Red Hat Enterprise Linux 8

Securing networks

Configuring secured networks and network communication

Last Updated: 2019-09-20

Red Hat Enterprise Linux 8 Securing networks

Configuring secured networks and network communication

Abstract

This title assists administrators with securing networks, connected machines, and network communication against various attacks.

Table of Contents

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. 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.

1.1. THE SSH PROTOCOL

1.1.1. Reasons for using 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.

1.1.2. 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.

1.1.3. Protocol versions

Two varieties of **SSH** currently exist: version 1, and newer version 2. The **OpenSSH** suite under Red Hat Enterprise Linux 8; 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, the **OpenSSH** suite does not support version 1 connections.

1.1.4. 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.

1.1.4.1. 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.

1.1.4.2. 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.

1.1.4.3. Channels

After a successful authentication over the **SSH** transport layer, multiple channels are opened using a technique called *multiplexing*[1]. 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.

1.2. SETTING UP OPENSSH 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 1.1. 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.
/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.

File	Description
/etc/ssh/ssh_host_ed25519_key	The Ed25519 private key used by the **sshd** daemon for version 2 of the SSH protocol.
/etc/ssh/ssh_host_ed25519_key.pub	The Ed25519 public key used by the **sshd** daemon for version 2 of the SSH protocol.
/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 1.2. 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_ed25519	The Ed25519 private key used by **ssh** for version 2 of the SSH protocol.
~/.ssh/id_ed25519.pub	The Ed25519 public key used by **ssh** for version 2 of the SSH protocol.
~/.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.

1.3. 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 1.1, "System-wide configuration files" for a complete list, and restore the files whenever you reinstall the system.

1.3.1. Configuring the OpenSSH server keys generation

OpenSSH creates RSA, ECDSA, and ED25519 server keys automatically if they are missing. Prior to RHEL 8, administrators have configured the automatic host keys creation in the **/etc/sysconfig/sshd** file when needed. To configure the host key creation in RHEL 8, use the **sshd-keygen@.service** instantiated service.

For example, the following commands disable the automatic RSA key creation and enable the automatic DSA key creation on the OpenSSH server:

```
# systemctl mask sshd-keygen@rsa.service
# systemctl enable sshd-keygen@dsa.service
```

> **WARNING**
>
> Note that DSA keys are insecure due to only small key size supported and thus deprecated.

1.4. 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**.

1.5. 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 login 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**.

1.5.1. 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:

```
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.

To generate an Ed25519 key pair for version 2 of the SSH protocol, use the **-t ed25519** parameter:

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/USER/.ssh/id_ed25519.
Your public key has been saved in /home/USER/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:QAraxfuP5se1CnGTcn7FKLJ++G6dtkzOaw/a46KU5CQ USER@penguin.example.com
The key's randomart image is:
+--[ED25519 256]--+
|  ... .          |
| o o.o           |
|. . ...          |
|    .  . . o     |
|    E.* S . o    |
|     =.X o..     |
|      *=.o+o     |
|     o+.BOB.     |
|     o+B=*O=.    |
+----[SHA256]-----+
```

The generated Ed25519 keys are handled in the same way as RSA or ECDSA keys.

IMPORTANT

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

1.6. RELATED INFORMATION

For more information on how to configure or connect to an **OpenSSH** server on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

- **sshd**(8) — The manual page for the **sshd** daemon documents available command-line options and provides a complete list of supported configuration files and directories.
- **ssh**(1) — The manual page for the **ssh** client application provides a complete list of available command-line options and supported configuration files and directories.
- **scp**(1) — The manual page for the **scp** utility provides a more detailed description of this utility and its usage.
- **sftp**(1) — The manual page for the **sftp** utility.
- **ssh-keygen**(1) — The manual page for the **ssh-keygen** utility documents in detail how to use it to generate, manage, and convert authentication keys used by **ssh**.
- **ssh_config**(5) — The manual page named **ssh_config** documents available SSH client configuration options.
- **sshd_config**(5) — The manual page named **sshd_config** provides a full description of available SSH daemon configuration options.

Online Documentation

- OpenSSH Home Page — The OpenSSH home page containing further documentation, frequently asked questions, links to the mailing lists, bug reports, and other useful resources.
- OpenSSL Home Page — The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

---

[1] 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 2. PLANNING AND IMPLEMENTING TLS

TLS (Transport Layer Security) is a cryptographic protocol used to secure network communications. When hardening system security settings by configuring preferred key-exchange protocols, authentication methods, and encryption algorithms, it is necessary to bear in mind that the broader the range of supported clients, the lower the resulting security. Conversely, strict security settings lead to limited compatibility with clients, which can result in some users being locked out of the system. Be sure to target the strictest available configuration and only relax it when it is required for compatibility reasons.

2.1. SSL AND TLS PROTOCOLS

The Secure Sockets Layer (SSL) protocol was originally developed by Netscape Corporation to provide a mechanism for secure communication over the Internet. Subsequently, the protocol was adopted by the Internet Engineering Task Force (IETF) and renamed to Transport Layer Security (TLS).

The TLS protocol sits between an application protocol layer and a reliable transport layer, such as TCP/IP. It is independent of the application protocol and can thus be layered underneath many different protocols, for example: HTTP, FTP, SMTP, and so on.

Protocol version	Usage recommendation
SSL v2	Do not use. Has serious security vulnerabilities. Removed from the core crypto libraries since RHEL 7.
SSL v3	Do not use. Has serious security vulnerabilities. Removed from the core crypto libraries since RHEL 8.
TLS 1.0	Not recommended to use. Has known issues that cannot be mitigated in a way that guarantees interoperability, and does not support modern cipher suites. Enabled only in the **LEGACY** system-wide cryptographic policy profile.
TLS 1.1	Use for interoperability purposes where needed. Does not support modern cipher suites. Enabled only in the **LEGACY** policy.
TLS 1.2	Supports the modern AEAD cipher suites. This version is enabled in all system-wide crypto policies, but optional parts of this protocol contain vulnerabilities and TLS 1.2 also allows outdated algorithms.
TLS 1.3	Recommended version. TLS 1.3 removes known problematic options, provides additional privacy by encrypting more of the negotiation handshake and can be faster thanks usage of more efficient modern cryptographic algorithms. TLS 1.3 is also enabled in all system-wide crypto policies.

Additional resources

- IETF: The Transport Layer Security (TLS) Protocol Version 1.3

2.2. SECURITY CONSIDERATIONS FOR TLS IN RHEL 8

In RHEL 8, cryptography-related considerations are significantly simplified thanks to the system-wide crypto policies. The **DEFAULT** crypto policy allows only TLS 1.2 and 1.3. To allow your system to negotiate connections using the earlier versions of TLS, you need to either opt out from following crypto policies in an application or switch to the **LEGACY** policy with the **update-crypto-policies** command. See Using system-wide cryptographic policies for more information.

The default settings provided by libraries included in RHEL 8 are secure enough for most deployments. The TLS implementations use secure algorithms where possible while not preventing connections from or to legacy clients or servers. Apply hardened settings in environments with strict security requirements where legacy clients or servers that do not support secure algorithms or protocols are not expected or allowed to connect.

The most straightforward way to harden your TLS configuration is switching the system-wide cryptographic policy level to **FUTURE** using the **update-crypto-policies --set FUTURE** command.

If you decide to not follow RHEL system-wide crypto policies, use the following recommendations for preferred protocols, cipher suites, and key lengths on your custom configuration:

2.2.1. Protocols

The latest version of TLS provides the best security mechanism. Unless you have a compelling reason to include support for older versions of TLS, allow your systems to negotiate connections using at least TLS version 1.2. Note that despite that RHEL 8 supports TLS version 1.3, not all features of this protocol are fully supported by RHEL 8 component. For example, the 0-RTT (Zero Round Trip Time) feature, which reduces connection latency, is not yet fully supported by Apache or Nginx web servers.

2.2.2. Cipher suites

Modern, more secure cipher suites should be preferred to old, insecure ones. Always disable the use of eNULL and aNULL cipher suites, which do not offer any encryption or authentication at all. If at all possible, ciphers suites based on RC4 or HMAC-MD5, which have serious shortcomings, should also be disabled. The same applies to the so-called export cipher suites, which have been intentionally made weaker, and thus are easy to break.

While not immediately insecure, cipher suites that offer less than 128 bits of security should not be considered for their short useful life. Algorithms that use 128 bits of security or more can be expected to be unbreakable for at least several years, and are thus strongly recommended. Note that while 3DES ciphers advertise the use of 168 bits, they actually offer 112 bits of security.

Always give preference to cipher suites that support (perfect) forward secrecy (PFS), which ensures the confidentiality of encrypted data even in case the server key is compromised. This rules out the fast RSA key exchange, but allows for the use of ECDHE and DHE. Of the two, ECDHE is the faster and therefore the preferred choice.

You should also give preference to AEAD ciphers, such as AES-GCM, before CBC-mode ciphers as they are not vulnerable to padding oracle attacks. Additionally, in many cases, AES-GCM is faster than AES in CBC mode, especially when the hardware has cryptographic accelerators for AES.

Note also that when using the ECDHE key exchange with ECDSA certificates, the transaction is even faster than pure RSA key exchange. To provide support for legacy clients, you can install two pairs of certificates and keys on a server: one with ECDSA keys (for new clients) and one with RSA keys (for legacy ones).

2.2.3. Public key length

When using RSA keys, always prefer key lengths of at least 3072 bits signed by at least SHA-256, which is sufficiently large for true 128 bits of security.

> **WARNING**
>
> The security of your system is only as strong as the weakest link in the chain. For example, a strong cipher alone does not guarantee good security. The keys and the certificates are just as important, as well as the hash functions and keys used by the Certification Authority (CA) to sign your keys.

Additional resources

- System-wide crypto policies in RHEL 8 .
- **update-crypto-policies(8)** man page

2.3. HARDENING TLS CONFIGURATION IN APPLICATIONS

In Red Hat Enterprise Linux 8, system-wide crypto policies provide a convenient way to ensure that your applications using cryptographic libraries do not allow known insecure protocols, ciphers, or algorithms.

If you want to harden your TLS-related configuration with your customized cryptographic settings, you can use the cryptographic configuration options described in this section, and override the system-wide crypto policies just in the minimum required amount.

Regardless of the configuration you choose to use, always make sure to mandate that your server application enforces *server-side cipher order* , so that the cipher suite to be used is determined by the order you configure.

2.3.1. Configuring the **Apache HTTP server**

The **Apache HTTP Server** can use both **OpenSSL** and **NSS** libraries for its TLS needs. Red Hat Enterprise Linux 8 provides the **mod_ssl** functionality through eponymous packages:

```
# yum install mod_ssl
```

The **mod_ssl** package installs the **/etc/httpd/conf.d/ssl.conf** configuration file, which can be used to modify the TLS-related settings of the **Apache HTTP Server**.

Install the **httpd-manual** package to obtain complete documentation for the **Apache HTTP Server**, including TLS configuration. The directives available in the **/etc/httpd/conf.d/ssl.conf** configuration file are described in detail in /usr/share/httpd/manual/mod/mod_ssl.html. Examples of various settings are in /usr/share/httpd/manual/ssl/ssl_howto.html.

When modifying the settings in the **/etc/httpd/conf.d/ssl.conf** configuration file, be sure to consider the following three directives at the minimum:

SSLProtocol

Use this directive to specify the version of TLS or SSL you want to allow.

SSLCipherSuite

Use this directive to specify your preferred cipher suite or disable the ones you want to disallow.

SSLHonorCipherOrder

Uncomment and set this directive to **on** to ensure that the connecting clients adhere to the order of ciphers you specified.

For example, to use only the TLS 1.2 and 1.3 protocol:

```
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
```

2.3.2. Configuring the Nginx HTTP and proxy server

To enable TLS 1.3 support in **Nginx**, add the **TLSv1.3** value to the **ssl_protocols** option in the **server** section of the **/etc/nginx/nginx.conf** configuration file:

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ....
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers
    ....
}
```

2.3.3. Configuring the Dovecot mail server

To configure your installation of the **Dovecot** mail server to use TLS, modify the **/etc/dovecot/conf.d/10-ssl.conf** configuration file. You can find an explanation of some of the basic configuration directives available in that file in the /usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt file, which is installed along with the standard installation of **Dovecot**.

When modifying the settings in the **/etc/dovecot/conf.d/10-ssl.conf** configuration file, be sure to consider the following three directives at the minimum:

ssl_protocols

Use this directive to specify the version of TLS or SSL you want to allow or disable.

ssl_cipher_list

Use this directive to specify your preferred cipher suites or disable the ones you want to disallow.

ssl_prefer_server_ciphers

Uncomment and set this directive to **yes** to ensure that the connecting clients adhere to the order of ciphers you specified.

For example, the following line in **/etc/dovecot/conf.d/10-ssl.conf** allows only TLS 1.1 and later:

```
ssl_protocols = !SSLv2 !SSLv3 !TLSv1
```

Additional resources

For more information about TLS configuration and related topics, see the resources listed below.

- **config(5)** man page describes the format of the **/etc/ssl/openssl.conf** configuration file.

- **ciphers(1)** man page includes a list of available **OpenSSL** keywords and cipher strings.
- Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
- Mozilla SSL Configuration Generator can help to create configuration files for **Apache** or **Nginx** with secure settings that disable known vulnerable protocols, ciphers, and hashing algorithms.
- SSL Server Test verifies that your configuration meets modern security requirements.

CHAPTER 3. CONFIGURING A VPN WITH IPSEC

In Red Hat Enterprise Linux 8, a virtual private network (VPN) can be configured using the **IPsec** protocol, which is supported by the **Libreswan** application.

3.1. LIBRESWAN AS AN IPSEC VPN IMPLEMENTATION

In Red Hat Enterprise Linux 8, a Virtual Private Network (VPN) can be configured using the **IPsec** protocol, which is supported by the **Libreswan** application. **Libreswan** is a continuation of the **Openswan** application, and many examples from the **Openswan** documentation are interchangeable with **Libreswan**.

The **IPsec** protocol for a VPN is configured using the Internet Key Exchange (**IKE**) protocol. The terms IPsec and IKE are used interchangeably. An IPsec VPN is also called an IKE VPN, IKEv2 VPN, XAUTH VPN, Cisco VPN or IKE/IPsec VPN. A variant of an IPsec VPN that also uses the Level 2 Tunneling Protocol (**L2TP**) is usually called an L2TP/IPsec VPN, which requires the Optional channel **xl2tpd** application.

Libreswan is an open-source, user-space **IKE** implementation. **IKE** v1 and v2 are implemented as a user-level daemon. The IKE protocol is also encrypted. The **IPsec** protocol is implemented by the Linux kernel, and **Libreswan** configures the kernel to add and remove VPN tunnel configurations.

The **IKE** protocol uses UDP port 500 and 4500. The **IPsec** protocol consists of two protocols:

- Encapsulated Security Payload (**ESP**), which has protocol number 50.
- Authenticated Header (**AH**), which has protocol number 51.

The **AH** protocol is not recommended for use. Users of **AH** are recommended to migrate to **ESP** with null encryption.

The **IPsec** protocol provides two modes of operation:

- **Tunnel Mode** (the default)
- **Transport Mode**.

You can configure the kernel with IPsec without IKE. This is called **Manual Keying**. You can also configure manual keying using the **ip xfrm** commands, however, this is strongly discouraged for security reasons. **Libreswan** interfaces with the Linux kernel using netlink. Packet encryption and decryption happen in the Linux kernel.

Libreswan uses the Network Security Services (**NSS**) cryptographic library. Both **Libreswan** and **NSS** are certified for use with the *Federal Information Processing Standard* (**FIPS**) Publication 140-2.

IMPORTANT

IKE/IPsec VPNs, implemented by **Libreswan** and the Linux kernel, is the only VPN technology recommended for use in Red Hat Enterprise Linux 8. Do not use any other VPN technology without understanding the risks of doing so.

In Red Hat Enterprise Linux 8, **Libreswan** follows **system-wide cryptographic policies** by default. This ensures that **Libreswan** uses secure settings for current threat models including **IKEv2** as a default protocol. See Using system-wide crypto policies for more information.

Libreswan does not use the terms "source" and "destination" or "server" and "client" because IKE/IPsec are peer to peer protocols. Instead, it uses the terms "left" and "right" to refer to end points (the hosts). This also allows you to use the same configuration on both end points in most cases. However, administrators usually choose to always use "left" for the local host and "right" for the remote host. :// included in configuring-a-vpn-with-ipsec

3.2. INSTALLING LIBRESWAN

This procedure describes the steps for installing and starting the **Libreswan** IPsec/IKE VPN implementation.

Prerequisites

- The **AppStream** repository is enabled.

Procedure

1. Install the **libreswan** packages:

```
# yum install libreswan
```

2. If you are re-installing **Libreswan**, remove its old database files:

```
# systemctl stop ipsec
# rm /etc/ipsec.d/*db
```

   To initialize the **NSS** database for **Libreswan** in FIPS mode, you have to enable password protection for it:

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

3. Start the **ipsec** service, and enable the service to be started automatically on boot:

```
# systemctl start ipsec
# systemctl enable ipsec
```

4. Configure the firewall to allow 500 and 4500/UDP ports for the IKE, ESP, and AH protocols by adding the **ipsec** service:

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

3.3. CREATING A HOST-TO-HOST VPN

To configure **Libreswan** to create a host-to-host **IPsec** VPN between two hosts referred to as *left* and *right*, enter the following commands on both of the hosts:

Procedure

1. Generate an RSA key pair on each host:

```
# ipsec newhostkey --output /etc/ipsec.d/hostkey.secrets
```

2. The previous step returned the generated key's **ckaid**. Use that **ckaid** with the following command on *left*, for example:

```
# ipsec showhostkey --left --ckaid 2d3ea57b61c9419dfd6cf43a1eb6cb306c0e857d
```

The output of the previous command generated the **leftrsasigkey=** line required for the configuration. Do the same on the second host (*right*):

```
# ipsec showhostkey --right --ckaid a9e1f6ce9ecd3608c24e8f701318383f41798f03
```

3. In the **/etc/ipsec.d/** directory, create a new **my_host-to-host.conf** file. Write the RSA host keys from the output of the **ipsec showhostkey** commands in the previous step to the new file. For example:

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

4. Start **ipsec**:

```
# ipsec setup start
```

5. Load the connection:

```
# ipsec auto --add mytunnel
```

6. Establish the tunnel:

```
# ipsec auto --up mytunnel
```

7. To automatically start the tunnel when the **ipsec** service is started, add the following line to the connection definition:

```
auto=start
```

3.4. CONFIGURING A SITE-TO-SITE VPN

To create a site-to-site **IPsec** VPN, by joining two networks, an **IPsec** tunnel between the two hosts, is created. The hosts thus act as the end points, which are configured to permit traffic from one or more subnets to pass through. Therefore you can think of the host as gateways to the remote portion of the network.

The configuration of the site-to-site VPN only differs from the host-to-host VPN in that one or more networks or subnets must be specified in the configuration file.

Prerequisites

- A host-to-host VPN is already configured.

Procedure

1. Copy the file with the configuration of your host-to-host VPN to a new file, for example:

```
# cp /etc/ipsec.d/my_host-to-host.conf /etc/ipsec.d/my_site-to-site.conf
```

2. Add the subnet configuration to the file created in the previous step, for example:

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

conn mysubnet6
     also=mytunnel
     leftsubnet=2001:db8:0:1::/64
     rightsubnet=2001:db8:0:2::/64
     auto=start

# the following part of the configuration file is the same for both host-to-host and site-to-site
connections:

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

3.5. CONFIGURING A REMOTE ACCESS VPN

Road warriors are traveling users with mobile clients with a dynamically assigned IP address, such as laptops. The mobile clients authenticate using certificates.

The following example shows configuration for **IKEv2**, and it avoids using the **IKEv1** XAUTH protocol.

On the server:

```
conn roadwarriors
    ikev2=insist
    # Support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    fragmentation=yes
    left=1.2.3.4
    # if access to the LAN is given, enable this, otherwise use 0.0.0.0/0
    # leftsubnet=10.10.0.0/16
    leftsubnet=0.0.0.0/0
    leftcert=gw.example.com
    leftid=%fromcert
    leftxauthserver=yes
    leftmodecfgserver=yes
    right=%any
```

```
    # trust our own Certificate Agency
    rightca=%same
    # pick an IP address pool to assign to remote users
    # 100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
    rightaddresspool=100.64.13.100-100.64.13.254
    # if you want remote clients to use some local DNS zones and servers
    modecfgdns="1.2.3.4, 5.6.7.8"
    modecfgdomains="internal.company.com, corp"
    rightxauthclient=yes
    rightmodecfgclient=yes
    authby=rsasig
    # optionally, run the client X.509 ID through pam to allow/deny client
    # pam-authorize=yes
    # load connection, don't initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

On the mobile client, the road warrior's device, use a slight variation of the previous configuration:

```
conn to-vpn-server
    ikev2=insist
    # pick up our dynamic IP
    left=%defaultroute
    leftsubnet=0.0.0.0/0
    leftcert=myname.example.com
    leftid=%fromcert
    leftmodecfgclient=yes
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
    # rightsubnet=10.10.0.0/16
    rightsubnet=0.0.0.0/0
    fragmentation=yes
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasig
    # allow narrowing to the server’s suggested assigned IP and remote subnet
    narrowing=yes
    # Support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    # Initiate connection
    auto=start
```

3.6. CONFIGURING A MESH VPN

A mesh VPN network, which is also known as an *any-to-any* VPN, is a network where all nodes communicate using **IPsec**. The configuration allows for exceptions for nodes that cannot use **IPsec**. The mesh VPN network can be configured in two ways:

- To require **IPsec**.
- To prefer **IPsec** but allow a fallback to clear-text communication.

Authentication between the nodes can be based on X.509 certificates or on DNS Security Extensions (DNSSEC).

The following procedure uses X.509 certificates. These certificates can be generated using any kind of Certificate Authority (CA) management system, such as the Dogtag Certificate System. Dogtag assumes that the certificates for each node are available in the PKCS #12 format (.p12 files), which contain the private key, the node certificate, and the Root CA certificate used to validate other nodes' X.509 certificates.

Each node has an identical configuration with the exception of its X.509 certificate. This allows for adding new nodes without reconfiguring any of the existing nodes in the network. The PKCS #12 files require a "friendly name", for which we use the name "node" so that the configuration files referencing the friendly name can be identical for all nodes.

Prerequisites

- **Libreswan** is installed, and the **ipsec** service is started on each node.

Procedure

1. On each node, import PKCS #12 files. This step requires the password used to generate the PKCS #12 files:

```
# ipsec import nodeXXX.p12
```

2. Create the following three connection definitions for the **IPsec required** (private), **IPsec optional** (private-or-clear), and **No IPsec** (clear) profiles:

```
# /etc/ipsec.d/mesh.conf
conn clear
 auto=ondemand
 type=passthrough
 authby=never
 left=%defaultroute
 right=%group

conn private
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=drop
 negotiationshunt=drop
 # left
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert
        leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup

conn private-or-clear
 auto=ondemand
 type=transport
 authby=rsasig
```

```
failureshunt=passthrough
negotiationshunt=passthrough
# left
left=%defaultroute
leftcert=nodeXXXX
leftid=%fromcert
        leftrsasigkey=%cert
# right
rightrsasigkey=%cert
rightid=%fromcert
right=%opportunisticgroup
```

3. Add the IP address of the network in the proper category. For example, if all nodes reside in the 10.15.0.0/16 network, and all nodes should mandate **IPsec** encryption:

```
# echo "10.15.0.0/16" >> /etc/ipsec.d/policies/private
```

4. To allow certain nodes, for example, 10.15.34.0/24, to work with and without **IPsec**, add those nodes to the private-or-clear group using:

```
# echo "10.15.34.0/24" >> /etc/ipsec.d/policies/private-or-clear
```

5. To define a host, for example, 10.15.1.2, that is not capable of **IPsec** into the clear group, use:

```
# echo "10.15.1.2/32" >> /etc/ipsec.d/policies/clear
```

The files in the **/etc/ipsec.d/policies** directory can be created from a template for each new node, or can be provisioned using Puppet or Ansible.

Note that every node has the same list of exceptions or different traffic flow expectations. Two nodes, therefore, might not be able to communicate because one requires **IPsec** and the other cannot use **IPsec**.

6. Restart the node to add it to the configured mesh:

```
# systemctl restart ipsec
```

7. Once you finish with the addition of nodes, a **ping** command is sufficient to open an **IPsec** tunnel. To see which tunnels a node has opened:

```
# ipsec trafficstatus
```

3.7. METHODS OF AUTHENTICATION USED IN LIBRESWAN

You can use the following methods for authentication of end points:

- *Pre-Shared Keys* (PSK) is the simplest authentication method. PSKs should consist of random characters and have a length of at least 20 characters. In FIPS mode, PSKs need to comply to a minimum strength requirement depending on the integrity algorithm used. It is recommended not to use PSKs shorter than 64 random characters.

- *Raw RSA keys* are commonly used for static host-to-host or subnet-to-subnet **IPsec** configurations. The hosts are manually configured with each other's public RSA key. This method does not scale well when dozens or more hosts all need to setup **IPsec** tunnels to each other.

- *X.509 certificates* are commonly used for large-scale deployments where there are many hosts that need to connect to a common **IPsec** gateway. A central *certificate authority* (CA) is used to sign RSA certificates for hosts or users. This central CA is responsible for relaying trust, including the revocations of individual hosts or users.

- *NULL authentication* is used to gain mesh encryption without authentication. It protects against passive attacks but does not protect against active attacks. However, since **IKEv2** allows asymmetrical authentication methods, NULL authentication can also be used for internet scale opportunistic IPsec, where clients authenticate the server, but servers do not authenticate the client. This model is similar to secure websites using **TLS**.

Protection against quantum computers

In addition to these authentication methods, you can use the *Postquantum Preshared Keys* (PPK) method to protect against possible attacks by quantum computers. Individual clients or groups of clients can use their own PPK by specifying a (PPKID) that corresponds to an out-of-band configured PreShared Key.

Using **IKEv1** with PreShared Keys provided protection against quantum attackers. The redesign of **IKEv2** does not offer this protection natively. **Libreswan** offers the use of *Postquantum Preshared Keys* (PPK) to protect **IKEv2** connections against quantum attacks.

To enable optional PPK support, add **ppk=yes** to the connection definition. To require PPK, add **ppk=insist**. Then, each client can be given a PPK ID with a secret value that is communicated out-of-band (and preferably quantum safe). The PPK's should be very strong in randomness and not be based on dictionary words. The PPK ID and PPK data itself are stored in **ipsec.secrets**, for example:

```
@west @east : PPKS "user1" "thestringismeanttobearandomstr"
```

The **PPKS** option refers to static PPKs. An experimental function uses one-time-pad based Dynamic PPKs. Upon each connection, a new part of a one-time pad is used as the PPK. When used, that part of the dynamic PPK inside the file is overwritten with zeroes to prevent re-use. If there is no more one-time-pad material left, the connection fails. See the **ipsec.secrets(5)** man page for more information.

> **WARNING**
>
> The implementation of dynamic PPKs is provided as a Technology Preview, and this functionality should be used with caution.

3.8. RELATED INFORMATION

The following resources provide additional information regarding **Libreswan** and the **ipsec** daemon.

Installed documentation

- **ipsec(8)** man page — Describes command options for **ipsec**.

- **ipsec.conf(5)** man page — Contains information on configuring **ipsec**.

- **ipsec.secrets(5)** man page – Describes the format of the **ipsec.secrets** file.
- **ipsec_auto(8)** man page – Describes the use of the **auto** command-line client for manipulating Libreswan IPsec connections established using automatic exchanges of keys.
- **ipsec_rsasigkey(8)** man page – Describes the tool used to generate RSA signature keys.
- **/usr/share/doc/libreswan-*version*/**

Online documentation

https://libreswan.org
: The website of the upstream project.

https://libreswan.org/wiki
: The Libreswan Project Wiki.

https://libreswan.org/man/
: All Libreswan man pages.

CHAPTER 4. CONFIGURING MACSEC

The following section provides information on how to configure **Media Control Access Security** (**MACsec**), which is an 802.1AE IEEE standard security technology for secure communication in all traffic on Ethernet links.

4.1. INTRODUCTION TO MACSEC

Media Access Control Security (**MACsec**, IEEE 802.1AE) encrypts and authenticates all traffic in LANs with the GCM-AES-128 algorithm. **MACsec** can protect not only **IP** but also Address Resolution Protocol (ARP), Neighbor Discovery (ND), or **DHCP**. While **IPsec** operates on the network layer (layer 3) and **SSL** or **TLS** on the application layer (layer 7), **MACsec** operates in the data link layer (layer 2). Combine **MACsec** with security protocols for other networking layers to take advantage of different security features that these standards provide.

4.2. USING MACSEC WITH NMCLI TOOL

This procedure shows how to configure **MACsec** with **nmcli** tool.

Prerequisites

- The **NetworkManager** must be running.
- You already have a 16-byte hexadecimal CAK (**$MKA_CAK**) and a 32-byte hexadecimal CKN (**$MKA_CKN**).

Procedure

```
~]# nmcli connection add type macsec \
  con-name test-macsec+ ifname macsec0 \
  connection.autoconnect no \
  macsec.parent eth0 macsec.mode psk \
  macsec.mka-cak $MKA_CAK \
  macsec.mka-ckn $MKA_CKN

~]# nmcli connection up test-macsec+
```

After this step, the *macsec0* device is configured and can be used for networking.

4.3. USING MACSEC WITH WPA_SUPPLICANT

This procedure shows how to enable **MACsec** with a switch that performs authentication using a pre-shared Connectivity Association Key/CAK Name (CAK/CKN) pair.

Procedure

1. Create a CAK/CKN pair. For example, the following command generates a 16-byte key in hexadecimal notation:

   ```
   ~]$ dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%02x"'
   ```

2. Create the **wpa_supplicant.conf** configuration file and add the following lines to it:

   ```
   ctrl_interface=/var/run/wpa_supplicant
   ```

```
eapol_version=3
ap_scan=0
fast_reauth=1

network={
    key_mgmt=NONE
    eapol_flags=0
    macsec_policy=1

    mka_cak=0011... # 16 bytes hexadecimal
    mka_ckn=2233... # 32 bytes hexadecimal
}
```

Use the values from the previous step to complete the **mka_cak** and **mka_ckn** lines in the **wpa_supplicant.conf** configuration file.

For more information, see the **wpa_supplicant.conf(5)** man page.

3. Assuming you are using *eth0* to connect to your network, start **wpa_supplicant** using the following command:

```
~]# wpa_supplicant -i eth0 -Dmacsec_linux -c wpa_supplicant.conf
```

4.4. RELATED INFORMATION

For more details, see the What’s new in MACsec: setting up MACsec using wpa_supplicant and (optionally) NetworkManager article. In addition, see the MACsec: a different solution to encrypt network traffic article for more information about the architecture of a **MACsec** network, use case scenarios, and configuration examples.

CHAPTER 5. 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.

5.1. GETTING STARTED WITH FIREWALLD

5.1.1. 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.

Additional resources

- **firewalld(1)** man page

5.1.2. 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**
- **firewall-config** tool
- **firewall-cmd** command-line tool
- The RHEL web console

The latter three can only edit the appropriate **NetworkManager** configuration files. If you change the zone of the interface using the web console, **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 default settings of the predefined zones are as follows:

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.

Additional resources

` **firewalld.zone(5)** man page

5.1.3. 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**.

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.

Additional resources

- **firewalld.service(5)** man page

5.2. INSTALLING THE FIREWALL-CONFIG GUI CONFIGURATION TOOL

To use the **firewall-config** GUI configuration tool, install the **firewall-config** package.

Procedure

1. Enter the following command 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.

2. 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**.

5.3. VIEWING THE CURRENT STATUS AND SETTINGS OF FIREWALLD

5.3.1. 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.

Procedure

1. To see the status of the service:

```
# firewall-cmd --state
```

2. 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
```

Additional resources

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 5.3.2, “Viewing current firewalld settings”

5.3.2. Viewing current firewalld settings

5.3.2.1. 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.

5.3.2.2. 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:
```

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
... [trimmed for clarity]
```

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
... [trimmed for clarity]
```

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

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

> **NOTE**
>
> 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.

5.4. STARTING FIREWALLD

Procedure

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

   ```
   # systemctl unmask firewalld
   # systemctl start firewalld
   ```

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

   ```
   # systemctl enable firewalld
   ```

5.5. STOPPING FIREWALLD

Procedure

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

```
# systemctl stop firewalld
```

2. To prevent **firewalld** from starting automatically at system start:

```
# systemctl disable firewalld
```

3. To make sure firewalld is not started by accessing the **firewalld D-Bus** interface and also if other services require **firewalld**:

```
# systemctl mask firewalld
```

5.6. 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.

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
> ```

5.7. CONTROLLING NETWORK TRAFFIC USING FIREWALLD

5.7.1. 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.

Procedure

1. 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
```

5.7.2. 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*.

Procedure

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 ...
[trimmed for clarity]
```

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
```

5.7.3. Controlling traffic with predefined services using GUI

To enable or disable a predefined or custom service:

1. Start the **firewall-config** tool and select the network zone whose services are to be configured.
2. Select the **Services** tab.
3. Select the check box for each type of service you want to trust or clear the check box to block a service.

To edit a service:

1. Start the **firewall-config** tool.
2. Select **Permanent** from the menu labeled **Configuration**. Additional icons and menu buttons appear at the bottom of the **Services** window.
3. 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.

5.7.4. 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.

Procedure

To add a new service in a terminal, use **firewall-cmd**, or **firewall-offline-cmd** in case of not active **firewalld**.

1. Enter the following command to add a new and empty service:

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

2. 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.

3. 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.

5.7.5. 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.

5.7.5.1. 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.

Procedure

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.

5.7.5.2. 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.

Procedure

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
```

5.7.6. Opening ports using GUI

To permit traffic through the firewall to a certain port:

1. Start the **firewall-config** tool and select the network zone whose settings you want to change.
2. Select the **Ports** tab and click the **Add** button on the right-hand side. The **Port and Protocol** window opens.
3. Enter the port number or range of ports to permit.
4. Select **tcp** or **udp** from the list.

5.7.7. Controlling traffic with protocols using GUI

To permit traffic through the firewall using a certain protocol:

1. Start the **firewall-config** tool and select the network zone whose settings you want to change.
2. Select the **Protocols** tab and click the **Add** button on the right-hand side. The **Protocol** window opens.
3. Either select a protocol from the list or select the **Other Protocol** check box and enter the protocol in the field.

5.7.8. Opening source ports using GUI

To permit traffic through the firewall from a certain port:

1. Start the firewall-config tool and select the network zone whose settings you want to change.
2. Select the **Source Port** tab and click the **Add** button on the right-hand side. The **Source Port** window opens.
3. Enter the port number or range of ports to permit. Select **tcp** or **udp** from the list.

5.8. WORKING WITH FIREWALLD 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.

5.8.1. Listing zones

Procedure

1. 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.

2. To see detailed information for all zones:

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

3. To see detailed information for a specific zone:

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

5.8.2. Modifying firewalld settings for a certain zone

The Section 5.7.2, “Controlling traffic with predefined services using CLI” and Section 5.7.5, “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.

Procedure

1. 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
```

5.8.3. 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.

Procedure

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.

5.8.4. 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.

Procedure

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.

5.8.5. 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.

Procedure

1. 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
```

5.8.6. 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.

Procedure
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
```

5.8.7. Zone configuration files

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.

Additional resources

- For more information, see the **firewalld.zone** manual pages.

5.8.8. 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.

Procedure

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>
```

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

5.9.1. 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.

5.9.2. 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.

- To set the source in the current zone:

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

- 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:

Procedure

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
```

5.9.3. Removing a source

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

Procedure

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
```

5.9.4. 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.

Procedure

1. To add a source port:

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

5.9.5. Removing a source port

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

Procedure

1. To remove a source port:

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

5.9.6. 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. The following procedure allows traffic from *192.168.1.0/24* to be able to reach the *HTTP* service while any other traffic is blocked.

Procedure

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
```

5.9.7. 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.

5.9.7.1. 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.

Procedure

1. 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.

5.9.7.2. 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.

Procedure

1. To remove a protocol from a zone:

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

5.10. CONFIGURING IP ADDRESS MASQUERADING

The following procedure describes how to enable IP masquerading on your system. IP masquerading hides individual machines behind a gateway when accessing the Internet.

Procedure

1. 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.

2. To enable IP masquerading, enter the following command as **root**:

```
# firewall-cmd --zone=external --add-masquerade
```

3. 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 --permanent
```

5.11. PORT FORWARDING

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

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

5.11.1. Adding a port to redirect

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.

Prerequisites

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

Procedure

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
```

5.11.2. Redirecting TCP port 80 to port 88 on the same machine

Follow the steps to redirect the TCP port 80 to port 88.

Procedure

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
```

5.11.3. 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, use the following procedure.

Procedure

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
```

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

To remove the port redirection:

Procedure

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
```

5.12. 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.

5.12.1. Listing and blocking ICMP requests

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>
```

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>
```

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 the block inversion 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
```

5.12.2. 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.

5.13. 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
```

5.13.1. Configuring IP set options using CLI

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.

5.14. 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.

5.14.1. Configuring lockdown with using CLI

- 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
  ```

5.14.2. Configuring lockdown whitelist options using CLI

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.

5.14.3. Configuring lockdown whitelist options using 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/libexec/platform-python -s /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.

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 entered 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** are 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 match. If the ***** is not there then the absolute command including arguments must match.

5.15. LOG 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.

5.16. RELATED INFORMATION

The following sources of information provide additional resources regarding **firewalld**.

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**.

Online documentation

- http://www.firewalld.org/ — **firewalld** home page.

CHAPTER 6. GETTING STARTED WITH NFTABLES

The **nftables** framework enables administrators to configure packet-filtering rules used by the Linux kernel firewall.

6.1. 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 RHEL 8, **nftables** serve as the default **firewalld** back end. Although the **nftables** back end 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.

Effect of the modules on the **nftables** ruleset can be observed using the **nft** list ruleset command. Since these tools add tables, chains, and rules to the **nftables** ruleset, be aware that **nftables** rule-set operations, such as the **nft flush ruleset** command, might affect rule sets installed using the formerly separate legacy commands.

To quickly identify which variant of the tool is present, version information has been updated to include the back-end name. In RHEL 8, the nftables-based **iptables** tool prints the following version string:

```
$ iptables --version
iptables v1.8.0 (nf_tables)
```

For comparison, the following version information is printed if legacy iptables tool is present:

```
$ iptables --version
iptables v1.8.0 (legacy)
```

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.

6.2. CONVERTING IPTABLES TO NFTABLES

Red Hat Enterprise Linux 8 provides the **iptables-translate** and **ip6tables-translate** tools to convert the existing **iptables** or **ip6tables** rules into the equivalent ones for **nftables**.

Note that some extensions lack translation support. If such an extension exists, the tool prints the untranslated rule prefixed with the **#** sign. For example:

```
| % iptables-translate -A INPUT -j CHECKSUM --checksum-fill
| nft # -A INPUT -j CHECKSUM --checksum-fill
```

Additionally, users can use the **iptables-restore-translate** and **ip6tables-restore-translate** tools to translate a dump of rules. Note that before that, users can use the **iptables-save** or **ip6tables-save** commands to print a dump of current rules. For example:

```
| % sudo iptables-save >/tmp/iptables.dump
| % iptables-restore-translate -f /tmp/iptables.dump
| # Translated by iptables-restore-translate v1.8.0 on Wed Oct 17 17:00:13 2018
| add table ip nat
| ...
```

For more information and a list of possible options and values, enter the **iptables-translate --help** command.

6.3. RELATED INFORMATION

- 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 back end for **firewalld**.