans must be turned on to allow certain behavior, such as allowing home directories and NFS file systems to be exported through Samba, as well as to allow Samba to act as a domain controller.

10.4.2. Types

Label files with the **samba_share_t** type to allow Samba to share them. Only label files you have created, and do not relabel system files with the **samba_share_t** type: Booleans can be turned on to share such files and directories. SELinux allows Samba to write to files labeled with the **samba_share_t** type, as long as **/etc/samba/smb.conf** and Linux permissions are set accordingly.

The **samba_etc_t** type is used on certain files in **/etc/samba/**, such as **smb.conf**. Do not manually label files with the **samba_etc_t** type. If files in **/etc/samba/** are not labeled correctly, run **restorecon -R -v /etc/samba** as the root user to restore such files to their default contexts. If **/etc/samba/smb.conf** is not labeled with the **samba_etc_t** type, the **service smb start** command may fail and an SELinux denial may be logged. The following is an example denial logged to **/var/log/messages** when **/etc/samba/smb.conf** was labeled with the **httpd_sys_content_t** type:

```

setroubleshoot: SELinux is preventing smbd (smbd_t) "read" to ./smb.conf
 (httpd_sys_content_t). For complete SELinux messages. run sealert -l deb33473-1069-482b-
bb50-e4cd05ab18af
```

10.4.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Booleans allow you to tell SELinux how you are running Samba:

allow_smbd_anon_write
: Having this Boolean enables allows smbd to write to a public directory, such as an area reserved for common files that otherwise has no special access restrictions.

samba_create_home_dirs
: Having this Boolean enabled allows Samba to create new home directories independently. This is often done by mechanisms such as PAM.

samba_domain_controller
: When enabled, this Boolean allows Samba to act as a domain controller, as well as giving it permission to execute related commands such as **useradd**, **groupadd** and **passwd**.

samba_enable_home_dirs
: Enabling this Boolean allows Samba to share users' home directories.

`samba_export_all_ro`
: Export any file or directory, allowing read-only permissions. This allows files and directories that are not labeled with the **`samba_share_t`** type to be shared through Samba. When the **`samba_export_all_ro`** Boolean is on, but the **`samba_export_all_rw`** Boolean is off, write access to Samba shares is denied, even if write access is configured in **`/etc/samba/smb.conf`**, as well as Linux permissions allowing write access.

`samba_export_all_rw`
: Export any file or directory, allowing read and write permissions. This allows files and directories that are not labeled with the **`samba_share_t`** type to be exported through Samba. Permissions in **`/etc/samba/smb.conf`** and Linux permissions must be configured to allow write access.

`samba_run_unconfined`
: Having this Boolean enabled allows Samba to run unconfined scripts in the **`/var/lib/samba/scripts`** directory.

`samba_share_fusefs`
: This Boolean must be enabled for Samba to share **`fusefs`** file systems.

`samba_share_nfs`
: Disabling this Boolean prevents `smbd` from having full access to NFS shares via Samba. Enabling this Boolean will allow Samba to share NFS file systems.

`use_samba_home_dirs`
: Enable this Boolean to use a remote server for Samba home directories.

`virt_use_samba`
: Allow virt to manage CIFS files.

10.4.4. Configuration examples

The following examples provide real-world demonstrations of how SELinux complements the Samba server and how full function of the Samba server can be maintained.

10.4.4.1. Sharing directories you create

The following example creates a new directory, and shares that directory through Samba:

1. Run **`rpm -q samba samba-common samba-client`** to confirm the *samba*, *samba-common*, and *samba-client* packages are installed. If any of these packages are not installed, install them by running **`yum install`** ***`package-name`*** as the root user.

2. Run **`mkdir /myshare`** as the root user to create a new top-level directory to share files through Samba.

3. Run **`touch /myshare/file1`** as the root user to create an empty file. This file is used later to verify the Samba share mounted correctly.

4. SELinux allows Samba to read and write to files labeled with the **`samba_share_t`** type, as long as **`/etc/samba/smb.conf`** and Linux permissions are set accordingly. Run the following command as the root user to add the label change to file-context configuration:

   ```
   semanage fcontext -a -t samba_share_t "/myshare(/.*)?"
   ```

5. Run **`restorecon -R -v /myshare`** as the root user to apply the label changes:

```
# restorecon -R -v /myshare
restorecon reset /myshare context unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
restorecon reset /myshare/file1 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
```

6. Edit **`/etc/samba/smb.conf`** as the root user. Add the following to the bottom of this file to share the /myshare/ directory through Samba:

   ```
   [myshare]
   comment = My share
   path = /myshare
   public = yes
   writeable = no
   ```

7. A Samba account is required to mount a Samba file system. Run **`smbpasswd -a`** ***`username`*** as the root user to create a Samba account, where *`username`* is an existing Linux user. For example, **`smbpasswd -a testuser`** creates a Samba account for the Linux testuser user:

   ```
   # smbpasswd -a testuser
   New SMB password: Enter a password
   Retype new SMB password: Enter the same password again
   Added user testuser.
   ```

   Running **`smbpasswd -a`** ***`username`***, where *`username`* is the username of a Linux account that does not exist on the system, causes a **`Cannot locate Unix account for '`*`username`*`'!`** error.

8. Run **`service smb start`** as the root user to start the Samba service:

   ```
   service smb start
   Starting SMB services:                                     [  OK  ]
   ```

9. Run **`smbclient -U`** ***`username`*** **`-L localhost`** to list the available shares, where *`username`* is the Samba account added in step 7. When prompted for a password, enter the password assigned to the Samba account in step 7 (version numbers may differ):

   ```
   $ smbclient -U username -L localhost
   Enter username's password:
   Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.5.2-59.fc13]

   Sharename       Type      Comment
   ---------       ----      -------
   myshare         Disk      My share
   IPC$            IPC       IPC Service (Samba Server Version 3.5.2-59.fc13)
   username        Disk      Home Directories
   Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.5.2-59.fc13]

   Server               Comment
   ---------            -------

   Workgroup            Master
   ```

```
---------            -------
```

10. Run **`mkdir /test/`** as the root user to create a new directory. This directory will be used to mount the **`myshare`** Samba share.

11. Run the following command as the root user to mount the **`myshare`** Samba share to **`/test/`**, replacing *`username`* with the username from step 7:

    ```
    mount //localhost/myshare /test/ -o user=username
    ```

    Enter the password for *`username`*, which was configured in step 7.

12. Run **`ls /test/`** to view the **`file1`** file created in step 3:

    ```
    $ ls /test/
    file1
    ```

10.4.4.2. Sharing a website

It may not be possible to label files with the **`samba_share_t`** type, for example, when wanting to share a website in **`/var/www/html/`**. For these cases, use the **`samba_export_all_ro`** Boolean to share any file or directory (regardless of the current label), allowing read only permissions, or the **`samba_export_all_rw`** Boolean to share any file or directory (regardless of the current label), allowing read and write permissions.

The following example creates a file for a website in **`/var/www/html/`**, and then shares that file through Samba, allowing read and write permissions. This example assumes the *httpd*, *samba*, *samba-common*, *samba-client*, and *wget* packages are installed:

1. As the root user, create a **`/var/www/html/file1.html`** file. Copy and paste the following content into **`/var/www/html/file1.html`**:

   ```
   <html>
   <h2>File being shared through the Apache HTTP Server and Samba.</h2>
   </html>
   ```

2. Run **`ls -Z /var/www/html/file1.html`** to view the SELinux context of **`file1.html`**:

   ```
   $ ls -Z /var/www/html/file1.html
   -rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
   file1.html
   ```

   `file1.index.html` is labeled with the **`httpd_sys_content_t`**. By default, the Apache HTTP Server can access this type, but Samba can not.

3. Run **`service httpd start`** as the root user to start the Apache HTTP Server:

   ```
   service httpd start
   Starting httpd:                                            [  OK  ]
   ```

4. Change into a directory your user has write access to, and run the **`wget http://localhost/file1.html`** command. Unless there are changes to the default configuration, this command succeeds:

   ```
   $ wget http://localhost/file1.html
   --2009-03-02 16:32:01--  http://localhost/file1.html
   Resolving localhost... 127.0.0.1
   Connecting to localhost|127.0.0.1|:80... connected.
   HTTP request sent, awaiting response... 200 OK
   Length: 84 [text/html]
   Saving to: `file1.html.1'

   100%[======================>] 84          --.-K/s   in 0s

   2009-03-02 16:32:01 (563 KB/s) - `file1.html.1' saved [84/84]
   ```

5. Edit **`/etc/samba/smb.conf`** as the root user. Add the following to the bottom of this file to share the **`/var/www/html/`** directory through Samba:

   ```
   [website]
   comment = Sharing a website
   path = /var/www/html/
   public = no
   writeable = no
   ```

6. The **`/var/www/html/`** directory is labeled with the **`httpd_sys_content_t`** type. By default, Samba can not access files and directories labeled with the **`httpd_sys_content_t`** type, even if Linux permissions allow it. To allow Samba access, run the following command as the root user to turn the **`samba_export_all_ro`** Boolean on:

   ```
   setsebool -P samba_export_all_ro on
   ```

   Do not use the **`-P`** option if you do not want the change to persist across reboots. Note: turning the **`samba_export_all_ro`** Boolean on allows Samba to access any type.

7. Run **`service smb start`** as the root user to start `smbd`:

   ```
   service smb start
   Starting SMB services:                                     [  OK  ]
   ```

10.5. File Transfer Protocol

From the *Red Hat Enterprise Linux 5 Deployment Guide*[13]:

File Transfer Protocol (FTP) is one of the oldest and most commonly used protocols found on the Internet today. Its purpose is to reliably transfer files between computer hosts on a network without

[13] http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/index.html

requiring the user to log directly into the remote host or have knowledge of how to use the remote system. It allows users to access files on remote systems using a standard set of simple commands.[14]

The Very Secure FTP Daemon (`vsftpd`) is designed from the ground up to be fast, stable, and, most importantly, secure. Its ability to handle large numbers of connections efficiently and securely is why `vsftpd` is the only stand-alone FTP distributed with Red Hat Enterprise Linux.[15]

In Fedora, the *vsftpd* package provides the Very Secure FTP daemon. Run **`rpm -q vsftpd`** to see if *vsftpd* is installed:

```
$ rpm -q vsftpd
```

If you want an FTP server and the *vsftpd* package is not installed, run the following command as the root user to install it:

```
yum install vsftpd
```

10.5.1. FTP and SELinux

When running SELinux, the FTP server, `vsftpd`, runs confined by default. SELinux policy defines how `vsftpd` interacts with files, processes, and with the system in general. For example, when an authenticated user logs in via FTP, they can not read from or write to files in their home directories: SELinux prevents `vsftpd` from accessing user home directories by default. Also, by default, `vsftpd` does not have access to NFS or CIFS file systems, and anonymous users do not have write access, even if such write access is configured in **`/etc/vsftpd/vsftpd.conf`**. Booleans can be turned on to allow the previously mentioned access.

The following example demonstrates an authenticated user logging in, and an SELinux denial when trying to view files in their home directory:

1. Run **`rpm -q vsftpd`** to see if the *vsftpd* package is installed. If it is not, run **`yum install vsftpd`** as the root user to install it.

2. In Fedora, `vsftpd` only allows anonymous users to log in by default. To allow authenticated users to log in, edit **`/etc/vsftpd/vsftpd.conf`** as the root user. Uncomment the **`local_enable=YES`** option:

   ```
   # Uncomment this to allow local users to log in.
   local_enable=YES
   ```

3. Run **`service vsftpd start`** as the root user to start `vsftpd`. If the service was running before editing **`vsftpd.conf`**, run **`service vsftpd restart`** as the root user to apply the configuration changes:

   ```
   service vsftpd start
   ```

[14] The first paragraph of "Chapter 23. FTP" of the Red Hat Enterprise Linux 5 Deployment Guide: *http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/ch-ftp.html*. Copyright © 2007 Red Hat, Inc.
[15] The first paragraph of the "23.2.1. vsftpd" section of the Red Hat Enterprise Linux 5 Deployment Guide: *http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/s1-ftp-servers.html#s2-ftp-servers-vsftpd*. Copyright © 2007 Red Hat, Inc.

```
Starting vsftpd for vsftpd:                                [  OK  ]
```

4. Run **ftp localhost** as the user you are currently logged in with. When prompted for your name, make sure your username is displayed. If the correct username is displayed, press **Enter**, otherwise, enter the correct username:

```
$ ftp localhost
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.1.0)
Name (localhost:username):
331 Please specify the password.
Password: Enter your password
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

5. Run the **ls** command from the **ftp** prompt. With the **ftp_home_dir** Boolean off, SELinux prevents vsftpd access to home directories, resulting in this command failing to return a directory listing:

```
ftp> ls
227 Entering Passive Mode (127,0,0,1,225,210).
150 Here comes the directory listing.
226 Transfer done (but failed to open directory).
```

An SELinux denial similar to the following is logged to **/var/log/messages**:

```
setroubleshoot: SELinux is preventing the ftp daemon from reading users home directories
 (username). For complete SELinux messages. run sealert -l c366e889-2553-4c16-
b73f-92f36a1730ce
```

6. Enable the **ftp_home_dir** Boolean by running the following command as the root user:

```
# setsebool -P ftp_home_dir=1
```

Note

Do not use the -P option if you do not want changes to persist across reboots.

Run the **ls** command again from the **ftp** prompt. Now that SELinux is allowing home directory browsing via the **ftp_home_dir** Boolean, the directory is displayed:

```
ftp> ls
227 Entering Passive Mode (127,0,0,1,56,215).
150 Here comes the directory listing.
-rw-rw-r--    1 501      501             0 Mar 30 09:22 file1
-rw-rw-r--    1 501      501             0 Mar 30 09:22 file2
```

```
226 Directory Send OK.
ftp>
```

10.5.2. Types

By default, anonymous users have read access to files in **`/var/ftp/`** when they log in via FTP. This directory is labeled with the **`public_content_t`** type, allowing only read access, even if write access is configured in **`/etc/vsftpd/vsftpd.conf`**. The **`public_content_t`** type is accessible to other services, such as Apache HTTP Server, Samba, and NFS.

Use one of the following types to share files through FTP:

`public_content_t`
: Label files and directories you have created with the **`public_content_t`** type to share them read-only through vsftpd. Other services, such as Apache HTTP Server, Samba, and NFS, also have access to files labeled with this type. Files labeled with the **`public_content_t`** type can not be written to, even if Linux permissions allow write access. If you require write access, use the **`public_content_rw_t`** type.

`public_content_rw_t`
: Label files and directories you have created with the **`public_content_rw_t`** type to share them with read and write permissions through `vsftpd`. Other services, such as Apache HTTP Server, Samba, and NFS, also have access to files labeled with this type; however, Booleans for each service must be turned on before such services can write to files labeled with this type.

10.5.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Booleans allow you to tell SELinux how you are running `vsftpd`:

`allow_ftpd_anon_write`
: When disabled, this Boolean prevents `vsftpd` from writing to files and directories labeled with the **`public_content_rw_t`** type. Turn this Boolean on to allow users to upload files via FTP. The directory where files are uploaded to must be labeled with the **`public_content_rw_t`** type and Linux permissions set accordingly.

`allow_ftpd_full_access`
: When this Boolean is on, only Linux permissions are used to control access, and authenticated users can read and write to files that are not labeled with the **`public_content_t`** or **`public_content_rw_t`** types.

`allow_ftpd_use_cifs`
: Having this Boolean enabled allows `vsftpd` to access files and directories labeled with the **`cifs_t`** type; therefore, having this Boolean enabled allows you to share file systems mounted via Samba through `vsftpd`.

`allow_ftpd_use_nfs`
: Having this Boolean enabled allows `vsftpd` to access files and directories labeled with the **`nfs_t`** type; therefore, having this Boolean enabled allows you to share file systems mounted via NFS through `vsftpd`.

`ftp_home_dir`
: Having this Boolean enabled allows authenticated users to read and write to files in their home directories. When this Boolean is off, attempting to download a file from a home directory results

in an error such as **`550 Failed to open file`**. An SELinux denial is logged to **`/var/log/messages`**.

`ftpd_connect_db`
: Allow FTP daemons to initiate a connection to a database.

`httpd_enable_ftp_server`
: Allow `httpd` to listen on the FTP port and act as a FTP server.

`tftp_anon_write`
: Having this Boolean enabled allows TFTP access to a public directory, such as an area reserved for common files that otherwise has no special access restrictions.

10.5.4. Configuration Examples

10.5.4.1. Uploading to an FTP site

The following example creates an FTP site that allows a dedicated user to upload files. It creates the directory structure and the required SELinux configuration changes:

1. Run **`mkdir -p /myftp/pub`** as the root user to create a new top-level directory.

2. Set Linux permissions on the **`/myftp/pub/`** directory to allow a Linux user write access. This example changes the owner and group from root to owner user1 and group root. Replace user1 with the user you want to give write access to:

   ```
   # chown user1:root /myftp/pub
   # chmod 775 /myftp/pub
   ```

   The **`chown`** command changes the owner and group permissions. The **`chmod`** command changes the mode, allowing the user1 user read, write, and execute permissions, and members of the root group read, write, and execute permissions. Everyone else has read and execute permissions: this is required to allow the Apache HTTP Server to read files from this directory.

3. When running SELinux, files and directories must be labeled correctly to allow access. Setting Linux permissions is not enough. Files labeled with the **`public_content_t`** type allow them to be read by FTP, Apache HTTP Server, Samba, and rsync. Files labeled with the **`public_content_rw_t`** type can be written to by FTP. Other services, such as Samba, require Booleans to be set before they can write to files labeled with the **`public_content_rw_t`** type. Label the top-level directory (**`/myftp/`**) with the **`public_content_t`** type, to prevent copied or newly-created files under **`/myftp/`** from being written to or modified by services. Run the following command as the root user to add the label change to file-context configuration:

   ```
   semanage fcontext -a -t public_content_t /myftp
   ```

4. Run **`restorecon -R -v /myftp/`** to apply the label change:

   ```
   # restorecon -R -v /myftp/
   restorecon reset /myftp context unconfined_u:object_r:default_t:s0-
   >system_u:object_r:public_content_t:s0
   ```

5. Confirm **`/myftp`** is labeled with the **`public_content_t`** type, and **`/myftp/pub/`** is labeled with the **`default_t`** type:

   ```
   $ ls -dZ /myftp/
   drwxr-xr-x. root root system_u:object_r:public_content_t:s0 /myftp/
   $ ls -dZ /myftp/pub/
   drwxrwxr-x. user1 root unconfined_u:object_r:default_t:s0 /myftp/pub/
   ```

6. FTP must be allowed to write to a directory before users can upload files via FTP. SELinux allows FTP to write to directories labeled with the **`public_content_rw_t`** type. This example uses **`/myftp/pub/`** as the directory FTP can write to. Run the following command as the root user to add the label change to file-context configuration:

   ```
   semanage fcontext -a -t public_content_rw_t "/myftp/pub(/.*)?"
   ```

7. Run **`restorecon -R -v /myftp/pub`** as the root user to apply the label change:

   ```
   # restorecon -R -v /myftp/pub
   restorecon reset /myftp/pub context system_u:object_r:default_t:s0-
   >system_u:object_r:public_content_rw_t:s0
   ```

8. The **`allow_ftpd_anon_write`** Boolean must be on to allow **`vsftpd`** to write to files that are labeled with the **`public_content_rw_t`** type. Run the following command as the root user to turn this Boolean on:

   ```
   setsebool -P allow_ftpd_anon_write on
   ```

   Do not use the **`-P`** option if you do not want changes to persist across reboots.

The following example demonstrates logging in via FTP and uploading a file. This example uses the user1 user from the previous example, where user1 is the dedicated owner of the **`/myftp/pub/`** directory:

1. Run **`cd ~/`** to change into your home directory. Then, run **`mkdir myftp`** to create a directory to store files to upload via FTP.

2. Run **`cd ~/myftp`** to change into the **`~/myftp/`** directory. In this directory, create an **`ftpupload`** file. Copy the following contents into this file:

   ```
   File upload via FTP from a home directory.
   ```

3. Run **`getsebool allow_ftpd_anon_write`** to confirm the **`allow_ftpd_anon_write`** Boolean is on:

   ```
   $ getsebool allow_ftpd_anon_write
   allow_ftpd_anon_write --> on
   ```

   If this Boolean is off, run **`setsebool -P allow_ftpd_anon_write on`** as the root user to turn it on. Do not use the **`-P`** option if you do not want the change to persist across reboots.

4. Run **`service vsftpd start`** as the root user to start `vsftpd`:

   ```
   # service vsftpd start
   Starting vsftpd for vsftpd:                                [  OK  ]
   ```

5. Run **`ftp localhost`**. When prompted for a username, enter the the username of the user who has write access, then, enter the correct password for that user:

   ```
   $ ftp localhost
   Connected to localhost (127.0.0.1).
   220 (vsFTPd 2.1.0)
   Name (localhost:username):
   331 Please specify the password.
   Password: Enter the correct password
   230 Login successful.
   Remote system type is UNIX.
   Using binary mode to transfer files.
   ftp>
   ```

10.6. Network File System

From the *Red Hat Linux Reference Guide*[16]:

NFS (Network File System) allows hosts to mount partitions on a remote system and use them as though they are local file systems. This allows the system administrator to store resources in a central location on the network, providing authorized users continuous access to them.

In Fedora, the *nfs-utils* package is required for full NFS support. Run **`rpm -q nfs-utils`** to see if the *nfs-utils* is installed. If it is not installed and you want to use NFS, run the following command as the root user to install it:

```
yum install nfs-utils
```

10.6.1. NFS and SELinux

When running SELinux, the `NFS` daemons are confined by default. SELinux policy does not allow NFS to share files by default. If you want to share NFS partitions, this can be configured via the **`nfs_export_all_ro`** and **`nfs_export_all_rw`** Booleans, as described below. These Booleans are however not required when files to be shared are labeled with the **`public_content_t`** or **`public_content_rw_t`** types. NFS can share files labeled with these types even if the **`nfs_export_all_ro`** and **`nfs_export_all_rw`** Booleans are off.

10.6.2. Types

By default, mounted NFS file systems on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **`nfs_t`** type.The following types are used with NFS. Different types allow you to configure flexible access:

[16] http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-nfs.html

`var_lib_nfs_t`
: This type is used for existing and new files copied to or created in the **`/var/lib/nfs`** directory. This type should not need to be changed in normal operation. To restore changes to the default settings, run the **`restorecon -R -v /var/lib/nfs`** command as the root user.

`nfsd_exec_t`
: The **`/usr/sbin/rpc.nfsd`** file is labeled with the **`nfsd_exec_t`**, as are other system executables and libraries related to NFS. Users should not label any files with this type. **`nfsd_exec_t`** will transition to **`nfs_t`**.

10.6.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Booleans allow you to tell SELinux how you are running NFS:

`allow_ftpd_use_nfs`
: When enabled, this Boolean allows `ftpd` access to NFS mounts.

`allow_nfsd_anon_write`
: When enabled, this Boolean allows `nfsd` to write to a public directory anonymously; such as to an area reserved for common files that otherwise has no special access restrictions.

`httpd_use_nfs`
: When enabled, this Boolean will allow `httpd` to access files stored on a NFS filesystem.

`nfs_export_all_ro`
: Export any file or directory via NFS, allowing read-only permissions.

`nfs_export_all_rw`
: Export any file or directory via NFS, allowing read and write permissions.

`qemu_use_nfs`
: Allow qemu to use NFS file systems.

`samba_share_nfs`
: When disabled, this Boolean prevents `smbd` from having full access to NFS shares via Samba. Enabling this Boolean will allow Samba to share NFS file systems.

`use_nfs_home_dirs`
: Having this Boolean enabled adds support for NFS home directories.

`virt_use_nfs`
: Allow virt to use NFS files.

`xen_use_nfs`
: Allow xen to manage NFS files.

10.6.4. Configuration Examples

10.6.4.1. Sharing directories using NFS

The example in this section creates a directory and shares it using NFS and SELinux. Two hosts are used in this example; a NFS server with a hostname of **`nfs-srv`** with an IP address of **`192.168.1.1`**, and a client with a hostname of **`nfs-client`** and an IP address of **`192.168.1.100`**.

Both hosts are on the same subnet (192.168.1.0/24). This is an example only and assumes that the *nfs-utils* package is installed, that the SELinux targeted policy is used, and that SELinux is running in enforced mode.

This example will show that while even with full network availability and Linux file permissions granting access to all users via NFS, SELinux is still able to block mounting of NFS file systems unless the proper permissions are given via SELinux Booleans.

10.6.4.1.1. Server setup

Steps 1-10 below should be performed on the NFS server, **`nfs-srv`**.

1. Run the **`setsebool`** command to disable read/write mounting of NFS file systems:

   ```
   setsebool -P nfs_export_all_rw off
   ```

   Note

   Do not use the **`-P`** option if you do not want **`setsebool`** changes to persist across reboots.

2. Run **`rpm -q nfs-utils`** to confirm the *nfs-utils* package is installed. The *nfs-utils* package provides support programs for using NFS and should be installed on a NFS server and on any clients in use. If this package is not installed, install it by running **`yum install`** ***`nfs-utils`*** as the root user.

3. Run **`mkdir /myshare`** as the root user to create a new top-level directory to share using NFS.

4. Run **`touch /myshare/file1`** as the root user to create a new empty file in the shared area. This file will be accessed later by the client.

5. To show that SELinux is still able to block access even when Linux permissions are completely open, give the **`/myshare`** directory full Linux access rights for all users:

   ```
   # chmod -R 777 /myshare
   ```

   Warning

   This is an example only and these permissions should not be used in a production system.

6. Edit the **`/etc/exports`** file and add the following line to the top of the file:

   ```
   /myshare  192.168.1.100(rw)
   ```

   This entry shows the full path on the server to the shared folder **`/myshare`**, the host or network range that **`nfs-srv`** will share to (in this case the IP address of a single host, **`nfs-client`** at

192.168.1.100), and finally the share permissions. Read and write permissions are given here, as indicated by **(rw)**.

7. The TCP and UDP ports used for NFS are assigned dynamically by rpcbind, which can cause problems when creating firewall rules. To simplify the process of allowing NFS traffic through the firewall in this example, edit the */etc/sysconfig/nfs* file and uncomment the **MOUNTD_PORT**,**STATD_PORT**,**LOCKD_TCPPORT** and **LOCKD_UDPPORT** variables. Changing the port numbers in this file is not required for this example.

   Ensure that incoming connections on TCP ports 111, 892 and 2049 are allowed through the server's firewall. This can be done via the *system-config-firewall* tool in Fedora.

8. Run **service nfs start** as the root user to start NFS and its related services:

   ```
   # service nfs start
   Starting NFS services:  [  OK  ]
   Starting NFS quotas:  [  OK  ]
   Starting NFS daemon:  [  OK  ]
   Starting NFS mountd:  [  OK  ]
   ```

9. To ensure that the NFS subsystem export table is updated, run **exportfs -rv** as the root user:

   ```
   # exportfs -rv
   exporting 192.168.1.100:/myshare
   ```

10. Run **showmount -e** as the root user to show all exported file systems:

    ```
    # showmount -e
    Export list for nfs-srv:
    /myshare 192.168.1.100
    ```

At this point the server **nfs-srv** has been configured to allow NFS communications to **nfs-client** at **192.168.1.100**, and full Linux file systems permissions are active. If SELinux were disabled, the client would be able to mount this share and have full access over it. However, as the **nfs_export_all_rw** Boolean is disabled, the client is currently not able to mount this file system, as shown below. This step should be performed on the client, **nfs-client**:

```
[nfs-client]# mkdir /myshare
[nfs-client]# mount.nfs 192.168.1.1:/myshare /myshare
mount.nfs: access denied by server while mounting 192.168.1.1:/myshare/
```

Enable the SELinux Boolean that was disabled in Step 1 above, and the client will be able to successfully mount the shared file system. This step should be performed on the NFS server, **nfs-srv**:

```
[nfs-srv]# setsebool -P nfs_export_all_rw on
```

Now try to mount the NFS file system again. This step should be performed on the NFS client, **nfs-client**:

```
[nfs-client]# mount.nfs 192.168.1.1:/myshare /myshare
[nfs-client]#
[nfs-client]# ls /myshare
total 0
-rwxrwxrwx.  1 root root 0 2009-04-16 12:07 file1
[nfs-client]#
```

The file system has been mounted successfully by the client. This example demonstrates how SELinux adds another layer of protection and can still enforce SELinux permissions even when Linux permissions are set to give full rights to all users.

10.7. Berkeley Internet Name Domain

BIND performs name resolution services via the `named` daemon. BIND lets users locate computer resources and services by name instead of numerical addresses.

In Fedora, the *bind* package provides a DNS server. Run **`rpm -q bind`** to see if the *bind* package is installed. If it is not installed and you want to use BIND, run the following command as the root user to install it:

```
yum install bind
```

10.7.1. BIND and SELinux

The default permissions on the **`/var/named/slaves`**,**`/var/named/dynamic`** and **`/var/named/data`** directories allow zone files to be updated via zone transfers and dynamic DNS updates. Files in **`/var/named`** are labeled with the **`name_zone_t`** type, which is used for master zone files.

For a slave server, configure **`/etc/named.conf`** to place slave zones in **`/var/named/slaves`**. The following is an example of a domain entry in **`/etc/named.conf`** for a slave DNS server that stores the zone file for **`testdomain.com`** in **`/var/named/slaves`**:

```
zone "testdomain.com" {
   type slave;
   masters { IP-address; };
   file "/var/named/slaves/db.testdomain.com";
         };
```

If a zone file is labeled **`name_zone_t`**, the **`named_write_master_zones`** Boolean must be enabled to allow zone transfers and dynamic DNS to update the zone file. Also, the mode of the parent directory has to be changed to allow the `named` user or group read, write and execue access.

If zone files in **`/var/named/`** are labeled with **`name_cache_t`** type, a file system relabel or running **`restorecon -R /var/`** will change their type to **`named_zone_t`**.

10.7.2. Types

The following types are used with BIND. Different types allow you to configure flexible access:

`named_zone_t`
: Used for master zone files. Other services can not modify files of this type. `named` can only modify files of this type if the **`named_write_master_zones`** Boolean is turned on.

`named_cache_t`

By default, `named` can write to files labeled with this type, without additional Booleans being set. Files copied or created in the **`/var/named/slaves`**,**`/var/named/dynamic`** and **`/var/named/data`** directories are automatically labeled with the **`named_cache_t`** type.

10.7.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Booleans allow you to tell SELinux how you are running NFS:

`named_write_master_zones`

When disabled, this Boolean prevents `named` from writing to zone files or directories labeled with the **`named_zone_t`** type. `named` does not usually need to write to zone files; but in the case that it needs to, or if a secondary server needs to write to zone files, enable this Boolean to allow this action.

10.7.4. Configuration Examples

10.7.4.1. Dynamic DNS

BIND allows hosts to update their records in DNS and zone files dynamically. This is used when a host computer's IP address changes frequently and the DNS record requires real-time modification.

Use the **`/var/named/dynamic`** directory for zone files you want updated via dynamic DNS. Files created in or copied into **`/var/named/dynamic`** inherit Linux permissions that allow `named` to write to them. As such files are labeled with the **`named_cache_t`** type, SELinux allows `named` to write to them.

If a zone file in **`/var/named/dynamic`** is labeled with the **`named_zone_t`** type, dynamic DNS updates may not be successful for a certain period of time as the update needs to be written to a journal first before being merged. If the zone file is labeled with the **`named_zone_t`** type when the journal attempts to be merged, an error such as the following is logged to **`/var/log/messages`**:

```
named[PID]: dumping master file: rename: /var/named/dynamic/zone-name: permission denied
```

Also, the following SELinux denial is logged to **`/var/log/messages`**:

```
setroubleshoot: SELinux is preventing named (named_t) "unlink" to zone-name (named_zone_t)
```

To resolve this labeling issue, run the **`restorecon -R -v /var/named/dynamic`** command as the Linux root user.

10.8. Concurrent Versioning System

The Concurrent Versioning System (CVS) is a free revision-control system. It is used to monitor and keep track of modifications to a central set of files which are usually accessed by several different users. It is commonly used by programmers to manage a source code repository and is widely used by open source programmers.

In Fedora, the *cvs* package provides CVS. Run **`rpm -q cvs`** to see if the *cvs* package is installed. If it is not installed and you want to use CVS, run the following command as the root user to install it:

```
yum install cvs
```

10.8.1. CVS and SELinux

The `cvs` daemon runs as **`cvs_t`**. By default in Fedora, CVS is only allowed to read and write certain directories. The label **`cvs_data_t`** defines which areas the `cvs` daemon has read and write access to. When using CVS with SELinux, assigning the correct label is essential for clients to have full access to the area reserved for CVS data.

10.8.2. Types

The following types are used with CVS. Different types allow you to configure flexible access:

`cvs_data_t`
: This type is used for data in a CVS repository. CVS can only gain full access to data if it has this type.

`cvs_exec_t`
: This type is used for the **`/usr/bin/cvs`** binary.

10.8.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Boolean allows you to tell SELinux how you are running CVS:

`allow_cvs_read_shadow`
: This Boolean allows the `cvs` daemon to access the **`/etc/shadow`** file for user authentication.

10.8.4. Configuration Examples

10.8.4.1. Setting up CVS

This example describes a simple CVS setup and an SELinux configuration which allows remote access. Two hosts are used in this example; a CVS server with a hostname of **`cvs-srv`** with an IP address of **`192.168.1.1`** and a client with a hostname of **`cvs-client`** and an IP address of **`192.168.1.100`**. Both hosts are on the same subnet (192.168.1.0/24). This is an example only and assumes that the *cvs* and *xinetd* packages are installed, that the SELinux targeted policy is used, and that SELinux is running in enforced mode.

This example will show that even with full DAC permissions, SELinux can still enforce policy rules based on file labels and only allow access to certain areas that have been specifically labeled for access by CVS.

10.8.4.2. Server setup

Note

Steps 1-9 should be performed on the CVS server, **`cvs-srv`**.

1. As the root user, install the *cvs* and *xinetd* packages. Run **`rpm -q cvs`** to see if the *cvs* package is installed. If it is not installed, run **`yum install cvs`** as the root user to install it. Run **`rpm -q xinetd`** to see if the *xinetd* package is installed. If it is not installed, run **`yum install xinetd`** as the root user to install it.

2. Create a group named **`CVS`**. This can be done via the **`groupadd CVS`** command as the root user, or by using the *system-config-users* tool.

3. Create a user with a username of **`cvsuser`** and make this user a member of the CVS group. This can be done using the *system-config-users* tool.

4. Edit the **`/etc/services`** file and make sure that the CVS server has uncommented entries looking similar to the following:

```
cvspserver 2401/tcp   # CVS client/server operations
cvspserver 2401/udp   # CVS client/server operations
```

5. Create the CVS repository in the root area of the file system. When using SELinux, it is best to have the repository in the root file system so that recursive labels can be given to it without affecting any other subdirectories. For example, as the root user, create a **`/cvs`** directory to house the repository:

```
[root@cvs-srv]# mkdir /cvs
```

6. Give full permissions to the **`/cvs`** directory to all users:

```
[root@cvs-srv]# chmod -R 777 /cvs
```

Warning

This is an example only and these permissions should not be used in a production system.

7. Edit the **`/etc/xinetd.d/cvs`** file and make sure that the CVS section is uncommented and configured to use the **`/cvs`** directory. The file should look similar to:

```
service cvspserver
{
 disable = no
 port   = 2401
 socket_type  = stream
 protocol  = tcp
 wait   = no
 user   = root
 passenv   = PATH
 server   = /usr/bin/cvs
 env   = HOME=/cvs
 server_args  = -f --allow-root=/cvs pserver
# bind   = 127.0.0.1
```

8. Start the `xinetd` daemon by running **`service xinetd start`** as the root user.

9. Add a rule which allows inbound connections using TCP on port 2401 by using the *system-config-firewall* tool.

10. As the **`cvsuser`** user, run the following command:

    ```
    [cvsuser@cvs-client]$ cvs -d /cvs init
    ```

11. At this point, CVS has been configured but SELinux will still deny logins and file access. To demonstrate this, set the $CVSROOT variable on **`cvs-client`** and try to log in remotely. The following step should be performed on **`cvs-client`**:

    ```
    [cvsuser@cvs-client]$ export CVSROOT=:pserver:cvsuser@192.168.1.1:/cvs
    [cvsuser@cvs-client]$
    [cvsuser@cvs-client]$ cvs login
    Logging in to :pserver:cvsuser@192.168.1.1:2401/cvs
    CVS password: ********
    cvs [login aborted]: unrecognized auth response from 192.168.100.1: cvs pserver: cannot
     open /cvs/CVSROOT/config: Permission denied
    ```

    SELinux has blocked access. In order to get SELinux to allow this access, the following step should be performed on **`cvs-srv`**:

12. Change the context of the **`/cvs`** directory as the root user in order to recursively label any existing and new data in the **`/cvs`** directory, giving it the **`cvs_data_t`** type:

    ```
    [root@cvs-srv]# semanage fcontext -a -t cvs_data_t '/cvs(/.*)?'
    [root@cvs-srv]# restorecon -R -v /cvs
    ```

13. The client, **`cvs-client`** should now be able to log in and access all CVS resources in this repository:

    ```
    [cvsuser@cvs-client]$ export CVSROOT=:pserver:cvsuser@192.168.1.1:/cvs
    [cvsuser@cvs-client]$
    [cvsuser@cvs-client]$ cvs login
    Logging in to :pserver:cvsuser@192.168.1.1:2401/cvs
    CVS password: ********
    [cvsuser@cvs-client]$
    ```

10.9. Squid Caching Proxy

From the *Squid Caching Proxy*[17] project page:

"Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator."

In Fedora, the *squid* package provides the Squid Caching Proxy. Run **`rpm -q squid`** to see if the *squid* package is installed. If it is not installed and you want to use squid, run the following command as the root user to install it:

```
# yum install squid
```

10.9.1. Squid Caching Proxy and SELinux

When SELinux is enabled, squid runs confined by default. Confined processes run in their own domains, and are separated from other confined processes. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited. The following example demonstrates the squid processes running in their own domain. This example assumes the squid package is installed:

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run **`service squid start`** as the root user to start `squid`:

   ```
   # service squid start
   Starting squid:                                            [  OK  ]
   ```

3. Run **`ps -eZ | grep squid`** to view the `squid` processes:

   ```
   $ ps -eZ | grep squid
   unconfined_u:system_r:squid_t:s0 2522 ?        00:00:00 squid
   unconfined_u:system_r:squid_t:s0 2524 ?        00:00:00 squid
   unconfined_u:system_r:squid_t:s0 2526 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2527 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2528 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2529 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2530 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2531 ?        00:00:00 unlinkd
   ```

   The SELinux context associated with the `squid` processes is **`unconfined_u:system_r:squid_t:s0`**. The second last part of the context, **`squid_t`**, is the

[17] http://www.squid-cache.org/

type. A type defines a domain for processes and a type for files. In this case, the `squid` processes are running in the **`squid_t`** domain.

SELinux policy defines how processes running in confined domains, such as squid_t, interact with files, other processes, and the system in general. Files must be labeled correctly to allow squid access to them.

When **`/etc/squid/squid.conf`** is configured so `squid` listens on a port other than the default TCP ports 3128, 3401 or 4827, the **`semanage port`** command must be used to add the required port number to the SELinux policy configuration. The following example demonstrates configuring `squid` to listen on a port that is not initially defined in SELinux policy configuration for `squid`, and, as a consequence, `squid` failing to start. This example also demonstrates how to then configure the SELinux system to allow `squid` to successfully listen on a non-standard port that is not already defined in the policy. This example assumes the *squid* package is installed. Run each command in the example as the root user:

1. Run **`service squid status`** to confirm `squid` is not running:

```
# service squid status
squid is stopped
```

   If the output differs, run **`service squid stop`** to stop the process:

```
# service squid stop
Stopping squid:                                            [  OK  ]
```

2. Run **`semanage port -l | grep -w squid_port_t`** to view the ports SELinux allows `squid` to listen on:

```
semanage port -l | grep -w -i squid_port_t
squid_port_t                   tcp      3128, 3401, 4827
squid_port_t                   udp      3401, 4827
```

3. Edit **`/etc/squid/squid.conf`** as the root user. Configure the **`http_port`** option so it lists a port that is not configured in SELinux policy configuration for `squid`. In this example, `squid` is configured to listen on port 10000:

```
# Squid normally listens to port 3128
http_port 10000
```

4. Run **`service squid start`** to start `squid`:

```
# service squid start
Starting squid: ....................                       [FAILED]
```

   An SELinux denial similar to the following is logged to **`/var/log/messages`**:

```
localhost setroubleshoot: SELinux is preventing the squid (squid_t) from binding to port
 1000. For complete SELinux messages. run sealert -l 97136444-4497-4fff-a7a7-c4d8442db982
```

5. For SELinux to allow `squid` to listen on port 10000, as used in this example, the following command is required:

```
# semanage port -a -t squid_port_t -p tcp 10000
```

6. Run **`service squid start`** again to start `squid` and have it listen on the new port:

```
# service squid start
Starting squid:          [  OK  ]
```

7. Now that SELinux has been configured to allow `squid` to listen on a non-standard port (TCP 10000 in this example), `squid` starts successfully on this port.

10.9.2. Types

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

The following types are used with `squid`. Different types allow you to configure flexible access:

`httpd_squid_script_exec_t`
: This type is used for utilities such as **`cachemgr.cgi`**, which provides a variety of statistics about squid and its configuration.

`squid_cache_t`
: Use this type for data that is cached by squid, as defined by the **`cache_dir`** directive in **`/etc/squid/squid.conf`**. By default, files created in or copied into **`/var/cache/squid`** and **`/var/spool/squid`** are labeled with the **`squid_cache_t`** type. Files for the *squidGuard*[18] URL redirector plugin for `squid` created in or copied to **`/var/squidGuard`** are also labeled with the **`squid_cache_t`** type. Squid is only able to use files and directories that are labeled with this type for its cached data.

`squid_conf_t`
: This type is used for the directories and files that `squid` uses for its configuration. Existing files, or those created in or copied to **`/etc/squid`** and **`/usr/share/squid`** are labeled with this type, including error messages and icons.

`squid_exec_t`
: This type is used for the squid binary, **`/usr/sbin/squid`**.

`squid_log_t`
: This type is used for logs. Existing files, or those created in or copied to **`/var/log/squid`** or **`/var/log/squidGuard`** must be labeled with this type.

[18] http://www.squidguard.org/

`squid_initrc_exec_t`
: This type is used for the initialization file required to start `squid` which is located at **`/etc/rc.d/init.d/squid`**.

`squid_var_run_t`
: This type is used by files in **`/var/run`**, especially the process id (PID) named **`/var/run/squid.pid`** which is created by squid when it runs.

10.9.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Boolean allows you to tell SELinux how you are running Squid:

`squid_connect_any`
: When enabled, this Boolean permits squid to initiate a connection to a remote host on any port.

`squid_use_tproxy`
: When enabled, this Boolean allows Squid to run as a transparent proxy.

10.9.4. Configuration Examples

10.9.4.1. Squid Connecting to Non-Standard Ports

The following example provides a real-world demonstration of how SELinux complements Squid by enforcing the above Boolean and by default only allowing access to certain ports. This example will then demonstrate how to change the Boolean and show that access is then allowed.

Note that this is an example only and demonstrates how SELinux can affect a simple configuration of Squid. Comprehensive documentation of Squid is beyond the scope of this document. Refer to the official *Squid documentation*[19] for further details. This example assumes that the Squid host has two network interfaces, Internet access, and that any firewall has been configured to allow access on the internal interface using the default TCP port on which Squid listens (TCP 3128).

1. As the root user, install the *squid* package. Run **`rpm -q squid`** to see if the *squid* package is installed. If it is not installed, run **`yum install squid`** as the root user to install it.

2. Edit the main configuration file, **`/etc/squid/squid.conf`** and confirm that the **`cache_dir`** directive is uncommented and looks similar to the following:

   ```
   cache_dir ufs /var/spool/squid 100 16 256
   ```

   This line specifies the default settings for the **`cache_dir`** directive to be used in this example; it consists of the Squid storage format (ufs), the directory on the system where the cache resides (/var/spool/squid), the amount of disk space in megabytes to be used for the cache (100), and finally the number of first-level and second-level cache directories to be created (16 and 256 respectively).

3. In the same configuration file, make sure the **`http_access allow localnet`** directive is uncommented. This allows traffic from the **`localnet`** ACL which is automatically configured in a

[19] http://www.squid-cache.org/Doc/

default installation of Squid on Fedora 13. It will allow client machines on any existing RFC1918 network to have access through the proxy, which is sufficient for this simple example.

4. In the same configuration file, make sure the **`visible_hostname`** directive is uncommented and is configured to the hostname of the machine. The value should be the fully qualified domain name (FQDN) of the host:

```
visible_hostname squid.example.com
```

5. As the root user, run **`service squid start`** to start `squid`. As this is the first time `squid` has started, this command will initialise the cache directories as specified above in the **`cache_dir`** directive and will then start the `squid` daemon. The output is as follows if `squid` starts successfully:

```
# /sbin/service squid start
init_cache_dir /var/spool/squid... Starting squid: .       [  OK  ]
```

6. Confirm that the `squid` process ID (PID) has started as a confined service, as seen here by the **`squid_var_run_t`** value:

```
# ls -lZ /var/run/squid.pid
-rw-r--r--. root squid unconfined_u:object_r:squid_var_run_t:s0 /var/run/squid.pid
```

7. At this point, a client machine connected to the **`localnet`** ACL configured earlier is successfully able to use the internal interface of this host as its proxy. This can be configured in the settings for all common web browsers, or system-wide. Squid is now listening on the default port of the target machine (TCP 3128), but the target machine will only allow outgoing connections to other services on the Internet via common ports. This is a policy defined by SELinux itself. SELinux will deny access to non-standard ports, as shown in the next step:

8. When a client makes a request using a non-standard port through the Squid proxy such as a website listening on TCP port 10000, a denial similar to the following is logged:

```
SELinux is preventing the squid daemon from connecting to network port 10000
```

9. To allow this access, the **`squid_connect_any`** Boolean must be modified, as it is disabled by default. To turn the **`squid_connect_any`** Boolean on, run the following command as the root user:

```
# setsebool -P squid_connect_any on
```

Note

Do not use the **`-P`** option if you do not want **`setsebool`** changes to persist across reboots.

10. The client will now be able to access non-standard ports on the Internet as Squid is now permitted to initiate connections to any port, on behalf of its clients.

10.10. MySQL

From the *MySQL*[20] project page:

"The MySQL® database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used on every continent -- Yes, even Antarctica! -- by individual Web developers as well as many of the world's largest and fastest-growing organizations to save time and money powering their high-volume Web sites, business-critical systems and packaged software -- including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, and Zappos.com."

In Fedora, the *mysql-server* package provides MySQL. Run **`rpm -q mysql-server`** to see if the *mysql-server* package is installed. If it is not installed, run the following command as the root user to install it:

```
yum install mysql-server
```

10.10.1. MySQL and SELinux

When MySQL is enabled, it runs confined by default. Confined processes run in their own domains, and are separated from other confined processes. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited. The following example demonstrates the MySQL processes running in their own domain. This example assumes the *mysql* package is installed:

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run **`service mysqld start`** as the root user to start `mysqld`:

   ```
   # service mysqld start
   Initializing MySQL database:  Installing MySQL system tables... [  OK  ]
   Starting MySQL:                                                [  OK  ]
   ```

3. Run **`ps -eZ | grep mysqld`** to view the `mysqld` processes:

   ```
   $ ps -eZ | grep mysqld
   unconfined_u:system_r:mysqld_safe_t:s0 6035 pts/1 00:00:00 mysqld_safe
   unconfined_u:system_r:mysqld_t:s0 6123 pts/1   00:00:00 mysqld
   ```

[20] http://www.mysql.com/why-mysql/

The SELinux context associated with the `mysqld` processes is **`unconfined_u:system_r:mysqld_t:s0`**. The second last part of the context, **`mysqld_t`**, is the type. A type defines a domain for processes and a type for files. In this case, the `mysqld` processes are running in the **`mysqld_t`** domain.

10.10.2. Types

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

The following types are used with `mysql`. Different types allow you to configure flexible access:

`mysqld_db_t`
: This type is used for the location of the MySQL database. In Fedora 12, the default location for the database is **`/var/lib/mysql`**, however this can be changed. If the location for the MySQL database is changed, the new location must be labeled with this type. Refer to the following example for instructions on how to change the default database location and how to label the new section appropriately.

`mysqld_etc_t`
: This type is used for the MySQL main configuration file **`/etc/my.cnf`** and any other configuration files in the **`/etc/mysql`** directory.

`mysqld_exec_t`
: This type is used for the `mysqld` binary located at **`/usr/libexec/mysqld`**, which is the default location for the MySQL binary on Fedora 12. Other systems may locate this binary at **`/usr/sbin/mysqld`** which should also be labeled with this type.

`mysqld_initrc_exec_t`
: This type is used for the initialization file for MySQL, located at **`/etc/rc.d/init.d/mysqld`** by default in Fedora 12.

`mysqld_log_t`
: Logs for MySQL need to be labeled with this type for proper operation. All log files in **`/var/log/`** matching the **`mysql.*`** wildcard must be labeled with this type.

`mysqld_var_run_t`
: This type is used by files in **`/var/run/mysqld`**, specifically the process id (PID) named **`/var/run/mysqld/mysqld.pid`** which is created by the `mysqld` daemon when it runs. This type is also used for related socket files such as **`/var/lib/mysql/mysql.sock`**. Files such as these must be labeled correctly for proper operation as a confined service.

10.10.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Boolean allows you to tell SELinux how you are running MySQL:

`exim_can_connect_db`
: When enabled, this Boolean allows the `exim` mailer to initiate connections to a database server.

`ftpd_connect_db`
: When enabled, this Boolean allows `ftp` daemons to initiate connections to a database server.

`httpd_can_network_connect_db`
Enabling this Boolean is required for a web server to communicate with a database server.

10.10.4. Configuration Examples

10.10.4.1. MySQL Changing Database Location

When using Fedora 12, the default location for MySQL to store its database is **`/var/lib/mysql`**. This is where SELinux expects it to be by default, and hence this area is already labeled appropriately for you, using the **`mysqld_db_t`** type.

The area where the database is located can be changed depending on individual environment requirements or preferences, however it is important that SELinux is aware of this new location - that it is labeled accordingly. This example explains how to change the location of a MySQL database and then how to label the new location so that SELinux can still provide its protection mechanisms to the new area based on its contents.

Note that this is an example only and demonstrates how SELinux can affect MySQL. Comprehensive documentation of MySQL is beyond the scope of this document. Refer to the official *MySQL documentation*[21] for further details. This example assumes that the *mysql-server* package is installed and that there is a valid database in the default location of **`/var/lib/mysql`**.

1. Run **`ls -lZ /var/lib/mysql`** to view the SELinux context of the default database location for `mysql`:

   ```
   # ls -lZ /var/lib/mysql
   drwx------. mysql mysql unconfined_u:object_r:mysqld_db_t:s0 mysql
   ```

   This shows **`mysqld_db_t`** which is the default context element for the location of database files. This context will have to be manually applied to the new database location that will be used in this example in order for it to function properly.

2. Enter **`mysqlshow -u root -p`** and enter the `mysqld` root password to show the available databases:

   ```
   # mysqlshow -u root -p
   Enter password: *******
   +--------------------+
   |     Databases      |
   +--------------------+
   | information_schema |
   | mysql              |
   | test               |
   | wikidb             |
   +--------------------+
   ```

3. Shut down the `mysqld` daemon with **`service mysqld stop`** as the root user:

   ```
   # service mysqld stop
   Stopping MySQL:                                            [  OK  ]
   ```

[21] http://dev.mysql.com/doc/

4. Create a new directory for the new location of the database(s). In this example, **`/opt/mysql`** is used:

   ```
   # mkdir -p /opt/mysql
   ```

5. Copy the database files from the old location to the new location:

   ```
   # cp -R /var/lib/mysql/* /opt/mysql/
   ```

6. Change the ownership of this location to allow access by the mysql user and group. This sets the traditional Unix permissions which SELinux will still observe.

   ```
   # chown -R mysql:mysql /opt/mysql
   ```

7. Run **`ls -lZ /opt`** to see the initial context of the new directory:

   ```
   # ls -lZ /opt
   drwxr-xr-x. mysql mysql unconfined_u:object_r:usr_t:s0   mysql
   ```

   The context **`usr_t`** of this newly created directory is not currently suitable to SELinux as a location for MySQL database files. Once the context has been changed, MySQL will be able to function properly in this area.

8. Open the main MySQL configuration file **`/etc/my.cnf`** with a text editor and modify the **`datadir`** option so that it refers to the new location. In this example the value that should be entered is **`/opt/mysql`**.

   ```
   [mysqld]
   datadir=/opt/mysql
   ```

   Save this file and exit.

9. Run **`service mysqld start`** as the root user to start `mysqld`. At this point a denial will be logged to **`/var/log/messages`**:

   ```
   # service mysqld start

   Timeout error occurred trying to start MySQL Daemon.
   Starting MySQL:                                            [FAILED]

   # tail -f /var/log/messages

   localhost setroubleshoot: SELinux is preventing mysqld (mysqld_t) "write" usr_t. For
    complete SELinux messages. run sealert -l 50d8e725-994b-499c-9caf-a676c50fb802
   ```

   The reason for this denial is that **`/opt/mysql`** is not labeled correctly for MySQL data files. SELinux is stopping MySQL from having access to the content labeled as **`usr_t`**. Perform the following steps to resolve this problem:

10. Run the **`semanage`** command to add a context mapping for **`/opt/mysql`**:

```
semanage fcontext -a -t mysqld_db_t "/opt/mysql(/.*)?"
```

11. This mapping is written to the **`/etc/selinux/targeted/contexts/files/file_contexts.local`** file:

```
# grep -i mysql /etc/selinux/targeted/contexts/files/file_contexts.local

/opt/mysql(/.*)?    system_u:object_r:mysqld_db_t:s0
```

12. Now use the **`restorecon`** command to apply this context mapping to the running system:

```
restorecon -R -v /opt/mysql
```

13. Now that the **`/opt/mysql`** location has been labeled with the correct context for MySQL, the `mysqld` daemon starts:

```
# service mysqld start
Starting MySQL:                                             [  OK  ]
```

14. Confirm the context has changed for **`/opt/mysql`**:

```
ls -lZ /opt
drwxr-xr-x. mysql mysql system_u:object_r:mysqld_db_t:s0 mysql
```

15. The location has been changed and labeled, and the `mysqld` daemon has started successfully. At this point all running services should be tested to confirm normal operation.

10.11. PostgreSQL

From the *PostgreSQL*[22] project page:

"PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness."

In Fedora, the *postgresql-server* package provides PostgreSQL. Run **`rpm -q postgresql-server`** to see if the *postgresql-server* package is installed. If it is not installed, run the following command as the root user to install it:

```
yum install postgresql-server
```

[22] http://www.postgresql.org/about/

10.11.1. PostgreSQL and SELinux

When PostgreSQL is enabled, it runs confined by default. Confined processes run in their own domains, and are separated from other confined processes. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited. The following example demonstrates the PostgreSQL processes running in their own domain. This example assumes the *postgresql-server* package is installed:

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run **`service postgresql start`** as the root user to start `postgresql`:

   ```
   service postgresql start
   Starting postgresql service:                               [  OK  ]
   ```

3. Run **`ps -eZ | grep postgres`** to view the `postgresql` processes:

   ```
   ps -eZ | grep postgres
   unconfined_u:system_r:postgresql_t:s0 395 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 397 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 399 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 400 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 401 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 402 ?    00:00:00 postmaster
   ```

   The SELinux context associated with the `postgresql` processes is **`unconfined_u:system_r:postgresql_t:s0`**. The second last part of the context, **`postgresql_t`**, is the type. A type defines a domain for processes and a type for files. In this case, the `postgresql` processes are running in the **`postgresql_t`** domain.

10.11.2. Types

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

The following types are used with `postgresql`. Different types allow you to configure flexible access:

`postgresql_db_t`

This type is used for several locations. The locations labeled with this type are used for data files for PostgreSQL:

- **`/usr/lib/pgsql/test/regres`**
- **`/usr/share/jonas/pgsql`**
- **`/var/lib/pgsql/data`**

- **`/var/lib/postgres(ql)?`**

`postgresql_etc_t`
This type is used for configuration files in **`/etc/postgresql`**.

`postgresql_exec_t`
This type is used for several locations. The locations labeled with this type are used for binaries for PostgreSQL:

- **`/usr/bin/initdb(.sepgsql)?`**
- **`/usr/bin/(se)?postgres`**
- **`/usr/lib(64)?/postgresql/bin/.*`**
- **`/usr/lib/phsql/test/regress/pg_regress`**

`postgresql_initrc_exec_t`
This type is used for the PostgreSQL initialization file located at **`/etc/rc.d/init.d/postgresql`**.

`postgresql_log_t`
This type is used for several locations. The locations labeled with this type are used for log files:

- **`/var/lib/pgsql/logfile`**
- **`/var/lib/pgsql/pgstartup.log`**
- **`/var/lib/sepgsql/pgstartup.log`**
- **`/var/log/postgresql`**
- **`/var/log/postgres.log.*`**
- **`/var/log/rhdb/rhdb`**
- **`/var/log/sepostgresql.log.*`**

`postgresql_var_run_t`
This type is used for run-time files for PostgreSQL, such as the process id (PID) in **`/var/run/postgresql`**.

10.11.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Boolean allows you to tell SELinux how you are running PostgreSQL:

`allow_user_postgresql_connect`
Having this Boolean enabled allows any user domain (as defined by PostgreSQL) to make connections to the database server.

10.11.4. Configuration Examples

10.11.4.1. PostgreSQL Changing Database Location

When using Fedora 12, the default location for PostgreSQL to store its database is **`/var/lib/pgsql/data`**. This is where SELinux expects it to be by default, and hence this area is already labeled appropriately for you, using the **`postgresql_db_t`** type.

The area where the database is located can be changed depending on individual environment requirements or preferences, however it is important that SELinux is aware of this new location - that it is labeled accordingly. This example explains how to change the location of a PostgreSQL database and then how to label the new location so that SELinux can still provide its protection mechanisms to the new area based on its contents.

Note that this is an example only and demonstrates how SELinux can affect PostgreSQL. Comprehensive documentation of PostgreSQL is beyond the scope of this document. Refer to the official *PostgreSQL documentation*[23] for further details. This example assumes that the *postgresql-server* package is installed.

1. Run **`ls -lZ /var/lib/pgsql`** to view the SELinux context of the default database location for `postgresql`:

   ```
   # ls -lZ /var/lib/pgsql
   drwx------. postgres postgres system_u:object_r:postgresql_db_t:s0 data
   ```

   This shows **`postgresql_db_t`** which is the default context element for the location of database files. This context will have to be manually applied to the new database location that will be used in this example in order for it to function properly.

2. Create a new directory for the new location of the database(s). In this example, **`/opt/postgresql/data`** is used. If you use a different location, replace the text in the following steps with your location:

   ```
   # mkdir -p /opt/postgresql/data
   ```

3. Perform a directory listing of the new location. Note that the initial context of the new directory is *usr_t*. This context is not sufficient for SELinux to offer its protection mechanisms to PostgreSQL. Once the context has been changed, it will be able to function properly in the new area.

   ```
   # ls -lZ /opt/postgresql/
   drwxr-xr-x. root root unconfined_u:object_r:usr_t:s0   data
   ```

4. Change the ownership of the new location to allow access by the postgres user and group. This sets the traditional Unix permissions which SELinux will still observe.

   ```
   # chown -R postgres:postgres /opt/postgresql
   ```

[23] http://www.postgresql.org/docs/

5. Open the PostgreSQL init file **`/etc/rc.d/init.d/postgresql`** with a text editor and modify all **`PGDATA`** and **`PGLOG`** variables to point to the new location:

   ```
   # vi /etc/rc.d/init.d/postgresql
   PGDATA=/opt/postgresql/data
   PGLOG=/opt/postgresql/data/pgstartup.log
   ```

   Save this file and exit the text editor.

6. Initialize the database in the new location.

   ```
   su - postgres -c "initdb -D /opt/postgresql/data"
   ```

7. Run the **`semanage`** command to add a context mapping for **`/opt/postgresql`** and any other directories/files within it:

   ```
   semanage fcontext -a -t postgresql_db_t "/opt/postgresql(/.*)?"
   ```

8. This mapping is written to the **`/etc/selinux/targeted/contexts/files/file_contexts.local`** file:

   ```
   # grep -i postgresql /etc/selinux/targeted/contexts/files/file_contexts.local

   /opt/postgresql(/.*)?    system_u:object_r:postgresql_db_t:s0
   ```

9. Now use the **`restorecon`** command to apply this context mapping to the running system:

   ```
   restorecon -R -v /opt/postgresql
   ```

10. Now that the **`/opt/postgresql`** location has been labeled with the correct context for PostgreSQL, the `mysqld` service will start successfully:

    ```
    # service postgresql start
    Starting postgreSQL service:                                            [  OK  ]
    ```

11. Confirm the context is correct for **`/opt/postgresql`**:

    ```
    ls -lZ /opt
    drwxr-xr-x. root root system_u:object_r:postgresql_db_t:s0 postgresql
    ```

12. Check with the **`ps`** command that the `postgresql` process displays the new location:

    ```
    # ps aux | grep -i postmaster

    postgres 21564  0.3  0.3  42308  4032 ?        S    10:13   0:00 /usr/bin/postmaster -p
     5432 -D /opt/postgresql/data
    ```

13. The location has been changed and labeled, and the `postgresql` daemon has started successfully. At this point all running services should be tested to confirm normal operation.

10.12. rsync

From the *Rsync*[24] project page:

"rsync is an open source utility that provides fast incremental file transfer."

When using Fedora, the *rsync* package provides rsync. Run **`rpm -q rsync`** to see if the *rsync* package is installed. If it is not installed, run the following command as the root user to install it:

```
yum install rsync
```

10.12.1. rsync and SELinux

From the Fedora 12 SELinux *rsync_selinux(8)* man page: "SELinux requires files to have an extended attribute to define the file type. Policy governs the access daemons have to these files. If you want to share files using the rsync daemon, you must label the files and directories *public_content_t*."

Like most services, correct labeling is required for SELinux to perform its protection mechanisms over `rsync`.

10.12.2. Types

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

The following types are used with **`rsync`**. Different types all you to configure flexible access:

`public_content_t`
: This is a generic type used for the location of files (and the actual files) to be shared via `rsync`. If a special directory is created to house files to be shared with `rsync`, the directory and its contents need to have this label applied to them.

`rsync_exec_t`
: This type is used for the **`/usr/bin/rsync`** system binary.

`rsync_log_t`
: This type is used for the `rsync` log file, located at **`/var/log/rsync.log`** by default. To change the location of the file rsync logs to, use the **`--log-file=FILE`** option to the **`rsync`** command at run-time.

`rsync_var_run_t`
: This type is used for the `rsyncd` lock file, located at **`/var/run/rsyncd.lock`**. This lock file is used by the `rsync` server to manage connection limits.

[24] http://www.samba.org/rsync/

10.12.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Boolean allows you to tell SELinux how you are running rsync:

`allow_rsync_anon_write`
: Having this Boolean enabled allows `rsync` in the *rsync_t* domain to manage files, links and directories that have a type of *public_content_rw_t*. Often these are public files used for public file transfer services. Files and directories must be labeled **`public_content_rw_t`**.

`rsync_client`
: Having this Boolean enabled aloows `rsync` to initiate connections to ports defined as *rsync_port_t*, as well as allowing `rsync` to manage files, links and directories that have a type of *rsync_data_t*. Note that the `rsync` daemon must be in the *rsync_t* domain in order for SELinux to enact its control over `rsync`. The configuration example in this chapter demonstrates `rsync` running in the *rsync_t* domain.

`rsync_export_all_ro`
: Having this Boolean enabled allows `rsync` in the *rsync_t* domain to export NFS and CIFS file systems with read-only access to clients.

10.12.4. Configuration Examples

10.12.4.1. Rsync as a daemon

When using Fedora, rsync can be used as a daemon so that multiple clients can directly communicate with it as a central server, in order to house centralized files and keep them synchronized. The following example will demonstrate running rsync as a daemon over a network socket in the correct domain, and how SELinux expects this daemon to be running on a pre-defined (in SELinux policy) TCP port. This example will then show how to modify SELinux policy to allow the `rsync` daemon to run normally on a non-standard port.

This example will be performed on a single system to demonstrate SELinux policy and its control over local daemons and processes. Note that this is an example only and demonstrates how SELinux can affect rsync. Comprehensive documentation of rsync is beyond the scope of this document. Refer to the official *rsync documentation*[25] for further details. This example assumes that the *rsync*, *setroubleshoot-server* and *audit* packages are installed, that the SELinux targeted policy is used and that SELinux is running in enforcing mode.

Getting rsync to launch as rsync_t

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run the **`which`** command to confirm that the rsync binary is in the system path:

   ```
   ```

[25] http://www.samba.org/rsync/documentation.html

```
$ which rsync
/usr/bin/rsync
```

3. When running `rsync` as a daemon, a configuration file should be used and saved as **`/etc/rsyncd.conf`**. Note that the following configuration file used in this example is very simple and is not indicative of all the possible options that are available, rather it is just enough to demonstrate the `rsync` daemon:

```
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
[files]
        path = /srv/files
        comment = file area
        read only = false
 timeout = 300
```

4. Now that a simple configuration file exists for rsync to operate in daemon mode, this step demonstrates that simply running **`rsync --daemon`** is not sufficient for SELinux to offer its protection over rsync. Refer to the following output:

```
# rsync --daemon

# ps x | grep rsync
 8231 ?        Ss     0:00 rsync --daemon
 8233 pts/3    S+     0:00 grep rsync

# ps -eZ | grep rsync
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 8231 ? 00:00:00 rsync
```

   Note that in the output from the final **`ps`** command, the context shows the `rsync` daemon running in the **`unconfined_t`** domain. This indicates that rsync has not transitioned to the **`rsync_t`** domain as it was launched by the **`rsync --daemon`** command. At this point SELinux can not enforce its rules and policy over this daemon. Refer to the following steps to see how to fix this problem. In the following steps, `rsync` will transition to the **`rsync_t`** domain by launching it from a properly-labeled init script. Only then can SELinux and its protection mechanisms have an effect over `rsync`. This `rsync` process should be killed before proceeding to the next step.

5. A custom init script for rsync is needed for this step. There is an example init script available at *http://www.fredshack.com/docs/rsync.html*. Save it to **`/etc/rc.d/init.d/rsyncd`**. The following steps show how to label this script as **`initrc_exec_t`**:

6. Run the **semanage** command to add a context mapping for **`/etc/rc.d/init.d/rsyncd`**:

```
semanage fcontext -a -t initrc_exec_t "/etc/rc.d/init.d/rsyncd"
```

7. This mapping is written to the **`/etc/selinux/targeted/contexts/files/file_contexts.local`** file:

```
# grep rsync /etc/selinux/targeted/contexts/files/file_contexts.local

/etc/rc.d/init.d/rsyncd    system_u:object_r:initrc_exec_t:s0
```

8. Now use the **restorecon** command to apply this context mapping to the running system:

   ```
   restorecon -R -v /etc/rc.d/init.d/rsyncd
   ```

9. Run the **ls** to confirm the script has been labeled appropriately. Note that in the following output the script has been labeled as **initrc_exec_t**:

   ```
    ls -lZ /etc/rc.d/init.d/rsyncd
   -rwxr-xr-x. root root system_u:object_r:initrc_exec_t:s0 /etc/rc.d/init.d/rsyncd
   ```

10. Launch `rsyncd` via the new script. Now that rsync has started from an init script that has been appropriately labeled, the process will start as **rsync_t**:

    ```
    # /etc/rc.d/init.d/rsync start
    Starting rsyncd:                                           [  OK  ]

    ps -eZ | grep rsync
    unconfined_u:system_r:rsync_t:s0 9794 ?        00:00:00 rsync
    ```

    SELinux can now enforce its protection mechanisms over the `rsync` daemon as it is now runing in the **rsync_t** domain.

This example demonstrated how to get `rsyncd` running in the **rsync_t** domain. The next example shows how to get this daemon successfully running on a non-default port. TCP port 10000 is used in the next example.

Running the rsync daemon on a non-default port

1. Modify the **/etc/rsyncd.conf** file and add the **port = 10000** line at the top of the file in the global configuration area (ie., before any file areas are defined). The new configuration file will look like:

   ```
   log file = /var/log/rsyncd.log
   pid file = /var/run/rsyncd.pid
   lock file = /var/run/rsync.lock
   port = 10000
   [files]
           path = /srv/files
           comment = file area
           read only = false
    timeout = 300
   ```

2. After launching rsync from the init script with this new setting, a denial similar to the following is logged by SELinux:

   ```
   Jul 22 10:46:59 localhost setroubleshoot: SELinux is preventing the rsync (rsync_t)
    from binding to port 10000. For complete SELinux messages. run sealert -l
    c371ab34-639e-45ae-9e42-18855b5c2de8
   ```

3. Run the **semanage** command to add TCP port 10000 to SELinux policy in **rsync_port_t**:

   ```
   # semanage port -a -t rsync_port_t -p tcp 10000
   ```

4. Now that TCP port 10000 has been added to SELinux policy for **`rsync_port_t`**, `rsyncd` will start and operate normally on this port:

   ```
   # /etc/rc.d/init.d/rsync start
   Starting rsyncd:                                            [  OK  ]
   ```

   ```
   # netstat -lnp | grep 10000
   tcp        0      0 0.0.0.0:10000   0.0.0.0:*      LISTEN      9910/rsync
   ```

SELinux has had its policy modified and is now permitting `rsyncd` to operate on TCP port 10000.

10.13. Postfix

From the *Postfix*[26] project page:

"What is Postfix? It is Wietse Venema's mailer that started life at IBM research as an alternative to the widely-used Sendmail program. Postfix attempts to be fast, easy to administer, and secure. The outside has a definite Sendmail-ish flavor, but the inside is completely different."

In Fedora, the *postfix* package provides postfix. Run **`rpm -q postfix`** to see if the *postfix* package is installed. If it is not installed, run the following command as the root user to install it:

```
yum install postfix
```

10.13.1. Postfix and SELinux

When Postfix is enabled, it runs confined by default. Confined processes run in their own domains, and are separated from other confined processes. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited. The following example demonstrates the Postfix and related processes running in their own domain. This example assumes the *postfix* package is installed and that the Postfix service has been started:

1. Run **`getenforce`** to confirm SELinux is running in enforcing mode:

   ```
   $ getenforce
   Enforcing
   ```

   The **`getenforce`** command returns **`Enforcing`** when SELinux is running in enforcing mode.

2. Run **`service postfix start`** as the root user to start `postfix`:

   ```
   service postfix start
   Starting postfix:                                [  OK  ]
   ```

3. Run **`ps -eZ | grep postfix`** to view the `postfix` processes:

[26] http://www.postfix.org/

```
ps -eZ | grep postfix
system_u:system_r:postfix_master_t:s0 1651 ?   00:00:00 master
system_u:system_r:postfix_pickup_t:s0 1662 ?   00:00:00 pickup
system_u:system_r:postfix_qmgr_t:s0 1663 ?     00:00:00 qmgr
```

For example, the SELinux context associated with the Postfix `master` process is **`unconfined_u:system_r:postfix_master_t:s0`**. The second last part of the context, **`postfix_master_t`**, is the type for this process. A type defines a domain for processes and a type for files. In this case, the `master` process is running in the **`postfix_master_t`** domain.

10.13.2. Types

Type Enforcement is the main permission control used in SELinux targeted policy. All files and processes are labeled with a type: types define a domain for processes and a type for files. SELinux policy rules define how types access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

The following types are used with **`Postfix`**. Different types all you to configure flexible access:

`postfix_etc_t`

This type is used for configuration files for Postfix in **`/etc/postfix`**.

`postfix_data_t`

This type is used for Postfix data files in **`/var/lib/postfix`**.

Note

To see the full list of files and their types for Postfix, run the following command:

```
$ grep postfix /etc/selinux/targeted/contexts/files/file_contexts
```

10.13.3. Booleans

SELinux is based on the least level of access required for a service to run. Services can be run in a variety of ways; therefore, you must tell SELinux how you are running services. The following Boolean allows you to tell SELinux how you are running Postfix:

`allow_postfix_local_write_mail_spool`

Having this Boolean enables Postfix to write to the local mail spool on the system. Postfix requires this Boolean to be enabled for normal operation when local spools are used.

10.13.4. Configuration Examples

10.13.4.1. SpamAssassin and Postfix

From the *SpamAssassin*[27] project page:

[27] http://spamassassin.apache.org/

"Open Source mail filter, written in Perl, to identify spam using a wide range of heuristic tests on mail headers and body text. Free software."

When using Fedora, the *spamassassin* package provides SpamAssassin. Run **`rpm -q spamassassin`** to see if the *spamassassin* package is installed. If it is not installed, run the following command as the root user to install it:

```
yum install spamassassin
```

SpamAssassin operates in tandom with a mailer such as Postfix to provide spam-filtering capabilities. In order for SpamAssassin to effectively intercept, analyze and filter mail, it must listen on a network interface. The default port for SpamAssassin is TCP/783, however this can be changed. The following example provides a real-world demonstration of how SELinux complements SpamAssassin by only allowing it access to a certain port by default. This example will then demonstrate how to change the port and have SpamAssassin operate on a non-default port.

Note that this is an example only and demonstrates how SELinux can affect a simple configuration of SpamAssassin. Comprehensive documentation of SpamAssassin is beyond the scope of this document. Refer to the official *SpamAssassin documentation*[28] for further details. This example assumes the *spamassassin* is installed, that any firewall has been configured to allow access on the ports in use, that the SELinux targeted policy is used, and that SELinux is running in enforcing mode:

Running SpamAssassin on a non-default port

1. Run the **`semanage`** command to show the port that SELinux allows `spamd` to listen on by default:

   ```
   # semanage port -l | grep spamd
   spamd_port_t  tcp 783
   ```

   This output shows that TCP/783 is defined in **`spamd_port_t`** as the port for SpamAssassin to operate on.

2. Edit the **`/etc/sysconfig/spamassassin`** configuration file and modify it so that it will start SpamAssassin on the example port TCP/10000:

   ```
   # Options to spamd
   SPAMDOPTIONS="-d -p 10000 -c m5 -H"
   ```

   This line now specifies that SpamAssassin will operate on port 10000. The rest of this example will show how to modify SELinux policy to allow this socket to be opened.

3. Start SpamAssassin and an error message similar to the following will appear:

   ```
   /etc/init.d/spamassassin start
   Starting spamd: [2203] warn: server socket setup failed, retry 1: spamd: could not create
    INET socket on 127.0.0.1:10000: Permission denied
   [2203] warn: server socket setup failed, retry 2: spamd: could not create INET socket on
    127.0.0.1:10000: Permission denied
   [2203] error: spamd: could not create INET socket on 127.0.0.1:10000: Permission denied
   spamd: could not create INET socket on 127.0.0.1:10000: Permission denied
   ```

[28] http://spamassassin.apache.org/doc.html

```
                                                            [FAILED]
```

This output means that SELinux has blocked access to this port.

4. A denial similar to the following will be logged by SELinux:

```
SELinux is preventing the spamd (spamd_t) from binding to port 10000.
```

5. As the root user, run the **`semanage`** command to modify SELinux policy in order to allow SpamAssassin to operate on the example port (TCP/10000):

```
semanage port -a -t spamd_port_t -p tcp 10000
```

6. Confirm that SpamAssassin will now start and is operating on TCP port 10000:

```
# /etc/init.d/spamassassin start
Starting spamd:     [ OK ]

# netstat -lnp | grep 10000
tcp 0 0 127.0.0.1:10000 0.0.0.0:* LISTEN 2224/spamd.pid
```

7. At this point, `spamd` is properly operating on TCP port 10000 as it has been allowed access to that port by SELinux policy.

Appendix A. 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).[1]

A.1.1.1. AES Uses

A.1.1.2. 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 in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details). 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 (see Security of AES, below).[2]

The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process. Rijndael (pronounced [r#inda#l]) is a portmanteau of the names of the two inventors.[3]

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.[4]

A.1.2.1. DES Uses

A.1.2.2. 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

[1] "Advanced Encryption Standard." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Advanced_Encryption_Standard*
[2] "Advanced Encryption Standard." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Advanced_Encryption_Standard*
[3] "Advanced Encryption Standard." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Advanced_Encryption_Standard*
[4] "Data Encryption Standard." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Data_Encryption_Standard*

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).[5]

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 (/#di##i###s/), or pronounced as a one-syllable acronym (/#d#z/).[6]

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.[7]

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.[8]

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.[9]

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.[10]

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.[11]

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

[5] "Data Encryption Standard." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Data_Encryption_Standard*
[6] "Data Encryption Standard." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Data_Encryption_Standard*
[7] "Public-key Encryption." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Public-key_cryptography*
[8] "Public-key Encryption." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Public-key_cryptography*
[9] "Public-key Encryption." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Public-key_cryptography*
[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*

communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.[12]

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).[13]

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).[14]

U.S. Patent 4,200,770, now expired, describes the algorithm and credits Hellman, Diffie, and Merkle as inventors.[15]

A.2.2. RSA

In cryptography, RSA (which stands for Rivest, Shamir and Adleman who first publicly described it; see below) 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.[16]

A.2.3. DSA

The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186, adopted in 1993. A minor revision was issued in 1996 as FIPS 186-1. The standard was expanded further in 2000 as FIPS 186-2 and again in 2009 as FIPS 186-3.[17]

A.2.4. SSL/TLS

Transport Layer Security (TLS) and its predecessor, Secure Socket 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). TLS is an IETF standards track protocol, last updated in RFC 5246, that was based on the earlier SSL specifications developed by Netscape Corporation.

The TLS protocol allows client/server applications to communicate across a network in a way designed to prevent eavesdropping and tampering. TLS provides endpoint authentication and communications confidentiality over the Internet using cryptography. TLS provides RSA security with 1024 and 2048 bit strengths.

[12] "Diffie-Hellman." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Diffie-Hellman*
[13] "Diffie-Hellman." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Diffie-Hellman*
[14] "Diffie-Hellman." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Diffie-Hellman*
[15] "Diffie-Hellman." *Wikipedia.* 14 November 2009 *http://en.wikipedia.org/wiki/Diffie-Hellman*
[16] "RSA" *Wikipedia* 14 April 2010 *http://en.wikipedia.org/wiki/RSA*
[17] "Digital Signature Algorithm" *Wikipedia* 14 April 2010 *http://en.wikipedia.org/wiki/Digital_Signature_Algorithm*

In typical end-user/browser usage, TLS authentication is unilateral: only the server is authenticated (the client knows the server's identity), but not vice versa (the client remains unauthenticated or anonymous).

TLS also supports the more secure bilateral connection mode (typically used in enterprise applications), in which both ends of the "conversation" can be assured with whom they are communicating (provided they diligently scrutinize the identity information in the other party's certificate). This is known as mutual authentication, or 2SSL. Mutual authentication requires that the TLS client-side also hold a certificate (which is not usually the case in the end-user/browser scenario). Unless, that is, TLS-PSK, the Secure Remote Password (SRP) protocol, or some other protocol is used that can provide strong mutual authentication in the absence of certificates.

Typically, the key information and certificates necessary for TLS are handled in the form of X.509 certificates, which define required fields and data formats.

SSL operates in modular fashion. It is extensible by design, with support for forward and backward compatibility and negotiation between peers.[18]

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.[19]

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.[1] ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm is a variant of the ElGamal signature scheme, which should not be confused with ElGamal encryption.[20]

[18] "Transport Layer Security" *Wikipedia* 14 April 2010 *http://en.wikipedia.org/wiki/Transport_Layer_Security*
[19] "Cramer–Shoup cryptosystem" *Wikipedia* 14 April 2010 *http://en.wikipedia.org/wiki/Cramer-Shoup_cryptosystem*
[20] "ElGamal encryption" *Wikipedia* 14 April 2010 *http://en.wikipedia.org/wiki/ElGamal_encryption*

Appendix B. Revision History

Revision 19.1-1	**Mon June 03 2013**	**Eric Christensen** *sparks@fedoraproject.org*

Branched for Fedora 19.
Added chapter on firewalls.

Revision 18.2-1	**Wed April 17 2013**	**Eric Christensen** *sparks@fedoraproject.org*

Added 'Manage Confined Services' Guide sections.

Revision 18.1-1	**Wed April 10 2013**	**Eric Christensen** *sparks@fedoraproject.org*

Added SELinux Guide sections.

Revision 18.0-1	**Sat October 6 2012**	**Eric Christensen** *sparks@fedoraproject.org*

Fixed Basic Hardening chapter (BZ 841825 and 693620).
Fixed broken LUKS link (BZ 846299).
Added GUI section to 7 Zip chapter (BZ 854781).
Fixed yum-plugin-security chapter (BZ 723282).
Fixed GPG CLI command screen (BZ 590493).
Improved Yubikey section (BZ 644238).
Fixed typos (BZ 863636).
Removed wiki markup found in some chapters.
Updated the Seahorse instructions.

Revision 17.0-1	**Tue January 24 2012**	**Eric Christensen** *sparks@fedoraproject.org*

Branched for Fedora 17.

Revision 16.0-1	**Fri September 09 2011**	**Eric Christensen** *sparks@fedoraproject.org*

Branched for Fedora 16.

Revision 14.3-1	**Sat Apr 02 2011**	**Eric Christensen** *sparks@fedoraproject.org*

Moved VPN text to the Encryption chapter and reformated.

Revision 14.2-1	**Wed Oct 20 2010**	**Zach Oglesby** *zoglesby@fedoraproject.org*

Added text for using Yubikey on Fedora with local authentication. (BZ 644999)

Revision 14.2-0	**Fri Oct 6 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Removed all variables in the document source.

Revision 14.1-2	**Fri Oct 1 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Corrected the link to the DISA Unix Checklist and updated link.

Revision 14.1-1	**Wed Jul 8 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Added CVE chapter.

Revision 14.0-1	**Fri May 28 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Branched for Fedora 14

Revision 13.0-7	**Fri May 14 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Removed "bug" text from 7-Zip chapter per bug 591980.

Revision 13.0-6	**Wed Apr 14 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Completed the encryption standards appendix.

Revision 13.0-5	**Fri Apr 09 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Added "Using GPG with Alpine".
Added "Using GPG with Evolution".

Revision 13.0-4	**Tue Apr 06 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Repaired issues regarding untranslatable text in para.

Revision 13.0-3	**Tue Apr 06 2010**	**Eric Christensen** *sparks@fedoraproject.org*

Removed the PackageKit vulnerability text seen in Fedora 12.

Revision 13.0-2	**Fri Nov 20 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Added the Revision History to the end of the document.

Added the Encryption Standards appendix.

Revision 13.0-1	**Fri Nov 20 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Fedora 13 branch.

Revision 1.0-23	**Thu Nov 19 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Updated the section "Local users may install trusted packages" to the latest fix, again.

Revision 1.0-22	**Thu Nov 19 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Updated the section "Local users may install trusted packages" to the latest fix.

Revision 1.0-21	**Wed Nov 18 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Added section "Local users may install trusted packages".

Revision 1.0-20	**Sat Nov 14 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Added information from Wikipedia to the Encryption Standards appendix.
Added Adam Ligas to the author page for his role in developing the 7-Zip portions.

Revision 1.0-19	**Mon Oct 26 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Updated license to CC-BY-SA.

Revision 1.0-18	**Wed Aug 05 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Fixed issues related to Bug 515043.

Revision 1.0-17	**Mon Jul 27 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Repaired vendor information in SPEC.

Revision 1.0-16	**Fri Jul 24 2009**	**Fedora Release Engineering** *rel-eng@lists.fedoraproject.org*

Rebuilt for https://fedoraproject.org/wiki/Fedora_12_Mass_Rebuild

Revision 1.0-15	**Tue Jul 14 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Added "desktop-file-utils" to BUILDREQUIRES on the spec

Revision 1.0-14	**Tue Mar 10 2009**	**Scott Radvan** *sradvan@redhat.com*

Remove more rhel specifics, major review and remove draft, ready for push

Revision 1.0-13	**Mon Mar 2 2009**	**Scott Radvan** *sradvan@redhat.com*

Lots of minor fixes

Revision 1.0-12	**Wed Feb 11 2009**	**Scott Radvan** *sradvan@redhat.com*

new screenshots from F11 replacing existing/older ones

Revision 1.0-11	**Tue Feb 03 2009**	**Scott Radvan** *sradvan@redhat.com*

LUKS specifics to Fedora 9 modified to include later releases as well.
Fix 404s in reference section, mainly bad NSA links.
minor formatting changes.

Revision 1.0-10	**Wed Jan 27 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Fixed missing firewall setup screenshot.

Revision 1.0-9	**Wed Jan 27 2009**	**Eric Christensen** *sparks@fedoraproject.org*

Repaired items found to be incorrect during validation. Many Red Hat references have been changed to Fedora references.