Red Hat Enterprise Linux 8.0 Beta

Configuring and managing security

A guide to securing Red Hat Enterprise Linux 8

Last Updated: 2019-04-03

Red Hat Enterprise Linux 8.0 Beta Configuring and managing security

A guide to securing Red Hat Enterprise Linux 8

Legal Notice

Abstract

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

Table of Contents

THIS IS A BETA VERSION!

Thank you for your interest in Red Hat Enterprise Linux 8.0 Beta. Be aware that:

- Beta code should not be used with production data or on production systems.
- Beta does not include a guarantee of support.
- Feedback and bug reports are welcome. Discussions with your account representative, partner contact, and Technical Account Manager (TAM) are also welcome.
- Upgrades to or from a Beta are not supported or recommended.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
    1. Go to the Bugzilla website.
    2. As the Component, use **Documentation**.
    3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
    4. Click **Submit Bug**.

CHAPTER 1. OVERVIEW OF SECURITY HARDENING IN RED HAT ENTERPRISE LINUX

1.1. WHAT IS COMPUTER SECURITY?

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

1.2. 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.

1.3. 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.

1.3.1. 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)

1.3.2. 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

1.3.3. 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

1.4. VULNERABILITY ASSESSMENT

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

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

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

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 1.2, “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.

1.4.1. 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 at an advantage since you are internal and your status is elevated to trusted. This is the viewpoint you and your co-workers have once logged on to your systems. You see print servers, file servers, databases, and other resources.

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

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

Assessing network infrastructure is a dynamic process. Security, both information and physical, is dynamic. Performing an assessment shows an overview, which can turn up false positives and false negatives. A false positive is a result, where the tool finds vulnerabilities which in reality do not exist. A false negative is when it omits actual vulnerabilities.

Security administrators are only as good as the tools they use and the knowledge they retain. Take any of the assessment tools currently available, run them against your system, and it is almost a guarantee that there are some false positives. Whether by program fault or user error, the result is the same. The tool may find false positives, or, even worse, false negatives.

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

WARNING

Do not attempt to exploit vulnerabilities on production systems. Doing so can have adverse effects on productivity and efficiency of your systems and network.

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

- Creates proactive focus on information security.
- Finds potential exploits before 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.

1.4.2. Establishing a methodology for vulnerability assessment

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, see the following website:

- https://www.owasp.org/ — *The Open Web Application Security Project*

1.4.3. Vulnerability assessment 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 tools. Additionally, look to the Internet for more information, such as articles, step-by-step guides, or even mailing lists specific to the tools.

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

1.4.3.1. Scanning hosts with Nmap

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

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

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

1.4.3.1.1. Using Nmap

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

```
nmap <hostname>
```

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

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

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

```
Interesting ports on foo.example.com:
Not shown: 1710 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
```

```
53/tcp  open   domain
80/tcp  open   http
113/tcp closed auth
```

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

For more information about using **Nmap**, see the official homepage at the following URL:

http://www.insecure.org/

1.4.3.2. Nessus

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

NOTE

The **Nessus** client and server software requires a subscription to use. It has been included in this document as a reference to users who may be interested in using this popular application.

For more information about **Nessus**, see the official website at the following URL:

http://www.nessus.org/

1.4.3.3. OpenVAS

OpenVAS (*Open Vulnerability Assessment System*) is a set of tools and services that can be used to scan for vulnerabilities and for a comprehensive vulnerability management. The **OpenVAS** framework offers a number of web-based, desktop, and command line tools for controlling the various components of the solution. The core functionality of **OpenVAS** is provided by a security scanner, which makes use of over 33 thousand daily-updated Network Vulnerability Tests (**NVT**). Unlike **Nessus** (see Section 1.4.3.2, “**Nessus**”), **OpenVAS** does not require any subscription.

For more information about OpenVAS, see the official website at the following URL:

http://www.openvas.org/

1.4.3.4. 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 **CGI** scripts, **Nikto** can be an excellent resource for checking the security of these servers.

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

http://cirt.net/nikto2

1.5. SECURITY THREATS

1.5.1. Threats to network security

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

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 someone exploits the opportunity **eventually**.

Broadcast networks

System administrators often fail to realize the importance of networking hardware in their security schemes. Simple hardware, such as hubs and routers, relies 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.

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 that allows access to the entire network.

1.5.2. 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.

Unused services and open ports

A full installation of Red Hat Enterprise Linux 8.0 Alpha contains more than 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.

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.

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 "assigning untrained people to maintain security and providing neither the training nor the time to make it possible to learn and do the job."footnote:[http://www.sans.org/security-resources/mistakes.php 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.

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 user names 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, Red Hat Enterprise Linux 8.0 Alpha is released with all such services turned off. However, since administrators often find themselves forced to use these services, careful configuration is critical.

1.5.3. 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.

Bad passwords

Bad passwords are one of the easiest ways for an attacker to gain access to a system.

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.

1.6. 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

Exploit	Description	Notes
Null or default passwords	Leaving administrative passwords blank or using a default password set by the product vendor. This is most common in hardware such as routers and firewalls, but some services that run on Linux can contain default administrator passwords as well (though Red Hat Enterprise Linux 8.0 Alpha does not ship with them).	Commonly associated with networking hardware such as routers, firewalls, VPNs, and network attached storage (NAS) appliances. Common in many legacy operating systems, especially those that bundle services (such as UNIX and Windows.) Administrators sometimes create privileged user accounts in a rush and leave the password null, creating a perfect entry point for malicious users who discover the account.
Default shared keys	Secure services sometimes package default security keys for development or evaluation testing purposes. If these keys are left unchanged and are placed in a production environment on the Internet, **all** users with the same default keys have access to that shared-key resource, and any sensitive information that it contains.	Most common in wireless access points and preconfigured secure server appliances.
IP spoofing	A remote machine acts as a node on your local network, finds vulnerabilities with your servers, and installs a backdoor program or Trojan horse to gain control over your network resources.	Spoofing is quite difficult as it involves the attacker predicting TCP/IP sequence numbers to coordinate a connection to target systems, but several tools are available to assist crackers in performing such a vulnerability. Depends on target system running services (such as **`rsh`**, **`telnet`**, FTP and others) that use *source-based* authentication techniques, which are not recommended when compared to PKI or other forms of encrypted authentication used in **`ssh`** or SSL/TLS.

Exploit	Description	Notes
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.

Exploit	Description	Notes
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.

Exploit	Description	Notes
Application vulnerabilities	Attackers find faults in desktop and workstation applications (such as email clients) and execute arbitrary code, implant Trojan horses for future compromise, or crash systems. Further exploitation can occur if the compromised workstation has administrative privileges on the rest of the network.	Workstations and desktops are more prone to exploitation as workers do not have the expertise or experience to prevent or detect a compromise; it is imperative to inform individuals of the risks they are taking when they install unauthorized software or open unsolicited email attachments. Safeguards can be implemented such that email client software does not automatically open or execute attachments. Additionally, the automatic update of workstation software using 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 rebroadcast), making investigation as to the true source of the attack difficult. Advances in ingress filtering (IETF rfc2267) using **`iptables`** and Network Intrusion Detection Systems such as **`snort`** assist administrators in tracking down and preventing distributed DoS attacks.

CHAPTER 2. SECURING RED HAT ENTERPRISE LINUX DURING INSTALLATION

Security begins even before you start the installation of Red Hat Enterprise Linux. Configuring your system securely from the beginning makes it easier to implement additional security settings later.

2.1. BIOS AND UEFI SECURITY

Password protection for the BIOS (or BIOS equivalent) and the boot loader can prevent unauthorized users who have physical access to systems from booting using removable media or obtaining root privileges through single user mode. The security measures you should take to protect against such attacks depends both on the sensitivity of the information on the workstation and the location of the machine.

For example, if a machine is used in a trade show and contains no sensitive information, then it may not be critical to prevent such attacks. However, if an employee's laptop with private, unencrypted SSH keys for the corporate network is left unattended at that same trade show, it could lead to a major security breach with ramifications for the entire company.

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

2.1.1. 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 CD-ROM or a flash drive. This makes it possible for them to enter rescue mode or single user mode, which in turn allows them to start arbitrary processes on the system or copy sensitive data.

2. **Preventing system booting** — Some BIOSes allow password protection of the boot process. When activated, an attacker is forced to enter a password before the BIOS launches the boot loader.

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

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

2.1.1.1. Non-BIOS-based systems security

Other systems and architectures use different programs to perform low-level tasks roughly equivalent to those of the BIOS on x86 systems. For example, the *Unified Extensible Firmware Interface* (*UEFI*) shell.

For instructions on password protecting BIOS-like programs, see the manufacturer's instructions.
:experimental:

2.2. DISK PARTITIONING

Red Hat recommends creating separate partitions for the **/boot**, **/**, **/home/tmp**, and **/var/tmp/** directories. The reasons for each are different, and we will address each partition.

/boot

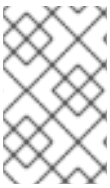
This partition is the first partition that is read by the system during boot up. The boot loader and kernel images that are used to boot your system into Red Hat Enterprise Linux 8.0 Alpha are stored in this partition. This partition should not be encrypted. If this partition is included in / and that partition is encrypted or otherwise becomes unavailable then your system will not be able to boot.

/home

When user data (**/home**) is stored in **/** instead of in a separate partition, the partition can fill up causing the operating system to become unstable. Also, when upgrading your system to the next version of Red Hat Enterprise Linux 8.0 Alpha 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 **/var/tmp/** directories are used to store data that does not need to be stored for a long period of time. However, if a lot of data floods one of these directories it can consume all of your storage space. If this happens and these directories are stored within **/** then your system could become unstable and crash. For this reason, moving these directories into their own partitions is a good idea.



NOTE

During the installation process, an option to encrypt partitions is presented to you. The user must supply a passphrase. This passphrase will be used as a key to unlock the bulk encryption key, which is used to secure the partition's data.

2.3. RESTRICTING NETWORK CONNECTIVITY DURING THE INSTALLATION PROCESS

When installing Red Hat Enterprise Linux 8.0 Alpha, the installation medium represents a snapshot of the system at a particular time. Because of this, it may not be up-to-date with the latest security fixes and may be vulnerable to certain issues that were fixed only after the system provided by the installation medium was released.

When installing a potentially vulnerable operating system, always limit exposure only to the closest necessary network zone. The safest choice is the "no network" zone, which means to leave your machine disconnected during the installation process. In some cases, a LAN or intranet connection is sufficient while the Internet connection is the riskiest. To follow the best security practices, choose the closest zone with your repository while installing Red Hat Enterprise Linux 8.0 Alpha from a network.

2.4. INSTALLING THE MINIMUM AMOUNT OF PACKAGES REQUIRED

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. If you find you need another package, you can always add it to the system later.

2.5. POST-INSTALLATION PROCEDURES

The following steps are the security-related procedures that should be performed immediately after installation of Red Hat Enterprise Linux.

1. Update your system. enter the following command as root:

```
~]# dnf update
```

2. Even though the firewall service, **firewalld**, is automatically enabled with the installation of Red Hat Enterprise Linux, there are scenarios where it might be explicitly disabled, for example in the kickstart configuration. In such a case, it is recommended to consider re-enabling the firewall.
   To start **firewalld** enter the following commands as root:

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

3. To enhance security, disable services you do not need. For example, if there are no printers installed on your computer, disable the **cups** service using the following command:

```
~]# systemctl disable cups
```

   To review active services, enter the following command:

```
~]$ systemctl list-units | grep service
```

[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.

CHAPTER 3. USING THE SYSTEM-WIDE CRYPTOGRAPHIC POLICIES

Crypto-policies is a component in Red Hat Enterprise Linux 8, which configures the core cryptographic subsystems, covering the TLS, IPSec, SSH, DNSSec, and Kerberos protocols. It provides a small set of policies, which the administrator can select.

3.1. SYSTEM-WIDE CRYPTOGRAPHIC POLICIES

Once a system-wide policy is set up, applications in RHEL follow it and refuse to use algorithms and protocols that do not meet the policy, unless the user has explicitly requested the application to do so. That is, the policy applies to the default behavior of applications when running with the system-provided configuration but it can be overridden by the user if required so.

Red Hat Enterprise Linux 8 contains the following policy levels:

DEFAULT	The default system-wide cryptographic policy is a reasonable policy offering secure settings for current threat models. This policy is also compatible with PCI-DSS requirements. It allows the TLS 1.2 and 1.3 protocols, as well as the IKEv2 and SSH2 protocols. The RSA keys and Diffie-Hellman parameters are accepted if they are at least 2048 bits long.
LEGACY	This policy ensures maximum compatibility with legacy systems; it is less secure due to an increased attack surface. In addition to the DEFAULT level algorithms and protocols, it includes support for the TLS 1.0, 1.1, and IKEv1 protocols. The algorithms DSA, 3DES, and RC4 are allowed, while RSA keys and Diffie-Hellman parameters are accepted if they are at least 1023 bits long.
FUTURE	A conservative security level that is believed to withstand any near-term future attacks. This level does not allow the use of SHA-1 in signature algorithms. The RSA keys and Diffie-Hellman parameters are accepted if they are at least 3072 bits long.
FIPS	A policy level that conforms with the FIPS140-2 requirements. This is used internally by the **`fips-mode-setup`** tool, which switches the RHEL system into FIPS140-2 compliance mode.

To view or change the current system-wide cryptographic policy, use the **`update-crypto-policies`** tool, for example:

```
$ update-crypto-policies --show
DEFAULT
```

```
# update-crypto-policies --set FUTURE
Setting system policy to FUTURE
```

To ensure that the change of the cryptographic policy is applied, restart the system.

Additional resources

- For more details, see the **`update-crypto-policies(8)`** man page.

3.2. SWITCHING THE SYSTEM-WIDE CRYPTOGRAPHIC POLICY TO MODE COMPATIBLE WITH PREVIOUS SYSTEMS

The default system-wide cryptographic policy in Red Hat Enterprise Linux 8 does not allow communication using older, currently considered insecure, protocols. For environments that need to be compatible with old releases of Red Hat Enterprise Linux or other legacy systems, a less secure policy level called **LEGACY** is available. Switching to another cryptographic policy level is a matter of a single command.

WARNING

The LEGACY cryptographic level enables compatibility with Red Hat Enterprise Linux 5, and in some cases also with older releases. By switching to this policy level, you make your system and applications less secure.

Procedure

1. To switch the system-wide cryptographic policy to the **LEGACY** level, enter the following command as **root**:

```
# update-crypto-policies --set LEGACY
Setting system policy to LEGACY
```

Additional resources

- For the list of available cryptographic policy levels, see the **update-crypto-policies(8)** man page.

3.3. RELATED INFORMATION

- See the Consistent security by crypto policies in Red Hat Enterprise Linux 8 article on the Red Hat Security Blog for more information.

CHAPTER 4. SCANNING THE SYSTEM FOR SECURITY COMPLIANCE AND VULNERABILITIES

A compliance audit is a process of figuring out whether a given object follows all the rules written out in a compliance policy. The compliance policy is defined by security professionals who specify required settings, often in the form of a checklist, that are to be used in the computing environment.

The compliance policy can vary substantially across organizations and even across different systems within the same organization. Differences among these policies are based on the purpose of these systems and its importance for the organization. The custom software settings and deployment characteristics also raise a need for custom policy checklists.

4.1. SECURITY COMPLIANCE TOOLS IN RHEL

Red Hat Enterprise Linux provides tools that allow for a fully automated compliance audit. These tools are based on the Security Content Automation Protocol (SCAP) standard and are designed for automated tailoring of compliance policies.

- **SCAP Workbench** - The **`scap-workbench`** graphical utility is designed to perform configuration and vulnerability scans on a single local or remote system. It can be also used to generate security reports based on these scans and evaluations.
- **OpenSCAP** - The **`oscap`** command-line utility is designed to perform configuration and vulnerability scans on a local system, to validate security compliance content, and to generate reports and guides based on these scans and evaluations.
- **SCAP Security Guide (SSG)** - The **`scap-security-guide`** package provides the latest collection of security policies for Linux systems. The guidance consists of a catalog of practical hardening advice, linked to government requirements where applicable. The project bridges the gap between generalized policy requirements and specific implementation guidelines.
- **Script Check Engine (SCE)** - SCE is an extension to the SCAP protocol that allows administrators to write their security content using a scripting language, such as Bash, Python, or Ruby. The SCE extension is provided in the **`openscap-engine-sce`** package.

If you require performing automated compliance audits on multiple systems remotely, you can utilize OpenSCAP solution for Red Hat Satellite.

Additional resources

- **`oscap(8)`** - The manual page for the **`oscap`** command-line utility provides a complete list of available options and their usage explanation.
- **`scap-workbench(8)`** - The manual page for the **`SCAP Workbench`** application provides a basic information about the application as well as some links to potential sources of SCAP content.
- **`scap-security-guide(8)`** - The manual page for the **`scap-security-guide`** project provides further documentation about the various available SCAP security profiles. Examples how to utilize the provided benchmarks using the OpenSCAP utility are provided as well.
- For more details about using OpenSCAP with Red Hat Satellite, see Security Compliance Management in the Administering Red Hat Satellite Guide.

4.2. RELATED INFORMATION

- The OpenSCAP project page - The home page to the OpenSCAP project provides detailed information about the oscap utility and other components and projects related to SCAP.
- The SCAP Workbench project page - The home page to the SCAP Workbench project provides detailed information about the scap-workbench application.
- The SCAP Security Guide (SSG) project page - The home page to the SSG project that provides the latest security content for Red Hat Enterprise Linux.
- National Institute of Standards and Technology (NIST) SCAP page - This page represents a vast collection of SCAP related materials, including SCAP publications, specifications, and the SCAP Validation Program.
- National Vulnerability Database (NVD) - This page represents the largest repository of SCAP content and other SCAP standards based vulnerability management data.
- Red Hat OVAL content repository - This is a repository containing OVAL definitions for vulnerabilities of Red Hat Enterprise Linux systems. This is the recommended source of vulnerability content.
- MITRE CVE - This is a database of publicly known security vulnerabilities provided by the MITRE corporation. For RHEL, using OVAL CVE content provided by Red Hat is recommended.
- MITRE OVAL - This page represents an OVAL related project provided by the MITRE corporation. Among other OVAL related information, these pages contain the latest version of the OVAL language and a repository of OVAL content with thousands of OVAL definitions. Note that for scanning RHEL, using OVAL CVE content provided by Red Hat is recommended.
- Red Hat Satellite documentation - This set of guides describes, among other topics, how to maintain system security on multiple systems by using OpenSCAP.

CHAPTER 5. ENCRYPTING BLOCK DEVICES USING LUKS

The Linux Unified Key Setup (LUKS) project enables you to encrypt block devices and it provides a set of tools that simplifies managing the encrypted devices.

5.1. LUKS DISK ENCRYPTION

Linux Unified Key Setup-on-disk-format (LUKS) enables 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.

What LUKS does

- LUKS encrypts entire block devices and is therefore well-suited for protecting the contents of mobile devices such as removable storage media or laptop disk drives.
- The underlying contents of the encrypted block device are arbitrary. This makes it useful for encrypting **swap** devices. This can also be useful with certain databases that use specially formatted block devices for data storage.
- LUKS uses the existing device mapper kernel subsystem.
- LUKS provides passphrase strengthening which protects against dictionary attacks.
- LUKS devices contain multiple key slots, allowing users to add backup keys or passphrases.

What LUKS does not do

- LUKS is not well-suited for applications requiring more than eight keyslots (users) to have distinct access keys to the same device. The LUKS2 format provides up to 32 keyslots.
- LUKS is not well-suited for applications requiring file-level encryption.

5.1.1. LUKS implementation in Red Hat Enterprise Linux

Red Hat Enterprise Linux utilizes LUKS to perform file system encryption. By default, the option to encrypt the file system is unchecked during the installation. If you select the option to encrypt your hard drive, you are prompted for a passphrase that is asked every time you boot the computer. This passphrase “unlocks” the bulk encryption key that is used to decrypt your partition. If you choose to modify the default partition table you can choose which partitions you want to encrypt. This is set in the partition table settings.

In Red Hat Enterprise Linux 8, the default format is LUKS2. The legacy LUKS (LUKS1) remains fully supported and it is provided as a backward compatible format. The LUKS2 format is inspired by LUKS1 and in certain situations can be converted from LUKS1. The conversion is not possible specifically in the following scenarios:

- A LUKS1 device is marked as being used by a Policy-Based Decryption (PBD - Clevis) solution. The **cryptsetup** tool refuses to convert the device when some **luksmeta** metadata are detected.
- A device is active. The device must be in the inactive state before any conversion is possible.

The LUKS2 format is designed to enable future updates of various parts without the need to modify binary structures and internally uses JSON text format for metadata. On-disk format provides redundancy of metadata, detection of metadata corruption and automatic repair from metadata copy.

IMPORTANT

Do not use LUKS2 in production systems that need to be compatible with older systems that support only LUKS1. Note that Red Hat Enterprise Linux 7 supports the LUKS2 format since version 7.6.

The default cipher used for LUKS is **`aes-xts-plain64`**. The default key size for LUKS is 256 bits. The default key size for LUKS with **Anaconda** (XTS mode) is 512 bits. Ciphers that are available are:

- AES - Advanced Encryption Standard - FIPS PUB 197
- Twofish (a 128-bit block cipher)
- Serpent

Additional resources

- LUKS Project Home Page
- LUKS On-Disk Format Specification

5.2. ENCRYPTING DATA ON A NOT YET ENCRYPTED DEVICE

The following procedure contains steps to encrypt data on a not yet encrypted device.

Prerequisites

- The **`cryptsetup-reencrypt`** package is installed.
- Your data are backed up.
- File systems on the device to be encrypted are not mounted.

Procedure

WARNING

You can lose your data during the encryption process; due to a hardware, kernel, or human failure. Ensure that you have a reliable backup before you start encrypting the data.

1. Back up the data from the device that is encrypted.
2. Unmount all file systems on the device, for example:

```
# umount /dev/sdb1
```

3. Make free space for storing a LUKS header. Choose one of the following options that suit your scenario:

   A. In the case of encrypting a logical volume, you can extend the logical volume without resizing the file system, for example:

   ```
   # lvextend -L+8M vg00/lv00
   ```

   B. Extend the partition using partition management tools, such as **parted**.

   C. Shrink the file system on the device. You can use the **resize2fs** utility for the ext2, ext3, or ext4 file systems. Note that the xfs file system cannot be shrunk.

4. Encrypt the file system while storing a new LUKS header in the head of the device. The following command asks you for a passphrase and starts the encryption process, for example:

   ```
   # cryptsetup-reencrypt --new --reduce-device-size 8M /dev/sdb1
   ```

Additional resources

- For more details, see the **cryptsetup-reencrypt(8)**, **cryptsetup(8)**, **lvextend(8)**, **resize2fs(8)**, and **parted(8)** man pages.

5.3. ENCRYPTING DATA ON A NOT YET ENCRYPTED DEVICE WHILE STORING A LUKS HEADER IN A DETACHED FILE

The following procedure describes the steps needed to encrypt a file system without creating free space for storing a LUKS header. The header is stored in a detached location, which can be also used as an additional layer of security.

Prerequisites

- The **cryptsetup-reencrypt** package is installed.

Procedure

WARNING

You can lose your data during the encryption process; due to a hardware, kernel, or human failure. Ensure that you have a reliable backup before you start encrypting the data.

1. Back up the data from the device that is encrypted.

2. Unmount all file systems on the device, for example:

   ```
   # umount /dev/sdb1
   ```

3. Use the **cryptsetup-reencrypt** to encrypt the file system while providing a path to the file with a detached LUKS header in the **--header** parameter. The following command asks you for a passphrase and starts the encryption process:

```
# cryptsetup-reencrypt --new --header /path/to/header /dev/sdb1
```

Note that the detached LUKS header has to be accessible so that encrypted device - /dev/sdb1 in this procedure - can be unlocked later, for example:

```
# cryptsetup open --header /path/to/header /dev/sdb1 my_crypt_device
```

Additional resources

- For more details, see the **cryptsetup-reencrypt(8)** and **cryptsetup(8)** man pages.

CHAPTER 6. CONFIGURING AUTOMATED UNLOCKING OF ENCRYPTED VOLUMES USING POLICY-BASED DECRYPTION

The Policy-Based Decryption (PBD) is a collection of technologies that enable unlocking encrypted root and secondary volumes of hard drives on physical and virtual machines. PBD uses a variety of unlocking methods, such as user passwords, a Trusted Platform Module (TPM) device, a PKCS#11 device connected to a system, for example, a smart card, or a special network server.

PBD allows combining different unlocking methods into a policy, which makes it possible to unlock the same volume in different ways. The current implementation of the PBD in Red Hat Enterprise Linux consists of the Clevis framework and plug-ins called *pins*. Each pin provides a separate unlocking capability. Currently, the following pins are available:

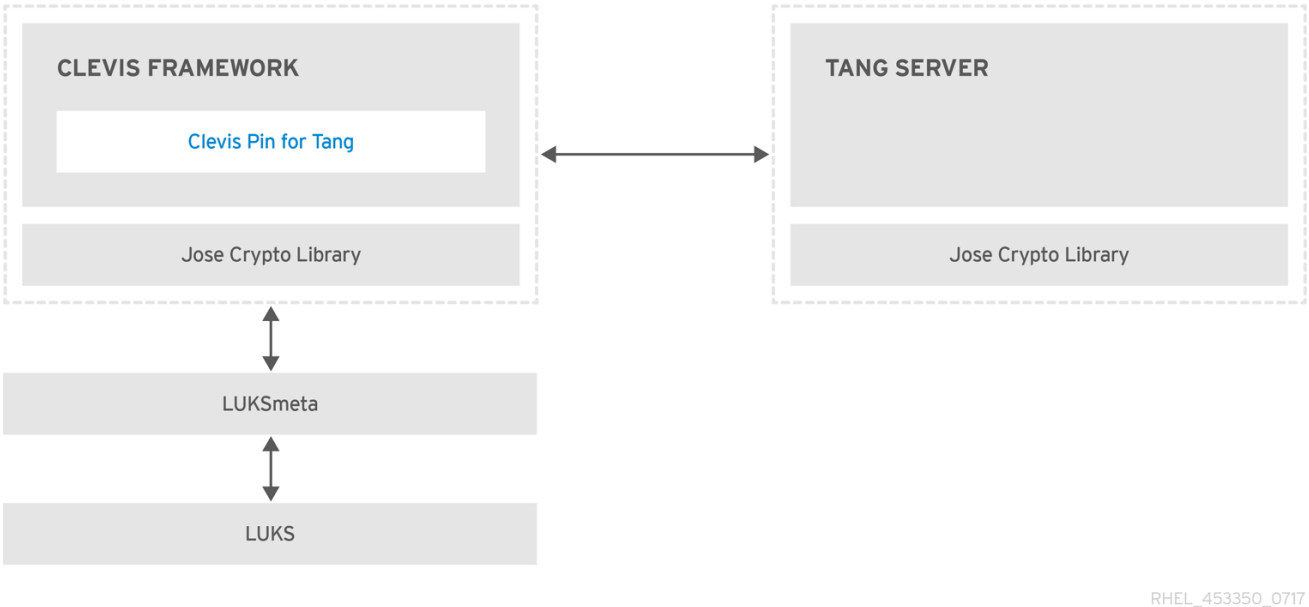
- **`tang`** - allows volumes to be unlocked using a network server
- **`tpm2`** - allows volumes to be unlocked using a TPM2 policy

The Network Bound Disc Encryption (NBDE) is a subcategory of PBD that allows binding encrypted volumes to a special network server. The current implementation of the NBDE includes a Clevis pin for Tang server and the Tang server itself.

6.1. NETWORK-BOUND DISK ENCRYPTION

In Red Hat Enterprise Linux, NBDE is implemented through the following components and technologies:

Figure 6.1. NBDE scheme when using a LUKS1-encrypted volume. The luksmeta package is not used for LUKS2 volumes.



Tang is a server for binding data to network presence. It makes a system containing your data available when the system is bound to a certain secure network. Tang is stateless and does not require TLS or authentication. Unlike escrow-based solutions, where the server stores all encryption keys and has knowledge of every key ever used, Tang never interacts with any client keys, so it never gains any identifying information from the client.

Clevis is a pluggable framework for automated decryption. In NBDE, Clevis provides automated unlocking of LUKS volumes. The **clevis** package provides the client side of the feature.

A *Clevis pin* is a plug-in into the Clevis framework. One of such pins is a plug-in that implements interactions with the NBDE server — Tang.

Clevis and Tang are generic client and server components that provide network-bound encryption. In Red Hat Enterprise Linux, they are used in conjunction with LUKS to encrypt and decrypt root and non-root storage volumes to accomplish Network-Bound Disk Encryption.

Both client- and server-side components use the *José* library to perform encryption and decryption operations.

When you begin provisioning NBDE, the Clevis pin for Tang server gets a list of the Tang server's advertised asymmetric keys. Alternatively, since the keys are asymmetric, a list of Tang's public keys can be distributed out of band so that clients can operate without access to the Tang server. This mode is called *offline provisioning*.

The Clevis pin for Tang uses one of the public keys to generate a unique, cryptographically-strong encryption key. Once the data is encrypted using this key, the key is discarded. The Clevis client should store the state produced by this provisioning operation in a convenient location. This process of encrypting data is the *provisioning step*.

Since the LUKS version 2 (LUKS2) is the default format in Red Hat Enterprise Linux 8, the provisioning state for NBDE is stored as a token in a LUKS2 header. The leveraging of provisioning state for NBDE by the **luksmeta** package is used only for volumes encrypted with LUKS1. The Clevis pin for Tang supports both LUKS1 and LUKS2 without specification need.

When the client is ready to access its data, it loads the metadata produced in the provisioning step and it responds to recover the encryption key. This process is the *recovery step*.

In NBDE, Clevis binds a LUKS volume using a pin so that it can be automatically unlocked. After successful completion of the binding process, the disk can be unlocked using the provided Dracut unlocker.

6.2. INSTALLING AN ENCRYPTION CLIENT - CLEVIS

To install the Clevis pluggable framework and its pins on a machine (client) with an encrypted volume, enter the following command as **root**:

```
~]# yum install clevis
```

To decrypt data, use the **clevis decrypt** command and provide a cipher text in the JSON Web Encryption (JWE) format, for example:

```
~]$ clevis decrypt < secret.jwe
```

Additional resources

- For a quick reference, see the built-in CLI help:

```
~]$ clevis
Usage: clevis COMMAND [OPTIONS]

  clevis decrypt      Decrypts using the policy defined at
encryption time
  clevis encrypt http Encrypts using a REST HTTP escrow server
policy
```

```
  clevis encrypt sss  Encrypts using a Shamir's Secret Sharing
policy
  clevis encrypt tang Encrypts using a Tang binding server policy
  clevis encrypt tpm2 Encrypts using a TPM2.0 chip binding policy

~]$ clevis decrypt
Usage: clevis decrypt < JWE > PLAINTEXT

Decrypts using the policy defined at encryption time

~]$ clevis encrypt tang
Usage: clevis encrypt tang CONFIG < PLAINTEXT > JWE

Encrypts using a Tang binding server policy

This command uses the following configuration properties:

  url: <string>   The base URL of the Tang server (REQUIRED)

  thp: <string>   The thumbprint of a trusted signing key

  adv: <string>   A filename containing a trusted advertisement
  adv: <object>   A trusted advertisement (raw JSON)

Obtaining the thumbprint of a trusted signing key is easy. If you
have access to the Tang server's database directory, simply do:

    $ jose jwk thp -i $DBDIR/$SIG.jwk

Alternatively, if you have certainty that your network connection
is not compromised (not likely), you can download the advertisement
yourself using:

    $ curl -f $URL/adv > adv.jws
```

- For more information , see the **clevis(1)** man page.

6.3. DEPLOYING A TANG SERVER

To install the **tang** package and its dependencies, enter the following command as **root**:

```
~]# yum install tang
```

Enable and start the **tangd** service using systemd:

```
~]# systemctl enable tangd.socket --now
Created symlink from /etc/systemd/system/multi-
user.target.wants/tangd.socket to /usr/lib/systemd/system/tangd.socket.
```

Since **tangd** uses the **systemd** socket activation mechanism, the server starts as soon as the first connection comes in. A new set of cryptographic keys is automatically generated at the first start.

To perform cryptographic operations such as manual key generation, use the **jose** utility. Enter the **jose -h** command or see the **jose(1)** man pages for more information.

Example 6.1. Rotating Tang keys

It is important to periodically rotate your keys. The precise interval at which you should rotate them depends upon your application, key sizes, and institutional policy. For some common recommendations, see the Cryptographic Key Length Recommendation page.

To rotate keys, start with the generation of new keys in the key database directory, typically **/var/db/tang**. For example, you can create new signature and exchange keys with the following commands:

```
~]# DB=/var/db/tang
~]# jose jwk gen -i '{"alg":"ES512"}' -o $DB/new_sig.jwk
~]# jose jwk gen -i '{"alg":"ECMR"}' -o $DB/new_exc.jwk
```

Rename the old keys to have a leading **.** to hide them from advertisement. Note that the file names in the following example differs from real and unique file names in the key database directory.

```
~]# mv $DB/old_sig.jwk $DB/.old_sig.jwk
~]# mv $DB/old_exc.jwk $DB/.old_exc.jwk
```

Tang immediately picks up all changes. No restart is required.

At this point, new client bindings pick up the new keys and old clients can continue to utilize the old keys. When you are sure that all old clients use the new keys, you can remove the old keys.

WARNING

Be aware that removing the old keys while clients are still using them can result in data loss.

Tang uses port 80 for communication. This port is also widely-used for web servers. To change Tang's port number, override the **tangd.socket** unit file using the standard **systemd** mechanisms.

6.3.1. Deploying high-availability systems

Tang provides two methods for building a high-availability deployment:

1. **Client redundancy (recommended)**
   Clients should be configured with the ability to bind to multiple Tang servers. In this setup, each Tang server has its own keys and clients are able to decrypt by contacting a subset of these servers. Clevis already supports this workflow through its **sss** plug-in.

   For more information about this setup, see the following man pages:

   - **tang(8)**, section High Availability
   - **clevis(1)**, section Shamir's Secret Sharing

- **`clevis-encrypt-sss(1)`**
  Red Hat recommends this method for a high-availability deployment.

2. **Key sharing**
   For redundancy purposes, more than one instance of Tang can be deployed. To set up a second or any subsequent instance, install the **tang** packages and copy the key directory to the new host using **`rsync`** over SSH. Note that Red Hat does not recommend this method because sharing keys increases the risk of key compromise and requires additional automation infrastructure.

6.4. DEPLOYING AN ENCRYPTION CLIENT FOR AN NBDE SYSTEM WITH TANG

The following procedure contains steps to configure automated unlocking of an encrypted volume with a Tang network server.

Prerequisites

- The Clevis framework is installed. See Section 6.2, “Installing an encryption client - Clevis”
- A Tang server is available. See Section 6.3, “Deploying a Tang server”

Procedure

1. To bind a Clevis encryption client to a Tang server, use the **`clevis encrypt tang`** sub-command:

```
~]$ clevis encrypt tang '{"url":"http://tang.srv"}' < input-plain.txt > secret.jwe
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
```

   Change the *http://tang.srv* URL in the previous example to match the URL of the server where **tang** is installed. The *secret.jwe* output file contains your encrypted cipher text in the JSON Web Encryption format. This cipher text is read from the *input-plain.txt* input file.

2. To decrypt data, use the **`clevis decrypt`** command and provide the cipher text (JWE):

```
~]$ clevis decrypt < secret.jwe > output-plain.txt
```

Additional resources

- For a quick reference, see the **`clevis-encrypt-tang(1)`** man page or use the built-in CLI help:

```
~]$ clevis
Usage: clevis COMMAND [OPTIONS]

  clevis decrypt      Decrypts using the policy defined at encryption time
  clevis encrypt http Encrypts using a REST HTTP escrow server policy
```

```
  clevis encrypt sss  Encrypts using a Shamir's Secret Sharing
policy
  clevis encrypt tang Encrypts using a Tang binding server policy
  clevis encrypt tang Encrypts using a Tang binding server policy
  clevis luks bind    Binds a LUKSv1 device using the specified
policy
  clevis luks unlock  Unlocks a LUKSv1 volume

~]$ clevis decrypt
Usage: clevis decrypt < JWE > PLAINTEXT

Decrypts using the policy defined at encryption time

~]$ clevis encrypt tang
Usage: clevis encrypt tang CONFIG < PLAINTEXT > JWE

Encrypts using a Tang binding server policy

This command uses the following configuration properties:

  url: <string>   The base URL of the Tang server (REQUIRED)

  thp: <string>   The thumbprint of a trusted signing key

  adv: <string>   A filename containing a trusted advertisement
  adv: <object>   A trusted advertisement (raw JSON)

Obtaining the thumbprint of a trusted signing key is easy. If you
have access to the Tang server's database directory, simply do:

    $ jose jwk thp -i $DBDIR/$SIG.jwk

Alternatively, if you have certainty that your network connection
is not compromised (not likely), you can download the advertisement
yourself using:

    $ curl -f $URL/adv > adv.jws
```

- For more information, see the following man pages:
- **clevis(1)**
- **clevis-luks-unlockers(7)**

6.5. DEPLOYING AN ENCRYPTION CLIENT WITH A TPM 2.0 POLICY

The following procedure contains steps to configure automated unlocking of an encrypted volume with a Trusted Platform Module 2.0 (TPM 2.0) policy.

Prerequisites

- The Clevis framework is installed. See Section 6.2, “Installing an encryption client - Clevis”
- A system with the 64-bit Intel or 64-bit AMD architecture

Procedure

1. To deploy a client that encrypts using a TPM 2.0 chip, use the **clevis encrypt tpm2** sub-command with the only argument in form of the JSON configuration object:

```
~]$ clevis encrypt tpm2 '{}' < input-plain.txt > secret.jwe
```

   To choose a different hierarchy, hash, and key algorithms, specify configuration properties, for example:

```
~]$ clevis encrypt tpm2 '{"hash":"sha1","key":"rsa"}' < input-
plain.txt > secret.jwe
```

2. To decrypt the data, provide the ciphertext in the JSON Web Encryption (JWE) format:

```
~]$ clevis decrypt < secret.jwe > output-plain.txt
```

The pin also supports sealing data to a Platform Configuration Registers (PCR) state. That way, the data can only be unsealed if the PCRs hashes values match the policy used when sealing.

For example, to seal the data to the PCR with index 0 and 1 for the SHA-1 bank:

```
~]$ clevis encrypt tpm2 '{"pcr_bank":"sha1","pcr_ids":"0,1"}' < input-
plain.txt > secret.jwe
```

Additional resources

- For more information and the list of possible configuration properties, see the **clevis-encrypt-tpm2(1)** man page.

6.6. CONFIGURING MANUAL ENROLLMENT OF LUKS-ENCRYPTED ROOT VOLUMES

1. To automatically unlock an existing LUKS-encrypted root volume, install the **clevis-luks** subpackage:

```
~]# yum install clevis-luks
```

2. Bind the volume to a Tang server using the **clevis luks bind** command:

```
~]# clevis luks bind -d /dev/sda tang '{"url":"http://tang.srv"}'
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
You are about to initialize a LUKS device for metadata storage.
Attempting to initialize it may result in data loss if data was
already written into the LUKS header gap in a different format.
A backup is advised before initialization is performed.

Do you wish to initialize /dev/sda? [yn] y
Enter existing LUKS password:
```

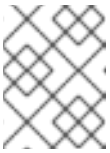
   This command performs four steps:

a. Creates a new key with the same entropy as the LUKS master key.

b. Encrypts the new key with Clevis.

c. Stores the Clevis JWE object in the LUKS2 header token or uses LUKSMeta if the non-default LUKS1 header is used.

d. Enables the new key for use with LUKS.



NOTE

The binding procedure assumes that there is at least one free LUKS password slot. The **`clevis luks bind`** command takes one of the slots.

3. The volume can now be unlocked with your existing password as well as with the Clevis policy.

4. To verify that the Clevis JWE object is successfully placed in a LUKS2 header token, use the **`cryptsetup luksDump`** command:

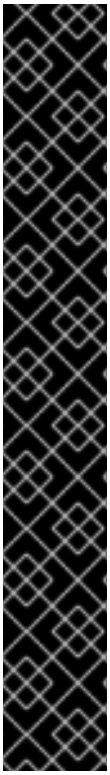
```
~]# cryptsetup luksDump /dev/sda
Tokens:
  0: clevis
    Keyslot:  1
```

In the case of a LUKS1 header, use the **`luksmeta show`** command:

```
~]# luksmeta show -d /dev/sda
0   active empty
1   active cb6e8904-81ff-40da-a84a-07ab9ab5715e
2 inactive empty
3 inactive empty
4 inactive empty
5 inactive empty
6 inactive empty
7 inactive empty
```

5. To enable the early boot system to process the disk binding, enter the following commands on an already installed system:

```
~]# yum install clevis-dracut
~]# dracut -fv --regenerate-all
```



IMPORTANT

To use NBDE for clients with static IP configuration (without DHCP), pass your network configuration to the dracut tool manually, for example:

```
# dracut -fv --regenerate-all --kernel-cmdline
"ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none:192.0.2.45"
```

Alternatively, create a .conf file in the **/etc/dracut.conf.d/** directory with the static network information. For example:

```
~]# cat /etc/dracut.conf.d/static_ip.conf
kernel_cmdline="ip=10.0.0.103::10.0.0.1:255.255.255.0::ens3:non
e:10.0.0.1"
```

Regenerate the initial RAM disk image:

```
~]# dracut -f
```

See the **dracut.cmdline(7)** man page for more information.

Additional resources

For more information, see the following man page:

- **clevis-luks-bind(1)**

6.7. CONFIGURING AUTOMATED ENROLLMENT OF LUKS-ENCRYPTED ROOT VOLUMES USING KICKSTART

Clevis can integrate with Kickstart to provide a fully automated enrollment process.

1. Instruct Kickstart to partition the disk such that the root partition has enabled LUKS encryption with a temporary password. The password is temporary for the enrollment process.

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --grow --encrypted --
passphrase=temppass
```

2. Install the related Clevis packages by listing them in the **%packages** section:

```
%packages
clevis-dracut
%end
```

3. Call **clevis luks bind** to perform binding in the **%post** section. Afterward, remove the temporary password:

```
%post
clevis luks bind -f -k- -d /dev/vda2 \
tang '{"url":"http://tang.srv","thp":"_OsIk0T-E2l6qjfdDiwVmidoZjA"}'
```

```
\ <<< "temppass"
cryptsetup luksRemoveKey /dev/vda2 - <<< "temppass"
%end
```

In the above example, note that we specify the thumbprint that we trust on the Tang server as part of our binding configuration, enabling binding to be completely non-interactive.

You can use an analogous procedure when using a TPM 2.0 policy instead of a Tang server.

6.8. CONFIGURING AUTOMATED UNLOCKING OF A LUKS-ENCRYPTED REMOVABLE STORAGE DEVICE

1. To automatically unlock a LUKS-encrypted removable storage device, such as a USB drive, install the **clevis-udisks2** package:

```
~]# yum install clevis-udisks2
```

2. Reboot the system, and then perform the binding step using the **`clevis luks bind`** command as described in Section 6.6, “Configuring manual enrollment of LUKS-encrypted root volumes”, for example:

```
~]# clevis luks bind -d /dev/sdb1 tang '{"url":"http://tang.srv"}'
```

3. The LUKS-encrypted removable device can be now unlocked automatically in your GNOME desktop session. The device bound to a Clevis policy can be also unlocked by the **`clevis luks unlock`** command:

```
~]# clevis luks unlock -d /dev/sdb1
```

You can use an analogous procedure when using a TPM 2.0 policy instead of a Tang server.

Additional resources

For more information, see the following man page:

- **`clevis-luks-unlockers(7)`**

6.9. CONFIGURING AUTOMATED UNLOCKING OF LUKS-ENCRYPTED NON-ROOT VOLUMES AT BOOT TIME

To use NBDE to also unlock LUKS-encrypted non-root volumes, perform the following steps:

1. Install the clevis-systemd package:

```
~]# yum install clevis-systemd
```

2. Enable the Clevis unlocker service:

```
~]# systemctl enable clevis-luks-askpass.path
Created symlink from /etc/systemd/system/remote-
fs.target.wants/clevis-luks-askpass.path to
/usr/lib/systemd/system/clevis-luks-askpass.path.
```

3. Perform the binding step using the **clevis luks bind** command as described in Section 6.6, “Configuring manual enrollment of LUKS-encrypted root volumes”.

4. To set up the encrypted block device during system boot, add the corresponding line with the **_netdev** option to the **/etc/crypttab** configuration file. See the **crypttab(5)** man page for more information.

5. Add the volume to the list of accessible filesystems in the **/etc/fstab** file. Use the **_netdev** option in this configuration file, too. See the **fstab(5)** man page for more information.

Additional resources

For more information, see the following man page:

- **clevis-luks-unlockers(7)**

6.10. DEPLOYMENT OF VIRTUAL MACHINES IN A NBDE NETWORK

The **clevis luks bind** command does not change the LUKS master key. This implies that if you create a LUKS-encrypted image for use in a virtual machine or cloud environment, all the instances that run this image will share a master key. This is extremely insecure and should be avoided at all times.

This is not a limitation of Clevis but a design principle of LUKS. If you wish to have encrypted root volumes in a cloud, you need to make sure that you perform the installation process (usually using Kickstart) for each instance of Red Hat Enterprise Linux in a cloud as well. The images cannot be shared without also sharing a LUKS master key.

If you intend to deploy automated unlocking in a virtualized environment, Red Hat strongly recommends that you use systems such as lorax or virt-install together with a Kickstart file (see Section 6.7, “Configuring automated enrollment of LUKS-encrypted root volumes using Kickstart”) or another automated provisioning tool to ensure that each encrypted VM has a unique master key.

Note that automated unlocking with a TPM 2.0 policy is not supported in a virtual machine.

Additional resources

For more information, see the following man page:

- **clevis-luks-bind(1)**

6.11. BUILDING AUTOMATICALLY-ENROLLABLE VM IMAGES FOR CLOUD ENVIRONMENTS USING NBDE

Deploying automatically-enrollable encrypted images in a cloud environment can provide a unique set of challenges. Like other virtualization environments detailed above, images should be instantiated at most once to avoid sharing the LUKS master key. Therefore, automated deployment systems systems such as lorax or virt-install together with a Kickstart file should be used to ensure master key uniqueness during the image building process.

Cloud environments enable two Tang server deployment options which we consider here. First, the Tang server can be deployed within the cloud environment itself. Second, the Tang server can be deployed outside of the cloud on independent infrastructure with a VPN link between the two infrastructures.

Deploying Tang natively in the cloud does allow for easy deployment. However, given that it shares infrastructure with the data persistence layer of ciphertext of other systems, it may be possible for both the Tang server’s private key and the Clevis metadata to be stored on the same physical disk. Access to this physical disk permits a full compromise of the ciphertext data.

IMPORTANT

For this reason, Red Hat strongly recommends maintaining a physical separation between the location where the data is stored and the system where Tang is running. This separation between the cloud and the Tang server ensures that the Tang server's private key cannot be accidentally combined with the Clevis metadata. It also provides local control of the Tang server if the cloud infrastructure is at risk.

6.12. RELATED INFORMATION

For more information, see the following man pages:

- **`tang(8)`**
- **`clevis(1)`**
- **`jose(1)`**
- **`clevis-luks-unlockers(1)`**

CHAPTER 7. USING SECURE COMMUNICATIONS BETWEEN TWO SYSTEMS WITH OPENSSH

SSH (Secure Shell) is a protocol which facilitates secure communications between two systems using a client-server architecture and allows users to log in to server host systems remotely. Unlike other remote communication protocols, such as **FTP** or **Telnet**, SSH encrypts the login session, rendering the connection difficult for intruders to collect unencrypted passwords.

The **ssh** program is designed to replace older, less secure terminal applications used to log in to remote hosts, such as **telnet** or **rsh**. A related program called **scp** replaces older programs designed to copy files between hosts, such as **rcp**. Because these older applications do not encrypt passwords transmitted between the client and the server, avoid them whenever possible. Using secure methods to log in to remote systems decreases the risks for both the client system and the remote host.

Red Hat Enterprise Linux includes the general **OpenSSH** package, **openssh**, as well as the **OpenSSH** server, **openssh-server**, and client, **openssh-clients**, packages. Note, the **OpenSSH** packages require the **OpenSSL** package **openssl-libs**, which installs several important cryptographic libraries, enabling **OpenSSH** to provide encrypted communications.

7.1. THE SSH PROTOCOL

7.1.1. Why use SSH?

Potential intruders have a variety of tools at their disposal enabling them to disrupt, intercept, and re-route network traffic in an effort to gain access to a system. In general terms, these threats can be categorized as follows:

Interception of communication between two systems

The attacker can be somewhere on the network between the communicating parties, copying any information passed between them. He may intercept and keep the information, or alter the information and send it on to the intended recipient.
This attack is usually performed using a *packet sniffer*, a rather common network utility that captures each packet flowing through the network, and analyzes its content.

Impersonation of a particular host

Attacker's system is configured to pose as the intended recipient of a transmission. If this strategy works, the user's system remains unaware that it is communicating with the wrong host.
This attack can be performed using a technique known as *DNS poisoning*, or via so-called *IP spoofing*. In the first case, the intruder uses a cracked DNS server to point client systems to a maliciously duplicated host. In the second case, the intruder sends falsified network packets that appear to be from a trusted host.

Both techniques intercept potentially sensitive information and, if the interception is made for hostile reasons, the results can be disastrous. If **SSH** is used for remote shell login and file copying, these security threats can be greatly diminished. This is because the **SSH** client and server use digital signatures to verify their identity. Additionally, all communication between the client and server systems is encrypted. Attempts to spoof the identity of either side of a communication does not work, since each packet is encrypted using a key known only by the local and remote systems.

7.1.2. Main features

The **SSH** protocol provides the following safeguards:

No one can pose as the intended server

After an initial connection, the client can verify that it is connecting to the same server it had connected to previously.

No one can capture the authentication information

The client transmits its authentication information to the server using strong, 128-bit encryption.

No one can intercept the communication

All data sent and received during a session is transferred using 128-bit encryption, making intercepted transmissions extremely difficult to decrypt and read.

Additionally, it also offers the following options:

It provides secure means to use graphical applications over a network

Using a technique called *X11 forwarding*, the client can forward *X11* (*X Window System*) applications from the server.

It provides a way to secure otherwise insecure protocols

The **SSH** protocol encrypts everything it sends and receives. Using a technique called *port forwarding*, an **SSH** server can become a conduit to securing otherwise insecure protocols, like **POP**, and increasing overall system and data security.

It can be used to create a secure channel

The **OpenSSH** server and client can be configured to create a tunnel similar to a virtual private network for traffic between server and client machines.

It supports the `Kerberos` authentication

OpenSSH servers and clients can be configured to authenticate using the **GSSAPI** (Generic Security Services Application Program Interface) implementation of the **`Kerberos`** network authentication protocol.

7.1.3. Protocol versions

Two varieties of **SSH** currently exist: version 1, and newer version 2. The **OpenSSH** suite under Red Hat Enterprise Linux 8.0 Alpha; uses **SSH** version 2, which has an enhanced key exchange algorithm not vulnerable to the known exploit in version 1. In Red Hat Enterprise Linux 8.0 Alpha, the **OpenSSH** suite does not support version 1 connections.

7.1.4. Event sequence of an SSH connection

The following series of events help protect the integrity of **SSH** communication between two hosts.

1. A cryptographic handshake is made so that the client can verify that it is communicating with the correct server.
2. The transport layer of the connection between the client and remote host is encrypted using a symmetric cipher.
3. The client authenticates itself to the server.
4. The client interacts with the remote host over the encrypted connection.

7.1.4.1. Transport layer

The primary role of the transport layer is to facilitate safe and secure communication between the two hosts at the time of authentication and during subsequent communication. The transport layer accomplishes this by handling the encryption and decryption of data, and by providing integrity protection of data packets as they are sent and received. The transport layer also provides compression, speeding the transfer of information.

Once an **SSH** client contacts a server, key information is exchanged so that the two systems can correctly construct the transport layer. The following steps occur during this exchange:

- Keys are exchanged
- The public key encryption algorithm is determined
- The symmetric encryption algorithm is determined
- The message authentication algorithm is determined
- The hash algorithm is determined

During the key exchange, the server identifies itself to the client with a unique *host key*. If the client has never communicated with this particular server before, the server's host key is unknown to the client and it does not connect. **OpenSSH** gets around this problem by accepting the server's host key. This is done after the user is notified and has both accepted and verified the new host key. In subsequent connections, the server's host key is checked against the saved version on the client, providing confidence that the client is indeed communicating with the intended server. If, in the future, the host key no longer matches, the user must remove the client's saved version before a connection can occur.

WARNING

It is possible for an attacker to masquerade as an **SSH** server during the initial contact since the local system does not know the difference between the intended server and a false one set up by an attacker. To help prevent this, verify the integrity of a new **SSH** server by contacting the server administrator before connecting for the first time or in the event of a host key mismatch.

SSH is designed to work with almost any kind of public key algorithm or encoding format. After an initial key exchange creates a hash value used for exchanges and a shared secret value, the two systems immediately begin calculating new keys and algorithms to protect authentication and future data sent over the connection.

After a certain amount of data has been transmitted using a given key and algorithm (the exact amount depends on the **SSH** implementation), another key exchange occurs, generating another set of hash values and a new shared secret value. Even if an attacker is able to determine the hash and shared secret value, this information is only useful for a limited period of time.

7.1.4.2. Authentication

Once the transport layer has constructed a secure tunnel to pass information between the two systems, the server tells the client the different authentication methods supported, such as using a private key-encoded signature or typing a password. The client then tries to authenticate itself to the server using one of these supported methods.

SSH servers and clients can be configured to allow different types of authentication, which gives each side the optimal amount of control. The server can decide which encryption methods it supports based on its security model, and the client can choose the order of authentication methods to attempt from the available options.

7.1.4.3. Channels

After a successful authentication over the **SSH** transport layer, multiple channels are opened using a technique called *multiplexing*[2]. Each of these channels handles communication for different terminal sessions and for forwarded X11 sessions.

Both clients and servers can create a new channel. Each channel is then assigned a different number on each end of the connection. When the client attempts to open a new channel, the clients sends the channel number along with the request. This information is stored by the server and is used to direct communication to that channel. This is done so that different types of sessions do not affect one another and so that when a given session ends, its channel can be closed without disrupting the primary **SSH** connection.

Channels also support *flow-control*, which allows them to send and receive data in an orderly fashion. In this way, data is not sent over the channel until the client receives a message that the channel is open.

The client and server negotiate the characteristics of each channel automatically, depending on the type of service the client requests and the way the user is connected to the network. This allows great flexibility in handling different types of remote connections without having to change the basic infrastructure of the protocol.

7.2. CONFIGURING OPENSSH

7.2.1. Setting OpenSSH up using configuration files

The **OpenSSH** suite uses two different sets of configuration files: those for client programs (that is, **ssh**, **scp**, and **sftp**), and those for the server (the **sshd** daemon).

System-wide **SSH** configuration information is stored in the **/etc/ssh/** directory. User-specific **SSH** configuration information is stored in **~/.ssh/** within the user's home directory.

Table 7.1. System-wide configuration files

File	Description
/etc/ssh/moduli	Contains Diffie-Hellman groups used for the Diffie-Hellman key exchange which is critical for constructing a secure transport layer. When keys are exchanged at the beginning of an SSH session, a shared, secret value is created which cannot be determined by either party alone. This value is then used to provide host authentication.
/etc/ssh/ssh_config	The default SSH client configuration file. Note that it is overridden by **~/.ssh/config** if it exists.
/etc/ssh/sshd_config	The configuration file for the **sshd** daemon.

File	Description
/etc/ssh/ssh_host_ecdsa_key	The ECDSA private key used by the **sshd** daemon.
/etc/ssh/ssh_host_ecdsa_key.pub	The ECDSA public key used by the **sshd** daemon.
/etc/ssh/ssh_host_rsa_key	The RSA private key used by the **sshd** daemon for version 2 of the SSH protocol.
/etc/ssh/ssh_host_rsa_key.pub	The RSA public key used by the **sshd** daemon for version 2 of the SSH protocol.
/etc/pam.d/sshd	The PAM configuration file for the **sshd** daemon.
/etc/sysconfig/sshd	Configuration file for the **sshd** service.

Table 7.2. User-specific configuration files

File	Description
~/.ssh/authorized_keys	Holds a list of authorized public keys for servers. When the client connects to a server, the server authenticates the client by checking its signed public key stored within this file.
~/.ssh/id_ecdsa	Contains the ECDSA private key of the user.
~/.ssh/id_ecdsa.pub	The ECDSA public key of the user.
~/.ssh/id_rsa	The RSA private key used by **ssh** for version 2 of the SSH protocol.
~/.ssh/id_rsa.pub	The RSA public key used by **ssh** for version 2 of the SSH protocol.
~/.ssh/known_hosts	Contains host keys of SSH servers accessed by the user. This file is very important for ensuring that the SSH client is connecting to the correct SSH server.

WARNING

If setting up an **SSH** server, do not turn off the **Privilege Separation** feature by using the **UsePrivilegeSeparation no** directive in the **/etc/ssh/sshd_config** file. Turning off **Privilege Separation** disables many security features and exposes the server to potential security vulnerabilities and targeted attacks. For more information about **UsePrivilegeSeparation**, see the **sshd_config**(5) manual page or the What is the significance of UsePrivilegeSeparation directive in /etc/ssh/sshd_config file and how to test it Red Hat Knowledgebase article.

For information about various directives that can be used in the SSH configuration files, see the **ssh_config**(5) and **sshd_config**(5) manual pages.

7.2.2. Starting an OpenSSH server

To run an **OpenSSH** server, install the **openssh-server** package. To start the **sshd** daemon in the current session:

```
~]# systemctl start sshd.service
```

To stop the running **sshd** daemon in the current session:

```
~]# systemctl stop sshd.service
```

To start the daemon automatically at boot time:

```
~]# systemctl enable sshd.service
Created symlink from /etc/systemd/system/multi-
user.target.wants/sshd.service to /usr/lib/systemd/system/sshd.service.
```

The **sshd** daemon depends on the **network.target** target unit, which is sufficient for static configured network interfaces and for default **ListenAddress 0.0.0.0** options. To specify different addresses in the **ListenAddress** directive and to use a slower dynamic network configuration, add dependency on the **network-online.target** target unit to the **sshd.service** unit file. To achieve this, create the **/etc/systemd/system/sshd.service.d/local.conf** file with the following options:

```
[Unit]
Wants=network-online.target
After=network-online.target
```

After this, reload the **systemd** manager configuration using the following command:

```
~]# systemctl daemon-reload
```

Note that if you reinstall the system, a new set of identification keys will be created. As a result, clients who had connected to the system with any of the **OpenSSH** tools before the reinstall will see the following message:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that the RSA host key has just been changed.
```

To prevent this, you can backup the relevant files from the **/etc/ssh/** directory. See Table 7.1, “System-wide configuration files” for a complete list, and restore the files whenever you reinstall the system.

7.2.3. Requiring SSH for remote connections

To make **SSH** truly effective, using insecure connection protocols should be prohibited. Otherwise, a user’s password may be protected using **SSH** for one session, only to be captured later while logging in using **Telnet**. Some services to disable include **telnet**, **rsh**, **rlogin**, and **vsftpd**.

7.3. USING KEY-BASED AUTHENTICATION

To improve the system security even further, generate **SSH** key pairs and then enforce key-based authentication by disabling password authentication. To do so, open the **/etc/ssh/sshd_config** configuration file in a text editor such as **vi** or **nano**, and change the **PasswordAuthentication** option as follows:

```
PasswordAuthentication no
```

On a system other than a new default installation, check that **PubkeyAuthentication no** has **not** been set. If connected remotely, not using console or out-of-band access, testing the key-based log in process before disabling password authentication is advised.

To be able to use **ssh**, **scp**, or **sftp** to connect to the server from a client machine, generate an authorization key pair by following the steps below. Note that keys must be generated for each user separately.

To use key-based authentication with NFS-mounted home directories, enable the **use_nfs_home_dirs** SELinux boolean first:

```
~]# setsebool -P use_nfs_home_dirs 1
```

IMPORTANT

If you complete the steps as **root**, only **root** is able to use the keys.

NOTE

If you reinstall your system and want to keep previously generated key pairs, backup the **~/.ssh/** directory. After reinstalling, copy it back to your home directory. This process can be done for all users on your system, including **root**.

7.3.1. Generating Key Pairs

To generate an RSA key pair for version 2 of the SSH protocol, follow these steps:

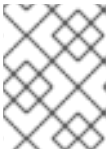
1. Generate an RSA key pair by typing the following at a shell prompt:

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_rsa):
```

2. Press **Enter** to confirm the default location, **~/.ssh/id_rsa**, for the newly created key.

3. Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.
   After this, you will be presented with a message similar to this:

```
Your identification has been saved in /home/USER/.ssh/id_rsa.
Your public key has been saved in /home/USER/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:UNIgIT4wfhdQH/K7yqmjsbZnnyGDKiDviv492U5z78Y
USER@penguin.example.com
The key's randomart image is:
+---[RSA 2048]----+
|o ..==o+.        |
|.+ . .=oo        |
| .o. ..o         |
|  ...  ..        |
|       .S        |
|o .     .        |
|o+ o .o+ ..      |
|+.++=o*.o .E     |
|BBBo+Bo.  oo     |
+----[SHA256]-----+
```



NOTE

To get an MD5 key fingerprint, which was the default fingerprint in previous versions, use the **ssh-keygen** command with the **-E md5** option.

4. By default, the permissions of the **~/.ssh/** directory are set to **rwx------** or **700** expressed in octal notation. This is to ensure that only the *USER* can view the contents. If required, this can be confirmed with the following command:

```
~]$ ls -ld ~/.ssh
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. To copy the public key to a remote machine, issue a command in the following format:

```
ssh-copy-id user@hostname
```

   This copies the most recently modified **~/.ssh/id*.pub** public key if it is not yet installed. Alternatively, specify the public key's file name as follows:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

This copies the content of **~/.ssh/id_rsa.pub** into the **~/.ssh/authorized_keys** file on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

To generate an ECDSA key pair for version 2 of the SSH protocol, follow these steps:

1. Generate an ECDSA key pair by typing the following at a shell prompt:

```
~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ecdsa):
```

2. Press **Enter** to confirm the default location, **~/.ssh/id_ecdsa**, for the newly created key.

3. Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.
After this, you will be presented with a message similar to this:

```
Your identification has been saved in /home/USER/.ssh/id_ecdsa.
Your public key has been saved in /home/USER/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:8BhZageKrLXM99z5f/AM9aPo/KAUd8ZZFPcPFWqK6+M
USER@penguin.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|      . .      +=|
| . . . =      o.o|
|  + . * .    o...|
| = . . *  . + +..|
|. + . . So o * ..|
|   . o . .+ =  ..|
|      o oo ..=. .|
|        ooo...+  |
|        .E++oo   |
+----[SHA256]-----+
```

4. By default, the permissions of the **~/.ssh/** directory are set to **rwx------** or **700** expressed in octal notation. This is to ensure that only the *USER* can view the contents. If required, this can be confirmed with the following command:

```
~]$ ls -ld ~/.ssh
              ~]$ ls -ld ~/.ssh/
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

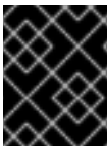
5. To copy the public key to a remote machine, issue a command in the following format:

```
ssh-copy-id USER@hostname
```

This copies the most recently modified **~/.ssh/id*.pub** public key if it is not yet installed. Alternatively, specify the public key's file name as follows:

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub USER@hostname
```

This copies the content of **~/.ssh/id_ecdsa.pub** into the **~/.ssh/authorized_keys** on the machine to which you want to connect. If the file already exists, the keys are appended to its end.



IMPORTANT

The private key is for your personal use only, and it is important that you never give it to anyone.

---

[2] A multiplexed connection consists of several signals being sent over a shared, common medium. With SSH, different channels are sent over a common secure connection.

CHAPTER 8. AUDITING THE SYSTEM

Audit does not provide additional security to your system; rather, it can be used to discover violations of security policies used on your system. These violations can further be prevented by additional security measures such as SELinux.

8.1. LINUX AUDIT

The Linux Audit system provides a way to track security-relevant information on your system. Based on pre-configured rules, Audit generates log entries to record as much information about the events that are happening on your system as possible. This information is crucial for mission-critical environments to determine the violator of the security policy and the actions they performed.

The following list summarizes some of the information that Audit is capable of recording in its log files:

- Date and time, type, and outcome of an event.
- Sensitivity labels of subjects and objects.
- Association of an event with the identity of the user who triggered the event.
- All modifications to Audit configuration and attempts to access Audit log files.
- All uses of authentication mechanisms, such as SSH, Kerberos, and others.
- Changes to any trusted database, such as **`/etc/passwd`**.
- Attempts to import or export information into or from the system.
- Include or exclude events based on user identity, subject and object labels, and other attributes.

The use of the Audit system is also a requirement for a number of security-related certifications. Audit is designed to meet or exceed the requirements of the following certifications or compliance guides:

- Controlled Access Protection Profile (CAPP)
- Labeled Security Protection Profile (LSPP)
- Rule Set Base Access Control (RSBAC)
- National Industrial Security Program Operating Manual (NISPOM)
- Federal Information Security Management Act (FISMA)
- Payment Card Industry — Data Security Standard (PCI-DSS)
- Security Technical Implementation Guides (STIG)

Audit has also been:

- Evaluated by National Information Assurance Partnership (NIAP) and Best Security Industries (BSI).
- Certified to LSPP/CAPP/RSBAC/EAL4+ on Red Hat Enterprise Linux 5.
- Certified to Operating System Protection Profile / Evaluation Assurance Level 4+ (OSPP/EAL4+) on Red Hat Enterprise Linux 6.

Use Cases

Watching file access

Audit can track whether a file or a directory has been accessed, modified, executed, or the file's attributes have been changed. This is useful, for example, to detect access to important files and have an Audit trail available in case one of these files is corrupted.

Monitoring system calls

Audit can be configured to generate a log entry every time a particular system call is used. This can be used, for example, to track changes to the system time by monitoring the `settimeofday`, `clock_adjtime`, and other time-related system calls.

Recording commands run by a user

Audit can track whether a file has been executed, so rules can be defined to record every execution of a particular command. For example, a rule can be defined for every executable in the `/bin` directory. The resulting log entries can then be searched by user ID to generate an audit trail of executed commands per user.

Recording execution of system pathnames

Aside from watching file access which translates a path to an inode at rule invocation, Audit can now watch the execution of a path even if it does not exist at rule invocation, or if the file is replaced after rule invocation. This allows rules to continue to work after upgrading a program executable or before it is even installed.

Recording security events

The `pam_faillock` authentication module is capable of recording failed login attempts. Audit can be set up to record failed login attempts as well, and provides additional information about the user who attempted to log in.

Searching for events

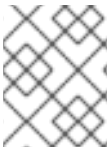
Audit provides the **ausearch** utility, which can be used to filter the log entries and provide a complete audit trail based on a number of conditions.

Running summary reports

The **aureport** utility can be used to generate, among other things, daily reports of recorded events. A system administrator can then analyze these reports and investigate suspicious activity further.

Monitoring network access

The **iptables** and **ebtables** utilities can be configured to trigger Audit events, allowing system administrators to monitor network access.



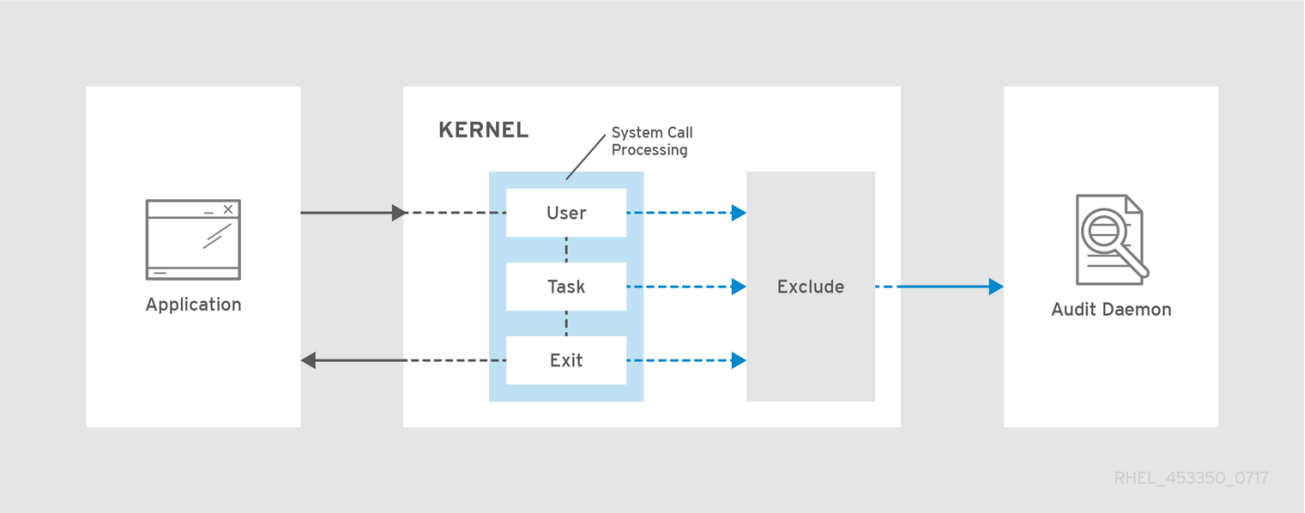
NOTE

System performance may be affected depending on the amount of information that is collected by Audit.

8.1.1. Audit system architecture

The Audit system consists of two main parts: the user-space applications and utilities, and the kernel-side system call processing. The kernel component receives system calls from user-space applications and filters them through one of the three filters: **user**, **task**, or **exit**. Once a system call passes the **exclude** filter, it is sent through one of the aforementioned filters, which, based on the Audit rule configuration, sends it to the Audit daemon for further processing. Figure 8.1, "Audit System Architecture" illustrates this process.

Figure 8.1. Audit System Architecture



The user-space Audit daemon collects the information from the kernel and creates entries in a log file. Other Audit user-space utilities interact with the Audit daemon, the kernel Audit component, or the Audit log files:

- **auditctl** — the Audit control utility interacts with the kernel Audit component to manage rules and to control a number of settings and parameters of the event generation process.
- The remaining Audit utilities take the contents of the Audit log files as input and generate output based on user's requirements. For example, the **aureport** utility generates a report of all recorded events.

The Audit dispatcher daemon (**audisp**) functionality is now integrated in the Audit daemon (**auditd**). Configuration files of plugins for interaction of real-time analytical programs with Audit events are located in the `/etc/audit/plugins.d/` directory by default.

8.2. RELATED INFORMATION

For more information about the Audit system, see the following sources.

Online Sources

- The Linux Audit Documentation Project page: https://github.com/linux-audit/audit-documentation/wiki.

Installed Documentation

Documentation provided by the **audit** package can be found in the `/usr/share/doc/audit/` directory.

Manual Pages

- **audispd.conf**(5)
- **auditd.conf**(5)
- **ausearch-expression**(5)
- **audit.rules**(7)

- **audispd**(8)
- **auditctl**(8)
- **auditd**(8)
- **aulast**(8)
- **aulastlog**(8)
- **aureport**(8)
- **ausearch**(8)
- **ausyscall**(8)
- **autrace**(8)
- **auvirt**(8)

CHAPTER 9. GETTING STARTED WITH SELINUX

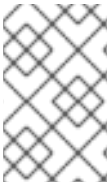
9.1. INTRODUCTION TO SELINUX

Security Enhanced Linux (SELinux) provides an additional layer of system security. SELinux fundamentally answers the question: *May <subject> do <action> to <object>?*, for example: *May a web server access files in users' home directories?*

The standard access policy based on the user, group, and other permissions, known as Discretionary Access Control (DAC), does not enable system administrators to create comprehensive and fine-grained security policies, such as restricting specific applications to only viewing log files, while allowing other applications to append new data to the log files.

SELinux implements Mandatory Access Control (MAC). Every process and system resource has a special security label called a *SELinux context*. A SELinux context, sometimes referred to as a *SELinux label*, is an identifier which abstracts away the system-level details and focuses on the security properties of the entity. Not only does this provide a consistent way of referencing objects in the SELinux policy, but it also removes any ambiguity that can be found in other identification methods; for example, a file can have multiple valid path names on a system that makes use of bind mounts.

The SELinux policy uses these contexts in a series of rules which define how processes can interact with each other and the various system resources. By default, the policy does not allow any interaction unless a rule explicitly grants access.



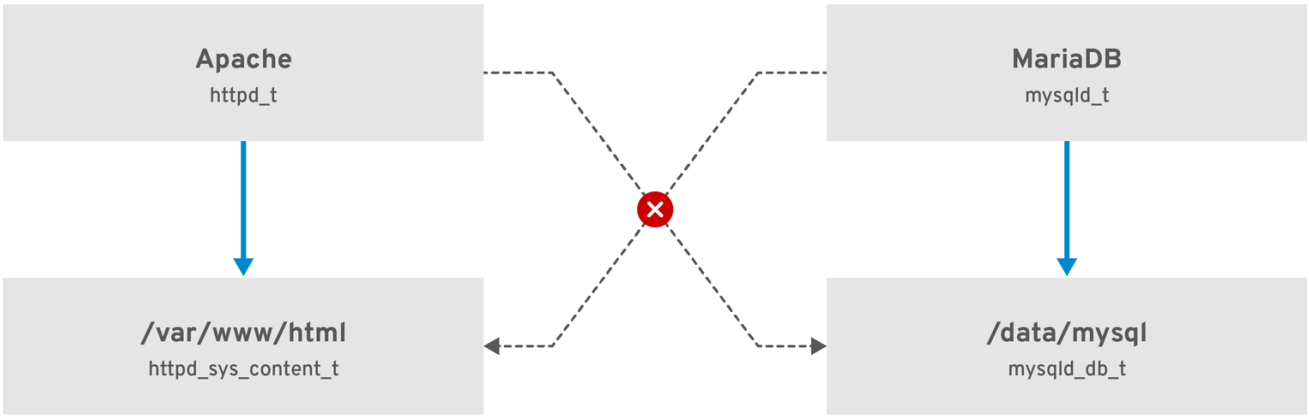
NOTE

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, which means that no SELinux denial is logged if the traditional DAC rules prevent the access.

SELinux contexts have several fields: user, role, type, and security level. The SELinux type information is perhaps the most important when it comes to the SELinux policy, as the most common policy rule which defines the allowed interactions between processes and system resources uses SELinux types and not the full SELinux context. SELinux types usually end with **_t**. For example, the type name for the web server is **httpd_t**. The type context for files and directories normally found in **/var/www/html/** is **httpd_sys_content_t**. The type contexts for files and directories normally found in **/tmp** and **/var/tmp/** is **tmp_t**. The type context for web server ports is **http_port_t**.

For example, there is a policy rule that permits Apache (the web server process running as **httpd_t**) to access files and directories with a context normally found in **/var/www/html/** and other web server directories (**httpd_sys_content_t**). There is no allow rule in the policy for files normally found in **/tmp** and **/var/tmp/**, so access is not permitted. With SELinux, even if Apache is compromised, and a malicious script gains access, it is still not able to access the **/tmp** directory.

Figure 9.1. SELinux allows the Apache process running as httpd_t to access the /var/www/html/ directory and it denies the same process to access the /data/mysql/ directory because there is no allow rule for the httpd_t and mysqld_db_t type contexts). On the other hand, the MariaDB process running as mysqld_t is able to access the /data/mysql/ directory and SELinux also correctly denies the process with the mysqld_t type to access the /var/www/html/ directory labeled as httpd_sys_content_t.



Additional resources

To better understand SELinux basic concepts, see the following documentation:

- The SELinux Coloring Book
- SELinux for Mere Mortals
- SELinux Wiki FAQ
- The SELinux Notebook

9.2. BENEFITS OF RUNNING SELINUX

SELinux provides the following benefits:

- All processes and files are labeled. 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 security level.
- SELinux policy is administratively-defined and enforced system-wide.
- Improved mitigation for privilege escalation attacks. Processes run in domains, and are therefore separated from each other. 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 cannot use that process to read files in user home directories, unless a specific SELinux policy rule was added or configured to allow such access.

- SELinux can be used to enforce data confidentiality and integrity, as well as protecting processes from untrusted inputs.

However, SELinux is not:

- antivirus software,
- replacement for passwords, firewalls, and other security systems,
- 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, and firewalls.

9.3. SELINUX EXAMPLES

The following examples demonstrate how SELinux increases security:

- The default action is deny. If an 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 - potentially malicious - files and applications in their home directory.
- Increased process and data separation. 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 cannot 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 MariaDB databases.
- SELinux helps mitigate the damage made by configuration mistakes. Domain Name System (DNS) 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 Domain (BIND) as a DNS server in Red Hat Enterprise Linux, even if an administrator forgets to limit which servers can perform a zone transfer, the default SELinux policy prevents zone files [3] from being updated using zone transfers, by the BIND **`named`** daemon itself, and by other processes.

9.4. SELINUX ARCHITECTURE AND PACKAGES

SELinux is a Linux Security Module (LSM) that is built into the Linux kernel. The SELinux subsystem in the kernel is driven by a security policy which is controlled by the administrator and loaded at boot. All security-relevant, kernel-level access operations on the system are intercepted by SELinux and examined in the context of the loaded security policy. If the loaded policy 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). When using these cached decisions, SELinux policy rules need to be checked less, which increases performance. Remember that SELinux policy rules have no effect if DAC

rules deny access first. Raw audit messages are logged to the **/var/log/audit/audit.log** and they start with the **type=AVC** string.

In Red Hat Enterprise Linux 8, system services are controlled by the **systemd** daemon; **systemd** starts and stops all services, and users and processes communicate with **systemd** using the **systemctl** utility. The **systemd** daemon has the ability to consult the SELinux policy and check the label of the calling process and the label of the unit file that the caller tries to manage, and then ask SELinux whether or not the caller is allowed the access. This approach strengthens access control to critical system capabilities, which include starting and stopping system services.

The **systemd** daemon also works as an SELinux Access Manager. It can retrieve the label of the process running **systemctl** or the process that sent a **D-Bus** message to **systemd**. The daemon then looks up the label of the unit file that the process wanted to configure. Finally, **systemd** can retrieve information from the kernel if the SELinux policy allows the specific access between the process label and the unit file label. This means a compromised application that needs to interact with **systemd** for a specific service can now be confined by SELinux. Policy writers can also use these fine-grained controls to confine administrators.

IMPORTANT

To avoid incorrect SELinux labeling and subsequent problems, ensure that you start services using the **systemctl start** command.

Red Hat Enterprise Linux 8 provides the following packages for working with SELinux:

- policies: **selinux-policy-targeted**, **selinux-policy-mls**
- tools: **policycoreutils**, **policycoreutils-gui**, **libselinux-utils**, **policycoreutils-python-utils**, **setools-console**, **checkpolicy**

9.5. SELINUX STATES AND MODES

SELinux can run in one of three modes: disabled, permissive, or enforcing.

Disabled mode is strongly discouraged; not only does the system avoid enforcing the SELinux policy, it also avoids labeling any persistent objects such as files, making it difficult to enable SELinux in the future.

In permissive mode, the system acts as if SELinux is enforcing the loaded security policy, including labeling objects and emitting access denial entries in the logs, but it does not actually deny any operations. While not recommended for production systems, permissive mode can be helpful for SELinux policy development.

Enforcing mode is the default, and recommended, mode of operation; in enforcing mode SELinux operates normally, enforcing the loaded security policy on the entire system.

Use the **setenforce** utility to change between enforcing and permissive mode. Changes made with **setenforce** do not persist across reboots. To change to enforcing mode, enter the **setenforce 1** command as the Linux root user. To change to permissive mode, enter the **setenforce 0** command. Use the **getenforce** utility to view the current SELinux mode:

```
~]# getenforce
Enforcing
```

```
~]# setenforce 0
~]# getenforce
Permissive
```

```
~]# setenforce 1
~]# getenforce
Enforcing
```

In Red Hat Enterprise Linux, you can set individual domains to permissive mode while the system runs in enforcing mode. For example, to make the *httpd_t* domain permissive:

```
~]# semanage permissive -a httpd_t
```

[3] Text files that include information, such as host name to IP address mappings, that are used by DNS servers.

CHAPTER 10. CHANGING SELINUX STATES AND MODES

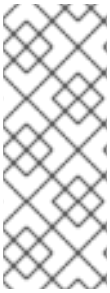
10.1. PERMANENT CHANGES IN SELINUX STATES AND MODES

As discussed in Section 9.5, “SELinux states and modes”, SELinux can be enabled or disabled. When enabled, SELinux has two modes: enforcing and permissive.

Use the **getenforce** or **sestatus** commands to check in which mode SELinux is running. The **getenforce** command returns **Enforcing**, **Permissive**, or **Disabled**.

The **sestatus** command returns the SELinux status and the SELinux policy being used:

```
~]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      31
```



NOTE

When systems run SELinux in permissive mode, users and processes can label various file-system objects incorrectly. File-system objects created while SELinux is disabled are not labeled at all. This behavior causes problems when changing to enforcing mode because SELinux relies on correct labels of file-system objects. To prevent incorrectly labeled and unlabeled files from causing problems, file systems are automatically relabeled when changing from the disabled state to permissive or enforcing mode.

10.2. ENABLING SELINUX

When enabled, SELinux can run in one of two modes: enforcing or permissive. The following sections show how to permanently change into these modes.

While enabling SELinux on systems that previously had it disabled, to avoid problems, such as systems unable to boot or process failures, follow this procedure:

1. Enable SELinux in permissive mode. For more information, see Section 10.2.1, “Changing to permissive mode”.
2. Reboot your system.
3. Check for SELinux denial messages.
4. If there are no denials, switch to enforcing mode. For more information, see Section 10.2.2, “Changing to enforcing mode”.

To run custom applications with SELinux in enforcing mode, choose one of the following scenarios:

- Run your application in the **unconfined_service_t** domain.

- Write a new policy for your application. See the Writing Custom SELinux Policy Knowledgebase article for more information.

Temporary changes in modes are covered in Section 9.5, “SELinux states and modes”.

10.2.1. Changing to permissive mode

When SELinux is running in permissive mode, SELinux policy is not enforced. The system remains operational and SELinux does not deny any operations but only logs AVC messages, which can be then used for troubleshooting, debugging, and SELinux policy improvements. Each AVC is logged only once in this case.

To permanently change mode to permissive, follow the procedure below:

1. Edit the **/etc/selinux/config** file as follows:

```
# 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
```

2. Reboot the system:

```
~]# reboot
```

10.2.2. Changing to enforcing mode

When SELinux is running in enforcing mode, it enforces the SELinux policy and denies access based on SELinux policy rules. In Red Hat Enterprise Linux, enforcing mode is enabled by default when the system was initially installed with SELinux.

Prerequisites
This procedure assumes that the **selinux-policy-targeted**, **libselinux-utils**, and **policycoreutils** packages are installed.

Procedure
If SELinux was disabled, follow the procedure below to change mode to enforcing again:

1. Edit the **/etc/selinux/config** file as follows:

```
# 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
```

2. Reboot the system:

```
~]# reboot
```

On the next boot, SELinux relabels all the files and directories within the system and adds SELinux context for files and directories that were created when SELinux was disabled.

NOTE

After changing to enforcing mode, SELinux may deny some actions because of incorrect or missing SELinux policy rules. To view what actions SELinux denies, enter the following command as root:

```
~]# ausearch -m AVC,USER_AVC,SELINUX_ERR -ts today
```

Alternatively, with the **setroubleshoot-server** package installed, enter the following command as root:

```
~]# grep "SELinux is preventing" /var/log/messages
```

10.3. DISABLING SELINUX

When SELinux is disabled, SELinux policy is not loaded at all; it is not enforced and AVC messages are not logged. Therefore, all benefits of running SELinux listed in Benefits of SELinux are lost.

IMPORTANT

Red Hat strongly recommends to use permissive mode instead of permanently disabling SELinux. See Section 10.2.1, “Changing to permissive mode” for more information about permissive mode.

To permanently disable SELinux, follow the procedure below:

1. Configure **`SELINUX=disabled`** in the **`/etc/selinux/config`** file:

```
# 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
```

2. Reboot your system. After reboot, confirm that the **`getenforce`** command returns **`Disabled`**:

```
~]$ getenforce
Disabled
```

10.4. CHANGING SELINUX MODES AT BOOT TIME

On boot, you can set several kernel parameters to change the way SELinux runs:

enforcing=0

Setting this parameter causes the machine to boot in permissive mode, which is useful when troubleshooting issues. Using permissive mode might be the only option to detect a problem if your file system is too corrupted. Moreover, in permissive mode the system continues to create the labels correctly. The AVC messages that are created in this mode can be different than in enforcing mode. In permissive mode, only the first denial is reported. However, in enforcing mode you might get a denial on reading a directory and an application stops. In permissive mode, you get the same AVC message, but the application continues reading files in the directory and you get an AVC for each denial in addition.

selinux=0

This parameter causes the kernel to not load any part of the SELinux infrastructure. The init scripts notice that the system booted with the **`selinux=0`** parameter and touch the **`/.autorelabel`** file. This causes the system to automatically relabel the next time you boot with SELinux enabled.

IMPORTANT

Red Hat does not recommend using the **`selinux=0`** parameter. To debug your system, prefer using permissive mode.

autorelabel=1

This parameter forces the system to relabel similarly to the following commands:

```
~]# touch /.autorelabel
~]# reboot
```

If a file system contains a large amount of mislabeled objects, you might need to boot in permissive mode in order to make the autorelabel process successful.

CHAPTER 11. USING AND CONFIGURING FIREWALLS

A *firewall* is a way to protect machines from any unwanted traffic from outside. It enables users to control incoming network traffic on host machines by defining a set of *firewall rules*. These rules are used to sort the incoming traffic and either block it or allow through.

11.1. GETTING STARTED WITH `FIREWALLD`

`firewalld` is a firewall service daemon that provides a dynamic customizable host-based firewall with a **`D-Bus`** interface. Being dynamic, it enables creating, changing, and deleting the rules without the necessity to restart the firewall daemon each time the rules are changed.

`firewalld` uses the concepts of *zones* and *services*, that simplify the traffic management. Zones are predefined sets of rules. Network interfaces and sources can be assigned to a zone. The traffic allowed depends on the network your computer is connected to and the security level this network is assigned. Firewall services are predefined rules that cover all necessary settings to allow incoming traffic for a specific service and they apply within a zone.

Services use one or more *ports* or *addresses* for network communication. Firewalls filter communication based on ports. To allow network traffic for a service, its ports must be *open*. **`firewalld`** blocks all traffic on ports that are not explicitly set as open. Some zones, such as *trusted*, allow all traffic by default.

11.1.1. Zones

`firewalld` can be used to separate networks into different zones according to the level of trust that the user has decided to place on the interfaces and traffic within that network. A connection can only be part of one zone, but a zone can be used for many network connections.

NetworkManager notifies **`firewalld`** of the zone of an interface. You can assign zones to interfaces with **NetworkManager**, with the **firewall-config** tool, or the **`firewall-cmd`** command-line tool. The latter two only edit the appropriate **NetworkManager** configuration files. If you change the zone of the interface using **`firewall-cmd`** or **firewall-config**, the request is forwarded to **NetworkManager** and is not handled by **`firewalld`**.

The predefined zones are stored in the **`/usr/lib/firewalld/zones/`** directory and can be instantly applied to any available network interface. These files are copied to the **`/etc/firewalld/zones/`** directory only after they are modified. The following table describes the default settings of the predefined zones:

`block`

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.

`dmz`

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

`drop`

Any incoming network packets are dropped without any notification. Only outgoing network connections are possible.

`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.

`home`

For use at home when you mostly trust the other computers on the network. Only selected incoming connections are accepted.

`internal`

For use on internal networks when you mostly trust the other computers on the network. Only selected incoming connections are accepted.

`public`

For use in public areas where you do not trust other computers on the network. Only selected incoming connections are accepted.

`trusted`

All network connections are accepted.

`work`

For use at work where you mostly trust the other computers on the network. Only selected incoming connections are accepted.

One of these zones is set as 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. The default zone can be changed.

NOTE

The network zone names have been chosen to be self-explanatory and to allow users to quickly make a reasonable decision. To avoid any security problems, review the default zone configuration and disable any unnecessary services according to your needs and risk assessments.

11.1.2. Predefined services

A service can be a list of local ports, protocols, source ports, and destinations, as well as a list of firewall helper modules automatically loaded if a service is enabled. Using services saves users time because they can achieve several tasks, such as opening ports, defining protocols, enabling packet forwarding and more, in a single step, rather than setting up everything one after another.

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`**. Protocol names are preferred over service or application names in **`firewalld`**.

11.1.3. Runtime and permanent settings

Any changes committed in *runtime* mode only apply while **`firewalld`** is running. When **`firewalld`** is restarted, the settings revert to their *permanent* values.

To make the changes persistent across reboots, apply them again using the **`--permanent`** option. Alternatively, to make changes persistent while **`firewalld`** is running, use the **`--runtime-to-permanent`** **`firewall-cmd`** option.

If you set the rules while **firewalld** is running using only the **--permanent** option, they do not become effective before **firewalld** is restarted. However, restarting **firewalld** closes all open ports and stops the networking traffic.

11.1.4. Modifying settings in runtime and permanent configuration using CLI

Using the CLI, you do not modify the firewall settings in both modes at the same time. You only modify either runtime or permanent mode. To modify the firewall settings in the permanent mode, use the **--permanent** option with the **firewall-cmd** command.

```
~]# firewall-cmd --permanent <other options>
```

Without this option, the command modifies runtime mode.

To change settings in both modes, you can use two methods:

1. Change runtime settings and then make them permanent as follows:

```
~]# firewall-cmd <other options>
~]# firewall-cmd --runtime-to-permanent
```

2. Set permanent settings and reload the settings into runtime mode:

```
~]# firewall-cmd --permanent <other options>
~]# firewall-cmd --reload
```

The first method allows you to test the settings before you apply them to the permanent mode.

NOTE

It is possible, especially on remote systems, that an incorrect setting results in a user locking themselves out of a machine. To prevent such situations, use the **--timeout** option. After a specified amount of time, any change reverts to its previous state. Using this options excludes the **--permanent** option.

For example, to add the **SSH** service for 15 minutes:

```
~]# firewall-cmd --add-service=ssh --timeout 15m
```

11.2. INSTALLING THE FIREWALL-CONFIG GUI CONFIGURATION TOOL

To use the **firewall-config** GUI configuration tool, install the **firewall-config** package as **root**:

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

Alternatively, in **GNOME**, use the **Super** key and type **Software** to launch the **Software Sources** application. Type **firewall** to the search box, which appears after selecting the search button in the top-right corner. Select the **Firewall** item from the search results, and click on the **Install** button.

To run **firewall-config**, use either the **firewall-config** command or press the **Super** key to enter the **Activities Overview**, type **firewall**, and press **Enter**.

11.3. VIEWING THE CURRENT STATUS AND SETTINGS OF FIREWALLD

11.3.1. Viewing the current status of firewalld

The firewall service, **firewalld**, is installed on the system by default. Use the **firewalld** CLI interface to check that the service is running.

To see the status of the service:

```
~]# firewall-cmd --state
```

For more information about the service status, use the **systemctl status** sub-command:

```
~]# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
vendor pr
   Active: active (running) since Mon 2017-12-18 16:05:15 CET; 50min ago
     Docs: man:firewalld(1)
 Main PID: 705 (firewalld)
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/firewalld.service
           └─705 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
```

Furthermore, it is important to know how **firewalld** is set up and which rules are in force before you try to edit the settings. To display the firewall settings, see Section 11.3.2, “Viewing current **firewalld** settings”

11.3.2. Viewing current firewalld settings

11.3.2.1. Viewing allowed services using GUI

To view the list of services using the graphical **firewall-config** tool, press the **Super** key to enter the Activities Overview, type **firewall**, and press **Enter**. The **firewall-config** tool appears. You can now view the list of services under the **Services** tab.

Alternatively, to start the graphical firewall configuration tool using the command-line, enter the following command:

```
~]$ firewall-config
```

The **Firewall Configuration** window opens. Note that this command can be run as a normal user, but you are prompted for an administrator password occasionally.

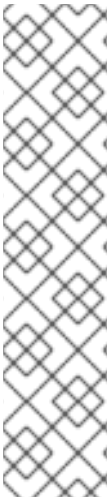
11.3.2.2. Viewing firewalld settings using CLI

With the CLI client, it is possible to get different views of the current firewall settings. The **--list-all** option shows a complete overview of the **firewalld** settings.

firewalld uses zones to manage the traffic. If a zone is not specified by the **--zone** option, the command is effective in the default zone assigned to the active network interface and connection.

To list all the relevant information for the default zone:

```
~]# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```



NOTE

To specify the zone for which to display the settings, add the **--zone=*zone-name*** argument to the **firewall-cmd --list-all** command, for example:

```
~]# firewall-cmd --list-all --zone=home
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh mdns samba-client dhcpv6-client
... [output truncated]
```

To see the settings for particular information, such as services or ports, use a specific option. See the **firewalld** manual pages or get a list of the options using the command help:

```
~]# firewall-cmd --help

Usage: firewall-cmd [OPTIONS...]

General Options
  -h, --help           Prints a short help text and exists
  -V, --version        Print the version string of firewalld
  -q, --quiet          Do not print status messages

Status Options
  --state              Return and print firewalld state
  --reload             Reload firewall and keep state information
... [output truncated]
```

For example, to see which services are allowed in the current zone:

```
~]# firewall-cmd --list-services
ssh dhcpv6-client
```

Listing the settings for a certain subpart using the CLI tool can sometimes be difficult to interpret. For example, you allow the **SSH** service and **firewalld** opens the necessary port (22) for the service. Later, if you list the allowed services, the list shows the **SSH** service, but if you list open ports, it does not show any. Therefore, it is recommended to use the **--list-all** option to make sure you receive a complete information.

=== To start **firewalld**, enter the following command as **root**:

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

To ensure **firewalld** starts automatically at system start, enter the following command as **root**:

```
~]# systemctl enable firewalld
```

11.4. STOPPING FIREWALLD

To stop **firewalld**, enter the following command as **root**:

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

To prevent **firewalld** from starting automatically at system start, enter the following command as **root**:

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

To make sure firewalld is not started by accessing the **firewalld D-Bus** interface and also if other services require **firewalld**, enter the following command as **root**:

```
~]# systemctl mask firewalld
```

11.5. CONTROLLING TRAFFIC

11.5.1. Predefined services

Services can be added and removed using the graphical **firewall-config** tool, **firewall-cmd**, and **firewall-offline-cmd**.

Alternatively, you can edit the XML files in the **/etc/firewalld/services/** directory. If a service is not added or changed by the user, then no corresponding XML file is found in **/etc/firewalld/services/**. The files in the **/usr/lib/firewalld/services/** directory can be used as templates if you want to add or change a service.

11.5.1.1. Disabling all traffic in case of emergency using CLI

In an emergency situation, such as a system attack, it is possible to disable all network traffic and cut off the attacker.

To immediately disable networking traffic, switch panic mode on:

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

IMPORTANT

Enabling panic mode stops all networking traffic. From this reason, it should be used only when you have the physical access to the machine or if you are logged in using a serial console.

Switching off panic mode reverts the firewall to its permanent settings. To switch panic mode off:

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

To see whether panic mode is switched on or off, use:

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

11.5.2. Controlling traffic with predefined services using CLI

The most straightforward method to control traffic is to add a predefined service to **firewalld**. This opens all necessary ports and modifies other settings according to the *service definition file*.

1. Check that the service is not already allowed:

```
~]# firewall-cmd --list-services
ssh dhcpv6-client
```

2. List all predefined services:

```
~]# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client
bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc ceph ceph-
mon cfengine condor-collector ctdb dhcp dhcpv6 dhcpv6-client dns
docker-registry ...
[output truncated]
```

3. Add the service to the allowed services:

```
~]# firewall-cmd --add-service=<service-name>
```

4. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.5.3. Controlling traffic with predefined services using GUI

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 select **Permanent** from the menu labeled **Configuration**. Additional icons and menu buttons appear at the bottom of the **Services** window. Select the service you want to configure.

The **Ports**, **Protocols**, and **Source Port** tabs enable adding, changing, and removing of ports, protocols, and source port 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**).

NOTE

It is not possible to alter service settings in **Runtime** mode.

11.5.4. Adding new services

Services can be added and removed using the graphical **firewall-config** tool, **firewall-cmd**, and **firewall-offline-cmd**. Alternatively, you can edit the XML files in **/etc/firewalld/services/**. If a service is not added or changed by the user, then no corresponding XML file are found in **/etc/firewalld/services/**. The files **/usr/lib/firewalld/services/** can be used as templates if you want to add or change a service.

To add a new service in a terminal, use **firewall-cmd**, or **firewall-offline-cmd** in case of not active **firewalld**. enter the following command to add a new and empty service:

```
~]$ firewall-cmd --new-service=service-name
```

To add a new service using a local file, use the following command:

```
~]$ firewall-cmd --new-service-from-file=service-name.xml
```

You can change the service name with the additional **--name=*service-name*** option.

As soon as service settings are changed, an updated copy of the service is placed into **/etc/firewalld/services/**.

As **root**, you can enter the following command to copy a service manually:

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

firewalld loads files from **/usr/lib/firewalld/services** in the first place. If files are placed in **/etc/firewalld/services** and they are valid, then these will override the matching files from **/usr/lib/firewalld/services**. The overriden files in **/usr/lib/firewalld/services** are used as soon as the matching files in **/etc/firewalld/services** have been removed or if **firewalld** has been asked to load the defaults of the services. This applies to the permanent environment only. A reload is needed to get these fallbacks also in the runtime environment.

11.5.5. Controlling ports using CLI

Ports are logical devices that enable an operating system to receive and distinguish network traffic and forward it accordingly to system services. These are usually represented by a daemon that listens on the port, that is it waits for any traffic coming to this port.

Normally, system services listen on standard ports that are reserved for them. The **httpd** daemon, for example, listens on port 80. However, system administrators by default configure daemons to listen on different ports to enhance security or for other reasons.

Opening a Port

Through open ports, the system is accessible from the outside, which represents a security risk. Generally, keep ports closed and only open them if they are required for certain services.

To get a list of open ports in the current zone:

1. List all allowed ports:

```
~]# firewall-cmd --list-ports
```

2. Add a port to the allowed ports to open it for incoming traffic:

```
~]# firewall-cmd --add-port=port-number/port-type
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

The port types are either **tcp**, **udp**, **sctp**, or **dccp**. The type must match the type of network communication.

Closing a Port

When an open port is no longer needed, close that port in **firewalld**. It is highly recommended to close all unnecessary ports as soon as they are not used because leaving a port open represents a security risk.

To close a port, remove it from the list of allowed ports:

1. List all allowed ports:

```
~]# firewall-cmd --list-ports
[WARNING]
====
This command will only give you a list of ports that have been
opened as ports. You will not be able to see any open ports that
have been opened as a service. Therefore, you should consider using
the --list-all option instead of --list-ports.
====
```

2. Remove the port from the allowed ports to close it for the incoming traffic:

```
~]# firewall-cmd --remove-port=port-number/port-type
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.5.6. Opening ports using GUI

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 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 list.

11.5.7. Controlling traffic with protocols using GUI

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

Either select a protocol from the list or select the **Other Protocol** check box and enter the protocol in the field.

11.5.8. Opening source ports using GUI

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

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

11.6. WORKING WITH ZONES

Zones represent a concept to manage incoming traffic more transparently. The zones are connected to networking interfaces or assigned a range of source addresses. You manage firewall rules for each zone independently, which enables you to define complex firewall settings and apply them to the traffic.

11.6.1. Listing zones

To see which zones are available on your system:

```
~]# firewall-cmd --get-zones
```

The **firewall-cmd --get-zones** command displays all zones that are available on the system, but it does not show any details for particular zones.

To see detailed information for all zones:

```
~]# firewall-cmd --list-all-zones
```

To see detailed information for a specific zone:

```
~]# firewall-cmd --zone=zone-name --list-all
```

11.6.2. Modifying `firewalld` settings for a certain zone

The Section 11.5.2, “Controlling traffic with predefined services using CLI” and Section 11.5.5, “Controlling ports using CLI” explain how to add services or modify ports in the scope of the current working zone. Sometimes, it is required to set up rules in a different zone.

To work in a different zone, use the **--zone=*zone-name*** option. For example, to allow the **SSH** service in the zone *public*:

```
~]# firewall-cmd --add-service=ssh --zone=public
```

11.6.3. Changing the default zone

System administrators assign a zone to a networking interface in its configuration files. If an interface is not assigned to a specific zone, it is assigned to the default zone. After each restart of the **firewalld** service, **firewalld** loads the settings for the default zone and makes it active.

To set up the default zone:

1. Display the current default zone:

```
~]# firewall-cmd --get-default-zone
```

2. Set the new default zone:

```
~]# firewall-cmd --set-default-zone zone-name
```

NOTE

Following this procedure, the setting is a permanent setting, even without the **--permanent** option.

11.6.4. Assigning a network interface to a zone

It is possible to define different sets of rules for different zones and then change the settings quickly by changing the zone for the interface that is being used. With multiple interfaces, a specific zone can be set for each of them to distinguish traffic that is coming through them.

To assign the zone to a specific interface:

1. List the active zones and the interfaces assigned to them:

```
~]# firewall-cmd --get-active-zones
```

2. Assign the interface to a different zone:

```
~]# firewall-cmd --zone=zone-name --change-interface=<interface-
name>
```

NOTE

You do not have to use the **--permanent** option to make the setting persistent across restarts. If you set a new default zone, the setting becomes permanent.

11.6.5. Assigning a default zone to a network connection

When the connection is managed by **NetworkManager**, it must be aware of a zone that it uses. For every network connection, a zone can be specified, which provides the flexibility of various firewall

settings according to the location of the computer with portable devices. Thus, zones and settings can be specified for different locations, such as company or home.

To set a default zone for an Internet connection, use either the **NetworkManager** GUI or edit the **/etc/sysconfig/network-scripts/ifcfg-*connection-name*** file and add a line that assigns a zone to this connection:

```
ZONE=zone-name
```

11.6.6. Creating a new zone

To use custom zones, create a new zone and use it just like a predefined zone. New zones require the **--permanent** option, otherwise the command does not work.

To create a new zone:

1. Create a new zone:

```
~]# firewall-cmd --new-zone=zone-name
```

2. Check if the new zone is added to your permanent settings:

```
~]# firewall-cmd --get-zones
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.6.7. Creating a new zone using a configuration file

Zones can also be created using a *zone configuration file*. This approach can be helpful when you need to create a new zone, but want to reuse the settings from a different zone and only alter them a little.

A **firewalld** zone configuration file contains the information for a zone. These are the zone description, services, ports, protocols, icmp-blocks, masquerade, forward-ports and rich language rules in an XML file format. The file name has to be ***zone-name*.xml** where the length of *zone-name* is currently limited to 17 chars. The zone configuration files are located in the **/usr/lib/firewalld/zones/** and **/etc/firewalld/zones/** directories.

The following example shows a configuration that allows one service (**SSH**) and one port range, for both the **TCP** and **UDP** protocols.:

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>My zone</short>
  <description>Here you can describe the characteristic features of the
zone.</description>
  <service name="ssh"/>
  <port port="1025-65535" protocol="tcp"/>
  <port port="1025-65535" protocol="udp"/>
</zone>
```

To change settings for that zone, add or remove sections to add ports, forward ports, services, and so on. For more information, see the **firewalld.zone** manual pages.

11.6.8. Using zone targets to set default behavior for incoming traffic

For every zone, you can set a default behavior that handles incoming traffic that is not further specified. Such behaviour is defined by setting the target of the zone. There are three options - **default**, **ACCEPT**, **REJECT**, and **DROP**. By setting the target to **ACCEPT**, you accept all incoming packets except those disabled by a specific rule. If you set the target to **REJECT** or **DROP**, you disable all incoming packets except those that you have allowed in specific rules. When packets are rejected, the source machine is informed about the rejection, while there is no information sent when the packets are dropped.

To set a target for a zone:

1. List the information for the specific zone to see the default target:

```
~]$ firewall-cmd --zone=zone-name --list-all
```

2. Set a new target in the zone:

```
~]# firewall-cmd --zone=zone-name --set-target=
<default|ACCEPT|REJECT|DROP>
```

11.7. USING ZONES TO MANAGE INCOMING TRAFFIC DEPENDING ON A SOURCE

You can use zones to manage incoming traffic based on its source. That enables you to sort incoming traffic and route it through different zones to allow or disallow services that can be reached by that traffic.

If you add a source to a zone, the zone becomes active and any incoming traffic from that source will be directed through it. You can specify different settings for each zone, which is applied to the traffic from the given sources accordingly. You can use more zones even if you only have one network interface.

11.7.1. Adding a source

To route incoming traffic into a specific source, add the source to that zone. The source can be an IP address or an IP mask in the Classless Inter-domain Routing (CIDR) notation.

1. To set the source in the current zone:

```
~]# firewall-cmd --add-source=<source>
```

2. To set the source IP address for a specific zone:

```
~]# firewall-cmd --zone=zone-name --add-source=<source>
```

The following procedure allows all incoming traffic from *192.168.2.15* in the **trusted** zone:

1. List all available zones:

```
~]# firewall-cmd --get-zones
```

2. Add the source IP to the trusted zone in the permanent mode:

```
~]# firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.7.2. Removing a source

Removing a source from the zone cuts off the traffic coming from it.

1. List allowed sources for the required zone:

```
~]# firewall-cmd --zone=zone-name --list-sources
```

2. Remove the source from the zone permanently:

```
~]# firewall-cmd --zone=zone-name --remove-source=<source>
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.7.3. Adding a source port

To enable sorting the traffic based on a port of origin, specify a source port using the **`--add-source-port`** option. You can also combine this with the **`--add-source`** option to limit the traffic to a certain IP address or IP range.

To add a source port:

```
~]# firewall-cmd --zone=zone-name --add-source-port=<port-
name>/<tcp|udp|sctp|dccp>
```

11.7.4. Removing a source port

By removing a source port you disable sorting the traffic based on a port of origin.

To remove a source port:

```
~]# firewall-cmd --zone=zone-name --remove-source-port=<port-
name>/<tcp|udp|sctp|dccp>
```

11.7.5. Using zones and sources to allow a service for only a specific domain

To allow traffic from a specific network to use a service on a machine, use zones and source.

For example, to allow traffic from *192.168.1.0/24* to be able to reach the *HTTP* service while any other traffic is blocked:

1. List all available zones:

```
~]# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

2. Add the source to the trusted zone to route the traffic originating from the source through the zone:

```
~]# firewall-cmd --zone=trusted --add-source=192.168.1.0/24
```

3. Add the *http* service in the trusted zone:

```
~]# firewall-cmd --zone=trusted -add-service=http
```

4. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

5. Check that the trusted zone is active and that the service is allowed in it:

```
~]# firewall-cmd --zone=trusted --list-all
trusted (active)
target: ACCEPT
sources: 192.168.1.0/24
services: http
```

11.7.6. Configuring traffic accepted by a zone based on a protocol

You can allow incoming traffic to be accepted by a zone based on a protocol. All traffic using the specified protocol is accepted by a zone, in which you can apply further rules and filtering.

Adding a protocol to a zone

By adding a protocol to a certain zone, you allow all traffic with this protocol to be accepted by this zone.

To add a protocol to a zone:

```
~]# firewall-cmd --zone=zone-name --add-protocol=port-
name/tcp|udp|sctp|dccp|igmp
```

NOTE

To receive multicast traffic, use the **`igmp`** value with the **`--add-protocol`** option.

Removing a protocol from a zone

By removing a protocol from a certain zone, you stop accepting all traffic based on this protocol by the zone.

To remove a protocol from a zone:

```
~]# firewall-cmd --zone=zone-name --remove-protocol=port-
name/tcp|udp|sctp|dccp|igmp
```

-

11.8. PORT FORWARDING

Using **firewalld**, you can set up ports redirection so that any incoming traffic that reaches a certain port on your system is delivered to another internal port of your choice or to an external port on another machine.

11.8.1. Adding a port to redirect

Before you redirect traffic from one port to another port, or another address, you need to know three things: which port the packets arrive at, what protocol is used, and where you want to redirect them.

To redirect a port to another port:

```
~]# firewall-cmd --add-forward-port=port=port-
number:proto=tcp|udp|sctp|dccp:toport=port-number
```

To redirect a port to another port at a different IP address:

1. Add the port to be forwarded:

```
~]# firewall-cmd --add-forward-port=port=port-
number:proto=tcp|udp:toport=port-number:toaddr=IP/mask
```

2. Enable masquerade:

```
~]# firewall-cmd -add-masquerade
```

Example 11.1. Redirecting TCP port 80 to port 88 on the same machine

To redirect the port:

1. Redirect the port 80 to port 88 for TCP traffic:

```
~]# firewall-cmd --add-forward-port=port=80:proto=tcp:toport=88
```

2. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

3. Check that the port is redirected:

```
~]# firewall-cmd --list-all
```

11.8.2. Removing a redirected Port

To remove a redirected port:

```
~]# firewall-cmd --remove-forward-port=port=port-number:proto=
<tcp|udp>:toport=port-number:toaddr=<IP/mask>
```

To remove a forwarded port redirected to a different address:

1. Remove the forwarded port:

```
~]# firewall-cmd --remove-forward-port=port=port-number:proto=
<tcp|udp>:toport=port-number:toaddr=<IP/mask>
```

2. Disable masquerade:

```
~]# firewall-cmd --remove-masquerade
```

NOTE

Redirecting ports using this method only works for IPv4-based traffic. For IPv6 redirecting setup, you need to use rich rules.

To redirect to an external system, it is necessary to enable masquerading. For more information, see Section 11.9, “Configuring IP address masquerading”.

Example 11.2. Removing TCP port 80 forwarded to port 88 on the same machine

To remove the port redirection:

1. List redirected ports:

```
~]# firewall-cmd --list-forward-ports
port=80:proto=tcp:toport=88:toaddr=
```

2. Remove the redirected port from the firewall::

```
~]# firewall-cmd  --remove-forward-
port=port=80:proto=tcp:toport=88:toaddr=
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.9. CONFIGURING 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
```

The command prints **yes** with exit status **0** if enabled. It prints **no** with exit status **1** otherwise. 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 persistent, repeat the command adding the **--permanent** option.

To disable IP masquerading, enter the following command as **root**:

```
~]# firewall-cmd --zone=external --remove-masquerade
```

To make this setting persistent, repeat the command adding the **--permanent** option.

11.10. MANAGING ICMP REQUESTS

The **Internet Control Message Protocol** (**ICMP**) is a supporting protocol that is used by various network devices to send error messages and operational information indicating a connection problem, for example, that a requested service is not available. **ICMP** differs from transport protocols such as TCP and UDP because it is not used to exchange data between systems.

Unfortunately, it is possible to use the **ICMP** messages, especially **echo-request** and **echo-reply**, to reveal information about your network and misuse such information for various kinds of fraudulent activities. Therefore, **firewalld** enables blocking the **ICMP** requests to protect your network information.

11.10.1. Listing ICMP requests

The **ICMP** requests are described in individual XML files that are located in the **/usr/lib/firewalld/icmptypes/** directory. You can read these files to see a description of the request. The **firewall-cmd** command controls the **ICMP** requests manipulation.

To list all available **ICMP** types:

```
~]# firewall-cmd --get-icmptypes
```

The **ICMP** request can be used by IPv4, IPv6, or by both protocols. To see for which protocol the **ICMP** request is used:

```
~]# firewall-cmd --info-icmptype=<icmptype>
```

The status of an **ICMP** request shows **yes** if the request is currently blocked or **no** if it is not. To see if an **ICMP** request is currently blocked:

```
~]# firewall-cmd --query-icmp-block=<icmptype>
```

11.10.2. Blocking or unblocking ICMP requests

When your server blocks **ICMP** requests, it does not provide the information that it normally would. However, that does not mean that no information is given at all. The clients receive information that the particular **ICMP** request is being blocked (rejected). Blocking the **ICMP** requests should be considered carefully, because it can cause communication problems, especially with IPv6 traffic.

To see if an **ICMP** request is currently blocked:

```
~]# firewall-cmd --query-icmp-block=<icmptype>
```

To block an **ICMP** request:

```
~]# firewall-cmd --add-icmp-block=<icmptype>
```

To remove the block for an **ICMP** request:

```
~]# firewall-cmd --remove-icmp-block=<icmptype>
```

11.10.3. Blocking ICMP requests without providing any information at all

Normally, if you block **ICMP** requests, clients know that you are blocking it. So, a potential attacker who is sniffing for live IP addresses is still able to see that your IP address is online. To hide this information completely, you have to drop all **ICMP** requests.

To block and drop all **ICMP** requests:

1. Set the target of your zone to **DROP**:

   ```
   ~]# firewall-cmd --set-target=DROP
   ```

2. Make the new settings persistent:

   ```
   ~]# firewall-cmd --runtime-to-permanent
   ```

Now, all traffic, including **ICMP** requests, is dropped, except traffic which you have explicitly allowed.

To block and drop certain **ICMP** requests and allow others:

1. Set the target of your zone to **DROP**:

   ```
   ~]# firewall-cmd --set-target=DROP
   ```

2. Add the ICMP block inversion to block all **ICMP** requests at once:

   ```
   ~]# firewall-cmd --add-icmp-block-inversion
   ```

3. Add the ICMP block for those **ICMP** requests that you want to allow:

   ```
   ~]# firewall-cmd --add-icmp-block=<icmptype>
   ```

4. Make the new settings persistent:

   ```
   ~]# firewall-cmd --runtime-to-permanent
   ```

The *block inversion* inverts the setting of the **ICMP** requests blocks, so all requests, that were not previously blocked, are blocked. Those that were blocked are not blocked. Which means that if you need to unblock a request, you must use the blocking command.

To revert this to a fully permissive setting:

1. Set the target of your zone to **default** or **ACCEPT**:

```
~]# firewall-cmd --set-target=default
```

2. Remove all added blocks for **ICMP** requests:

```
~]# firewall-cmd --remove-icmp-block=<icmptype>
```

3. Remove the **ICMP** block inversion:

```
~]# firewall-cmd --remove-icmp-block-inversion
```

4. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

11.10.4. Configuring the ICMP filter using GUI

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** type, start the **firewall-config** tool and select **Permanent** mode from the menu labeled **Configuration**. Additional icons appear at the bottom of the **Services** window. Select **Yes** in the following dialog to enable masquerading and to make forwarding to another machine working.

To enable inverting the **ICMP Filter**, click the **Invert Filter** check box on the right. Only marked **ICMP** types are now accepted, all other are rejected. In a zone using the DROP target, they are dropped.

11.11. SETTING AND CONTROLLING IP SETS USING FIREWALLD

To see the list of IP set types supported by **firewalld**, enter the following command as root.

```
~]# firewall-cmd --get-ipset-types
hash:ip hash:ip,mark hash:ip,port hash:ip,port,ip hash:ip,port,net
hash:mac hash:net hash:net,iface hash:net,net hash:net,port
hash:net,port,net
```

11.11.1. Configuring IP set options with the command-line client

IP sets can be used in **firewalld** zones as sources and also as sources in rich rules. In Red Hat Enterprise Linux, the preferred method is to use the IP sets created with **firewalld** in a direct rule.

To list the IP sets known to **firewalld** in the permanent environment, use the following command as **root**:

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

To add a new IP set, use the following command using the permanent environment as **root**:

```
~]# firewall-cmd --permanent --new-ipset=test --type=hash:net
success
```

The previous command creates a new IP set with the name *test* and the **hash:net** type for **IPv4**. To create an IP set for use with **IPv6**, add the **--option=family=inet6** option. To make the new setting effective in the runtime environment, reload **firewalld**. List the new IP set with the following command as **root**:

```
~]# firewall-cmd --permanent --get-ipsets
test
```

To get more information about the IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --info-ipset=test
test
type: hash:net
options:
entries:
```

Note that the IP set does not have any entries at the moment. To add an entry to the *test* IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --add-entry=192.168.0.1
success
```

The previous command adds the IP address *192.168.0.1* to the IP set. To get the list of current entries in the IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

Generate a file containing a list of IP addresses, for example:

```
~]# cat > iplist.txt <<EOL
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
EOL
```

The file with the list of IP addresses for an IP set should contain an entry per line. Lines starting with a hash, a semi-colon, or empty lines are ignored.

To add the addresses from the *iplist.txt* file, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --add-entries-from-
file=iplist.txt
success
```

To see the extended entries list of the IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
192.168.0.2
```

```
192.168.0.3
192.168.1.0/24
192.168.2.254
```

To remove the addresses from the IP set and to check the updated entries list, use the following commands as **root**:

```
~]# firewall-cmd --permanent --ipset=test --remove-entries-from-
file=iplist.txt
success
~]# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

You can add the IP set as a source to a zone to handle all traffic coming in from any of the addresses listed in the IP set with a zone. For example, to add the *test* IP set as a source to the *drop* zone to drop all packets coming from all entries listed in the *test* IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --zone=drop --add-source=ipset:test
success
```

The **ipset:** prefix in the source shows **firewalld** that the source is an IP set and not an IP address or an address range.

Only the creation and removal of IP sets is limited to the permanent environment, all other IP set options can be used also in the runtime environment without the **--permanent** option.

WARNING

Red Hat does not recommend using IP sets that are not managed through **firewalld**. To use such IP sets, a permanent direct rule is required to reference the set, and a custom service must be added to create these IP sets. This service needs to be started before firewalld starts, otherwise **firewalld** is not able to add the direct rules using these sets. You can add permanent direct rules with the **/etc/firewalld/direct.xml** file.

11.12. CONFIGURING FIREWALL LOCKDOWN

Local applications or services are able to change the firewall configuration if they are running as **root** (for example, **libvirt**). With this feature, the administrator can lock the firewall configuration so that either no applications or only applications that are added to the lockdown whitelist are able to request firewall changes. The lockdown settings default to disabled. If enabled, the user can be sure that there are no unwanted configuration changes made to the firewall by local applications or services.

11.12.1. Configuring lockdown with the command-line client

To query whether lockdown is enabled, use the following command as **root**:

```
~]# firewall-cmd --query-lockdown
```

The command prints **yes** with exit status **0** if lockdown is enabled. It prints **no** with exit status **1** otherwise.

To enable lockdown, enter the following command as **root**:

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

To disable lockdown, use the following command as **root**:

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

11.12.2. Configuring lockdown whitelist options with the command-line client

The lockdown whitelist can contain commands, security contexts, users and user IDs. If a command entry on the whitelist ends with an asterisk "*", then all command lines starting with that command will match. If the "*" is not there then the absolute command including arguments must match.

The context is the security (SELinux) context of a running application or service. To get the context of a running application use the following command:

```
~]$ ps -e --context
```

That command returns all running applications. Pipe the output through the **grep** tool to get the application of interest. For example:

```
~]$ ps -e --context | grep example_program
```

To list all command lines that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-commands
```

To add a command *command* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

To remove a command *command* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

To query whether the command *command* is on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --query-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

The command prints **yes** with exit status **0** if true. It prints **no** with exit status **1** otherwise.

To list all security contexts that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-contexts
```

To add a context *context* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-context=context
```

To remove a context *context* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-context=context
```

To query whether the context *context* is on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --query-lockdown-whitelist-context=context
```

Prints **yes** with exit status **0**, if true, prints **no** with exit status **1** otherwise.

To list all user IDs that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-uids
```

To add a user ID *uid* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-uid=uid
```

To remove a user ID *uid* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-uid=uid
```

To query whether the user ID *uid* is on the whitelist, enter the following command:

```
~]$ firewall-cmd --query-lockdown-whitelist-uid=uid
```

Prints **yes** with exit status **0**, if true, prints **no** with exit status **1** otherwise.

To list all user names that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-users
```

To add a user name *user* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-user=user
```

To remove a user name *user* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-user=user
```

To query whether the user name *user* is on the whitelist, enter the following command:

```
~]$ firewall-cmd --query-lockdown-whitelist-user=user
```

Prints **yes** with exit status **0**, if true, prints **no** with exit status **1** otherwise.

11.12.3. Configuring lockdown whitelist options with configuration files

The default whitelist configuration file contains the **NetworkManager** context and the default context of **libvirt**. The user ID 0 is also on the list.

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
   <user id="0"/>
 </whitelist>
```

Following is an example whitelist configuration file enabling all commands for the **firewall-cmd** utility, for a user called *user* whose user ID is **815**:

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <command name="/usr/bin/python3 -Es /bin/firewall-cmd*"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <user id="815"/>
   <user name="user"/>
 </whitelist>
```

This example shows both **user id** and **user name**, but only one option is required. Python is the interpreter and is prepended to the command line. You can also use a specific command, for example:

```
/usr/bin/python3 /bin/firewall-cmd --lockdown-on
```

In that example, only the **--lockdown-on** command is allowed.

NOTE

In Red Hat Enterprise Linux , all utilities are placed in the **/usr/bin/** directory and the **/bin/** directory is sym-linked to the **/usr/bin/** directory. In other words, although the path for **firewall-cmd** when run as **root** might resolve to **/bin/firewall-cmd**, **/usr/bin/firewall-cmd** can now be used. All new scripts should use the new location. But be aware that if scripts that run as **root** have been written to use the **/bin/firewall-cmd** path, then that command path must be whitelisted in addition to the **/usr/bin/firewall-cmd** path traditionally used only for non-**root** users.

The "*" at the end of the name attribute of a command means that all commands that start with this string will match. If the "*" is not there then the absolute command including arguments must match.

11.13. CONFIGURING LOGGING FOR DENIED PACKETS

With the **LogDenied** option in the **firewalld**, it is possible to add a simple logging mechanism for denied packets. These are the packets that are rejected or dropped. To change the setting of the logging, edit the **/etc/firewalld/firewalld.conf** file or use the command-line or GUI configuration tool.

If **LogDenied** is enabled, logging rules are added right before the reject and drop rules in the INPUT, FORWARD and OUTPUT chains for the default rules and also the final reject and drop rules in zones.

The possible values for this setting are: **all**, **unicast**, **broadcast**, **multicast**, and **off**. The default setting is **off**. With the **unicast**, **broadcast**, and **multicast** setting, the **pkttype** match is used to match the link-layer packet type. With **all**, all packets are logged.

To list the actual **LogDenied** setting with firewall-cmd, use the following command as **root**:

```
~]# firewall-cmd --get-log-denied
off
```

To change the **LogDenied** setting, use the following command as **root**:

```
~]# firewall-cmd --set-log-denied=all
success
```

To change the **LogDenied** setting with the **firewalld** GUI configuration tool, start **firewall-config**, click the **Options** menu and select **Change Log Denied**. The **LogDenied** window appears. Select the new **LogDenied** setting from the menu and click OK.

11.14. ADDITIONAL RESOURCES FOR FIREWALLD

The following sources of information provide additional resources regarding **firewalld**.

11.14.1. 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.
- **firewall-config(1)** man page — Describes settings for the **firewall-config** tool.
- **firewall-offline-cmd(1)** man page — Describes command options for the **firewalld** offline command-line client.
- **firewalld.icmptype(5)** man page — Describes XML configuration files for **ICMP** filtering.
- **firewalld.ipset(5)** man page — Describes XML configuration files for the **firewalld IP** sets.
- **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.
- **firewalld.direct(5)** man page — Describes the **firewalld** direct interface configuration file.
- **firewalld.lockdown-whitelist(5)** man page — Describes the **firewalld** lockdown whitelist configuration file.

- **firewalld.richlanguage(5)** man page — Describes the **firewalld** rich language rule syntax.
- **firewalld.zones(5)** man page — General description of what zones are and how to configure them.
- **firewalld.dbus(5)** man page — Describes the **D-Bus** interface of **firewalld**.

11.14.2. Online documentation

- http://www.firewalld.org/ — **firewalld** home page.

11.15. INTRODUCTION TO NFTABLES

The **nftables** framework provides packet classification facilities and it is the designated successor to the **iptables**, **ip6tables**, **arptables**, and **ebtables** tools. It offers numerous improvements in convenience, features, and performance over previous packet-filtering tools, most notably:

- lookup tables instead of linear processing
- a single framework for both the **IPv4** and **IPv6** protocols
- rules all applied atomically instead of fetching, updating, and storing a complete ruleset
- support for debugging and tracing in the ruleset (**nftrace**) and monitoring trace events (in the **nft** tool)
- more consistent and compact syntax, no protocol-specific extensions
- a Netlink API for third-party applications

Similarly to **iptables**, **nftables** use tables for storing chains. The chains contain individual rules for performing actions. The **nft** tool replaces all tools from the previous packet-filtering frameworks. The **libnftnl** library can be used for low-level interaction with **nftables** Netlink API over the **libmnl** library.

In Red Hat Enterprise Linux 8, **nftables** serve as the default **firewalld** backend. Although the **nftables** backend is backward-compatible with the previous **iptables** backend in firewall configurations, you can switch back to **iptables** by setting the **FirewallBackend** option to the **iptables** value in the **/etc/firewalld/firewalld.conf** file.

Additional resources

- The **nft(8)** man page provides a comprehensive reference documentation for configuring and inspecting packet filtering with nftables using the **nft** command-line tool.
- The What comes after iptables? Its successor, of course: nftables article explains why **nftables** replaces **iptables**.
- The Firewalld: The Future is nftables article provides additional information on **nftables** as a default backend for **firewalld**.