Red Hat Enterprise Linux 8

安全网络

配置安全网络和网络通信

Last Updated: 2021-09-14

Red Hat Enterprise Linux 8 安全网络

配置安全网络和网络通信

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

此标题可协助管理员保护网络、连接的计算机和网络通信，免受各种攻击。

目录

使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始： master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。如需了解更多详细信息，请参阅 CTO Chris Wright 信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请让我们了解如何改进文档。要做到这一点：

- 关于特定内容的简单评论：
  1. 请确定您使用 *Multi-page HTML* 格式查看文档。另外，确定 **Feedback** 按钮出现在文档页的右上方。
  2. 用鼠标指针高亮显示您想评论的文本部分。
  3. 点在高亮文本上弹出的 **Add Feedback**。
  4. 按照显示的步骤操作。
- 要提交更复杂的反馈，请创建一个 Bugzilla ticket：
  1. 进入 Bugzilla 网站。
  2. 在 Component 中选择 **Documentation**。
  3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
  4. 点 **Submit Bug**。

第 1 章 使用 OPENSSH 的两个系统间使用安全通讯

SSH(Secure Shell)是一种协议，它使用客户端-服务器架构在两个系统之间提供安全通信，并允许用户远程登录服务器主机系统。和其它远程沟通协议，如 FTP 或 Telnet 不同，SSH 会加密登录会话，它会阻止入侵者从连接中收集未加密的密码。

Red Hat Enterprise Linux 8 包括基本的 **OpenSSH** 软件包：常规 **openssh** 软件包、open **ssh-server** 软件包和 **openssh-clients** 软件包。请注意，**OpenSSH** 软件包需要 **OpenSSL** 软件包 **openssl-libs**，它会安装几个重要的加密库来启用 **OpenSSH** 对通讯进行加密。

1.1. SSH 和 OPENSSH

SSH（安全 Shell）是一个登录远程机器并在该机器上执行命令的程序。SSH 协议通过不安全的网络在两个不可信主机间提供安全加密的通讯。您还可以通过安全频道转发 X11 连接和任意 TCP/IP 端口。

当 SSH 协议用于远程 shell 登录或文件复制时，SSH 协议可缓解拦截两个系统之间的通信和特定主机模仿等安全威胁。这是因为 SSH 客户端和服务器使用数字签名来验证其身份。另外，所有客户端和服务器系统之间的沟通都是加密的。

主机密钥验证使用 SSH 协议的主机。主机密钥是首次安装 **OpenSSH** 时或主机第一次引导时自动生成的加密密钥。

OpenSSH 是很多 Linux、UNIX 和类似操作系统支持的 SSH 协议的实现。它包括 OpenSSH 客户端和服务器需要的核心文件。OpenSSH 组件由以下用户空间工具组成：

- **ssh** 是一个远程登录程序（SSH 客户端）
- **sshd** 是一个 **OpenSSH** SSH 守护进程
- **scp** 是一个安全的远程文件复制程序
- **sftp** 是一个安全的文件传输程序
- **ssh-agent** 是用于缓存私钥的身份验证代理
- **ssh-add** 为 **ssh-agent**添加私钥身份
- **ssh-keygen** 生成、管理并转换 **ssh** 验证密钥
- **ssh-copy-id** 是一个在远程 SSH 服务器的 **authorized_keys** 文件中添加本地公钥的脚本
- **ssh-keyscan** - 收集 SSH 公共主机密钥

现有两个 SSH 版本： 版本 1 和较新的版本 2。Red Hat Enterprise Linux 8 中的 **OpenSSH** 套件只支持 SSH 版本 2，其增强的密钥交换算法不会受到版本 1 中已知漏洞的影响。

OpenSSH，作为 RHEL 核心加密子系统之一使用系统范围的加密策略。这样可确保在默认配置中禁用弱密码套件和加密算法。要调整策略，管理员必须使用 **update-crypto-policies** 命令更严格或者更松一些设置，或者手动选择不使用系统范围的加密策略。

OpenSSH 套件使用两组不同的配置文件：用于客户端程序（即 **ssh**、**scp** 和 **sftp**）的配置文件，和用于服务器（**sshd** 守护进程）的配置文件。系统范围的 SSH 配置信息保存在 **/etc/ssh/** 目录中。用户特定的 SSH 配置信息保存在用户主目录中的 **~/.ssh/** 中。有关 OpenSSH 配置文件的详细列表，请查看 **sshd(8)** man page 中的 **FILES** 部分。

其它资源

- 使用 man **-k ssh** 命令列出 **man page**。
- 使用系统范围的加密策略。

1.2. 配置并启动 OPENSSH 服务器

使用以下步骤进行您的环境以及启动 **OpenSSH** 服务器所需的基本配置。请注意，在默认 RHEL 安装后，**sshd** 守护进程已经启动，服务器主机密钥会自动被创建。

先决条件

- 已安装 **openssh-server** 软件包。

流程

1. 在当前会话中启动 **sshd** 守护进程，并在引导时自动启动：

```
# systemctl start sshd
# systemctl enable sshd
```

2. 要为 **/etc/ssh/sshd_config** 配置文件中的 **ListenAddress** 指令指定默认地址 **0.0.0.0**（IPv4）或 **::**（IPv6），并使用较慢的动态网络配置，将 **network-online.target** 目标单元的依赖关系添加到 **sshd.service** 单元文件中。要做到这一点，使用以下内容创建 **/etc/systemd/system/sshd.service.d/local.conf** 文件：

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

3. 查看 **/etc/ssh/sshd_config** 配置文件中的 **OpenSSH** 服务器设置是否满足您的情况要求。
4. 另外，还可通过编辑 **/etc/issue** 文件来更改您的 **OpenSSH** 服务器在客户端验证前显示的欢迎信息，例如：

```
Welcome to ssh-server.example.com
Warning: By accessing this server, you agree to the referenced terms and conditions.
```

确保 **/etc/ssh/sshd_config** 中未注释掉 **Banner** 选项，其值包含 **/etc/issue** ：

```
# less /etc/ssh/sshd_config | grep Banner
Banner /etc/issue
```

请注意：要在成功登录后改变显示的信息，您必须编辑服务器上的 **/etc/motd** 文件。详情请查看 **pam_motd** man page。

5. 重新载入 **systemd** 配置并重启 **sshd** 以应用更改：

```
# systemctl daemon-reload
# systemctl restart sshd
```

验证

1. 检查 **sshd** 守护进程是否正在运行：

```
# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-11-18 14:59:58 CET; 6min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1149 (sshd)
    Tasks: 1 (limit: 11491)
   Memory: 1.9M
   CGroup: /system.slice/sshd.service
           └─1149 /usr/sbin/sshd -D -oCiphers=aes128-ctr,aes256-ctr,aes128-cbc,aes256-cbc -
oMACs=hmac-sha2-256,>

Nov 18 14:59:58 ssh-server-example.com systemd[1]: Starting OpenSSH server daemon...
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on 0.0.0.0 port 22.
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on :: port 22.
Nov 18 14:59:58 ssh-server-example.com systemd[1]: Started OpenSSH server daemon.
```

2. 使用 SSH 客户端连接到 SSH 服务器。

```
# ssh user@ssh-server-example.com
ECDSA key fingerprint is SHA256:dXbaS0RG/UzlTTku8GtXSz0S1++lPegSy31v3L/FAEc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ssh-server-example.com' (ECDSA) to the list of known hosts.

user@ssh-server-example.com's password:
```

其它资源

- ***sshd(8)*** and ***sshd_config(5)*** man pages.

1.3. 为基于密钥的身份验证设置 OPENSSH 服务器

要提高系统安全性，请通过在 OpenSSH 服务器上禁用密码身份验证来强制进行基于密钥的身份验证。

先决条件

- *已安装* ***openssh-server*** *软件包。*
- ***sshd*** *守护进程正在服务器中运行。*

流程

1. *在文本编辑器中打开* ***/etc/ssh/sshd_config*** *配置，例如：*

```
# vi /etc/ssh/sshd_config
```

2. *将* ***PasswordAuthentication*** *选项改为* ***no***:

```
PasswordAuthentication no
```

在新默认安装以外的系统中，检查 **PubkeyAuthentication** 没有被设置，并且将 **ChallengeResponseAuthentication** 指令设为 **no**。如果您要进行远程连接，而不使用控制台或带外访问，在禁用密码验证前测试基于密钥的登录过程。

3. 要在 NFS 挂载的主目录中使用基于密钥的验证，启用 **use_nfs_home_dirs** SELinux 布尔值：

```
# setsebool -P use_nfs_home_dirs 1
```

4. 重新载入 **sshd** 守护进程以应用更改：

```
# systemctl reload sshd
```

其它资源

- ***sshd(8)***, ***sshd_config(5)***, and ***setsebool(8)*** man pages.

1.4. 生成 SSH 密钥对

使用这个流程在本地系统中生成 SSH 密钥对，并将生成的公钥复制到 **OpenSSH** 服务器中。如果正确配置了服务器，您可以在不提供任何密码的情况下登录到 **OpenSSH** 服务器。

重要

如果以 **root** 用户身份完成以下步骤，则只有 **root** 用户可以使用密钥。

流程

1. 为 SSH 协议的版本 2 生成 ECDSA 密钥对：

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/joesec/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joesec/.ssh/id_ecdsa.
Your public key has been saved in /home/joesec/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joesec@localhost.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++        |
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+.  .  . .    |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

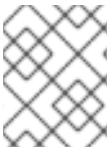
您还可以通过输入 **ssh-keygen -t ed25519** 命令，在 **ssh-keygen** 命令或 Ed25519 密钥对中使用 **-t rsa** 选项生成 RSA 密钥对。

2. 要将公钥复制到远程机器中：

```
$ ssh-copy-id joesec@ssh-server-example.com
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
joesec@ssh-server-example.com's password:
...
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'joesec@ssh-server-example.com'" and check to make sure that only the key(s) you wanted were added.
```

如果您没有在会话中使用 **ssh-agent** 程序，上一个命令会复制最新修改的 **~/.ssh/id*.pub** 公钥。要指定另一个公钥文件，或在 **ssh-agent** 内存中缓存的密钥优先选择文件中的密钥，使用带有 **-i** 选项的 **ssh-copy-id** 命令。



注意

如果重新安装您的系统并希望保留之前生成的密钥对，备份 **~/.ssh/** 目录。重新安装后，将其复制到主目录中。您可以为系统中的所有用户（包括 **root** 用户）进行此操作。

验证

1. 在不提供任何密码的情况下登录到 OpenSSH 服务器：

```
$ ssh joesec@ssh-server-example.com
Welcome message.
...
Last login: Mon Nov 18 18:28:42 2019 from ::1
```

其它资源

- **ssh-keygen(1)和 ssh-copy-id(1)** man page.

1.5. 使用保存在智能卡中的SSH 密钥

Red Hat Enterprise Linux 可让您使用保存在 OpenSSH 客户端智能卡中的 RSA 和 ECDSA 密钥。使用这个步骤使用智能卡而不是使用密码启用验证。

先决条件

- 在客户端中安装了 **opensc** 软件包，**pcscd** 服务正在运行。

流程

1. 列出所有由 OpenSC PKCS #11 模块提供的密钥，包括其 PKCS #11 URIs，并将输出保存到 key.pub 文件：

```
$ ssh-keygen -D pkcs11: > keys.pub
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3NzaC1yc2E...KKZMzcQZzx
pkcs11:id=%02;object=SIGN%20pubkey;token=SSH%20key;manufacturer=piv_II?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

```
ecdsa-sha2-nistp256 AAA...J0hkYnnsM=
pkcs11:id=%01;object=PIV%20AUTH%20pubkey;token=SSH%20key;manufacturer=piv_II?
module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

2. 要使用远程服务器上的智能卡（example.com）启用验证，将公钥传送到远程服务器。使用带有上一步中创建的 key.pub 的 **ssh-copy-id** 命令：

```
$ ssh-copy-id -f -i keys.pub username@example.com
```

3. 要使用在第 1 步的 **ssh-keygen -D** 命令输出中的 ECDSA 密钥连接到 example.com，您只能使用 URI 中的一个子集，它是您的密钥的唯一参考，例如：

```
$ ssh -i "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so" example.com
Enter PIN for 'SSH key':
[example.com] $
```

4. 您可以使用 **~/.ssh/config** 文件中的同一 URI 字符串使配置持久：

```
$ cat ~/.ssh/config
IdentityFile "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so"
$ ssh example.com
Enter PIN for 'SSH key':
[example.com] $
```

因为 OpenSSH 使用 **p11-kit-proxy** wrapper 和 OpenSC PKCS #11 模块注册到 PKCS#11 Kit，所以您可以简化前面的命令：

```
$ ssh -i "pkcs11:id=%01" example.com
Enter PIN for 'SSH key':
[example.com] $
```

如果您跳过 PKCS #11 URI 的 **id=** 部分，则 OpenSSH 会加载代理模块中可用的所有密钥。这可减少输入所需的数量：

```
$ ssh -i pkcs11: example.com
Enter PIN for 'SSH key':
[example.com] $
```

其它资源

- Fedora 28：OpenSSH 中更出色的智能卡支持.
- **p11-kit(8)**、**opensc.conf(5)**、**pcd(8)**、**ssh(1)和 ssh -keygen(1)** 手册页.

1.6. 使 OPENSSH 更安全

以下提示可帮助您在使用 OpenSSH 时提高安全性。请注意，**/etc/ssh/sshd_config** OpenSSH 配置文件的更改需要重新载入 **sshd** 守护进程才能生效：

```
# systemctl reload sshd
```

重要

大多数安全强化配置更改降低了与不支持最新算法或密码套件的客户端的兼容性。

禁用不安全的连接协议

- 要使 SSH 生效，防止使用由 **OpenSSH** 套件替代的不安全连接协议。否则，用户的密码可能只会在一个会话中被 SSH 保护，可能会在以后使用 Telnet 登录时被捕获。因此，请考虑禁用不安全的协议，如 telnet、rsh、rlogin 和 ftp。

启用基于密钥的身份验证并禁用基于密码的身份验证

- 禁用密码验证并只允许密钥对可减少安全攻击面，还可节省用户的时间。在客户端中，使用 **ssh-keygen** 工具生成密钥对，并使用 **ssh-copy-id** 工具从 **OpenSSH** 服务器的客户端复制公钥。要在 OpenSSH 服务器中禁用基于密码的验证，请编辑 **/etc/ssh/sshd_config**，并将 **PasswordAuthentication** 选项改为 **no**：

```
PasswordAuthentication no
```

密钥类型

- 虽然 **ssh-keygen** 命令会默认生成一组 RSA 密钥，但您可以使用 **-t** 选项指定它生成 ECDSA 或者 Ed25519 密钥。ECDSA(Elliptic Curve Digital Signature Algorithm)能够以等效的对称密钥强度比 RSA 提供更好的性能。它还会生成较短的密钥。Ed25519 公钥算法是 TWisted Edwards 曲线的实现，其安全性也比 RSA、DSA 和 ECDSA 更快。
  如果没有这些密钥，OpenSSH 会自动创建 RSA、ECDSA 和 Ed25519 服务器主机密钥。要在 RHEL 8 中配置主机密钥创建，使用 **sshd-keygen@.service** 实例化服务。例如，禁用自动创建 RSA 密钥类型：

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

- 要排除 SSH 连接的特定密钥类型，注释 **/etc/ssh/sshd_config** 中的相关行，并重新载入 **sshd** 服务。例如，只允许 Ed25519 主机密钥：

```
# HostKey /etc/ssh/ssh_host_rsa_key
# HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```

非默认端口

- 默认情况下，**sshd** 守护进程侦听 TCP 端口 22。更改此端口可降低系统因自动网络扫描而受到攻击的风险，并可以提高安全性。您可以使用 **/etc/ssh/sshd_config** 配置文件中的 **Port** 指令指定端口。
  您还必须更新默认 SELinux 策略以允许使用非默认端口。要做到这一点，使用 **policycoreutils-python-utils** 软件包中的 **semanage** 工具：

```
# semanage port -a -t ssh_port_t -p tcp port_number
```

另外，更新 **firewalld** 配置：

```
# firewall-cmd --add-port port_number/tcp
# firewall-cmd --runtime-to-permanent
```

在前面的命令中，将 port_number 替换为使用 **Port** 指令指定的新端口号。

没有 root 登录

- 如果您的特定用例不需要作为 root 用户登录，应该考虑在 **/etc/ssh/sshd_config** 文件中将 **PermitRootLogin** 配置指令设置为 **no**。通过禁止以 root 用户身份登录，管理员可以审核哪些用户在以普通用户身份登录后运行哪些特权命令，然后获得 root 权限。
  或者，将 **PermitRootLogin** 设置为 **prohibit-password**：

```
PermitRootLogin prohibit-password
```

这强制使用基于密钥的身份验证，而不使用密码以 root 身份登录，并通过防止暴力攻击来降低风险。

使用 X 安全扩展

- Red Hat Enterprise Linux 客户端中的 X 服务器不提供 X 安全性扩展。因此，当连接到带有 X11 转发的不可信 SSH 服务器时，客户端无法请求另一个安全层。大多数应用程序都无法在启用此扩展时运行。
  默认情况下，**/etc/ssh/ssh_config.d/05-redhat.conf** 文件中的 **ForwardX 11Trusted** 选项被设置为 **yes**，且 **ssh -X remote_machine**（不信任主机）和 **ssh -Y remote_machine**（可信主机）命令之间没有区别。

  如果您的场景根本不需要 X11 转发功能，请将 **/etc/ssh/sshd_config** 配置文件中的 **X11Forwarding** 指令设置为 **no**。

限制对特定用户、组群或者域的访问

- **/etc/ssh/sshd_config** 配置文件服务器中的 **AllowUsers** 和 **AllowGroups** 指令可让您只允许某些用户、域或组连接到您的 OpenSSH 服务器。您可以组合 **AllowUsers 和 Allow Groups** 来更精确地限制访问，例如：

```
AllowUsers *@192.168.1.*,*@10.0.0.*,!*@192.168.1.2
AllowGroups example-group
```

以上配置行接受来自 192.168.1.* 和 10.0.0.* 子网中所有用户的连接，但具有 192.168.1.2 地址的系统除外。所有用户都必须位于 **example-group** 组。OpenSSH 服务器拒绝所有其他连接。

请注意，使用允许列表（以 Allow 开头的指令）比使用 blocklists 更安全（以 Deny 开始的选项），因为 allowlists 也会阻止新的未授权用户或组。

更改系统范围的加密策略

- **OpenSSH** 使用 RHEL 系统范围的加密策略，默认的系统范围的加密策略级别为当前威胁模型提供了安全设置。要使您的加密设置更严格，请更改当前的策略级别：

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

- 要为您的 **OpenSSH** 服务器选择不使用系统范围的加密策略，请使用 **/etc/sysconfig/sshd** 文件中的 **CRYPTO_POLICY=** 变量取消注释这一行。更改后，您在 **/etc/ssh/sshd_config** 文件中的 **Ciphers**、**MAC**、**KexAlgoritm s 和 GSSAPIKexAlgorithms** 部分指定的值不会被覆盖。请注意，此任务在配置加密选项时需要深入了解。

- 如需更多信息，请参阅 RHEL 8 安全强化 文档中的使用系统范围的加密策略。

其它资源

- **sshd_config(5)**、**ssh-keygen(1)**、 crypto **-policies(7)** 和 **update-crypto-policies(8)man** page.

1.7. 使用 SSH 跳过主机连接到远程服务器

使用这个步骤通过中间服务器（也称为跳过主机）将本地系统连接到远程服务器。

先决条件

- 跳过主机接受来自本地系统的 SSH 连接。
- 远程服务器只接受来自跳过主机的 SSH 连接。

流程

1. 通过编辑本地系统中的 **~/.ssh/config** 文件来定义跳过主机，例如：

```
Host jump-server1
  HostName jump1.example.com
```

   - **Host** 参数定义您可以在 **ssh** 命令中使用的主机的名称或别名。该值可以匹配真实的主机名，但也可以是任意字符串。
   - **HostName** 参数设置跳过主机的实际主机名或 IP 地址。

2. 使用 **ProxyJump** 指令将远程服务器跳过配置添加到本地系统上的 **~/.ssh/config** 文件中，例如：

```
Host remote-server
  HostName remote1.example.com
  ProxyJump jump-server1
```

3. 使用您的本地系统通过跳过服务器连接到远程服务器：

```
$ ssh remote-server
```

   如果省略了配置步骤 1 和 2，则上一命令等同于 **ssh -J skip-server1 remote-server** 命令。

注意

您可以指定更多跳过服务器，您也可以在提供其完整主机名时跳过在配置文件中添加主机定义，例如：

```
$ ssh -J jump1.example.com,jump2.example.com,jump3.example.com
remote1.example.com
```

如果跳过服务器上的用户名或 SSH 端口与远程服务器上的名称和端口不同，请更改上一命令中的主机名或 SSH 端口，例如：

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@jump3.ex
ample.com:75 joesec@remote1.example.com:220
```

其它资源

- ***ssh_config(5)*** *和* ***ssh(1)*** *手册页.*

1.8. 使用 SSH-AGENT 使用 SSH 密钥连接到远程机器

为了避免每次发起 SSH 连接时输入密语，您可以使用 ***ssh-agent*** *实用程序缓存 SSH 私钥。私钥和密语保持安全。*

先决条件

- *您有一个远程主机正在运行 SSH 守护进程，并可通过网络访问。*
- *您知道登录到远程主机的 IP 地址或者主机名以及凭证。*
- *您已用密码生成了 SSH 密钥对，并将公钥传送到远程机器。如需更多信息，请参阅生成 SSH 密钥对。*

流程

1. *可选：验证您可以使用密钥在远程主机中进行身份验证：*
   a. *使用 SSH 连接到远程主机：*

   ```
   $ ssh example.user1@198.51.100.1 hostname
   ```

   b. *输入您在创建密钥时设定的密码短语以授予对私钥的访问权限。*

   ```
   $ ssh example.user1@198.51.100.1 hostname
    host.example.com
   ```

2. *启动* ***ssh-agent***。

   ```
   $ eval $(ssh-agent)
   Agent pid 20062
   ```

3. *将密钥添加到* ***ssh-agent***。

```
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for ~/.ssh/id_rsa:
Identity added: ~/.ssh/id_rsa (example.user0@198.51.100.12)
```

验证

- *可选：使用 SSH 登录到主机机器。*

```
$ ssh example.user1@198.51.100.1

Last login: Mon Sep 14 12:56:37 2020
```

请注意您不必输入密码短语。

1.9. 其它资源

- ***sshd(8)*、*ssh(1)*、*sftp(1)*、*sftp(1)*、*ssh* *-keygen(1)*、*ssh -copy-id(1)*、*ssh _config(5)*、*ssh _config(5)*、*update* *-crypto-policies(8)和 crypto-policies(7)* *man page.***
- *OpenSSH 主页.*
- *为使用非标准配置的应用程序和服务配置 SELinux。*
- *使用 firewalld 控制网络流量.*

第 2 章 配置与 SSH 系统角色的安全通信

作为管理员，您可以使用 SSHD 系统角色配置 SSH 服务器和 SSH 系统角色，通过使用 Red Hat Ansible Automation Platform 在任意数量的 RHEL 系统上一致地配置 SSH 客户端。

2.1. SSHD 系统角色变量

在 SSHD 系统角色 playbook 中，您可以根据您的偏好和限制定义 SSH 配置文件的参数。

如果没有配置这些变量，系统角色将生成与 RHEL 默认值匹配的 **sshd_config** 文件。

在所有情况下，布尔值在 **sshd** 配置中都正确呈现为 **yes** 和 **no**。您可以使用 list 定义多行配置项。例如：

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

呈现为：

```
ListenAddress 0.0.0.0
ListenAddress ::
```

SSHD 系统角色的变量

sshd_enable

如果设置为 **False**，则角色将被完全禁用。默认值为 **True**。

sshd_skip_defaults

如果设置为 **True**，则系统角色不会应用默认值。相反，您可以使用 sshd dict 或 **sshd _Key** 变量指定完整的配置默认值集合。默认值为 **False**。

sshd_manage_service

如果设置为 **False**，则服务不会被管理，这意味着它不会在引导时启用，也不会启动或重新加载。除非在容器内或 AIX 中运行，否则默认为 **True**，因为 Ansible 服务模块目前不支持 **AIX**。

sshd_allow_reload

如果设置为 **False**，则 **sshd** 不会在配置更改后重新加载。这可帮助进行故障排除。要应用更改后的配置，请手动重新加载 **sshd**。默认为与 **sshd_manage_service** 相同的值，但 AIX 除外，其中 **sshd_manage_service** 默认为 **False**，但 **sshd_allow_reload** 默认为 **True**。

sshd_install_service

如果设置为 **True**，该角色将安装 **sshd** 服务的服务文件。这会覆盖操作系统中提供的文件。除非您要配置第二个实例，而且您还要更改 **sshd_service** 变量，否则不要设置为 **True**。默认值为 **False**。
该角色使用以下变量指向的文件作为模板：

```
sshd_service_template_service (default: templates/sshd.service.j2)
sshd_service_template_at_service (default: templates/sshd@.service.j2)
sshd_service_template_socket (default: templates/sshd.socket.j2)
```

sshd_service

此变量更改 **sshd** 服务名称，这对于配置第二个 **sshd** 服务实例非常有用。

sshd

包含配置的字典。例如：

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

sshd_OptionName

您可以使用由 **sshd_** 前缀和选项名称而不是 dict 组成的简单变量来定义选项。简单的变量覆盖 **sshd** 字典中的值。例如：

```
sshd_Compression: no
```

sshd_match 和 sshd_match_1 到 sshd_match_9

Match 部分的字典或字典列表。请注意，这些变量不会覆盖 **sshd** 字典中定义的匹配块。所有源都会反映在生成的配置文件中。

SSHD 系统角色的辅助变量

您可以使用这些变量覆盖与每个支持的平台对应的默认值。

sshd_packages

您可以使用此变量覆盖安装的软件包的默认列表。

sshd_config_owner、sshd_config_group 和 sshd_config_mode

您可以使用这些变量设置此角色 **生成的 openssh** 配置文件的所有权和权限。

sshd_config_file

此角色保存生成的 **openssh** 服务器配置的路径。

sshd_binary

openssh 的 **sshd** 可执行文件的路径。

sshd_service

sshd 服务的名称。默认情况下，此变量包含目标平台使用的 **sshd** 服务名称。当角色使用 **sshd _install_service** 变量时，**您还可以使用它来设置自定义 sshd** 服务的名称。

sshd_verify_hostkeys

默认值为 **auto**。当设置为 **auto** 时，这将列出生成的配置文件中存在的所有主机密钥，并生成所有不存在的路径。此外，权限和文件所有者被设置为默认值。如果部署阶段使用该角色来确保服务能够在第一次尝试时启动，这非常有用。若要禁用此检查，可将此变量设置为空列表 **[]**。

sshd_hostkey_owner,sshd_hostkey_group,sshd_hostkey_mode

使用这些变量设置来自 **sshd_verify_hostkeys** 的主机密钥的所有权和权限。

sshd_sysconfig

在基于 RHEL 的系统上，这个变量配置 **sshd** 服务的详情。如果设置为 **true，则** 此角色还根据以下配置管理 **/etc/sysconfig/sshd** 配置文件：默认值为 **false**。

sshd_sysconfig_override_crypto_policy

在 RHEL 8 中，当设置为 **true** 时，这个变量会覆盖系统范围的加密策略。默认值为 **false**。

sshd_sysconfig_use_strong_rng

在基于 RHEL 的系统上，此变量可以强制 **sshd** 重新密封 **openssl** 随机数字生成器，并将字节数指定为 参数。默认值为 **0**，它会禁用此功能。如果系统没有硬件随机数字生成器，请不要打开此项。

2.2. 使用 SSHD 系统角色配置 OPENSSH 服务器

您可以通过运行 Ansible playbook，使用 SSHD 系统角色来配置多个 SSH 服务器。

先决条件

- 访问一个或多个 受管节点，它们是您要使用 SSHD 系统角色配置的系统。
- 对控制节点的访问和权限，这是 Red Hat Ansible Engine 配置其他系统的系统。
  在控制节点上：
  - 已安装 Red Hat Ansible Engine。
  - 已安装 **rhel-system-roles** 软件包。
  - 列出受管节点的清单文件。

流程

1. 为 SSHD 系统角色复制示例 playbook:

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml path/custom-playbook.yml
```

2. 使用文本编辑器打开复制的 playbook，例如：

```
# vim path/custom-playbook.yml

---
- hosts: all
  tasks:
  - name: Configure sshd to prevent root and password login except from particular subnet
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd:
        # root login and password login is enabled only from a particular subnet
        PermitRootLogin: no
        PasswordAuthentication: no
        Match:
        - Condition: "Address 192.0.2.0/24"
          PermitRootLogin: yes
          PasswordAuthentication: yes
```

playbook 将受管节点配置为 SSH 服务器，以便：

- 禁用密码和 **root** 用户登录
- 仅从子网 **192.0.2.0/24** 中启用密码和 **root** 用户登录

您可以根据您的偏好修改变量。如需了解更多详细信息，请参阅 SSHD 服务器系统角色变量。

3. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file path/custom-playbook.yml

...

PLAY RECAP
**************************************************

localhost : ok=12 changed=2 unreachable=0 failed=0
skipped=10 rescued=0 ignored=0
```

验证

1. *登录到SSH 服务器：*

```
$ ssh user1@10.1.1.1
```

其中：

- ***user1** 是SSH 服务器上的用户。*
- ***10.1.1.1** 是SSH 服务器的IP 地址。*

2. *检查SSH 服务器上的 **sshd_config** 文件的内容：*

```
$ vim /etc/ssh/sshd_config

# Ansible managed
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
AuthorizedKeysFile .ssh/authorized_keys
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
PasswordAuthentication no
PermitRootLogin no
PrintMotd no
Subsystem sftp /usr/libexec/openssh/sftp-server
SyslogFacility AUTHPRIV
UsePAM yes
X11Forwarding yes
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

3. *检查您是否可以以root 用户身份从 **192.0.2.0/24** 子网连接到服务器：*

   a. *确定您的IP 地址：*

```
$ hostname -I
192.0.2.1
```

如果 IP 地址在 **192.0.2.1** - **192.0.2.254** 范围内，您可以连接到服务器。

b. 以 **root** 用户身份连接到服务器：

```
$ ssh root@10.1.1.1
```

其它资源

- **/usr/share/doc/rhel-system-roles/sshd/README.md** file.
- **ansible-playbook(1)** man page.

2.3. SSH 系统角色变量

在 SSH 系统角色 playbook 中，您可以根据您的偏好和限制定义客户端 SSH 配置文件的参数。

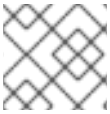
如果没有配置这些变量，系统角色将生成与 RHEL 默认值匹配的全局 **ssh_config** 文件。

在所有情况下，布尔值在 **ssh** 配置中都正确呈现为 **yes** 或 **no**。您可以使用 list 定义多行配置项。例如：

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

呈现为：

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```



注意

配置选项区分大小写。

SSH 系统角色的变量

ssh_user

您可以定义一个现有用户名，供系统角色修改用户特定的配置。用户特定配置保存在给定用户的 **~/.ssh/config** 中。默认值为 null，它会修改所有用户的全局配置。

ssh_skip_defaults

默认值为 **auto**。如果设置为 **auto**，系统角色将写入系统范围的配置文件 **/etc/ssh/ssh_config**，并保留在其中定义的 RHEL 默认值。通过定义 **ssh_drop_in_name 变量创建置入** 配置文件，将自动禁用 **ssh_skip_defaults** 变量。

ssh_drop_in_name

定义置入配置文件的名称，该文件放置在系统范围的置入目录中。该名称在模板 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 中使用，以引用要修改的配置文件。如果系统不支持置入目录，则默认值为 null。如果系统支持置入目录，则默认值为 **00-ansible**。

警告

如果系统不支持置入目录，设置此选项将使 play 失败。

建议的格式是 **NN-name**，其中 **NN** 是用于订购配置文件和 **名称** 的两位数字，是内容或文件所有者的任何描述性名称。

ssh

包含配置选项和对应的值的字典。

ssh_OptionName

您可以使用由 **ssh_** 前缀和选项名称而不是 dict 组成的简单变量来定义选项。简单的变量覆盖 **ssh** 字典中的值。

ssh_additional_packages

此角色会自动安装 **openssh** 和 **openssh-clients** 软件包，这是最常见的用例所需要的。如果您需要安装其他软件包，例如 **openssh-keysign** 以用于基于主机的身份验证，您可以在此变量中指定它们。

ssh_config_file

角色保存配置文件的路径。默认值：

- 如果系统有一个置入目录，则默认值由模板 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 定义。
- 如果系统没有置入目录，则默认值为 **/etc/ssh/ssh_config**。
- 如果定义了 **ssh_user** 变量，则默认值为 **~/.ssh/config**。

ssh_config_owner,ssh_config_group,ssh_config_mode

创建的配置文件的所有者、组和模式。默认情况下，文件的所有者是 **root:root**，模式是 **0644**。如果定义了 **ssh_user**，则模式为 **0600**，所有者和组派生自 **ssh_user** 变量中指定的用户名。

2.4. 使用 SSH 系统角色配置 OPENSSH 客户端

您可以通过运行 Ansible playbook，使用 SSH 系统角色来配置多个 SSH 客户端。

先决条件

- 访问一个或多个 受管节点，它们是您要使用 SSH 系统角色配置的系统。
- 对控制节点的访问和权限，这是 Red Hat Ansible Engine 配置其他系统的系统。
  在控制节点上：
  - 已安装 Red Hat Ansible Engine。
  - 已安装 **rhel-system-roles** 软件包。
  - 列出受管节点的清单文件。

流程

1. 使用以下内容创建新 **playbook.yml** 文件：

```
---
- hosts: all
  tasks:
  - name: "Configure ssh clients"
    include_role:
      name: rhel-system-roles.ssh
    vars:
      ssh_user: root
      ssh:
        Compression: true
        GSSAPIAuthentication: no
        ControlMaster: auto
        ControlPath: ~/.ssh/.cm%C
        Host:
          - Condition: example
            Hostname: example.com
            User: user1
      ssh_ForwardX11: no
```

此 playbook 使用以下配置在受管节点上配置 **root** 用户的 SSH 客户端首选项：

- 压缩已启用。
- ControlMaster 多路复用设置为 **auto**。
- 连接到 **example.com** 主机的示例 别名 是 **user1**。
- 已创建 **示例** 主机别名，它表示与 **example.com** 主机的 连接 **使用 user1** 用户名。
- X11 转发被禁用。

另外，您还可以根据您的偏好修改这些变量。如需了解更多详细信息，请参阅 SSH 客户端角色变量。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

验证

- 通过在文本编辑器中打开 SSH 配置文件来验证受管节点是否具有正确的配置，例如：

```
# vi ~root/.ssh/config
```

在应用了上面显示的示例 playbook 后，配置文件应包含以下内容：

```
# Ansible managed
Compression yes
ControlMaster auto
```

```
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

第 3 章 计划并使用 TLS

TLS（传输层安全）是用来保护网络通信的加密协议。在通过配置首选密钥交换协议、身份验证方法和加密算法来强化系统安全设置时，需要记住支持的客户端的范围越宽，进而降低由此产生的安全性。相反，严格的安全设置会导致与客户端的兼容性受限，这可能导致某些用户被锁定在系统之外。务必以最严格的可用配置为目标，且仅在出于兼容性原因需要时才放松。

3.1. SSL 和 TLS 协议

安全套接字层(SSL)协议最初由 Netscape Corporation 开发，以提供通过互联网进行安全通信的机制。因此，该协议被互联网工程任务组(IETF)采纳，并重命名为传输层安全(TLS)。

TLS 协议位于应用协议层和可靠的传输层，如 TCP/IP。它独立于应用程序协议,因此可在很多不同的协议下分层，如 HTTP、FTP、SMTP 等等。

协议版本	用法建议
SSL v2	不要使用。具有严重的安全漏洞。从 RHEL 7 开始从核心加密库中删除。
SSL v3	不要使用。具有严重的安全漏洞。从 RHEL 8 开始的核心加密库中删除。
TLS 1.0	不建议使用。有无法以保证互操作性的方式缓解的已知问题，且不支持现代密码套件。仅在 **系统范围的加密策略** 配置集中启用。
TLS 1.1	在需要时用于互操作性.不支持现代加密套件。仅在 cri **ACY 策略** 中启用。
TLS 1.2	支持现代 AEAD 密码组合。此版本在所有系统范围的加密策略中启用，但此协议的可选部分包含漏洞，TLS 1.2 也允许过时的算法。
TLS 1.3	推荐的版本。TLS 1.3 删除了已知有问题的选项，通过加密更多协商握手来提供额外的隐私，由于使用了更有效的现代加密算法，可以更快地提供隐私。在所有系统范围的加密策略中也启用了 TLS 1.3。

其它资源

- IETF：传输层安全(TLS)协议版本 1.3.

3.2. RHEL 8 中 TLS 的安全注意事项

在 RHEL 8 中，由于系统范围的加密策略，与加密相关的注意事项显著简化。The **DEFAULT** 加密策略仅允许 TLS 1.2 和 1.3。要允许您的系统使用早期版本的 TLS 协商连接，您需要选择不使用应用程序中的以下加密策略，或使用 **update-crypto-policies** 命令切换到 cri **ACY** 策略。如需更多信息，请参阅使用系统范围的加密策略。

RHEL 8 中包含的库提供的默认设置足以满足大多数部署的需要。TLS 实施尽可能使用安全算法，而不阻止与旧客户端或服务器的连接。在具有严格安全要求的环境中应用强化设置，使不支持安全算法或协议的旧客户端或服务器不预期或允许连接。

强化 TLS 配置的最简单方法是使用 **update-crypto-policies --set FUTURE 命令将系统范围的加密策略级别切换到 FUTURE**。

如果您决定不遵循 RHEL 系统范围的加密策略，请在自定义配置中使用以下建议进行首选协议、密码套件和密钥长度：

3.2.1. 协议

TLS 的最新版本提供了最佳安全机制。除非有充分的理由包含对旧版本的 TLS 的支持，请允许您的系统使用至少 TLS 版本 1.2 协商连接。请注意，虽然 RHEL 8 支持 TLS 版本 1.3，但 RHEL 8 组件并不完全支持此协议的所有功能。例如，Apache 或 Nginx web 服务器尚不完全支持 0-RTT(Zero R Trip Time)功能，它可降低连接延迟。

3.2.2. 密码套件

现代、更安全的密码套件应该优先于旧的不安全密码套件。总是禁用 eNULL 和 aNULL 密码套件的使用，它们根本不提供任何加密或身份验证。如果有可能，基于 RC4 或 HMAC-MD5 的密码套件也必须被禁用。这同样适用于所谓的出口密码套件，其本意为较弱，因此容易中断。

虽然不会立即变得不安全，但提供安全性少于 128 位的密码套件在它们的短使用期中不应该被考虑。使用 128 位或者更高安全性的算法可以预期在至少数年内不会被破坏，因此我们强烈推荐您使用此算法。请注意，虽然 3DES 密码公告使用 168 位但它们实际只提供了 112 位的安全性。

始终优先使用支持(perfect)转发保密(PFS)的密码套件，这样可确保加密数据的机密性，即使在服务器密钥泄露时也是如此。此规则退出了快速 RSA 密钥交换，但允许使用 ECDHE 和 DHE。在两者中，ECDHE 是速度更快，因此是首选。

您还应该在 CBC 模式密码之前优先使用 AEAD 密码，如 AES-GCM，因为它们不会受到 padding oracle 攻击。此外，在很多情况下，AES-GCM 比 CBC 模式的 AES 快，特别是在硬件为 AES 加密加速器的情况下。

另请注意，在使用 ECDSA 证书的 ECDHE 密钥交换时，事务的速度甚至要快于纯 RSA 密钥交换。为了支持旧客户端，您可以在服务器上安装两对证书和密钥：一台带有 ECDSA 密钥（用于新客户端），另一个用于 RSA 密钥（用于旧密钥）。

3.2.3. 公钥长度

在使用 RSA 密钥时，总是首选使用至少由 SHA-256 签名的 3072 位的密钥长度，对于真实的 128 位安全性来说，这个值已经足够大。

警告

您的系统安全性仅与链中最弱的连接相同。例如，只是一个强大的密码不能保证良好安全性。密钥和证书以及认证机构(CA)用来签署您的密钥的哈希功能和密钥同样重要。

其它资源

- RHEL 8 中的系统范围的加密策略。
- **update-crypto-policies(8)** man page。

3.3. 在应用程序中强化 TLS 配置

在 Red Hat Enterprise Linux 8 中，系统范围的加密策略 提供了一种便捷的方法，可确保使用加密库的应用程序不允许已知的不安全协议、密码或算法。

如果要使用自定义加密设置强化与 TLS 相关的配置，您可以使用本节中描述的加密配置选项，并在最小数量时覆盖系统范围的加密策略。

无论您选择使用什么配置，请始终确保您的服务器应用程序强制实施服务器端密码顺序，以便使用的密码套件由您配置的顺序决定。

3.3.1. 配置 Apache HTTP 服务器

Apache HTTP 服务器 可以使用 **OpenSSL** 和 **NSS** 库 来满足其 TLS 的需求。Red Hat Enterprise Linux 8 通过 eponymous 软件包提供 **mod_ssl** 功能：

```
# yum install mod_ssl
```

mod_ssl 软件包将安装 **/etc/httpd/conf.d/ssl.conf** 配置文件，该文件可用于修改 **Apache HTTP 服务器的** 与 TLS 相关的设置。

安装 **httpd-manual** 软件包以获取 **Apache HTTP 服务器** 的完整文档，包括 TLS 配置。**/etc/httpd/conf.d/ssl.conf** 配置文件中的指令在 /usr/share/httpd/manual/mod_ssl.html 中有详细介绍。各种设置示例位于 /usr/share/httpd/manual/ssl/ssl_howto.html。

修改 **/etc/httpd/conf.d/ssl.conf** 配置文件中的设置时，请确定至少考虑以下三个指令：

SSLProtocol

使用这个指令指定您要允许的 TLS 或者 SSL 版本。

SSLCipherSuite

使用这个指令指定首选的密码套件或禁用您要禁止的密码套件。

SSLHonorCipherOrder

取消注释并将此指令设置为 **on**，以确保连接的客户端遵循您指定的密码顺序。

例如，只使用 TLS 1.2 和 1.3 协议：

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

如需更多信息，请参阅 部署不同类型的 服务器文档中的在 Apache HTTP 服务器上配置 TLS 加密 一章。

3.3.2. 配置 Nginx HTTP 和代理服务器

要在 **Nginx** 中启用 TLS 1.3 支持，将 **TLSv1.3** 值添加到 **/etc/nginx/nginx.conf** 配置文件的 **server** 部分的 **ssl_protocols** 选项：

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

如需更多信息，请参阅 部署不同类型的服务器 文档中的 向 Nginx web 服务器添加 TLS 加密 一章。

3.3.3. 配置 Dovecot 邮件服务器

要将 **Dovecot** 邮件服务器的安装配置为使用 TLS，请修改 **/etc/dovecot/conf.d/10-ssl.conf** 配置文件。您可以在 /usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt 文件中找到该文件中提供的一些基本配置指令的说明，该文件与 **Dovecot** 的标准安装一同安装。

修改 **/etc/dovecot/conf.d/10-ssl.conf** 配置文件中的设置时，请确保至少考虑以下三个指令：

ssl_protocols

使用这个指令指定您要允许或者禁用的 TLS 或者 SSL 版本。

ssl_cipher_list

使用这个指令指定首选的密码套件或禁用您要禁止的密码套件。

ssl_prefer_server_ciphers

取消注释并将此指令设置为 **yes**，以确保连接的客户端遵循您指定的密码顺序。

例如，**/etc/dovecot/conf.d/10-ssl.conf** 中的以下行只允许 TLS 1.1 及之后的版本：

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

其它资源

- 在 RHEL 8 中部署不同类型的服务器
- **config(5)** 和 **ciphers(1)** man page.
- 安全使用传输层安全性(TLS)和数据报传输层安全(DTLS)的建议。
- Mozilla SSL 配置生成器.
- SSL 服务器测试.

第 4 章 使用 IPSEC 配置 VPN

在 Red Hat Enterprise Linux 8 中，可以使用 **IPsec** 协议配置虚拟专用网络(VPN)，该协议受到 **Libreswan** 应用程序的支持。

4.1. LIBRESWAN 作为 IPSEC VPN 的实现

在 Red Hat Enterprise Linux 8 中，可以使用 **IPsec** 协议配置虚拟专用网络(VPN)，该协议受到 **Libreswan** 应用程序的支持。**Libreswan** 是 **Openswan** 应用的延续，Openswan **文档中的**许多示例可与 **Libreswan** 互换。

VPN 的 **IPsec** 协议使用互联网密钥交换(**IKE**)协议进行配置。术语 IPsec 和 IKE 可互换使用。IPsec VPN 也称为 IKE VPN、IKEv2 VPN、XAUTH VPN、Cisco VPN 或 IKE/IPsec VPN。使用级别 2 隧道协议(**L2TP**)的 IPsec VPN 变体通常称为 L2TP/IPsec VPN，这需要可选通道 **xl2tpd** 应用程序。

Libreswan 是一种开源用户空间 **IKE** 实施。**IKE** v1 和 v2 作为用户级守护进程实施。IKE 协议也加密。**IPsec** 协议由 Linux 内核实施，**Libreswan** 配置内核以添加和删除 VPN 隧道配置。

TheIKE 协议使用 UDP 端口 500 和 4500。**IPsec** 协议由两个协议组成：

- 封装的安全支付(**ESP**)，其协议编号为 50。
- 经过身份验证的标头(**AH**)，其协议编号为 51。

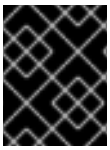
不建议使用 **AH** 协议。建议 **AH** 用户迁移到使用 null 加密的 **ESP**。

IPsec 协议提供了两种操作模式：

- **隧道模式** （默认）
- **传输模式.**

您可以用没有 IKE 的 IPsec 来配置内核。这称为 **手动密钥。**您还可以使用 **ip xfrm** 命令配置手动密钥，但为了安全起见，强烈建议您这样做。**Libreswan** 使用 netlink 与 Linux 内核接口.在 Linux 内核中进行数据包加密和解密。

libreswan 使用网络安全服务(**NSS**)加密库。**Libreswan** 和 **NSS** 均通过 联邦信息处理标准 (**FIPS**)出版物 140-2 认证。



重要

由 **Libreswan** 和 Linux 内核实施的 **IKE/IPsec** VPN 是 Red Hat Enterprise Linux 8 中推荐使用的唯一 VPN 技术。在不了解这样做风险的情况下不要使用任何其他 VPN 技术。

在 Red Hat Enterprise Linux 8 中，**Libreswan** 默认遵循 **系统范围的加密策略**。这样可确保 **Libreswan** 将安全设置用于当前威胁模型，包括 **IKEv2** 作为默认协议。如需更多信息，请参阅使用系统范围的加密策略。

Libreswan 不使用术语"来源"和"目标"或"服务器"和"客户端"，因为 IKE/IPsec 对等协议。相反，它使用术语"左"和"right"指点（主机）。这也允许您在大多数情况下在两个端点使用相同的配置。但是，管理员通常选择始终对本地主机使用"左转"，而"右"用于远程主机。

4.2. 安装 LIBRESWAN

这个步骤描述了安装和启动 **Libreswan** IPsec/IKE VPN 实现的步骤。

先决条件

- **AppStream**存储库已启用。

流程

1. 安装 **libreswan** 软件包：

```
# yum install libreswan
```

2. 如果您要重新安装 **Libreswan**，请删除其旧数据库文件：

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

3. 启动 **ipsec** 服务，并在引导时启用自动启动该服务：

```
# systemctl enable ipsec --now
```

4. 通过添加 **ipsec** 服务，将防火墙配置为允许 IKE、ESP 和 AH 协议的 500 和 4500/UDP 端口：

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

4.3. 创建主机到主机的 VPN

要将 **Libreswan** 配置为在两个主机（称为 *左* 和 *右*）之间创建一个主机到主机的 **IPsec** VPN，请在两台主机上输入以下命令：

流程

1. 在每个主机上生成 RSA 密钥对：

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

2. 上一步返回生成的密钥's **ckaid**。将该 **ckaid** 与 左面的以下命令一起使用，例如：

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

上一命令的输出生成了配置所需的 **leftrsasigkey=** 行。在第二主机上执行相同的操作（右侧）：

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

3. 在 **/etc/ipsec.d/** 目录中，创建一个新的 **my_host-to-host.conf** 文件。将上一步中 **ipsec showhostkey** 命令的输出中的 RSA 主机密钥写入新文件。例如：

```
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. 导入密钥后，重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

5. 启动 **Libreswan** ：

```
# ipsec setup start
```

6. 加载连接：

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

7. 建立隧道：

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

8. 要在 **ipsec** 服务启动时自动启动隧道，请在连接定义中添加以下行：

```
auto=start
```

4.4. 配置站点到站点的 VPN

要创建一个站点到站点 **IPsec** VPN（加入两个网络），在两个主机之间创建一个 **IPsec** 隧道。主机因此充当端点，它们配置为允许来自一个或多个子网的流量通过。因此您可以将主机视为到网络远程部分的网关。

站点到站点 VPN 的配置只能与主机到主机 VPN 不同，同时必须在配置文件中指定一个或多个网络或子网。

先决条件

- 已配置了主机到主机的 VPN。

流程

1. 使用主机到主机的 VPN 配置将该文件复制到新文件中，例如：

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

2. 在上一步创建的文件中添加子网配置，例如：

```
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
```

4.5. 配置远程访问 VPN

路战员利用动态分配的 IP 地址（如笔记本电脑）移动客户。使用证书进行移动客户端验证。

以下示例显示了 **IKEv2** 的配置，并且避免使用 **IKEv1** XAUTH 协议。

在服务器中：

```
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
```

在移动客户端（即路径战器设备）上，使用之前配置的稍有变化：

```
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
```

4.6. 配置网格 VPN

网格 VPN 网络（也称为 任意 VPN） 是一个网络，所有节点都使用 **IPsec** 进行通信。该配置允许在无法使用 **IPsec** 的节点中出现例外。可使用两种方式配置网格 VPN 网络：

- 需要 **IPsec**。
- 首选 **IPsec**，但允许回退清除文本通信。

节点之间的身份验证可以基于 X.509 证书或 DNS 安全扩展(DNSSEC)。

以下流程使用 X.509 证书。这些证书可以使用任何类型的证书颁发机构(CA)管理系统生成，如 Dogtag 证书系统。Dogtag 假定每个节点的证书可用 PKCS #12 格式（.p12 文件），其中包含私钥、节点证书和用于验证其他节点的 X.509 证书的 Root CA 证书。

每个节点的配置与其 X.509 证书不同。这允许在不重新配置网络中的任何现有节点的情况下添加新节点。PKCS #12 文件需要一个"友好名称"，对于它，我们使用名称"node"，以便引用友好名称的配置文件对所有节点都相同。

先决条件

- **Libreswan** 已安装，在每个节点上启动 **ipsec** 服务。

流程

1. 在每个节点中导入 PKCS #12 文件。此步骤需要用于生成 PKCS #12 文件的密码：

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

2. 为 **需要** （专用）、IPsec **可选** (private-or-clear)和 **No IPsec** (clear)配置集创建以下三个连接定义：

```
# cat /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. 以适当的类别添加网络的 IP 地址。例如，如果所有节点都驻留在 10.15.0.0/16 网络中，所有节点都应强制进行 **IPsec** 加密：

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

4. 要允许某些节点（如 10.15.34.0/24）使用或不使用 **IPsec，请使用以下** 方法将这些节点添加到 private-or-clear 组中：

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

5. 要定义 **IPsec** 无法到清除组中的主机（如 10.15.1.2），请使用：

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

/etc/ipsec.d/policies 目录中的文件可以从每个新节点的模板创建，也可以使用 Puppet 或 Ansible 来调配。

请注意，每个节点都有相同的异常列表或不同流量流预期的列表。因此，两个节点可能无法通信，因为一个节点需要 **IPsec**，另一个节点无法使用 **IPsec**。

6. 重启节点将其添加到配置的网格中：

```
# systemctl restart ipsec
```

7. 完成添加节点后，**ping** 命令就足以打开 **IPsec** 隧道。查看节点已打开的隧道：

```
# ipsec trafficstatus
```

4.7. LIBRESWAN 中使用的验证方法

您可以使用以下方法验证端点：

- 预共享密钥 (PSK) 是最简单的验证方法。PSK 应该由随机字符组成，长度至少为 20 个字符。在 FIPS 模式中，PSK 需要根据所使用的完整性算法满足最低强度要求。建议您不要使用小于 64 个随机字符的 PSK。
- 原始 RSA 密钥 通常用于静态主机到主机或子网到子网 **IPsec** 配置。主机使用其他的公共 RSA 密钥进行手动配置。当 dozens 或更多主机都需要相互设置 **IPsec** 隧道时，此方法无法很好地扩展。
- X.509 证书 通常用于大型部署，其中有很多主机需要连接到通用 **IPsec** 网关。中央 证书颁发机构（CA）用于为主机或用户签名 RSA 证书。此中央 CA 负责转发信任关系，包括单个主机或用户的吊销。
- NULL 身份验证 用于在没有身份验证的情况下获得网格加密。它可防止被动攻击，但不会防止主动攻击。但是，由于 **IKEv2** 允许非对称身份验证方法，因此 NULL 身份验证也可用于互联网扩展机会 IPsec，其中客户端验证服务器，但服务器不验证客户端的身份验证。此模型类似于使用 **TLS** 的安全网站。

保护量子计算机

除了这些验证方法外，您还可以使用 Postquantum Preshared Keys (PPK) 方法来防止 Quantum 计算机可能受到的攻击。单个客户端或客户端组可以通过指定与带外配置的 PreShared 密钥对应的 (PPKID) 来使用自己的 PPK。

使用带预共享密钥的 **IKEv1** 可为量级攻击者提供保护。重新设计 **IKEv2** 不原生提供这种保护。**Libreswan** 提供使用 Postquantum Preshared Keys (PPK) 来保护 **IKEv2** 连接免受量子攻击。

要启用可选的 PPK 支持，请在连接定义中添加 **ppk=yes**。要需要 PPK，请添加 **ppk=insist**。然后，可为每个客户端分配一个带有一个 secret 值的 PPK ID，其 secret 值会被传递到带外（最好是使用半字节安全）。PPK 的随机性应该非常强大，而且不能基于字典的单词。PPK ID 和 PPK 数据本身存储在 **ipsec.secrets** 中，例如：

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

PPKS 选项指的是静态 PPK。实验性功能使用基于单时间的 Dynamic PPK。在每个连接中，会将一次性 pad 的新部分用作 PPK。当使用时，文件中的该部分动态 PPK 被零覆盖，以防止重复使用。如果没有剩下一次性资源，连接会失败。详情请查看 **ipsec.secrets(5)** man page。

警告

动态 PPK 的实现是作为技术预览提供的，这个功能应该小心使用。

4.8. 部署 FIPS 兼容 IPSEC VPN

使用此流程基于 Libreswan 部署 FIPS 兼容 IPsec VPN 解决方案。以下步骤还允许您识别哪些加密算法可用，并在 FIPS 模式的 Libreswan 中禁用了哪些加密算法。

先决条件

- **AppStream** 存储库已启用。

流程

1. 安装 **libreswan** 软件包：

```
# yum install libreswan
```

2. 如果您要重新安装 **Libreswan**，请删除其旧的 NSS 数据库：

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

3. 启动 **ipsec** 服务，并在引导时启用自动启动该服务：

```
# systemctl enable ipsec --now
```

4. 通过添加 **ipsec** 服务，将防火墙配置为允许 IKE、ESP 和 AH 协议的 500 和 4500/UDP 端口：

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

5. 在 RHEL 8 中将系统切换到 FIPS 模式：

```
# fips-mode-setup --enable
```

6. 重启您的系统以允许内核切换到 FIPS 模式：

```
# reboot
```

验证

验证

1. 确认 Libreswan 在 FIPS 模式下运行：

```
# ipsec whack --fipsstatus
000 FIPS mode enabled
```

2. 或者，检查 **systemd** 日志中的 **ipsec** 单元条目：

```
$ journalctl -u ipsec
...
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Product: YES
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Kernel: YES
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Mode: YES
```

3. 以 FIPS 模式查看可用算法：

```
# ipsec pluto --selftest 2>&1 | head -11
FIPS Product: YES
FIPS Kernel: YES
FIPS Mode: YES
NSS DB directory: sql:/etc/ipsec.d
Initializing NSS
Opening NSS database "sql:/etc/ipsec.d" read-only
NSS initialized
NSS crypto library initialized
FIPS HMAC integrity support [enabled]
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity verification self-test passed
```

4. 使用 FIPS 模式查询禁用的算法：

```
# ipsec pluto --selftest 2>&1 | grep disabled
Encryption algorithm CAMELLIA_CTR disabled; not FIPS compliant
Encryption algorithm CAMELLIA_CBC disabled; not FIPS compliant
Encryption algorithm SERPENT_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_SSH disabled; not FIPS compliant
Encryption algorithm NULL disabled; not FIPS compliant
Encryption algorithm CHACHA20_POLY1305 disabled; not FIPS compliant
Hash algorithm MD5 disabled; not FIPS compliant
PRF algorithm HMAC_MD5 disabled; not FIPS compliant
PRF algorithm AES_XCBC disabled; not FIPS compliant
Integrity algorithm HMAC_MD5_96 disabled; not FIPS compliant
Integrity algorithm HMAC_SHA2_256_TRUNCBUG disabled; not FIPS compliant
Integrity algorithm AES_XCBC_96 disabled; not FIPS compliant
DH algorithm MODP1024 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant
```

5. 在 FIPS 模式中列出所有允许的算法和密码：

```
# ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
{256,192,*128}  aes_ccm, aes_ccm_c
```

```
{256,192,*128}  aes_ccm_b
{256,192,*128}  aes_ccm_a
[*192]  3des
{256,192,*128}  aes_gcm, aes_gcm_c
{256,192,*128}  aes_gcm_b
{256,192,*128}  aes_gcm_a
{256,192,*128}  aesctr
{256,192,*128}  aes
{256,192,*128}  aes_gmac
sha, sha1, sha1_96, hmac_sha1
sha512, sha2_512, sha2_512_256, hmac_sha2_512
sha384, sha2_384, sha2_384_192, hmac_sha2_384
sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
null, dh0
dh14
dh15
dh16
dh17
dh18
ecp_256, ecp256
ecp_384, ecp384
ecp_521, ecp521
```

其它资源

- 使用系统范围的加密策略。

4.9. 使用密码保护 IPSEC NSS 数据库

默认情况下，IPsec 服务在第一次启动时使用空密码创建其网络安全服务(NSS)数据库。使用以下命令添加密码保护。

注意

在 RHEL 6.6 和之前的版本中，您必须使用一个密码来保护 IPsec NSS 数据库，以满足 FIPS 140-2 标准的要求，因为 NSS 加密库是针对 FIPS 140-2 Level 2 标准认证的。在 RHEL 8 中，NIS 将 NSS 认证为标准级别 1，并且该状态不需要对数据库进行密码保护。

先决条件

- **/etc/ipsec.d** 目录包含 NSS 数据库文件。

流程

1. 为 **Libreswan** 的 **NSS** 数据库启用密码保护：

```
# certutil -N -d sql:/etc/ipsec.d
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
```

```
and should contain at least one non-alphabetic character.

Enter new password:
```

2. 创建包含您在上一步中设置的密码的 **/etc/ipsec.d/nsspassword** 文件，例如：

```
# cat /etc/ipsec.d/nsspassword
NSS Certificate DB:MyStrongPasswordHere
```

请注意，**thensspassword** 文件使用以下语法：

```
token_1_name:the_password
token_2_name:the_password
```

默认 NSS 软件令牌是 **NSS 证书 DB**。如果您的系统以 FIPS 模式运行，则令牌的名称为 **NSS FIPS 140-2 证书数据库**。

3. 根据您的场景，在完成后启动或重启 **ipsec 服务** ：

```
# systemctl restart ipsec
```

验证

1. 检查 **ipsec** 服务在在其 NSS 数据库中添加非空密码后是否正在运行：

```
# systemctl status ipsec
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
   Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor preset: disable>
   Active: active (running)...
```

2. （可选）检查 **Journal** 日志是否包含确认成功初始化的条目：

```
# journalctl -u ipsec
...
pluto[23001]: NSS DB directory: sql:/etc/ipsec.d
pluto[23001]: Initializing NSS
pluto[23001]: Opening NSS database "sql:/etc/ipsec.d" read-only
pluto[23001]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate
DB" with length 20 passed to NSS
pluto[23001]: NSS crypto library initialized
...
```

其它资源

- **certutil(1)** 手册页.
- 政府标准 知识库文章.

4.10. 配置 IPSEC VPN 以使用 TCP

Libreswan 支持 IKE 和 IPsec 数据包的 TCP 封装，如 RFC 8229 所述。使用这个功能，您可以在网络上建立 IPsec VPN，以防止通过 UDP 传输的流量并封装安全负载(ESP)。您可以将 VPN 服务器和客户端配置为使用 TCP 作为回退，或者作为主 VPN 传输协议。由于 TCP 封装的性能成本较高，因此只有在您的场

景中永久阻止 UDP 时，才使用 TCP 作为主 VPN 协议。

先决条件

- *已配置了 远程访问 VPN。*

流程

1. *在 **config setup** 部分的 **/etc/ipsec.conf** 文件中添加以下选项：*

```
listen-tcp=yes
```

2. *要在第一次尝试 UDP 失败时使用 TCP 封装作为回退选项，请在客户端的连接定义中添加以下两个选项：*

```
enable-tcp=fallback
tcp-remoteport=4500
```

另外，如果您知道 UDP 被永久阻止，请在客户端的连接配置中使用以下选项：

```
enable-tcp=yes
tcp-remoteport=4500
```

其它资源

- *IETF RFC 8229：IKE 和 IPsec Packet 的 TCP 封装.*

4.11. 在绑定中配置 ESP 硬件卸载以加快 IPSEC 连接

将安全负载(ESP)卸载到硬件可加快 IPsec 连接。如果您使用网络绑定作为故障切换的原因，配置 ESP 硬件卸载的要求和步骤与使用常规以太网设备的不同。例如，在这种情况下，您可以在绑定中启用卸载支持，内核会将设置应用到绑定的端口。

先决条件

- *绑定中的所有网卡都支持 ESP 硬件卸载。*
- *网络驱动程序支持绑定设备中的 ESP 硬件卸载。在 RHEL 中，只有 **ixgbe** 驱动程序支持此功能。*
- *绑定已配置且正常工作。*
- *该绑定使用 **active-backup** 模式。绑定驱动程序不支持此功能的任何其他模式。*
- *IPsec 连接已配置且可以正常工作。*

流程

1. *在网络绑定中启用 ESP 硬件卸载支持：*

```
# nmcli connection modify bond0 ethtool.feature-esp-hw-offload on
```

*这个命令在 **bond0** 连接中启用 ESP 硬件卸载支持。*

2. 重新激活 **bond0** 连接：

```
# nmcli connection up bond0
```

3. 编辑应使用 ESP 硬件卸载的连接的 **/etc/ipsec.d/** 目录中的 Libreswan 配置文件，并将 **nic-offload=yes** 语句附加到连接条目：

```
conn example
    ...
    nic-offload=yes
```

4. 重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

验证

1. 显示绑定的活动端口：

```
# grep "Currently Active Slave" /proc/net/bonding/bond0
Currently Active Slave: enp1s0
```

2. 显示活跃端口的 **tx_ipsec** 和 **rx_ipsec** 计数器：

```
# ethtool enp1s0 | egrep "_ipsec"
     tx_ipsec: 10
     rx_ipsec: 10
```

3. 通过 IPsec 隧道发送流量。例如，ping 远程 IP 地址：

```
# ping -c 5 remote_ip_address
```

4. 再次显示活跃端口的 **tx _ipsec 和rx_ipsec** 计数器：

```
# ethtool enp1s0 | egrep "_ipsec"
     tx_ipsec: 15
     rx_ipsec: 15
```

如果计数器值增加，ESP 硬件卸载可以正常工作。

其它资源

- 配置网络绑定
- 使用 IPsec 配置 VPN
- 在保护网络文档中使用 IPsec 配置 VPN 章节。

4.12. 配置选择不使用系统范围的加密策略的 IPSEC 连接

为连接覆盖系统范围的加密策略

RHEL 系统范围的加密策略会创建一个名为 **%default** 的特殊连接。此连接包含 **ikev2**、**esp** 和 **ike** 选项的默认值。但是，您可以通过在连接配置文件中指定上述选项来覆盖默认值。

例如，以下配置允许使用 IKEv1 和 AES 和 SHA-1 或 SHA-2 的连接，以及使用 AES-GCM 或 AES-CBC 的 IPsec(ESP)：

```
conn MyExample
  ...
  ikev2=never
  ike=aes-sha2,aes-sha1;modp2048
  esp=aes_gcm,aes-sha2,aes-sha1
  ...
```

请注意，AES-GCM 可用于 IPsec(ESP) 和 IKEv2，但不适用于 IKEv1。

为所有连接禁用系统范围的加密策略

要禁用所有 IPsec 连接的系统范围的加密策略，在 **/etc/ipsec.conf** 文件中注释掉以下行：

```
include /etc/crypto-policies/back-ends/libreswan.config
```

然后将 **ikev2=never** 选项添加到连接配置文件。

其它资源

- 使用系统范围的加密策略。

4.13. IPSEC VPN 配置故障排除

与 IPsec VPN 配置相关的问题通常是由于几个主要原因造成的。如果您遇到此类问题，您可以检查问题的原因是否与以下任何情况对应，并应用相应的解决方案。

基本连接故障排除

VPN 连接的大多数问题发生在新部署中，管理员在新部署中配置了不匹配配置选项的端点。此外，由于新引入的不兼容值，工作配置可能会突然停止工作。这可能是管理员更改配置的结果。或者，管理员可能已安装了固件更新，或者使用特定选项的不同默认值（如加密算法）安装了软件包更新。

要确认已建立 IPsec VPN 连接：

```
# ipsec trafficstatus
006 #8: "vpn.example.com"[1] 192.0.2.1, type=ESP, add_time=1595296930, inBytes=5999,
outBytes=3231, id='@vpn.example.com', lease=100.64.13.5/32
```

如果输出为空或者没有显示具有连接名称的条目，则隧道将断开。

检查连接中的问题：

1. 重新载入 vpn.example.com 连接：

```
# ipsec auto --add vpn.example.com
002 added connection description "vpn.example.com"
```

2. 下一步，启动 VPN 连接：

```
# ipsec auto --up vpn.example.com
```

与防火墙相关的问题

最常见的问题是，其中一个 IPsec 端点或端点之间的路由器上的防火墙是丢弃所有互联网密钥交换(IKE)数据包。

- 对于 IKEv2，类似以下示例的输出说明防火墙出现问题：

```
# ipsec auto --up vpn.example.com
181 "vpn.example.com"[1] 192.0.2.2 #15: initiating IKEv2 IKE SA
181 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: sent v2I1, expected v2R1
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 0.5
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 1
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 2
seconds for
...
```

- 对于 IKEv1，启动命令的输出如下：

```
# ipsec auto --up vpn.example.com
002 "vpn.example.com" #9: initiating Main Mode
102 "vpn.example.com" #9: STATE_MAIN_I1: sent MI1, expecting MR1
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 0.5 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 1 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 2 seconds for
response
...
```

由于用于设置 IPsec 的 IKE 协议已经加密，您只能使用 **tcpdump** 工具排除一小部分问题。如果防火墙丢弃 IKE 或 IPsec 数据包，您可以尝试使用 **tcpdump** 实用程序查找原因。但是， **tcp** dump 无法通过 IPsec VPN 连接诊断其他问题。

- 捕获 VPN 协商以及 **eth0** 接口上的所有加密数据：

```
# tcpdump -i eth0 -n -n esp or udp port 500 or udp port 4500 or tcp port 4500
```

不匹配的算法、协议和策略

VPN 连接要求端点具有匹配的 IKE 算法、IPsec 算法和 IP 地址范围。如果发生不匹配，连接会失败。如果您使用以下方法之一发现不匹配，请通过匹配算法、协议或策略来修复它。

- 如果远程端点没有运行 IKE/IPsec，您可以看到一个 ICMP 数据包来指示它。例如：

```
# ipsec auto --up vpn.example.com
...
000 "vpn.example.com"[1] 192.0.2.2 #16: ERROR: asynchronous network error report on
wlp2s0 (192.0.2.2:500), complainant 198.51.100.1: Connection refused [errno 111, origin
ICMP type 3 code 3 (not authenticated)]
...
```

- *不匹配 IKE 算法示例：*

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com"[1] 193.110.157.148 #3: dropping unexpected IKE_SA_INIT message containing NO_PROPOSAL_CHOSEN notification; message payloads: N; missing payloads: SA,KE,Ni
```

- *不匹配 IPsec 算法示例：*

```
# ipsec auto --up vpn.example.com
...
182 "vpn.example.com"[1] 193.110.157.148 #5: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2 cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_256 group=MODP2048}
002 "vpn.example.com"[1] 193.110.157.148 #6: IKE_AUTH response contained the error notification NO_PROPOSAL_CHOSEN
```

不匹配的 IKE 版本还可导致远程端点在没有响应的情况下丢弃请求。这与丢弃所有 IKE 数据包的防火墙相同。

- *IKEv2 不匹配的 IP 地址范围示例（称为流量选择器 - TS）：*

```
# ipsec auto --up vpn.example.com
...
1v2 "vpn.example.com" #1: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2 cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_512 group=MODP2048}
002 "vpn.example.com" #2: IKE_AUTH response contained the error notification TS_UNACCEPTABLE
```

- *IKEv1 的不匹配 IP 地址范围示例：*

```
# ipsec auto --up vpn.example.com
...
031 "vpn.example.com" #2: STATE_QUICK_I1: 60 second timeout exceeded after 0 retransmits.  No acceptable response to our first Quick Mode message: perhaps peer likes no proposal
```

- *当在 IKEv1 中使用 PreSharedKeys(PSK)时，如果两端没有放入同一个 PSK 中，整个 IKE 信息将变得不可读取：*

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com" #1: received Hash Payload does not match computed value
223 "vpn.example.com" #1: sending notification INVALID_HASH_INFORMATION to 192.0.2.23:500
```

- *在 IKEv2 中，不匹配-PSK 错误会导致 AUTHENTICATION_FAILED 信息：*

```
# ipsec auto --up vpn.example.com
...
002 "vpn.example.com" #1: IKE SA authentication request rejected by peer: AUTHENTICATION_FAILED
```

最大传输单元

除阻止 IKE 或 IPsec 数据包的防火墙外，联网问题的最常见原因是加密数据包的数据包大小增加。大于最大传输单元(MTU)的网络硬件片段，例如 1500 字节。通常，片段会丢失，数据包无法重新集合。当使用小数据包但其他流量失败的 ping 测试时，这会导致间歇性失败。在这种情况下，您可以建立 SSH 会话，但在使用时立即停止终端，例如，通过在远程主机上输入 'ls -al /usr' 命令。

要临时解决这个问题，请通过将 **mtu=1400** 选项添加到隧道配置文件来减小 MTU 大小。

另外，对于 TCP 连接，启用更改 MSS 值的 iptables 规则：

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

如果上一命令没有解决您的场景中的问题，请在 **set-mss** 参数中直接指定较小的大小：

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1380
```

网络地址转换(NAT)

当 IPsec 主机也充当 NAT 路由器时，可能会意外重新映射数据包。以下示例配置演示了这个问题：

```
conn myvpn
    left=172.16.0.1
    leftsubnet=10.0.2.0/24
    right=172.16.0.2
    rightsubnet=192.168.0.0/16
…
```

地址为 172.16.0.1 的系统有 NAT 规则：

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

如果地址 10.0.2.33 上的系统将数据包发送到 192.168.0.1，则路由器会在应用 IPsec 加密前将源 10.0.2.33 转换为 172.16.0.1。

然后，源地址为 10.0.2.33 的数据包不再与 **conn myvpn** 配置匹配，并且 IPsec 不会加密此数据包。

要解决这个问题，插入排除路由器中目标 IPsec 子网范围的 NAT 的规则，在本例中为：

```
iptables -t nat -I POSTROUTING -s 10.0.2.0/24 -d 192.168.0.0/16 -j RETURN
```

内核 IPsec 子系统错误

例如，当错误导致 IKE 用户空间和 IPsec 内核取消同步时，内核 IPsec 子系统可能会失败。检查此问题：

```
$ cat /proc/net/xfrm_stat
XfrmInError               0
XfrmInBufferError         0
...
```

上一命令的输出中的任何非零值都表示问题。如果您遇到这个问题，请打开一个新 支持问题单，并将上一命令的输出与对应的 IKE 日志连接。

libreswan 日志

默认情况下，**Libreswan** 日志使用 **syslog** 协议。您可以使用 **journalctl** 命令查找与 IPsec 相关的日志条目。因为与日志对应的条目由 **pluto** IKE 守护进程发送，因此请搜索 "pluto" 关键字，例如：

```
$ journalctl -b | grep pluto
```

显示 **ipsec** 服务的实时日志：

```
$ journalctl -f -u ipsec
```

如果默认日志记录级别没有显示您的配置问题，请通过将 **plutodebug=all** 选项添加到 **/etc/ipsec.conf** 文件的配置设置 **部分来** 启用调试日志。

请注意，调试日志记录会生成大量条目，**journald** 或 **syslogd** 服务速率可能会限制 **syslog** 消息。要确保您有完整的日志，请将日志记录重定向到文件中。编辑 **/etc/ipsec.conf**，并在 **config setup** 部分中添加 **logfile=/var/log/pluto.log**。

其它资源

- 使用日志文件对问题进行故障排除.
- 使用和配置 firewalld.
- ***tcpdump(8)*** and ***ipsec.conf(5)*** man pages.

4.14. 相关信息

- ***ipsec(8)***、***ipsec.conf(5)***、***ipsec.secrets(5)***、***ipsec_auto(8)*** 和 ***ipsec_rsasigkey(8)man*** page.
- ***/usr/share/doc/libreswan-版本/*** 目录.
- 上游项目的网站。
- Libreswan Project Wiki.
- 所有 Libreswan man page.
- NIST 特殊发布 800-77：IPsec VPN 指南.

第 5 章 使用 MACSEC 加密同一物理网络中的第 2 层流量

这部分论述了如何为以太网链路中的所有流量配置 MACsec 以安全通信。

介质访问控制安全(MACsec)是一个第 2 层协议，可通过以太网链接保护不同类型的流量类型，包括：

- 动态主机配置协议(DHCP)
- 地址解析协议(ARP)
- 互联网协议版本 4 / 6(**IPv4** / **IPv6**)和
- 任何使用 IP 的流量（如 TCP 或 UDP）

MACsec 默认使用 GCM-AES-128 算法加密并验证 LAN 中的所有流量，并使用预共享密钥在参与者主机之间建立连接。如果要更改预共享密钥，您需要更新网络中使用 MACsec 的所有主机上的 NM 配置。

MACsec 连接将以太网设备（如以太网网卡、VLAN 或隧道设备）用作父设备。您只能在 MACsec 设备上设置 IP 配置，以仅使用加密连接与其他主机通信，也可以在父设备中设置 IP 配置。在后者的情况下，您可以使用父设备使用未加密连接和 MACsec 设备加密连接与其他主机通信。

macsec 不需要任何特殊硬件。例如，您可以使用任何交换机，除非您只想加密主机和交换机之间的流量。在这种情况下，交换机还必须支持 MACsec。

换句话说，有两种常用方法来配置 MACsec：

- 主机到主机，和
- 主机到交换机，然后交换机到其他主机

重要

您只能在相同（物理或虚拟）LAN 的主机间使用 MACsec。

下例演示了如何使用预共享密钥在 2 个主机之间配置 MACsec。

5.1. 使用 NMCLI 配置 MACSEC 连接

您可以使用 nmcli 工具将以太网接口配置为使用 MACsec。这个步骤描述了如何创建使用以太网接口加密网络流量的 MACsec 连接。

在所有应在此 MACsec 保护的网络中进行通信的所有主机上运行此步骤。

流程

在主机 A 中：

- 在您配置 MACsec 的第一个主机上，为预共享密钥创建连接关联密钥(CAK)和连接关联密钥名称(CKN)：
  a. 创建 16 字节十六进制 CAK：

```
dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
50b71a8ef0bd5751ea76de6d6c98c03a
```

  b. 创建 32 字节十六进制 CKN：

```
dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

在主机 A 和 B 中：

1. 创建 MACsec 连接：

```
# nmcli connection add type macsec con-name macsec0 ifname macsec0
connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-cak
50b71a8ef0bd5751ea76de6d6c98c03a  macsec.mka-ckn
f2b4297d39da7330910a7abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

使用 macsec. **mka-cak 和 macsec.mka- ckn** 参数中在上一步中生成的 CAK 和 CKN。在 MACsec-protected 网络的每个主机上，这些值必须相同。

2. 配置 MACsec 连接中的 IP 设置。

   a. 配置 **IPv4** 设置。例如，要为 **macsec0** 连接设置静态 **IPv4** 地址、网络掩码、默认网关和 DNS 服务器，请输入：

   ```
   # nmcli connection modify macsec0 ipv4.method manual ipv4.addresses
   '192.0.2.1/24' ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253'
   ```

   b. 配置 **IPv6** 设置。例如，要为 **macsec0** 连接设置静态 **IPv6** 地址、网络掩码、默认网关和 DNS 服务器，请输入：

   ```
   # nmcli connection modify macsec0 ipv6.method manual ipv6.addresses
   '2001:db8:1::1/32' ipv6.gateway '2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd'
   ```

3. 激活连接：

```
# nmcli connection up macsec0
```

验证步骤

1. 要验证流量是否已加密，请输入：

```
tcpdump -nn -i enp1s0
```

2. 要查看未加密的流量，请输入：

```
tcpdump -nn -i macsec0
```

3. 显示 MACsec 统计数据：

```
# ip macsec show
```

4. 针对每种保护类型显示单个计数器：仅完整性（加密关闭）和加密（加密）

```
# ip -s macsec show
```

5.2. 其它资源

- *MACsec：加密网络流量博客的不同解决方案*。

第 6 章 使用和配置 FIREWALLD

防火墙是保护机器不受来自外部的、不需要的网络数据的一种方式。它允许用户通过定义一组防火墙规则来控制主机上的入站网络流量。这些规则用于对进入的流量进行排序，并可以阻断或允许流量。

firewalld 是一个防火墙服务守护进程，通过 D-Bus 接口提供动态可定制的主机防火墙。如果是动态的，它可在每次修改规则时启用、修改和删除规则，而不需要在每次修改规则时重启防火墙守护进程。

firewalld 使用区域和服务的概念来简化流量管理。zones 是预定义的规则集。网络接口和源可以分配给区。允许的流量取决于您计算机连接到的网络，并分配了这个网络的安全级别。防火墙服务是预定义的规则，覆盖了允许特定服务进入流量的所有必要设置，并在区中应用。

服务使用一个或多个端口或地址进行网络通信。防火墙会根据端口过滤通讯。要允许服务的网络流量，必须打开其端口。**firewalld** 会阻止未明确设置为打开的端口上的所有流量。一些区（如可信区）默认允许所有流量。

请注意，带有 **nftables** 后端的 **firewalld** 不支持使用 **--direct** 选项将自定义 **nftables** 规则传递到 **firewalld**。

6.1. FIREWALLD 入门

本节提供有关 **firewalld** 的信息。

6.1.1. 使用 firewalld、nftables 或者 iptables 时

以下是您应该使用以下工具之一的概述：

- **firewalld** ：将 **firewalld** 实用程序用于简单的防火墙用例。实用程序易于使用，并涵盖这些情况下的典型用例。
- **nftables** ：使用 **nftables** 实用程序设置复杂和性能关键的防火墙，如整个网络。
- **iptables** ：Red Hat Enterprise Linux 上的 **iptables** 实用程序使用 the **nf_tables** 内核 API 而不是旧 后端。The **nf_tables** API 提供向后兼容性，因此使用 **iptables** 命令的脚本仍可在 Red Hat Enterprise Linux 上工作。对于新的防火墙脚本，红帽建议使用 **nftables**。

重要

要避免不同的防火墙服务相互影响，在 RHEL 主机中只有一个服务，并禁用其他服务。

6.1.2. Zones

firewalld 可以用来根据用户决定放置在该网络中的接口和流量级别的信任级别将网络划分为不同的区域。一个连接只能是一个区的一部分，但一个区可以被用来进行很多网络连接。

NetworkManager 通知接口区域的 **firewalld**。您可以为接口分配区：

- **NetworkManager**
- **firewall-config** 工具
- **firewall-cmd** 命令行工具
- RHEL web 控制台

后三个只能编辑适当的 **NetworkManager** 配置文件。如果您使用 web 控制台 **firewall-cmd** 或 **firewall-config** 更改界面区域，则请求将转发到 **NetworkManager**，且不会由 **firewalld** 处理。

预定义区域存储在 **/usr/lib/firewalld/zones/** 目录中，并可立即应用于任何可用的网络接口。只有在修改后，这些文件才会复制到 **/etc/firewalld/zones/** 目录中。预定义区的默认设置如下：

block

任何传入的网络连接都将通过适用于 **IPv4** 的 icmp-host-prohibited 消息和 icmp6-adm-prohibited（适用于 **IPv6**）来拒绝。只有从系统启动的网络连接才能进行。

dmz

对于您的非企业化区里的计算机来说，这些计算机可以被公开访问，且有限访问您的内部网络。只接受所选的入站连接。

drop

所有传入的网络数据包都会丢失，没有任何通知。只有外发网络连接也是可行的。

external

适用于启用了伪装的外部网络，特别是路由器。您不信任网络中的其他计算机不会损害您的计算机。只接受所选的入站连接。

home

用于家用，因为您可以信任其他计算机。只接受所选的入站连接。

internal

当您主要信任网络中的其他计算机时，供内部网络使用。只接受所选的入站连接。

public

可用于您不信任网络中其他计算机的公共区域。只接受所选的入站连接。

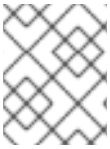
trusted

所有网络连接都被接受。

work

可用于您主要信任网络中其他计算机的工作。只接受所选的入站连接。

这些区中的一个被设置为 default 区。当接口连接添加到 **NetworkManager** 时，它们会被分配给默认区域。**安装时，firewalld** 中的默认区域设置为 **public** 区域。默认区可以被修改。



注意

网络区名称应该自我解释，并允许用户迅速做出合理的决定。要避免安全问题，请查看默认区配置并根据您的需要和风险禁用任何不必要的服务。

其它资源

- **firewalld.zone(5)** 手册页.

6.1.3. 预定义的服务

服务可以是本地端口、协议、源端口和目的地列表，并在启用了服务时自动载入防火墙帮助程序模块列表。使用服务可节省用户时间，因为它们可以完成一些任务，如打开端口、定义协议、启用数据包转发等等，而不必在另外的步骤中设置所有任务。

firewalld.service(5) man page 中介绍了服务配置选项和通用文件信息。服务通过单独的 XML **配置文件来指定，这些文件采用以下格式命名：service-name.xml.**协议名称优先于 **firewalld** 中的服务或应用程序名称。

可以使用图形化的 **firewall-config 工具**、**firewall-cmd 和 firewall-offline-cmd** 添加和删除服务。

或者，您可以编辑 **/etc/firewalld/services/** 目录中的 XML 文件。如果用户未添加或更改服务，则在 **/etc/firewalld/services/** 中找不到对应的 XML 文件。**如果要添加或更改服务，/usr/lib/firewalld/services/** 目录中的文件可作为模板使用。

其它资源

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

6.1.4. 启动 firewalld

流程

1. 要启动 **firewalld**，以 **root** 用户身份输入以下命令：

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

2. 要确保 **firewalld** 在系统启动时自动启动，以 **root** 用户身份输入以下命令：

```
# systemctl enable firewalld
```

6.1.5. 停止 firewalld

流程

1. 要停止 **firewalld**，以 **root** 用户身份输入以下命令：

```
# systemctl stop firewalld
```

2. 要防止 **firewalld** 在系统启动时自动启动：

```
# systemctl disable firewalld
```

3. 访问 firewalld **D-Bus** 接口以及其它服务需要 **firewalld** 以确保 **firewalld** 没有启动：

```
# systemctl mask firewalld
```

6.1.6. 验证永久 firewalld 配置

在某些情况下，例如在手动编辑 **firewalld** 配置文件后，管理员希望验证更改是否正确。本节论述了如何验证 **firewalld** 服务的永久配置。

先决条件

- **firewalld** 服务正在运行。

流程

1. 验证 **firewalld** 服务的永久配置：

-

```
# firewall-cmd --check-config
success
```

如果永久配置有效，该命令将返回 **成功**。在其他情况下，命令返回一个带有更多详情的错误，如下所示：

```
# firewall-cmd --check-config
Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}
```

6.2. 查看 FIREWALLD的当前状态和设置

本节论述了有关查看当前状态、允许的服务以及 **firewalld** 当前设置的信息。

6.2.1. 查看 firewalld的当前状态

默认情况下，防火墙服务 **firewalld** 安装在系统上。使用 **firewalld** CLI 界面检查该服务是否正在运行。

流程

1. 查看服务的状态：

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

2. 如需有关服务状态的更多信息，请使用 **systemctl status** 子命令：

```
# 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
```

6.2.2. 使用 GUI 查看允许的服务

要使用图形化的 **firewall-config** 工具查看服务列表，请按 **Super** 键进入"活动概览"，键入 **firewall**，然后按 **Enter** 键。**firewall-config** 工具会出现。现在，您可以在"服务"选项卡下查看 **服务** 列表。

您可以使用命令行启动图形防火墙配置工具。

先决条件

- 已安装 **firewall-config** 软件包。

流程

- 使用命令行启动图形防火墙配置工具：

```
$ firewall-config
```

此时将打开防火墙配置窗口。请注意，这个命令可以以普通用户身份运行，但偶尔会提示您输入管理员密码。

6.2.3. 使用 CLI 查看 firewalld 设置

使用 CLI 客户端可能会对当前防火墙设置有不同的视图。**list-all** 选项显示 **firewalld** 设置的完整概述。

Firewalld 使用区域来管理流量。如果 **--zone** 选项没有指定区域，该命令将在分配给活跃网络接口和连接的默认区域中有效。

流程

- 要列出默认区的所有相关信息：

```
# 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:
```

- 要指定显示设置的区域，请在 **firewall-cmd --list-all 命令中添加 --zone= zone-name** 参数，例如：

```
# 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]
```

- 要查看特定信息（如服务或端口）的设置，请使用特定选项。使用命令帮助查看 **firewalld** 手册页或获取选项列表：

```
# firewall-cmd --help
```

- 查看当前区中允许哪些服务：

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

注意

使用 CLI 工具列出某个子部分的设置有时会比较困难。例如，您允许 **SSH 服务**，**firewalld** 会打开该服务所需的端口(22)。之后，如果您列出允许的服务，列表将显示 **SSH** 服务，但如果列出开放端口，则不会显示任何服务。因此，建议您使用 **--list-all** 选项来确保您收到完整信息。

6.3. 使用 FIREWALLD控制网络流量

本节介绍使用 **firewalld** 控制网络流量的信息。

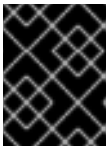
6.3.1. 使用 CLI 禁用紧急事件的所有流量

在紧急情况下，如系统攻击，可以禁用所有网络流量并关闭攻击者。

流程

1. 要立即禁用网络流量，请切换 panic 模式：

```
# firewall-cmd --panic-on
```



重要

启用 panic 模式可停止所有网络流量。因此，它应该只在对机器有物理访问权限或使用串行控制台登录时才使用。

2. 关闭 panic 模式会使防火墙恢复到其永久设置。要关闭 panic 模式，请输入：

```
# firewall-cmd --panic-off
```

验证

- 要查看是否打开或关闭 panic 模式，请使用：

```
# firewall-cmd --query-panic
```

6.3.2. 使用 CLI 控制预定义服务的流量

控制流量的最简单方法是添加预定义服务到 **firewalld**。这会打开所有必需的端口并根据 服务定义文件 修改其他设置。

流程

1. 检查该服务是否还未被允许：

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

2. 列出所有预定义的服务：

```
# 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. 在允许的服务中添加服务：

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

4. 使新设置持久：

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

6.3.3. 通过 GUI，使用预定义服务控制流量

这个步骤描述了如何使用图形用户界面控制预定义服务的网络流量。

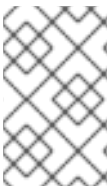
先决条件

- 已安装 **firewall-config** 软件包

流程

1. 启用或禁用预定义或自定义服务：
   a. 启动 **firewall-config** 工具并选择要配置的服务的网络区。
   b. 选择" **服务"** 选项卡。
   c. 选择您要信任的每种服务类型的复选框，或者清除要阻断服务的复选框。
2. 编辑服务：
   a. 启动 **firewall-config** 工具。
   b. 从标记为 **Configuration** 的菜单中选择 **永久**。其它图标和菜单按钮会出现在服务窗口底部。
   c. 选择您要配置的服务。

端口、协议和源端口 选项卡可为所选服务启用、更改和删除端口、协议和源端口。模块标签是用来配置 **Netfilter helper** 模块。**Destination** 选项卡启用限制到特定目标地址和互联网协议（**IPv4** 或 **IPv 6**）的流量。



注意

在运行时 **模式中 无法更改服**务设置。

6.3.4. 添加新服务

可以使用图形化的 **firewall-config** 工具、**firewall- cmd** 和 **firewall- offline-cmd** 添加和删除服务。或

者，您可以编辑 /etc/firewalld/services/ 中的 XML 文件。如果用户未添加或更改服务，则在 /etc/firewalld/services/ 中没有找到对应的 XML 文件。如果要添加或更改服务，则 /usr/lib/firewalld/services/ 文件可用作模板。

注意

服务名称必须是字母数字，还可仅包含 _ （下划线）和 - （短划线）字符。

流程

要在终端中添加新服务，请使用 firewall-cmd 或 firewall-offline-cmd （如果未激活 firewalld ）。

1. 运行以下命令以添加新和空服务：

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

2. 要使用本地文件添加新服务，请使用以下命令：

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

您可以使用 additional --name=service-name 选项来更改服务名称。

3. 更改服务设置后，服务的更新副本将置于 /etc/firewalld/services/ 中。

作为 root 用户，您可以输入以下命令手动复制服务：

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

firewalld 最初从 /usr/lib/firewalld/services 加载文件。如果文件放置在 /etc/firewalld/services 中并且有效，则这些文件将覆盖 /usr/lib/firewalld/services 中的匹配文件。一旦删除了 /etc/firewalld/services 中的匹配文件，或者要求 firewalld 加载服务的默认值，则将立即使用 /usr/lib/firewalld/services 中的覆盖文件。这只适用于永久性环境。要在运行时环境中获取这些回退，则需要重新载入。

6.3.5. 使用 GUI 打开端口

要允许通过防火墙到特定端口的流量，您可以在 GUI 中打开端口。

先决条件

- 已安装 firewall-config 软件包

流程

1. 启动 firewall-config 工具并选择要更改的网络区。
2. 选择 端口 选项卡，然后单击右侧的 添加 按钮。此时会打开 端口和协议 窗口。
3. 输入要允许的端口号或者端口范围。
4. 从列表中选择 tcp 或 udp。

6.3.6. 使用 GUI 控制协议的流量

要允许使用特定协议通过防火墙的流量，您可以使用 GUI。

先决条件

- 已安装 firewall-config 软件包

流程

1. 启动 firewall-config 工具并选择要更改的网络区。
2. 选择 协议 选项卡，然后单击右侧的 添加 按钮。此时会打开 协议 窗口。
3. 从列表中选择协议，或者选择"其他协议 "复选框并在字段中输入协议。

6.3.7. 使用 GUI 打开源端口

要允许通过防火墙的流量来自特定端口，您可以使用 GUI。

先决条件

- 已安装 firewall-config 软件包

流程

1. 启动 firewall-config 工具并选择要更改的网络区。
2. 选择"源端口 "选项卡，然后单击右侧的添加按钮。Source Port 窗口将打开。
3. 输入要允许的端口号或者端口范围。从列表中选择 tcp orudp。

6.4. 使用 CLI 控制端口

端口是可让操作系统接收和区分网络流量并将其转发到系统服务的逻辑设备。它们通常由侦听端口的守护进程来表示，它会等待到达这个端口的任何流量。

通常，系统服务侦听为它们保留的标准端口。例如，httpd 守护进程侦听端口 80。但默认情况下，系统管理员会将守护进程配置为在不同端口上侦听以便增强安全性或出于其他原因。

6.4.1. 打开端口

通过打开端口，系统可从外部访问，这代表了安全风险。通常，让端口保持关闭，且只在某些服务需要时才打开。

流程

要获得当前区的打开端口列表：

1. 列出所有允许的端口：

```
# firewall-cmd --list-ports
```

2. 在允许的端口中添加一个端口，以便为入站流量打开这个端口：

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

端口类型为 tcp、udp、sctp 或 dccp。这个类型必须与网络通信的类型匹配。

3. 使新设置具有持久性：

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

端口类型为 tcp、udp、sctp 或 dccp。这个类型必须与网络通信的类型匹配。

6.4.2. 关闭端口

当不再需要打开端口时，在 firewalld 中关闭该端口。强烈建议您尽快关闭所有不必要的端口，因为端口处于打开状态会存在安全隐患。

流程

要关闭某个端口，请将其从允许的端口列表中删除：

1. 列出所有允许的端口：

```
# firewall-cmd --list-ports
```

警告

这个命令只为您提供已打开作为端口的端口列表。您将无法看到作为服务打开的任何打开端口。因此，您应该考虑使用 **--list-all** 选项而不是 **--list-ports**。

2. 从允许的端口中删除端口，以便对传入的流量关闭：

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

3. 使新设置具有持久性：

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

6.5. 使用 FIREWALLD 区

zones 代表一种更透明管理传入流量的概念。这些区域连接到联网接口或者分配一系列源地址。您可以独立为每个区管理防火墙规则，这样就可以定义复杂的防火墙设置并将其应用到流量。

6.5.1. 列出区域

这个步骤描述了如何使用命令行列出区。

流程

1. 查看系统中有哪些可用区：

```
# firewall-cmd --get-zones
```

firewall-cmd --get-zones 命令显示系统上可用的所有区域，但不显示特定区域的任何详情。

2. 查看所有区的详细信息：

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

3. 查看特定区的详细信息：

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

6.5.2. 更改特定区的 firewalld 设置

使用 cli 和 控制端口通过预定义服务控制流量，使用 cli 说明如何在当前工作区范围内添加服务或修改端口。有时，需要在不同区内设置规则。

流程

- 要在不同区域中工作，请使用 --zone=zone-name 选项。例如，允许在区 public 中使用 SSH 服务：

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

6.5.3. 更改默认区

系统管理员在其配置文件中为网络接口分配区域。如果接口没有被分配给指定区，它将被分配给默认区。每次重启 firewalld 服务后，firewalld 加载默认区域的设置并使它活跃。

流程

设置默认区：

1. 显示当前的默认区：

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

2. 设置新的默认区：

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

注意

遵循此过程后，该设置是永久设置，即使没有 --permanent 选项。

6.5.4. 将网络接口分配给区

可以为不同区定义不同的规则集，然后通过更改所使用的接口的区来快速改变设置。使用多个接口，可以为每个具体区设置一个区来区分通过它们的网络流量。

流程

要将区分配给特定的接口：

1. 列出活跃区以及分配给它们的接口：

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

2. 为不同的区分配接口：

```
# firewall-cmd --zone=zone_name --change-interface=interface_name --permanent
```

6.5.5. 使用 nmcli 为连接分配区域

这个步骤描述了如何使用 nmcli 实用程序将 firewalld 区域添加到 NetworkManager 连接中。

流程

1. 为 NetworkManager 连接配置集分配区：

```
# nmcli connection modify profile connection.zone zone_name
```

2. 重新加载连接：

```
# nmcli connection up profile
```

6.5.6. 在 ifcfg 文件中手动将区分配给网络连接

当连接由 网络管理器（NetworkManager）管理时，必须了解它使用的区域。为每个网络连接指定区域，根据计算机有可移植设备的位置提供各种防火墙设置的灵活性。因此，可以为不同的位置（如公司或家）指定区域和设置。

流程

- 要为连接设置区域，请编辑 /etc/sysconfig/network-scripts/ifcfg-connection_name 文件，并添加为这个连接分配区的行：

```
ZONE=zone_name
```

6.5.7. 创建一个新区

要使用自定义区，创建一个新的区并使用它像预定义区一样。新区域需要 --permanent 选项，否则 命令不起作用。

流程

1. 创建一个新区：

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

2. 检查是否在您的永久设置中添加了新的区：

```
# firewall-cmd --get-zones
```

3. 使新设置具有持久性：

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

6.5.8. 区配置文件

区也可以通过区配置文件创建。如果您需要创建新区，但想从不同区重复使用设置，这种方法就很有用了。

firewalld 区域配置文件包含区域的信息。这些区描述、服务、端口、协议、icmp-blocks、masquerade、forward-ports 和丰富的语言规则采用 XML 文件格式。文件名必须是zone-name.xml，其中 zone-name 的长度限制为 17 个字符。区域配置文件位于 /usr/lib/firewalld/zones/ 和 /etc/firewalld/zones/ 目录中。

以下示例显示了允许一个服务(SSH)和一个端口范围的配置，适用于 TCP 和 UDP 协议：

```
<?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 protocol="udp" port="1025-65535"/>
  <port protocol="tcp" port="1025-65535"/>
</zone>
```

要更改那个区的设置，请添加或者删除相关的部分来添加端口、转发端口、服务等等。

其它资源

- **firewalld.zone 手册页**

6.5.9. 使用区目标设定传入流量的默认行为

对于每个区，您可以设置一种处理尚未进一步指定的传入流量的默认行为。这种行为是通过设置区目标来定义的。有四个选项 - default、ACCEPT、REJECT 和 DROP。通过将目标设置为 ACCEPT，您接受除特定规则禁用的数据包外的所有传入数据包。如果将目标设置为 REJECT 或 DROP，则禁用除特定规则中允许的数据包之外的所有传入数据包。拒绝数据包时，会通知源机器，但丢弃数据包时不会发送任何信息。

流程

为区设置目标：

1. 列出特定区的信息以查看默认目标：

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

2. 在区中设置一个新目标：

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

6.6. 根据源使用区管理传入流量

您可以使用区管理传入的流量，根据其源管理传入的流量。这可让您对进入的流量进行排序，并将其路由到不同的区，以允许或禁止该流量可访问的服务。

如果您给区添加一个源，区就会成为活跃的，来自该源的所有进入流量都会被定向到它。您可以为每个区指定不同的设置，这些设置相应地应用于来自给定源的网络流量。即使只有一个网络接口，您可以使用更多区域。

6.6.1. 添加源

要将传入的流量路由到特定区域，请将源添加到那个区。源可以是一个使用 CIDR 格式的 IP 地址或 IP 掩码。

注意

如果您添加多个带有重叠网络范围的区域，则根据区名称排序，且只考虑第一个区。

- 在当前区中设置源：

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

- 要为特定区设置源 IP 地址：

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

以下流程允许来自 可信 区 192.168.2.15 的所有传入流量：

流程

1. 列出所有可用区：

```
# firewall-cmd --get-zones
```

2. 将源 IP 添加到持久性模式的信任区中：

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

3. 使新设置具有持久性：

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

6.6.2. 删除源

从区中删除源会关闭来自它的网络流量。

流程

1. 列出所需区的允许源：

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

2. 从区永久删除源：

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

3. 使新设置具有持久性：

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

6.6.3. 添加源端口

要启用根据原始端口对流量排序，请使用 **--add-source-port** 选项指定一个源端口。您还可以将其与 **--add-source** 选项组合使用，将流量限制为特定的 IP 地址或 IP 范围。

流程

- 添加源端口：

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

6.6.4. 删除源端口

通过删除源端口，您可以根据原始端口禁用对流量排序。

流程

- 要删除源端口：

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

6.6.5. 使用区和源来允许一个服务只适用于一个特定的域

要允许特定网络的流量在机器上使用服务，请使用区和源。以下步骤只允许来自 192.0.2.0/24 网络的 HTTP 流量，而任何其他流量都被阻断。

警告

配置此场景时，请使用具有默认目标的区域。使用将目标设置为 ACCEPT 的区域存在安全风险，因为对于来自 192.0.2.0/24 的流量，所有网络连接都将被接受。

流程

1. *列出所有可用区：*

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

2. *将 IP 范围添加到内部区，以通过区路由来自源的流量：*

```
# firewall-cmd --zone=internal --add-source=192.0.2.0/24
```

3. *在内部区中添加 http 服务：*

```
# firewall-cmd --zone=internal --add-service=http
```

4. *使新设置具有持久性：*

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

验证

- *检查内部区是否活跃，且该服务是否允许：*

```
# firewall-cmd --zone=internal --list-all
internal (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: 192.0.2.0/24
  services: cockpit dhcpv6-client mdns samba-client ssh http
  ...
```

其它资源

- firewalld.zones(5) man page

6.7. 使用 FIREWALLD 配置 NAT

使用 firewalld，您可以配置以下网络地址转换(NAT)类型：

- 伪装
- 源 NAT（SNAT）
- 目标 NAT（DNAT）
- 重定向

6.7.1. 不同的 NAT 类型： masquerading、source NAT、destination NAT 和 redirect

这些是不同的网络地址转换（NAT）类型：

伪装和源 NAT（SNAT）

使用以上 NAT 类型之一更改数据包的源 IP 地址。例如，互联网服务提供商不路由专用 IP 范围，如 10.0.0.0/8。如果您在网络中使用专用 IP 范围，并且用户应该能够访问 Internet 上的服务器，请将这些范围内的数据包源 IP 地址映射到公共 IP 地址。

伪装和 SNAT 都非常相似。不同之处是：

- 伪装自动使用传出接口的 IP 地址。因此，如果传出接口使用了动态 IP 地址，则使用伪装。
- SNAT 将数据包的源 IP 地址设置为指定的 IP 地址，且不会动态查找传出接口的 IP 地址。因此，SNAT 要比伪装更快。如果传出接口使用了固定 IP 地址，则使用 SNAT。

目标 NAT（DNAT）

使用此 NAT 类型重写传入数据包的目标地址和端口。例如，如果您的 Web 服务器使用私有 IP 范围内的 IP 地址，因此无法直接从互联网访问，您可以在路由器上设置 DNAT 规则以将传入的流量重定向到此服务器。

重定向

这个类型是 IDT 的特殊示例，它根据链 hook 将数据包重定向到本地机器。例如，如果服务在其标准端口的不同端口上运行，您可以将从标准端口传入的流量重定向到此特定端口。

6.7.2. 配置 IP 地址伪装

以下流程描述了如何在系统中启用 IP 伪装。IP 伪装会在访问互联网时隐藏网关后面的独立机器。

流程

1. 要检查 IP 伪装是否已启用（例如，对于 外部 区），以 root 用户身份输入以下命令：

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

如果已启用，命令将打印 yes，退出状态为 0。否则，将不会 打印退出状态 1。如果省略 区域，则将使用默认区域。

2. 要启用 IP 伪装，以 root 用户身份输入以下命令：

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

3. 要使此设置持久，请重复添加 **--permanent** 选项的命令。

要禁用 IP 伪装，以 root 身份输入以下命令：

```
# firewall-cmd --zone=external --remove-masquerade --permanent
```

6.8. 端口转发

使用此方法重定向端口只可用于基于 IPv4 的流量。对于 IPv6 重定向设置，您必须使用丰富的规则。

要重定向到外部系统，需要启用伪装。如需更多信息，请参阅配置 IP 地址伪装。

6.8.1. 添加一个端口来重定向

使用 firewalld，您可以设置端口重定向，以便到达您系统中特定端口的任何传入流量都将传送到您选择的其他内部端口或另一台计算机上的外部端口。

先决条件

- 在您将从一个端口的流量重新指向另一个端口或另一个地址前，您必须了解 3 个信息：数据包到达哪个端口，使用什么协议，以及您要重定向它们的位置。

流程

1. 将端口重新指向另一个端口：

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

2. 将端口重定向到不同 IP 地址的另一个端口：

   a. 添加要转发的端口：

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

   b. 启用伪装：

```
# firewall-cmd --add-masquerade
```

6.8.2. 将 TCP 端口 80 重定向到同一台机器中的 88 端口

按照以下步骤将 TCP 端口 80 重定向到端口 88。

流程

1.

1. 将端口 80 重定向到 TCP 流量的端口 88:

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

2. 使新设置具有持久性：

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

3. 检查是否重定向了端口：

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

6.8.3. 删除重定向的端口

这个步骤描述了如何删除重定向的端口。

流程

1. 要删除重定向的端口：

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

2. 要删除重定向到不同地址的转发端口：

   a. 删除转发的端口：

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

   b. 禁用伪装：

```
# firewall-cmd --remove-masquerade
```

6.8.4. 在同一台机器上将 TCP 端口 80 转发到端口 88

这个步骤描述了如何删除端口重定向。

流程

1. 列出重定向的端口：

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

2. 从防火墙中删除重定向的端口：

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

3. 使新设置具有持久性：

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

6.9. 管理 ICMP 请求

Internet 控制消息协议 (ICMP)是一种支持协议，供各种网络设备用于发送错误消息和显示连接问题的操作信息，例如，请求的服务不可用。ICMP 与 TCP 和 UDP 等传输协议不同，因为它不用于在系统之间交换数据。

不幸的是，可以使用 ICMP 消息（特别是 echo-request 和 echo-reply ）揭示关于您的网络的信息，并将此类信息滥用于各种类型的活动。因此，firewalld 允许阻止 ICMP 请求来保护您的网络信息。

6.9.1. 列出和阻塞 ICMP 请求

列出 ICMP 请求

位于 /usr/lib/firewalld/icmptypes/ 目录中的独立 XML 文件中描述了 ICMP 请求。您可以阅读这些文件来查看请求的描述。firewall-cmd 命令控制 ICMP 请求操作。

- 要列出所有可用的 ICMP 类型：

```
# firewall-cmd --get-icmptypes
```

- IPv4、IPv6 或两个协议都可以使用 ICMP 请求。要查看 ICMP 请求使用了哪些协议：

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

- 如果请求当前为 blocked 或 no，则 ICMP 请求的状态将显示 yes。查看 ICMP 请求当前是否被阻断：

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

阻塞或取消阻塞 ICMP 请求

当您的服务器阻断 ICMP 请求时，它不会提供通常会提供的信息。但这并不意味着根本不给出任何信息。客户端收到特定 ICMP 请求被阻止（拒绝）的信息。应仔细考虑阻止 ICMP 请求，因为它可能会导致通信问题，特别是 IPv6 流量。

- 查看 ICMP 请求当前是否被阻断：

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

- 阻止 ICMP 请求：

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

- 删除 ICMP 请求的块：

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

在不提供任何信息的情况下阻塞 ICMP 请求

通常，如果您阻止 ICMP 请求，客户端会知道您在阻止 ICMP 请求。这样潜在的攻击者仍然可以看到您的 IP 地址在线。要完全隐藏此信息，您必须丢弃所有 ICMP 请求。

- 阻止和丢弃所有 ICMP 请求：

- 将区的目标设置为 DROP ：

```
# firewall-cmd --permanent --set-target=DROP
```

-

现在，除您明确允许的流量外，所有流量（包括 ICMP 请求）将被丢弃。

阻塞和丢弃某些 ICMP 请求并允许其他请求：

1. 将区的目标设置为 DROP ：

```
# firewall-cmd --permanent --set-target=DROP
```

2. 添加 ICMP 块 inversion 以阻止所有 ICMP 请求：

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

3. 为您要允许的 ICMP 请求添加 ICMP 块：

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

4. 使新设置具有持久性：

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

block inversion 会颠倒 ICMP 请求块的设置，因此所有之前没有被阻止的请求都会被阻止，因为区域的目标更改为 DROP。被阻断的请求不会被阻断。这意味着，如果您想要取消阻塞请求，则必须使用 blocking 命令。

将块 inversion 恢复到完全 permissive 设置：

1. 将区的目标设置为 default 或 ACCEPT:

```
# firewall-cmd --permanent --set-target=default
```

2. 删除 ICMP 请求添加的所有块：

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

3. 删除 ICMP 块 inversion：

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

4. 使新设置具有持久性：

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

6.9.2. 使用 GUI 配置 ICMP 过滤器

- 要启用或禁用 ICMP 过滤器，请启动 firewall-config 工具并选择过滤消息的网络区域。选择 ICMP Filter 选项卡，再选中您要过滤的每种 ICMP 消息的复选框。清除复选框以禁用过滤器。这个设置按方向设置，默认允许所有操作。

- 若要编辑 ICMP 类型，可启动 firewall-config 工具，然后从标记为 Configuration 的菜单中选择 永久 模式。在 Services 窗口的底部会显示附加图标。选择 Yes 启用伪装并转发到另一台机器工作。

- 若要启用反向 ICMP Filter，可单击右侧的 Invert Filter 复选框。现在仅接受标记的 ICMP 类型，所有其他均被拒绝。在使用 DROP 目标的区域里它们会被丢弃。

6.10. 使用 FIREWALLD设置和控制 IP 集

要查看 firewalld 支持的 IP 设置类型列表，以 root 用户身份输入以下命令。

```
~]# 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
```

6.10.1. 使用 CLI 配置 IP 设置选项

可以在 firewalld 区域中使用 IP 集作为源，也可以用作富规则的来源。在 Red Hat Enterprise Linux 中，首选的方法是在直接规则中使用通过 firewalld 创建的 IP 集合。

- 要列出永久环境中 firewalld 已知的 IP 集，以 root 用户身份运行以下命令：

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

- 要添加新 IP 集，以 root 用户身份使用永久环境运行以下命令：

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

上一命令为 IPv4 创建了名称 test 和 hash:net 类型的新 IP 设置。要创建用于 IPv6 的 IP 集，请添加 --option=family=inet6 选项。要使新设置在运行时环境中有效，请重新加载 firewalld。

- 使用以下命令以 root 用户身份列出新 IP 设置：

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

- 要获取有关 IP 集的更多信息，以 root 用户身份运行以下命令：

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

请注意，IP 集目前没有任何条目。

- 要在 test IP 集中添加一个条目，以 root 用户身份运行以下命令：

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

前面的命令将 IP 地址 192.168.0.1 添加到 IP 集合中。

- 要获取 IP 集合中当前条目列表，以 root 用户身份运行以下命令：

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

- 生成包含 IP 地址列表的文件，例如：

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

包含 IP 集合 IP 地址列表的文件应该每行包含一个条目。以 hash、分号或空行开头的行将被忽略。

- 要添加 iplist.txt 文件中的地址，以 root 用户身份运行以下命令：

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

- 要查看 IP 集合的扩展条目列表，以 root 用户身份运行以下命令：

```
# 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
```

- 要从 IP 集合中删除地址并检查更新的条目列表，以 root 用户身份运行以下命令：

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

- 您可以将 IP 集合作为一个源添加到区，以便处理所有来自 IP 集合中列出的任意地址的网络流量。例如，要将 test IP 集作为源添加到 drop 区域，以丢弃来自测试 IP 集合中列出的所有条目的所有数据包，以 root 用户身份运行以下命令 ：

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

源中的 ipset: 前缀显示 firewalld 表明源是 IP 集，而不是 IP 地址或地址范围。

只有 IP 集的创建和删除仅限于永久环境，所有其他 IP 设置选项也可以在运行时环境中使用，而无需 --

permanent 选项。

警告

红帽不推荐使用不通过 firewalld 管理的 IP 集。要使用这样的 IP 组，需要一个永久直接规则来引用集合，且必须添加自定义服务来创建这些 IP 组件。此服务需要在 firewalld 启动前启动，否则 firewalld 无法使用这些集合添加直接规则。您可以使用 /etc/firewalld/direct.xml 文件添加永久直接规则。

6.11. 丰富规则的优先级

默认情况下，会根据其规则操作对富规则进行组织。例如，拒绝 规则优先于 allow 规则。优先级 参数丰富的规则可让管理员对丰富的规则及其执行顺序进行精细的控制。

6.11.1. priority 参数如何将规则组织为不同的链

您可以将 priority 参数设置为在 -32768 和 32767 之间的任意数字，较低值具有更高的优先级。

firewalld 服务根据其优先级值在不同的链中组织规则：

- 优先级低于 0：规则被重定向到带有 _pre 后缀的 链中。
- 优先级高于 0：规则被重定向到带有 _post 后缀的链中。
- 优先级等于 0：基于操作，规则将通过 _log、_deny 或 _ allow 重定向到链中。

在这些子链中，firewall d 会根据其优先级值对规则进行排序。

6.11.2. 设置丰富的规则的优先级

该流程描述了如何创建丰富的规则，该规则使用 priority 参数记录其他规则不允许或拒绝的所有流量。您可以使用此规则标记意非预期的流量。

流程

1. 添加一个带有非常低优先级的丰富规则来记录未由其他规则匹配的所有流量：

```
# firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit
value="5/m"'
```

命令还会将日志条目的数量限制为每分钟 5 个。

2. 另外，还可显示上一步中命令创建的 nftables 规则：

```
# nft list chain inet firewalld filter_IN_public_post
table inet firewalld {
  chain filter_IN_public_post {
    log prefix "UNEXPECTED: " limit rate 5/minute
  }
}
```

6.12. 配置防火墙锁定

如果本地应用或服务以 root 身份运行（如 libvirt），则可以更改防火墙配置。使用这个特性，管理员可以锁定防火墙配置，从而达到没有应用程序或只有添加到锁定白名单中的应用程序可以请求防火墙更改的目的。锁定设置默认会被禁用。如果启用，用户就可以确定，防火墙没有被本地的应用程序或服务进行了不必要的配置更改。

6.12.1. 使用 CLI 配置锁定

这个步骤描述了如何使用命令行启用或禁用锁定。

- 要查询是否启用了锁定，以 root 用户身份运行以下命令：

```
# firewall-cmd --query-lockdown
```

如果启用锁定，该命令将输出 yes，退出状态为 0。否则，将不会 打印退出状态 1。

- 要启用锁定，以 root 用户身份输入以下命令：

```
# firewall-cmd --lockdown-on
```

- 要禁用锁定，以 root 用户身份使用以下命令：

```
# firewall-cmd --lockdown-off
```

6.12.2. 使用 CLI 配置锁定允许列表选项

锁定允许名单中可以包含命令、安全上下文、用户和用户 ID。如果允许列表中的命令条目以星号"*"结尾，则以该命令开头的所有命令行都将匹配。如果没有 "*"，那么包括参数的绝对命令必须匹配。

- 上下文是正在运行的应用程序或服务的安全（SELinux）上下文。要获得正在运行的应用程序的上下文，请使用以下命令：

```
$ ps -e --context
```

该命令返回所有正在运行的应用程序。通过 grep 工具管道输出以便获取您感兴趣的应用程序。例如：

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

- 要列出允许列表中的所有命令行，以 root 用户身份输入以下命令：

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

- 要在允许列表中添加 命令 command，以 root 用户身份输入以下命令：

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

- 要从允许列表中删除 命令 命令，以 root 用户身份输入以下命令：

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

- 要查询命令命令是否在 允许列表中，以 root 用户身份输入以下命令：

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

如果为 **true**，该命令将输出 **yes**，退出状态为 **0**。否则，将不会 打印退出状态 **1**。

- 要列出允许列表中的所有安全上下文，以 **root** 用户身份输入以下命令：

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

- 要在允许列表中添加 上下文 **context**，以 **root** 用户身份输入以下命令：

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

- 要从允许列表中删除 上下文 **context**，以 **root** 用户身份输入以下命令：

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

- 要查询上下文 上下文 是否在允许列表中，以 **root** 用户身份输入以下命令：

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

如果为 **true**，则打印 **yes** （退出状态为 **0** ），否则打印 **no**，退出状态为 **1**。

- 要列出允许列表中的所有用户 **ID**，以 **root** 用户身份输入以下命令：

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

- 要在允许列表中添加用户 **ID uid**，以 **root** 用户身份输入以下命令：

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

- 要从允许列表中删除用户 **ID uid**，以 **root** 用户身份输入以下命令：

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

- 要查询用户 ID uid 是否在 allowlist 中，请输入以下命令：

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

如果为 true，则打印 yes （退出状态为 0 ），否则打印 no，退出状态为 1。

- 要列出允许列表中的所有用户名，以 root 用户身份输入以下命令：

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

- 要在允许列表中添加用户名 user，以 root 用户身份输入以下命令：

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

- 要从允许列表中删除用户名 user，以 root 用户身份输入以下命令：

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

- 要查询用户名 user 是否在 allowlist 中，请输入以下命令：

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

如果为 true，则打印 yes （退出状态为 0 ），否则打印 no，退出状态为 1。

6.12.3. 使用配置文件配置锁定的 allowlist 选项

默认的 allowlist 配置文件包含 NetworkManager 上下文和 libvirt 的默认上下文。用户 ID 0 也位于列表中。

```
<?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>
```

以下是一个 allowlist 配置文件示例，为 firewall-cmd 工具程序启用所有命令，对于用户 ID 为 815 的

名为 **user** 的用户 ：

```
<?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>
```

此示例显示用户 ID 和用户名，但只需要一个选项。Python 是程序解释器，它位于命令行的前面。您还可以使用特定的命令，例如：

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

在该示例中，只允许 **--lockdown-on** 命令。

在 **Red Hat Enterprise Linux** 中，所有实用程序都放置在 **/usr/bin/** 目录中，**/bin/** 目录则链接到 **/usr/bin/** 目录。换句话说，尽管以 **root** 身份输入时输入 **firewall-cmd** 的路径可能会解析为 **/bin/firewall-cmd**，但现在可以使用/usr/bin/firewall-cmd。所有新脚本都应该使用新位置。但请注意，如果以 **root** 身份运行的脚本被写为使用 **/bin/firewall-cmd** 路径，那么除了通常仅用于非root 用户的 **/usr/bin/firewall-cmd** 路径外，还必须在允许列表中添加该命令路径。

命令 **name** 属性末尾的 * 表示所有以这个字符串开头的命令都匹配。如果没有 *，则包括参数的绝对命令必须匹配。

6.13. 其它资源

- **firewalld(1) man page**
- **firewalld.conf(5)** 手册页
- **firewall-cmd(1) man page**
- **firewall-config(1) man page**

- ***firewall-offline-cmd(1) man page***
- ***firewalld.icmptype(5) man page***
- ***firewalld.ipset(5) man page***
- ***firewalld.service(5) man page***
- ***firewalld.zone(5) 手册页***
- ***firewalld.direct(5) man page***
- ***firewalld.lockdown-whitelist(5)***
- ***firewalld.richlanguage(5)***
- ***firewalld.zones(5) man page***
- ***firewalld.dbus(5) man page***

第 7 章 NFTABLES 入门

nftables 框架提供数据包分类功能。最显著的功能是：

- 内置查找表而不是线性处理
- IPv4 和 IPv6 使用同一个协议框架
- 规则会以一个整体被应用，而不是分为抓取、更新和存储完整的规则集的步骤
- 支持在规则集(nftrace)和监控追踪事件（nft）中调试和追踪
- 更加一致和压缩的语法，没有特定协议的扩展
- 用于第三方应用程序的 Netlink API

nftables 框架使用表来存储链。链包含执行动作的独立规则。libnftnl 库可用于通过 libmnl 库与 nftables Netlink API 进行低级交互。

要显示规则集更改的影响，请使用 nft list ruleset 命令。由于这些工具将表、链、规则、集合和其他对象添加到 nftables 规则集，请注意 nftables 规则集操作（如 nft flush ruleset 命令）可能会影响使用之前独立的旧命令安装的规则集。

7.1. 从 IPTABLES 迁移到 NFTABLES

如果您的防火墙配置仍然使用 iptables 规则，您可以将 iptables 规则 迁移到 nftables。

7.1.1. 使用 firewalld、nftables 或者 iptables 时

以下是您应该使用以下工具之一的概述：

- firewalld ：将 firewalld 实用程序用于简单的防火墙用例。实用程序易于使用，并涵盖这些

情况下的典型用例。

- nftables ：使用 nftables 实用程序设置复杂和性能关键的防火墙，如整个网络。

- iptables ：Red Hat Enterprise Linux 上的 iptables 实用程序使用 the nf_tables 内核 API 而不是 旧 后端。The nf_tables API 提供向后兼容性，因此使用 iptables 命令的脚本仍可在 Red Hat Enterprise Linux 上工作。对于新的防火墙脚本，红帽建议使用 nftables。

重要

要避免不同的防火墙服务相互影响，在 RHEL 主机中只有一个服务，并禁用其他服务。

7.1.2. 将 iptables 规则转换为 nftables 规则

Red Hat Enterprise Linux 提供了 iptables-translate 和 ip6tables-translate 工具，可以将现有的 iptables 或 ip6tables 规则转换为与 nftables 相同的规则。

请注意，一些扩展可能缺少响应的转换支持。如果存在这样的扩展，工具会输出带有 # 符号前缀的未转换的规则。例如：

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

此外，用户可以使用 iptables-restore-translate 和 ip6tables-restore-translate 工具来转换规则转储。请注意，在此之前，用户可以使用 iptables-save 或 ip6tables-save 命令来打印当前规则的转储。例如：

```
# 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
...
```

如需更多信息，以及可能的选项和值列表，请输入 iptables-translate --help 命令。

7.1.3. 常见 iptables 和 nftables 命令的比较

以下是常见 iptables 和 nftables 命令的比较：

- 列出所有规则：

iptables	nftables
iptables-save	nft list ruleset

- 列出特定的表和链：

iptables	nftables
iptables -L	nft list table ip filter
iptables -L INPUT	nft list chain ip filter INPUT
iptables -t nat -L PREROUTING	nft list chain ip nat PREROUTING

nft 命令不会预先创建表和链。只有在用户手动创建它们时才会出现它们。

示例：列出 firewalld 生成的规则

```
# nft list table inet firewalld
# nft list table ip firewalld
# nft list table ip6 firewalld
```

7.2. 编写和执行 NFTABLES 脚本

nftables 框架提供了一个原生脚本环境，它比使用 shell 脚本维护防火墙规则具有主要优势：执行脚本是原子的。这意味着，系统会应用整个脚本，或者在出现错误时防止执行。这样可保证防火墙始终处于一致状态。

另外，nftables 脚本环境使管理员能够：

- 添加评论
- 定义变量
- 包含其他规则集文件

本节介绍如何使用这些功能，以及创建和执行 nftables 脚本。

安装 nftables 软件包时，Red Hat Enterprise Linux 会在 /etc/ nftables/ 目录中自动创建 *.nf t 脚本。这些脚本包含为不同目的创建表和空链的命令。

7.2.1. 支持的 nftables 脚本格式

nftables 脚本环境支持以下格式的脚本：

- 您可以使用与 nft list ruleset 命令相同的格式编写脚本，显示规则集：

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

table inet example_table {
  chain example_chain {
    # Chain for incoming packets that drops all packets that
    # are not explicitly allowed by any rule in this chain
    type filter hook input priority 0; policy drop;

    # Accept connections to port 22 (ssh)
    tcp dport ssh accept
  }
}
```

- 您可以使用与 nft 命令相同的语法：

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset
```

```
# Create a table
add table inet example_table

# Create a chain for incoming packets that drops all packets
# that are not explicitly allowed by any rule in this chain
add chain inet example_table example_chain { type filter hook input priority 0 ; policy
drop ; }

# Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

7.2.2. 运行 nftables 脚本

您可以通过传递至 nft 实用程序或直接执行脚本来运行 nftables 脚本。

先决条件

- 本节的步骤假设您在 /etc/nftables/example_firewall.nft 文件中存储了 nftables 脚本。

流程

- 要运行 nftables 脚本，请将其传递给 nft 实用程序，请输入：

```
# nft -f /etc/nftables/example_firewall.nft
```

- 要直接运行 nftables 脚本：

  a. 只需要执行一次的步骤：

     i. 确保脚本以以下 shebang 序列开头：

```
#!/usr/sbin/nft -f
```

重要

如果省略 -f 参数，nf t 实用程序不会读取脚本并显示： Error: syntax error, unexpected newline, expecting string.

ii. 可选：将脚本的所有者设置为 root ：

```
# chown root /etc/nftables/example_firewall.nft
```

iii. 使脚本可以被其所有者执行：

```
# chmod u+x /etc/nftables/example_firewall.nft
```

b. 运行脚本：

```
# /etc/nftables/example_firewall.nft
```

如果没有输出结果，系统将成功执行该脚本。

重要

即使 nft 成功执行脚本，在脚本中错误地放置规则、缺少参数或其他问题都可能导致防火墙的行为不如预期。

其它资源

- chown(1) man page
- chmod(1) man page
- 系统引导时自动载入 nftables 规则

7.2.3. 使用 nftables 脚本中的注释

nftables 脚本环境将 # 字符右侧的所有内容都视为注释。

例 7.1. nftables 脚本中的注释

注释可在一行的开始，也可以在命令后：

```
...
# Flush the rule set
flush ruleset

add table inet example_table  # Create a table
...
```

7.2.4. 使用 nftables 脚本中的变量

要在 nftables 脚本中定义变量，请使用 define 关键字。您可以在变量中存储单个值和匿名集合。对于更复杂的场景，请使用 set 或 verdict 映射。

只有一个值的变量

以下示例定义了一个名为 INET_DEV 的变量，其值为 enp1s0 ：

```
define INET_DEV = enp1s0
```

您可以通过在变量名称后写入 $ 符号来使用脚本中的变量：

```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

包含匿名集合的变量

以下示例定义了一个包含匿名集合的变量：

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

您可以通过在变量名称后写入 $ 符号来使用脚本中的变量：

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```

注意

请注意，在规则中使用大括号时具有特殊的意义，因为它们表示变量代表一个集合。

其中资源

其它资源

- 使用 nftables 命令中的设置
- 在 nftables 命令中使用 verdict 映射

7.2.5. 在 nftables 脚本中包含文件

借助 nftables 脚本环境，管理员可以使用 include 语句包含 其他脚本。

如果您只指定没有绝对路径或相对路径的文件名，nftables 包括默认搜索路径中的文件，该路径设置为 Red Hat Enterprise Linux 上的 /etc。

例 7.2. 包含默认搜索目录中的文件

从默认搜索目录中包含一个文件：

```
include "example.nft"
```

例 7.3. 包含目录中的所有 *.nft 文件

包含以 *.nft 结尾且存储在 /etc/nftables/rulesets/ 目录中的所有文件：

```
include "/etc/nftables/rulesets/*.nft"
```

请注意，include 语句不匹配以点开头的文件。

其它资源

- nft(8)man page 中的 Include files 部分

7.2.6. 系统引导时自动载入 nftables 规则

nftables systemd 服务加载 /etc/sysconfig/nftables.conf 文件中包含的防火墙脚本。这部分论述了如何在系统引导时载入防火墙规则。

先决条件

- nftables 脚本存储在 /etc/nftables/ 目录中。

流程

1. 编辑 /etc/sysconfig/nftables.conf 文件。
   - 如果您在安装 nftables 软件包时增强了在 /etc/nftables/ 中创建的 *.nft 脚本，请取消注释这些脚本的 include 语句。
   - 如果您从头开始编写脚本，请添加 include 语句以包含这些脚本。例如，要在 nftables 服务启动时载入 /etc/nftables/example.nft 脚本，请添加：

     ```
     include "/etc/nftables/example.nft"
     ```

2. （可选）启动 nftables 服务在不重启系统的情况下载入防火墙规则：

   ```
   # systemctl start nftables
   ```

3. 启用 nftables 服务。

   ```
   # systemctl enable nftables
   ```

其它资源

- 支持的 nftables 脚本格式

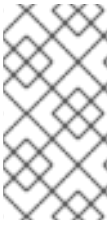
7.3. 创建和管理 NFTABLES 表、链和规则

本节介绍如何显示 nftables 规则集以及如何管理它们。

7.3.1. 标准链优先级值和文本名称

当您创建链时，优先级 可以设置整数值或标准名称，以指定具有相同 hook 值链的顺序。

名称和值的定义取决于 xtables 在注册其默认链时使用的优先级。



注意

nft list chain 命令默认显示文本优先级值。您可以通过将 -y 选项传递给命令来查看数字值。

例 7.4. 使用文本值设定优先级

以下命令使用标准优先级值 50 在 example_ table 中创建一个名为 example_ chain 的链：

```
# nft add chain inet example_table example_chain { type filter input priority 50 \; policy accept \; }
```

因为优先级是一个标准值，所以您可以使用文本值：

```
# nft add chain inet example_table example_chain { type filter input priority security \; policy accept \; }
```

表 7.1. 标准优先级名称、系列和 hook 兼容性列表

名称	值	系列	Hook
Raw	-300	**ip、ip6、inet**	all
mangle	-150	**ip、ip6、inet**	all
dstnat	-100	**ip、ip6、inet**	prerouting
filter	0	**ip、ip6、inet、arp、netdev**	all
security	50	**ip、ip6、inet**	all
srcnat	100	**ip、ip6、inet**	postrouting

所有系列都使用相同的值，但 网桥 系列使用以下值：

表 7.2. 网桥系列的标准优先级名称和 hook 兼容性

名称	值	Hook
dstnat	-300	prerouting
filter	-200	all
out	100	output
srcnat	300	postrouting

其它资源

- nft(8)man page 中的 Chains 部分

7.3.2. 显示 nftables 规则集

nftables 的规则集包含表、链和规则。本节介绍如何显示规则集。

流程

- 要显示规则集，请输入：

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport http accept
    tcp dport ssh accept
  }
}
```

注意

默认情况下，nftables 不预先创建表。因此，在没有表的情况下显示主机上设置的规则，nft list ruleset 命令不会显示任何输出。

7.3.3. 创建 nftables 表

nftables 中的表是包含链、规则、集合和其他对象的集合的命名空间。本节介绍如何创建表。

每个表都必须定义一个地址系列。表的地址系列定义了表进程的类型。在创建表时，您可以设置以下地址系列之一：

- ip ：仅匹配 IPv4 数据包。如果没有指定地址系列，这是默认设置。
- ip6 ：仅与 IPv6 数据包匹配。
- iNet ：匹配 IPv4 和 IPv6 数据包.
- ARP：匹配 IPv4 地址解析协议(ARP)数据包。
- 网桥 ：匹配遍历网桥设备的数据包。
- netdev ：匹配来自 ingress 的数据包。

流程

1. 使用 nft add table 命令创建新表。例如，要创建一个名为 example_table 的表，用于处理 IPv4 和 IPv6 数据包：

```
# nft add table inet example_table
```

2. 另外，还可列出规则集中的所有表：

```
# nft list tables
table inet example_table
```

其它资源

- nft(8)man page 中的 地址系列部分

- nft(8)man page 中的 Tables 部分

7.3.4. 创建 nftables 链

chains 是规则的容器。存在以下两种规则类型：

- 基本链：您可以使用基础链作为来自网络堆栈的数据包的入口点。
- 常规链：您可以使用常规链作为跳过目标，并更好地组织规则。

这个步骤描述了如何在现有表中添加基本链。

先决条件

- 已存在您要添加新链的表。

流程

1. 使用 nft add chain 命令创建新链。例如，要在 example_table 中创建一个名为 example_chain 的链：

```
# nft add chain inet example_table example_chain { type filter hook input priority 0 \; policy accept \; }
```

重要

为避免 shell 将分号解析为命令结尾，请在分号前加上 \ 转义字符。

这个链过滤传入的数据包。priority 参数指定 nftables 进程使用相同 hook 值链的顺序。较低优先级的值优先于优先级更高的值。policy 参数设置此链中规则的默认操作。请注意，如果您远程登录服务器，并将默认策略设置为 drop，如果没有其他规则允许远程访问，您将立即断开连接。

2. 另外，还可以显示所有链：

```
# nft list chains
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
  }
}
```

其它资源

- nft(8)man page 中的 地址系列部分
- nft(8)man page 中的 Chains 部分

7.3.5. 在 nftables 链末尾附加规则

本节介绍如何将规则附加到现有 nftables 链末尾。

先决条件

- 您要添加该规则的链已存在。

流程

1. 要添加新的规则，请使用 nft add rule 命令。例如，在 example_table 中的 example_chain 中添加一条规则，允许端口 22 上的 TCP 流量：

```
# nft add rule inet example_table example_chain tcp dport 22 accept
```

您可以选择指定服务名称而不是端口号。在该示例中，您可以使用 ssh 而不是端口号 22。请注意，服务名称会根据在 /etc/services 文件中的条目解析为端口号。

2. 另外，还可在 example_table 中显示所有链及其规则：

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    ...
```

```
    tcp dport ssh accept
  }
}
```

其它资源

- nft(8)man page 中的 地址系列部分
- n ft(8)man page 中的 Rules 部分

7.3.6. 在 nftables 链的开头插入规则

本节介绍如何在现有 nftables 链的开头插入规则。

先决条件

- 您要添加该规则的链已存在。

流程

1. 要插入新规则，请使用 nft 插入规则命令。例如，要在 example_ table 中将规则插入到 example_chain，该规则允许端口 22 上的 TCP 流量：

```
# nft insert rule inet example_table example_chain tcp dport 22 accept
```

您还可以指定服务名称而不是端口号。在该示例中，您可以使用 ssh 而不是端口号 22。请注意，服务名称会根据在 /etc/services 文件中的条目解析为端口号。

2. 另外，还可在 example_table 中显示所有链及其规则：

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept
    ...
  }
}
```

其它资源

- nft(8)man page 中的 地址系列部分
- n ft(8)man page 中的 Rules 部分

7.3.7. 在 nftables 链的特定位置插入规则

本节介绍如何在 nftables 链中现有规则前后插入规则。这样，您可以将新规则置于正确的位置。

先决条件

- 存在您要添加规则的链。

流程

1. 使用 nft -a list ruleset 命令显示 example_table 中的所有链及其规则，包括其句柄：

```
# nft -a list table inet example_table
table inet example_table { # handle 1
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 443 accept # handle 3
    tcp dport 389 accept # handle 4
  }
}
```

使用 -a 可显示句柄。您需要此信息才能在后续步骤中定位新规则。

2. 在 example_table 中的 example_chain 链中插入新规则 ：

- 要在处理 3 前插入允许 port636 上的 TCP 流量的规则，请输入 ：

```
# nft insert rule inet example_table example_chain position 3 tcp dport 636 accept
```

- 要在句柄 3 后添加允许端口 80 上的 TCP 流量的规则，请输入：

```
# nft add rule inet example_table example_chain position 3 tcp dport 80 accept
```

-

3. 另外，还可在 example_table 中显示所有链及其规则：

```
# nft -a list table inet example_table
table inet example_table { # handle 1
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    tcp dport 80 accept # handle 6
    tcp dport 389 accept # handle 4
  }
}
```

其它资源

- nft(8)man page 中的 地址系列部分
- n ft(8)man page 中的 Rules 部分

7.4. 使用 NFTABLES 配置 NAT

使用 nftables，您可以配置以下网络地址转换(NAT)类型：

- 伪装
- 源 NAT（SNAT）
- 目标 NAT（DNAT）
- 重定向

7.4.1. 不同的 NAT 类型： masquerading、source NAT、destination NAT 和 redirect

这些是不同的网络地址转换（NAT）类型：

伪装和源 NAT（SNAT）

使用以上 NAT 类型之一更改数据包的源 IP 地址。例如，互联网服务提供商不路由专用 IP 范围，如 10.0.0.0/8。如果您在网络中使用专用 IP 范围，并且用户应该能够访问 Internet 上的服务器，请将这些范围内的数据包源 IP 地址映射到公共 IP 地址。

伪装和 SNAT 都非常相似。不同之处是：

- 伪装自动使用传出接口的 IP 地址。因此，如果传出接口使用了动态 IP 地址，则使用伪装。
- SNAT 将数据包的源 IP 地址设置为指定的 IP 地址，且不会动态查找传出接口的 IP 地址。因此，SNAT 要比伪装更快。如果传出接口使用了固定 IP 地址，则使用 SNAT。

目标 NAT（DNAT）

使用此 NAT 类型重写传入数据包的目标地址和端口。例如，如果您的 Web 服务器使用私有 IP 范围内的 IP 地址，因此无法直接从互联网访问，您可以在路由器上设置 DNAT 规则以将传入的流量重定向到此服务器。

重定向

这个类型是 IDT 的特殊示例，它根据链 hook 将数据包重定向到本地机器。例如，如果服务在其标准端口的不同端口上运行，您可以将从标准端口传入的流量重定向到此特定端口。

7.4.2. 使用 nftables 配置伪装

伪装使路由器动态地更改通过接口到接口 IP 地址发送的数据包的源 IP。这意味着，如果接口被分配了新的 IP，nftables 会在替换源 IP 时自动使用新的 IP。

以下流程描述了如何将通过 ens3 接口离开主机的数据包源 IP 替换为 ens3 上设置的 IP。

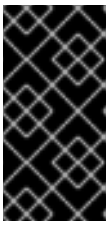
流程

1. 创建一个表：

```
# nft add table nat
```

2. 在表中添加 prerouting 和 postrouting 链：

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



重要

即使您没有向抢占链添加规则，nf tables 框架也要求此链与传入数据包回复匹配。

请注意，您必须将 -- 选项传递给 nft 命令，以避免 shell 将负优先级值解析为 nft 命令的选项。

3. 在 postrouting 链中添加与 ens3 接口上的传出数据包匹配的规则：

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

7.4.3. 使用 nftables 配置源 NAT

在路由器中，源 NAT（SNAT）可让您将通过接口发送的数据包 IP 改为专门的 IP 地址。

以下流程描述了如何将通过 ens3 接口离开路由器的数据包源 IP 替换为 192.0.2.1。

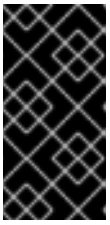
流程

1. 创建一个表：

```
# nft add table nat
```

2. 在表中添加 prerouting 和 postrouting 链：

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



重要

即使您没有向 postrouting 链添加规则，nf tables 框架也要求此链与传出数据包回复匹配。

请注意，您必须将 -- 选项传递给 nft 命令，以避免 shell 将负优先级值解析为 nft 命令的选项。

3. 在 postrouting 链中添加一条规则，通过 ens3 将传出数据包的源 IP 替换为 192.0.2.1 ：

```
# nft add rule nat postrouting oifname "ens3" snat to 192.0.2.1
```

其它资源

- Lin:https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/getting-started-with-nftables_configuring-and-managing-networking#forwarding-incoming-packets-on-a-specific-local-port-to-a-different-host_configuring-port-forwarding-using-nftables[Forwarding 在特定本地端口上传入到不同主机的数据包]

7.4.4. 使用 nftables 配置目标 NAT

目标 NAT 可让您将路由器中的流量重新指向无法直接从互联网访问的主机。

以下流程描述了如何将发送到路由器的端口 80 和 443 的传入流量重定向到使用 192.0.2.1 IP 地址的主机。

流程

1. 创建一个表：

```
# nft add table nat
```

2. 在表中添加 prerouting 和 postrouting 链：

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```

重要

即使您没有向 postrouting 链添加规则，nf tables 框架也要求此链与传出数据包回复匹配。

请注意，您必须将 -- 选项传递给 nft 命令，以避免 shell 将负优先级值解析为 nft 命令的选项。

3. 在 prerouting 链中添加一条规则，将发送到端口 80 和 443 的 ens3 接口上传入的流量重定向到具有 192.0.2.1 IP 的主机：

```
# nft add rule nat prerouting iifname ens3 tcp dport { 80, 443 } dnat to 192.0.2.1
```

4. 根据您的环境，添加 SNAT 或伪装规则以更改源地址：

   a. 如果 ens3 接口使用动态 IP 地址，请添加一个伪装规则：

   ```
   # nft add rule nat postrouting oifname "ens3" masquerade
   ```

   b. 如果 ens3 接口使用静态 IP 地址，请添加 SNAT 规则。例如，如果 ens3 使用 198.51.100.1 IP 地址：

   ```
   # nft add rule nat postrouting oifname "ens3" snat to 198.51.100.1
   ```

其它资源

- 不同的 NAT 类型： masquerading、source NAT、destination NAT 和 redirect

7.4.5. 使用 nftables 配置重定向

重定向 功能是目标网络地址转换(DNAT)的一种特殊情况，它根据链 hook 将数据包重定向到本地计算机。

以下流程描述了如何将发送到本地主机的端口 22 的传入和转发的流量重定向到端口 2222。

流程

1. 创建一个表：

```
# nft add table nat
```

2. 在表中添加 prerouting chain:

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

请注意，您必须将 -- 选项传递给 nft 命令，以避免 shell 将负优先级值解析为 nft 命令的选项。

3. 在 prerouting 链中添加一条规则，将端口 22 上传入的流量重定向到端口 2222 ：

```
# nft add rule nat prerouting tcp dport 22 redirect to 2222
```

其它资源

- 不同的 NAT 类型： masquerading、source NAT、destination NAT 和 redirect

7.5. 使用 NFTABLES 命令中的设置

nftables 框架原生支持集合。您可以使用一个集合，例如，规则匹配多个 IP 地址、端口号、接口或其他匹配标准。

7.5.1. 在 nftables 中使用匿名集合

匿名集合包含以逗号分开的值，用逗号分开，如 { 22、80、443 }，它们直接在规则中使用。您还可以将匿名集合用于 IP 地址或其他匹配标准。

匿名集合的缺陷是，如果要更改集合，则需要替换规则。对于动态解决方案，使用命名的集合，如在 nftables 中使用命名集中所述。

先决条件

- inet 系列中的 example_chain 链和 example_table 表存在。

流程

1. 例如，在 example_ table 中的 example_chain 中添加允许传入流量到端口 22、80 和 443 的规则：

```
# nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

2. 另外，还可在 example_table 中显示所有链及其规则：

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport { ssh, http, https } accept
  }
}
```

7.5.2. 在 nftables 中使用命名集

nftables 框架支持 mutable 命名集。命名集是一个列表或一组元素，您可以在表中的多个规则中使用。匿名集合的另外一个好处在于，您可以更新命名的集合而不必替换使用集合的规则。

当您创建一个命名集时，必须指定集合包含的元素类型。您可以设置以下类型：

- 包含 IPv4 地址或范围的集合的 ipv4_addr，如 192.0.2.1 或 192.0.2.0/24。
- 包含 IPv 6 地址或范围的集合的 ipv6_addr，如 2001:db8:1::1 或 2001:db8:1::1/64。
- ether_addr，用于包含介质访问控制(MAC)地址列表的集合，如 52:54:00:6b:66:42。
- inet_proto，用于包含 Internet 协议类型列表的集合，如 tcp。
- 包含互联网服务列表集合的 inet_service，如 ssh。
- 包含数据包标记列表的集合标记。数据包标记可以是任意正 32 位整数值（0 到 2147483647）。

先决条件

- example_chain 链和 example_table 表存在。

流程

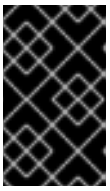
1. 创建一个空集。以下示例为 IPv4 地址创建了一个集合：

   - 要创建可存储多个独立 IPv4 地址的集合：

     ```
     # nft add set inet example_table example_set { type ipv4_addr \; }
     ```

   - 要创建可存储 IPv4 地址范围的集合：

     ```
     # nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
     ```

   重要

   要避免 shell 认为分号作为命令结尾，您必须用反斜杠转义分号。

2. 另外，还可创建使用该集合的规则。例如，以下命令向 example_ table 中的 example_chain 添加一条规则，该规则将丢弃来自 example_ set 中的 IPv4 地址的所有数据包。

   ```
   # nft add rule inet example_table example_chain ip saddr @example_set drop
   ```

   因为 example_set 仍然为空，因此该规则目前无效。

3. 在 example_set 中添加 IPv4 地址：

   - 如果您创建存储单个 IPv4 地址的集合，请输入：

     ```
     # nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
     ```

- 如果您创建存储 IPv4 范围的集合，请输入：

```
# nft add element inet example_table example_set { 192.0.2.0-192.0.2.255 }
```

当您指定 IP 地址范围时，也可以使用无类别域间路由(CIDR)标记，如上例中的 192.0.2.0/24。

7.5.3. 其它资源

- nft(8)man page 中的 Sets 部分

7.6. 在 NFTABLES 命令中使用 VERDICT 映射

平均字典映射（也称为字典）使 nft 通过映射匹配到操作条件来基于数据包信息执行操作。

7.6.1. 在 nftables 中使用匿名映射

匿名映射是您直接在规则中使用的 { match_criteria : action } 语句。这个语句可以包含多个用逗号分开的映射。

匿名映射的缺点是，如果要更改映射，则必须替换规则。对于动态解决方案，请使用命名映射，如在 nftables 中使用命名映射中所述。

这个示例描述了如何使用匿名映射将 IPv4 和 IPv6 协议的 TCP 和 UDP 数据包路由到不同的链，以分别计算传入的 TCP 和 UDP 数据包。

流程

1. 创建 example_table:

```
# nft add table inet example_table
```

2. 在 example_table 中创建 tcp_packets 链：

```
# nft add chain inet example_table tcp_packets
```

3. 在 tcp_packets 中添加计算此链中流量的规则：

```
# nft add rule inet example_table tcp_packets counter
```

4. 在 example _table 中创建 theudp_ packets链

```
# nft add chain inet example_table udp_packets
```

5. 添加一条计算此链中流量的规则 toudp_packets ：

```
# nft add rule inet example_table udp_packets counter
```

6. 为传入的流量创建一个链。例如，在 example_table 中创建一个名为 incoming_traffic 的链，用于过滤传入的流量：

```
# nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7. 将带有匿名映射的规则添加到 incoming_traffic ：

```
# nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump
tcp_packets, udp : jump udp_packets }
```

匿名映射区分数据包，并根据它们的协议将它们发送到不同的计数链。

8. 要列出流量计数器，显示 example_table:

```
# nft list table inet example_table
table inet example_table {
  chain tcp_packets {
    counter packets 36379 bytes 2103816
  }

  chain udp_packets {
    counter packets 10 bytes 1559
  }

  chain incoming_traffic {
    type filter hook input priority filter; policy accept;
```

```
    ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
  }
}
```

tcp_packets 和udp_packets 链中的计数器同时显示收到的数据包数和字节数。

7.6.2. 在 nftables 中使用命名映射

nftables 框架支持命名映射。您可以在表中的多个规则中使用这些映射。匿名映射的另一个优势在于，您可以更新命名映射而不替换使用它的规则。

在创建命名映射时，您必须指定元素类型：

- ipv4_addr 用于匹配部分包含 IPv4 地址的映射，如 192.0.2.1。
- ipv6_addr 用于匹配部分包含 IPv6 地址的映射，如 2001:db8:1::1。
- 匹配部分包含介质访问控制(MAC)地址的映射的 ether_addr，如 52:54:00:6b:66:42。
- inet_proto 用于匹配部分包含 Internet 协议类型（如 tcp ）的映射。
- 匹配部分包含互联网服务名称端口号（如 ssh 或 22 ）的映射的 inet_service。
- 为匹配部分包含数据包标记的映射添加标记。数据包标记可以是任意正 32 位整数值（0 到 2147483647）。
- 映射的计数器，其匹配部分包含计数器值。计数器值可以是任意正 64 位整数值。
- 匹配部分包含配额值的映射配额。配额值可以是任意正 64 位整数值。

这个示例论述了如何根据源 IP 地址允许或丢弃传入的数据包。使用命名映射时，您只需要一条规则来配置这种情况，同时 IP 地址和操作会动态存储在映射中。此流程还描述了如何从映射中添加和删除条

目。

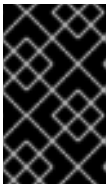
流程

1. 创建表。例如，要创建一个名为 example_table 的表来处理 IPv4 数据包：

```
# nft add table ip example_table
```

2. 创建链。例如，要在 example_table 中创建一个名为 example_chain 的链：

```
# nft add chain ip example_table example_chain { type filter hook input priority 0 \; }
```



重要

要避免 shell 认为分号作为命令结尾，您必须用反斜杠转义分号。

3. 创建一个空的映射。例如，要为 IPv4 地址创建映射：

```
# nft add map ip example_table example_map { type ipv4_addr : verdict \; }
```

4. 创建使用该映射的规则。例如，以下命令为 example_ table 中的 example_chain 添加了一个规则，它把操作应用到 example_map 中定义的 IPv4 地址：

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. 在 example_map 中添加 IPv4 地址和对应的操作：

```
# nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

这个示例定义了 IPv4 地址到操作的映射。根据上述规则，防火墙接受来自 192.0.2.1 的数据包并丢弃来自 192.0.2 的数据包。

6. 另外，还可添加另一个 IP 地址和 action 语句来增强映射：

```
# nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7. （可选）从映射中删除条目：

```
# nft delete element ip example_table example_map { 192.0.2.1 }
```

8. 另外，还可显示规则集：

```
# nft list ruleset
table ip example_table {
  map example_map {
    type ipv4_addr : verdict
    elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
  }

  chain example_chain {
    type filter hook input priority filter; policy accept;
    ip saddr vmap @example_map
  }
}
```

7.6.3. 其它资源

- nft(8)man page 中的 Maps 部分

7.7. 使用 NFTABLES 配置端口转发

端口转发可让管理员将发送到特定目的端口的数据包转发到不同的本地或者远程端口。

例如，如果您的 Web 服务器没有公共 IP 地址，您可以在防火墙上设置端口转发规则，该规则将在防火墙的端口 80 和 443 上转发传入的数据包到 Web 服务器。使用这个防火墙规则，互联网中的用户可以使用防火墙的 IP 或主机名访问网页服务器。

7.7.1. 将传入的数据包转发到不同的本地端口

这部分论述了如何在端口 8022 上将传入的 IPv4 数据包转发到本地系统上的端口 22 的示例。

流程

1. 使用 ip 地址系列创建一个名为 nat 的表：

```
# nft add table ip nat
```

2. 在表中添加 prerouting 和 postrouting 链：

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
```

注意

将 -- 选项传递给 nft 命令，以避免 shell 将负优先级值解析为 nft 命令的选项。

3. 在 prerouting 链中添加一条规则，将端口 8022 上传入的数据包重定向到本地端口 22 ：

```
# nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

7.7.2. 将特定本地端口上传入的数据包转发到不同主机

您可以使用目标网络地址转换（DNAT）规则将本地端口上传入的数据包转发到远程主机。这可让互联网中的用户访问使用专用 IP 地址在主机上运行的服务。

这个步骤描述了如何将本地端口 443 中传入的 IPv4 数据包转发到 IP 地址为 192.0.2.1 的远程系统上的相同端口号。

先决条件

- 您以 root 用户身份登录应该转发数据包。

流程

1. 使用 ip 地址系列创建一个名为 nat 的表：

```
# nft add table ip nat
```

2. 在表中添加 prerouting 和 postrouting 链：

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain ip nat postrouting { type nat hook postrouting priority 100 \; }
```

注意

将 -- 选项传递给 nft 命令，以避免 shell 将负优先级值解析为 nft 命令的选项。

3. 在 prerouting chain 中添加一条规则，将端口 443 上传入的数据包重新指向 192.0.2.1 上的同一端口：

```
# nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4. 为 postrouting 链添加一条规则伪装出站流量：

```
# nft add rule ip daddr 192.0.2.1 masquerade
```

5. 启用数据包转发：

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

7.8. 使用 NFTABLES 来限制连接数量

您可以使用 nftables 来限制连接数量或阻止建立给定连接量的 IP 地址，以防止它们使用过多的系统资源。

7.8.1. 使用 nftables 限制连接数量

nft 实用程序的 ct count 参数允许管理员限制连接数量。这个步骤描述了如何限制进入的连接的基本示例。

先决条件

- example_table 中的基本 example_chain 存在。

流程

1. 添加一条规则，该规则只允许从 IPv4 地址同时连接到 SSH 端口（22），并从同一 IP 拒绝所有后续连接：

```
# nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
```

2. 另外，还可以显示上一步中创建的 meter：

```
# nft list meter ip example_table example_meter
table ip example_table {
  meter example_meter {
    type ipv4_addr
    size 65535
    elements = { 192.0.2.1 : ct count over 2 , 192.0.2.2 : ct count over 2  }
  }
}
```

元素 条目显示目前与该规则匹配的地址。在本例中，元素 列出了已连接到 SSH 端口的 IP 地址。请注意，输出不会显示活跃连接的数量，或者连接是否被拒绝。

7.8.2. 在一分钟内尝试超过十个进入的 TCP 连接的 IP 地址

nftables 框架可让管理员动态更新设置。本节解释了如何使用这个功能临时阻止在一分钟内建立十个 IPv4 TCP 连接的主机。五分钟后，nftables 会自动从拒绝列表中删除 IP 地址。

流程

1. 使用 ip 地址系列创建 过滤器 表：

```
# nft add table ip filter
```

2. 在 过滤器 表中添加 输入 链：

```
# nft add chain ip filter input { type filter hook input priority 0 \; }
```

3. 在 过滤器 表中添加名为 denylist 的集合：

```
# nft add set ip filter denylist { type ipv4_addr \; flags dynamic, timeout \; timeout 5m \; }
```

这个命令为 IPv4 地址创建动态设置。timeout 5m 参数定义 nftables 在 5 分钟后自动删除条目。

4. 添加一条规则，该规则会在一分钟内试图建立十个新的 TCP 连接的主机源 IP 地址添加到 denylist 集合中：

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked limit rate over 10/minute add @denylist { ip saddr }
```

5. 添加一条规则，丢弃来自 denylist 集中 IP 地址的所有连接：

```
# nft add rule ip filter input ip saddr @denylist drop
```

其它资源

- 在 nftables 中使用命名集

7.9. 调试 NFTABLES 规则

nftables 框架为管理员提供了不同的选项来调试规则，并在数据包匹配时提供不同的选项。本节描述了这些选项。

7.9.1. 创建带有计数器的规则

在识别规则是否匹配时，可以使用计数器。本节描述了如何创建带有计数器的新规则。

- 有关在现有规则中添加计数器的步骤的更多信息，请参阅 在配置和管理网络中在现有规则中添加计数器

先决条件

- 您要添加该规则的链已存在。

流程

1. 向 链添加带有 counter 参数的新规则。以下示例添加一个带有计数器的规则，它允许端口 22 上的 TCP 流量，并计算与这个规则匹配的数据包和网络数据的数量：

```
# nft add rule inet example_table example_chain tcp dport 22 counter accept
```

2. 显示计数器值：

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

7.9.2. 在现有规则中添加计数器

在识别规则是否匹配时，可以使用计数器。本节论述了如何在现有规则中添加计数器。

- 有关使用计数器添加新规则的步骤的更多信息，请参阅配置和管理网络中的计数器创建规则

先决条件

- 您要添加计数器的规则已存在。

流程

1. 在链中显示规则及其句柄：

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. 通过替换规则但替换为 counter 参数来添加计数器。以下示例替换了上一步中显示的规则并添加计数器：

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter
accept
```

3. 显示计数器值：

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

7.9.3. 监控与现有规则匹配的数据包

nftables 中的追踪功能与 nft monitor 命令相结合，使管理员可以显示与规则匹配的数据包。该流程描述了如何为规则启用追踪以及与本规则匹配的监控数据包。

先决条件

- 您要添加计数器的规则已存在。

流程

1. 在链中显示规则及其句柄：

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. 通过替换规则但使用 meta nftrace 设置 1 参数来添加追踪功能。以下示例替换了上一步中显示的规则并启用追踪：

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace set 1 accept
```

3. 使用 nft monitor 命令显示追踪。以下示例过滤命令的输出，仅显示包含 inet example_table example_chain 的条目：

```
# nft monitor | grep "inet example_table example_chain"
trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr
```

```
52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip
dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728
tcp dport ssh tcp flags == syn tcp window 64240
trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nftrace set 1
accept (verdict accept)
...
```

警告

根据启用追踪的规则数量以及匹配的流量数量，nft monitor 命令可以显示大量输出。使用 grep 或其他实用程序过滤输出。

7.10. 备份和恢复 NFTABLES 规则集

本节论述了如何将 nftables 规则备份到文件，以及从文件中恢复规则。

管理员可以使用具有规则的文件将规则传送到不同的服务器。

7.10.1. 将 nftables 规则设置为文件

本节论述了如何将 nftables 规则集备份到文件。

流程

- 备份 nftables 规则：
  - nft 列表规则集 格式：

```
# nft list ruleset > file.nft
```

  - JSON 格式：

```
# nft -j list ruleset > file.json
```

7.10.2. 从文件中恢复 nftables 规则集

本节论述了如何恢复 nftables 规则集。

流程

- 恢复 nftables 规则：
  - 如果要恢复的文件采用 nft 列表规则集 格式，或者包含 nft 命令：

```
# nft -f file.nft
```

  - 如果要恢复的文件采用 JSON 格式：

```
# nft -j -f file.json
```

7.11. 其它资源

- 在 Red Hat Enterprise Linux 8 中使用 nftables
- iptables 之后会发生什么？其后续，当然：nftables
- firewalld：未来是 nftables