Red Hat Enterprise Linux 8

安全网络

配置安全网络和网络通信

Last Updated: 2024-07-14

Red Hat Enterprise Linux 8 安全网络

配置安全网络和网络通信

法律通告

摘要

了解工具和技术以提高网络安全性，并降低数据泄露和入侵的风险。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您的改进建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

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到服务器主机系统。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 包括基本的 **OpenSSH** 软件包：通用的 **openssh** 软件包、**openssh-server** 软件包以及 **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。

当使用 SSH 协议进行远程 shell 登录或文件复制时，SSH 协议可以缓解威胁，例如，拦截两个系统之间的通信和模拟特定主机。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。当首次安装 OpenSSH 或主机第一次引导时，主机密钥是自动生成的加密密钥。

OpenSSH 是 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** 是一个将本地公钥添加到远程 SSH 服务器上的 **authorized_keys** 文件中的脚本。
- **ssh-keyscan** 可以收集 SSH 公共主机密钥。

RHEL 中的 OpenSSH 套件仅支持 SSH 版本 2。它有一个增强的密钥交换算法，其不会受到较旧版本 1 中已知的漏洞的影响。

OpenSSH 作为 RHEL 的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。要修改策略，管理员必须使用 **update-crypto-policies** 命令来调整设置，或者手动选择不使用系统范围的加密策略。

OpenSSH 套件使用两组配置文件：一个用于客户端程序（即 **ssh**、**scp** 和 **sftp**），另一个用于服务器（**sshd** 守护进程）。

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。

其他资源

其他资源

- 使用 **man -k ssh** 命令显示 man page
- 使用系统范围的加密策略

1.2. 配置并启动 OPENSSH 服务器

您可以更改 OpenSSH 服务器的欢迎横幅和 IP 地址。您还可以切换到较慢的动态网络配置。

请注意，在默认 RHEL 安装后，**sshd** 守护进程已经启动，服务器主机密钥会被自动创建。

先决条件

- 已安装 **openssh-server** 软件包。

流程

1. 如果 **sshd** 服务还没有运行，请在当前会话中启动它，并将其设置为在引导时自动启动：

```
# systemctl enable --now 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，并且其值包含 **/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** 以应用修改：

```
# 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:
```

其他资源

- **sshd(8)** 和 **sshd_config(5)** 手册页。

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，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
```

其他资源

- **sshd(8)**, **sshd_config(5)** 和 **setsebool(8)** 手册页。

1.4. 生成 SSH 密钥对

您可以通过在本地系统上生成 SSH 密钥对并将生成的公钥复制到 OpenSSH 服务器来在没有提供密码的情况下登录到 OpenSSH 服务器。必须将服务器配置为允许此选项。

先决条件

- 您以 Linux 用户身份登录，该用户将连接到 OpenSSH 服务器。如果以 **root** 身份完成以下步骤，则只有 **root** 用户才能使用该密钥。

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/<username>/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_ecdsa.
Your public key has been saved in /home/<username>/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
<username>@<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. 将公钥复制到远程机器中：

```
$ ssh-copy-id <username>@<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
<username>@<ssh-server-example.com>'s password:
...
Number of key(s) added: 1

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

将 & ***lt;username> @ <ssh-server-example.com&*** gt; 替换为您的凭证。

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。

3. 可选：要在重新安装系统之间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 <username>@<ssh-server-example.com>
Welcome message.
...
Last login: Mon Nov 18 18:28:42 2019 from ::1
```

其他资源

- **ssh-keygen(1)** 和 **ssh-copy-id(1)** 手册页。

1.5. 使用保存在智能卡中的 SSH 密钥

您可以使用保存在 OpenSSH 客户端智能卡中的 RSA 和 ECDSA 密钥。使用智能卡进行验证替换了默认密码验证。

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** 包装器，并且 OpenSC PKCS #11 模块是注册到 PKCS#11 Kit 的，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:在 OpenSSH 中提供更强大的智能卡
- **p11-kit (8)**, **opensc.conf (5)**, **pcscd (8)**, **ssh (1)**, 和 **ssh-keygen (1)** man page

1.6. 使 OPENSSH 更安全

在使用 OpenSSH 时，您可以调整系统以提高安全性。

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
```

-

警告

大多数安全强化配置更改会降低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)能够在同等的对称密钥强度下，提供比 RSA 更好的性能。它还会生成较短的密钥。Ed25519 公钥算法是 一种变形的 Edwards 曲线的实现，其比 RSA、DSA 和 ECDSA 更安全，也更快。
  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 中配置主机密钥创建，请使用 **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
```

注意

在启用了 **cloud-init** 的镜像中，**ssh-keygen** 单元会自动禁用。这是因为 **ssh-keygen 模板** 服务可能会干扰 **cloud-init** 工具，并导致主机密钥生成问题。要防止这些问题 **etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** drop-in 配置文件禁用 **ssh-keygen** 单元（如果 **cloud-init** 正在运行）。

- 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
```

重要

Ed25519 算法不符合 FIPS-140，OpenSSH 在 FIPS 模式下无法使用 Ed25519 密钥。

非默认端口

- 默认情况下，**sshd** 守护进程侦听 TCP 端口 22。更改端口可降低系统因自动网络扫描而受到攻击的风险，从而提高安全性。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 --remove-port=22/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以 root 用户身份登录，您可以在 **/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了哪些特权命令，然后获得了 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，而不是使用密码以 root 身份登录，并通过防止暴力攻击来降低风险。

使用 X 安全性扩展

- 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** 来更准确地限制访问，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 192.168.1.2 地址的系统除外。所有用户都必须在 **example-group** 组中。OpenSSH 服务器拒绝所有其他连接。

OpenSSH 服务器仅允许通过 **/etc/ssh/sshd_config** 中所有 Allow 和 Deny 指令的连接。例如，如果 **AllowUsers** 指令列出的用户不是 **AllowGroups** 指令中列出的组的一部分，则用户无法登录。

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

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
```

警告

如果您的系统在互联网上进行通信，则可能会因为 **FUTURE** 策略的严格设置而面临互操作性问题。

您还可以只通过系统范围的加密策略为 SSH 协议禁用特定的密码。如需更多信息，请参阅 安全强化 文档中的 使用子策略自定义系统范围的加密策略 部分。

要为您的 OpenSSH 服务器选择不使用系统范围的加密策略，请在 **/etc/sysconfig/sshd** 文件中取消具有 **CRYPTO_POLICY=** 变量的行的注释。更改后，您在 **/etc/ssh/sshd_config** 文件中的 **Ciphers** 、**MAC** 、**KexAlgoritms** 和 **GSSAPIKexAlgorithms** 部分指定的值不会被覆盖。

详情请查看 **sshd_config(5)** 手册页。

要为您的 OpenSSH 客户端选择不使用系统范围的加密策略，请执行以下任务之一：

- 对于给定的用户，使用 **~/.ssh/config** 文件中特定于用户的配置覆盖全局 **ssh_config**。
- 对于整个系统，在 **/etc/ssh/ssh_config.d/** 目录中的置入配置文件中指定加密策略，使用小于 5 的两位数字前缀，以便其在字典顺序上位于 **05-redhat.conf** 文件之前，并带有 **.conf** 后缀，例如 **04-crypto-policy-override.conf**。

其他资源

- **sshd_config(5)**、**ssh-keygen(1)**、 **crypto-policies(7)** 和 **update-crypto-policies(8)** 手册页。
- 安全强化 文档中的 使用系统范围的加密策略 。
- 如何只为 ssh 服务禁用特定的算法和密码 文章。

1.7. 使用 SSH 跳过主机连接到远程服务器

您可以通过中间服务器（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** 文件来定义跳板主机，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板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。

4. 您可以指定更多的跳板服务器，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
```

   如果跳板服务器上的用户名或 SSH 端口与远程服务器上的用户名和端口不同，请只修改上一命令中的主机名表示法，例如：

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

其他资源

- **ssh_config(5)** 和 **ssh(1)** 手册页.

1.8. 通过 SSH-AGENT，使用 SSH 密钥连接到远程机器

为了避免在每次发起 SSH 连接时输入密语，您可以使用 **ssh-agent** 工具缓存 SSH 私钥。确保私钥和密语安全。

先决条件

- 您有一个运行 SSH 守护进程的远程主机，并且可通过网络访问。
- 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。

流程

1. 可选：验证您是否可以使用密钥来对远程主机进行身份验证：
   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. 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 系统上同时配置 SSH 客户端。

1.9.1. sshd RHEL 系统角色的变量

在 **sshd** 系统角色 playbook 中，您可以根据您的首选项和限制定义 SSH 配置文件的参数。

如果没有配置这些变量，系统角色会生成一个与 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 来定义多行配置项。例如：

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

呈现为：

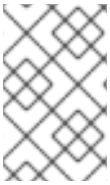
```
ListenAddress 0.0.0.0
ListenAddress ::
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** directory

1.9.2. 使用 sshd RHEL 系统角色配置 OpenSSH 服务器

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



注意

您可以将 **sshd** RHEL 系统角色用于更改 SSH 和 SSHD 配置的其他 RHEL 系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **sshd** 角色使用命名空间(RHEL 8 和更早的版本)或 drop-in 目录(RHEL 9)。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
- name: SSH server configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure sshd to prevent root and password login except from particular subnet
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd:
          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
- 只对子网 **192.0.2.0/24** 启用密码和 **root** 用户登录

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 登录到 SSH 服务器：

```
$ ssh <username>@<ssh_server>
```

2. 验证 SSH 服务器中的 **sshd_config** 文件的内容：

```
$ cat /etc/ssh/sshd_config
...
PasswordAuthentication no
PermitRootLogin no
...
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@<ssh_server>
   ```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件

- **/usr/share/doc/rhel-system-roles/sshd/** directory

1.9.3. ssh RHEL 系统角色的变量

在 **ssh** 系统角色 playbook 中，您可以根据您的首选项和限制定义客户端 SSH 配置文件的参数。

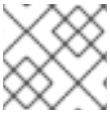
如果没有配置这些变量，系统角色会生成一个与 RHEL 默认值匹配的全局 **ssh_config** 文件。

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

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.ssh/README.md** file
- **/usr/share/doc/rhel-system-roles/ssh/** directory

1.9.4. 使用 ssh RHEL 系统角色配置 OpenSSH 客户端

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

注意

您可以将 **ssh** RHEL 系统角色用于更改 SSH 和 SSHD 配置的其他系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **ssh** 角色使用置入目录（在 RHEL 8 及更新的版本中使用）。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
```

```
- name: SSH client configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: "Configure ssh clients"
      ansible.builtin.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: server.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 **server. example.com** 主机的示例别名是 **user1**。
- 创建 **示例** 主机别名，它代表使用 **user1** 用户名连接到 **server.example.com** 主机。
- X11 转发被禁用。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 通过显示 SSH 配置文件来验证受管节点是否具有正确的配置：

```
# cat ~/root/.ssh/config
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.ssh/README.md** file
- **/usr/share/doc/rhel-system-roles/ssh/** directory

1.9.5. 将 sshd RHEL 系统角色用于非独占配置

通常，应用 **sshd** 系统角色会覆盖整个配置。如果您之前已调整了配置，例如使用不同的系统角色或 playbook，这可能会出现问题。要只对所选配置选项应用 **sshd** 系统角色，同时保留其他选项，您可以使用非排除配置。

您可以应用非独占配置：

- 在 RHEL 8 及更早版本中，使用配置片断。
- 在 RHEL 9 及更高版本中，使用置入目录中的文件。默认配置文件已放入随时可访问的目录中，存为 **/etc/ssh/sshd_config.d/00-ansible_system_role.conf**。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：
   - 对于运行 RHEL 8 或更早版本的受管节点：

```
---
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure SSHD to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_namespace: <my-application>
        sshd:
          # Environment variables to accept
          AcceptEnv:
            LANG
            LS_COLORS
            EDITOR
```

   - 对于运行 RHEL 9 或更高版本的受管节点：

```
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure sshd to accept some useful environment variables>
```

```
ansible.builtin.include_role:
  name: rhel-system-roles.sshd
vars:
  sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
  sshd:
    # Environment variables to accept
    AcceptEnv:
      LANG
      LS_COLORS
      EDITOR
```

在 **sshd_config_file** 变量中，定义 **sshd** 系统角色在其中写入配置选项的 **.conf** 文件。使用两位前缀，例如 ***42-*** 来指定应用配置文件的顺序。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 验证 SSH 服务器上的配置：
  - 对于运行 RHEL 8 或更早版本的受管节点：

```
# cat /etc/ssh/sshd_config.d/42-my-application.conf
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

  - 对于运行 RHEL 9 或更高版本的受管节点：

```
# cat /etc/ssh/sshd_config
...
# BEGIN sshd system role managed block: namespace <my-application>
Match all
  AcceptEnv LANG LS_COLORS EDITOR
# END sshd system role managed block: namespace <my-application>
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** directory

1.10. 其他资源

- **sshd(8)**、**ssh(1)**、**scp(1)**、**sftp(1)**、**ssh-keygen(1)**、**ssh-copy-id(1)**、**ssh_config(5)**、**ssh_config(5)**、**update-crypto-policies(8)** 和 **crypto-policies(7)** 手册页
- 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 创建和管理 TLS 密钥和证书

您可以使用 TLS （传输层安全）协议加密在两个系统间传输的通信。此标准将非对称加密用于私钥和公钥、数字签名和证书。

2.1. TLS 证书

TLS （传输层安全）是一个协议，它允许客户端-服务器应用程序安全地传递信息。TLS 使用公钥和私钥对系统来加密在客户端和服务器间传输的通信。TLS 是到 SSL 的后继协议（安全套接字层）。

TLS 使用 X.509 证书将主机名或机构等身份绑定到使用数字签名的公共密钥。X.509 是一个定义公钥证书格式的标准。

安全应用程序的身份验证取决于应用证书中公钥值的完整性。如果攻击者用自己的公钥替换公钥，它可以模拟真正的应用程序，并获得安全数据的访问权限。为防止此类攻击，所有证书都必须由证书颁发机构(CA)签名。CA 是一个可信节点，用于确认证书中公钥值的完整性。

CA 通过添加公钥签名并发布证书来签署公钥。数字签名是用 CA 的私钥编码的消息。通过分发 CA 的证书，CA 的公钥可供应用程序使用。应用程序使用 CA 的公钥解码 CA 的数字签名来验证证书是否有效签名。

要让证书由 CA 签名，您必须生成公钥，并将其发送到 CA 以进行签名。这称为证书签名请求(CSR)。CSR 也包含证书的可分辨名称(DN)。您可以为任一证书提供的 DN 信息可以包括您所在国家/地区的两字母国家/地区代码、您的州全名或省、您的城市或 town，即您的机构名称、电子邮件地址，也可以为空。许多当前商业 CA 首选 Subject Alternative Name 扩展，并在 CSR 中忽略 DN。

RHEL 为使用 TLS 证书提供了两个主要工具包：GnuTLS 和 OpenSSL。您可以使用 **openssl** 软件包中的 **openssl** 实用程序创建、读取、签名和验证证书。**gnutls-utils** 软件包提供的 **certtool** 工具可以使用不同的语法以及后端中的所有不同库集合执行相同的操作。

其他资源

- RFC 5280：互联网 X.509 公钥基础架构证书和证书撤销列表(CRL)配置文件
- **openssl (1)**, **x509 (1)**, **ca (1)**, **req (1)**, 和 **certtool (1)** man page

2.2. 使用 OPENSSL 创建私有 CA

当您的情况需要验证您的内部网络内实体时，私有证书颁发机构(CA)非常有用。例如，当基于您控制下的 CA 签名的证书或通过身份验证创建 VPN 网关时，或者您不想支付商业 CA 时，使用私有 CA。要在这样的用例中签署证书，私有 CA 使用自签名证书。

前提条件

- 您有 **root** 权限或使用 **sudo** 输入管理命令的权限。需要此类权限的命令标记为 **#**。

流程

1. 为您的 CA 生成私钥。例如，以下命令会创建一个 256 位 Elliptic Curve 数字签名算法(ECDSA)密钥：

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <ca.key>
```

密钥生成过程的时间取决于主机的硬件和熵、所选算法以及密钥长度。

2. 使用上一个命令生成的私钥创建证书：

```
$ openssl req -key <ca.key> -new -x509 -days 3650 -addext
keyUsage=critical,keyCertSign,cRLSign -subj "/CN=<Example CA>" -out <ca.crt>
```

生成的 **ca.crt** 文件是一个自签名 CA 证书，可用来为其他证书签名 10 年。对于私有 CA，您可以将 *<Example CA>* 替换为任何字符串作为通用名称(CN)。

3. 对 CA 的私钥设置安全权限，例如：

```
# chown <root>:<root> <ca.key>
# chmod 600 <ca.key>
```

后续步骤

- 要将自签名 CA 证书用作客户端系统上的信任锚，请将 CA 证书复制到客户端，并以 **root** 用户身份将其添加到客户端的系统范围信任存储中：

```
# trust anchor <ca.crt>
```

如需更多信息，请参阅 第 3 章 *使用共享的系统证书*。

验证

1. 创建证书签名请求(CSR)，并使用您的 CA 为请求签名。CA 必须成功基于 CSR 创建证书，例如：

```
$ openssl x509 -req -in <client-cert.csr> -CA <ca.crt> -CAkey <ca.key> -CAcreateserial -
days 365 -extfile <openssl.cnf> -extensions <client-cert> -out <client-cert.crt>
Signature ok
subject=C = US, O = Example Organization, CN = server.example.com
Getting CA Private Key
```

如需更多信息，请参阅 第 2.5 节 “使用私有 CA 使用 OpenSSL 为 CSR 发布证书” 。

2. 显示有关自签名 CA 的基本信息：

```
$ openssl x509 -in <ca.crt> -text -noout
Certificate:
…
        X509v3 extensions:
            …
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
…
```

3. 验证私钥的一致性：

```
$ openssl pkey -check -in <ca.key>
Key is valid
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgcagSaTEBn74xZAwO
```

```
18wRpXoCVC9vcPki7WlT+gnmCI+hRANCAARb9NxIvkaVjFhOoZbGp/HtIQxbM78E
lwbDP0BI624xBJ8gK68ogSaq2x4SdezFdV1gNeKScDcU+Pj2pELldmdF
-----END PRIVATE KEY-----
```

其他资源

- **openssl (1)**, **ca (1)**, **genpkey (1)**, **x509 (1)**, 和 **req (1)** man page

2.3. 使用 OPENSSL 为 TLS 服务器证书创建私钥和 CSR

只有在来自证书颁发机构(CA)的有效 TLS 证书时才可以使用 TLS 加密通信频道。要获取证书，您必须首先为您的服务器创建私钥和证书签名请求(CSR)。

流程

1. 在服务器系统中生成私钥，例如：

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <server-private.key>
```

2. 可选：使用您选择的文本编辑器准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_server.cnf>
[server-cert]
keyUsage = critical, digitalSignature, keyEncipherment, keyAgreement
extendedKeyUsage = serverAuth
subjectAltName = @alt_name

[req]
distinguished_name = dn
prompt = no

[dn]
C = <US>
O = <Example Organization>
CN = <server.example.com>

[alt_name]
DNS.1 = <example.com>
DNS.2 = <server.example.com>
IP.1 = <192.168.0.1>
IP.2 = <::1>
IP.3 = <127.0.0.1>
```

extendedKeyUsage = serverAuth 选项限制证书的使用。

3. 使用之前创建的私钥创建 CSR：

```
$ openssl req -key <server-private.key> -config <example_server.cnf> -new -out <server-cert.csr>
```

如果省略 **-config** 选项，**req** 工具会提示您输入更多信息，例如：

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]: <US>
State or Province Name (full name) []: <Washington>
Locality Name (eg, city) [Default City]: <Seattle>
Organization Name (eg, company) [Default Company Ltd]: <Example Organization>
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: <server.example.com>
Email Address []: <server@example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 以签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。如需更多信息，请参阅 第 2.5 节 “使用私有 CA 使用 OpenSSL 为 CSR 发布证书”。

验证

1. 从 CA 获取请求的证书后，检查证书的人类可读部分是否与您的要求匹配，例如：

```
$ openssl x509 -text -noout -in <server-cert.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

其它资源

- **openssl (1)**, **x509 (1)**, **genpkey (1)**, **req (1)**, 和 **config (5)** man page

2.4. 使用 OPENSSL 为 TLS 客户端证书创建私钥和 CSR

只有在来自证书颁发机构(CA)的有效 TLS 证书时才可以使用 TLS 加密通信频道。要获取证书，您必须首先为您的客户端创建私钥和证书签名请求(CSR)。

流程

1. 在客户端系统中生成私钥，例如：

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <client-private.key>
```

2. 可选：使用您选择的文本编辑器准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_client.cnf>
[client-cert]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
subjectAltName = @alt_name

[req]
distinguished_name = dn
prompt = no

[dn]
CN = <client.example.com>

[clnt_alt_name]
email= <client@example.com>
```

extendedKeyUsage = clientAuth 选项限制证书的使用。

3. 使用之前创建的私钥创建 CSR：

```
$ openssl req -key <client-private.key> -config <example_client.cnf> -new -out <client-cert.csr>
```

如果省略 **-config** 选项，**req** 工具会提示您输入更多信息，例如：

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
…
Common Name (eg, your name or your server's hostname) []: <client.example.com>
Email Address []: <client@example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 以签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。如需更多信息，请参阅 第 2.5 节 “使用私有 CA 使用 OpenSSL 为 CSR 发布证书”。

验证

1. 检查证书人类可读的部分是否与您的要求匹配，例如：

```
$ openssl x509 -text -noout -in <client-cert.crt>
Certificate:
…
            X509v3 Extended Key Usage:
```

```
            TLS Web Client Authentication
            X509v3 Subject Alternative Name:
              email:client@example.com
…
```

其它资源

- **openssl (1)**, **x509 (1)**, **genpkey (1)**, **req (1)**, 和 **config (5)** man page

2.5. 使用私有 CA 使用 OPENSSL 为 CSR 发布证书

要让系统建立 TLS 加密通信频道，证书颁发机构(CA)必须为它们提供有效的证书。如果您有私有 CA，您可以通过从系统签署证书签名请求(CSR)来创建请求的证书。

前提条件

- 您已配置了私有 CA。如需更多信息，请参阅 第 2.2 节 “使用 OpenSSL 创建私有 CA”。
- 您有一个包含 CSR 的文件。您可以在 第 2.3 节 “使用 OpenSSL 为 TLS 服务器证书创建私钥和 CSR” 中找到创建 CSR 的示例。

流程

1. 可选：使用您选择的文本编辑器准备 OpenSSL 配置文件，以便为证书添加扩展，例如：

```
$ vim <openssl.cnf>
[server-cert]
extendedKeyUsage = serverAuth

[client-cert]
extendedKeyUsage = clientAuth
```

2. 使用 **x509** 工具来基于 CSR 创建证书，例如：

```
$ openssl x509 -req -in <server-cert.csr> -CA <ca.crt> -CAkey <ca.key> -CAcreateserial -
days 365 -extfile <openssl.cnf> -extensions <server-cert> -out <server-cert.crt>
Signature ok
subject=C = US, O = Example Organization, CN = server.example.com
Getting CA Private Key
```

要提高安全性，请在从 CSR 创建另一个证书前删除 serial-number 文件。这样，您可以确保序列号始终随机。如果您省略了 **CAserial** 选项用于指定自定义文件名，则 serial-number 文件名与证书的文件名相同，但其扩展名被替换为 **.srl** 扩展名（上例中的**server-cert.srl** ）。

其他资源

- **openssl (1)**, **ca (1)**, 和 **x509 (1)** man page

2.6. 使用 GNUTLS 创建私有 CA

当您的情况需要验证您的内部网络内实体时，私有证书颁发机构(CA)非常有用。例如，当基于您控制下的 CA 签名的证书或通过身份验证创建 VPN 网关时，或者您不想支付商业 CA 时，使用私有 CA。要在这样的用例中签署证书，私有 CA 使用自签名证书。

前提条件

- 您有 **root** 权限或使用 **sudo** 输入管理命令的权限。需要此类权限的命令标记为 **#**。
- 您已在系统上安装了 GnuTLS。如果没有，您可以使用这个命令：

```
$ yum install gnutls-utils
```

流程

1. 为您的 CA 生成私钥。例如，以下命令会创建一个 256 位 ECDSA (Elliptic Curve Digital Signature Algorithm)密钥：

```
$ certtool --generate-privkey --sec-param High --key-type=ecdsa --outfile <ca.key>
```

密钥生成过程的时间取决于主机的硬件和熵、所选算法以及密钥长度。

2. 为证书创建一个模板文件。

   a. 使用您选择的文本编辑器创建一个文件，例如：

```
$ vi <ca.cfg>
```

   b. 编辑文件以包含必要的认证详情：

```
organization = "Example Inc."
state = "Example"
country = EX
cn = "Example CA"
serial = 007
expiration_days = 365
ca
cert_signing_key
crl_signing_key
```

3. 使用在第 1 步中生成的私钥创建一个签名的证书：
   生成的 *<ca.crt>* 文件是一个自签名 CA 证书，可用于为其他证书签名一年。*<ca.crt>* 文件是公钥(证书)。加载的文件 *<ca.key>* 是私钥。您应该将此文件保存在安全的地方。

```
$ certtool --generate-self-signed --load-privkey <ca.key> --template <ca.cfg> --outfile
<ca.crt>
```

4. 对 CA 的私钥设置安全权限，例如：

```
# chown <root>:<root> <ca.key>
# chmod 600 <ca.key>
```

后续步骤

- 要将自签名 CA 证书用作客户端系统上的信任锚，请将 CA 证书复制到客户端，并以 **root** 用户身份将其添加到客户端的系统范围信任存储中：

```
# trust anchor <ca.crt>
```

如需更多信息，请参阅 第 3 章 *使用共享的系统证书*。

验证

1. 显示有关自签名 CA 的基本信息：

```
$ certtool --certificate-info --infile <ca.crt>
Certificate:
…
    X509v3 extensions:
        …
        X509v3 Basic Constraints: critical
            CA:TRUE
        X509v3 Key Usage: critical
            Certificate Sign, CRL Sign
```

2. 创建证书签名请求(CSR)，并使用您的 CA 为请求签名。CA 必须成功基于 CSR 创建证书，例如：

   a. 为您的 CA 生成一个私钥：

```
$ certtool --generate-privkey --outfile <example-server.key>
```

   b. 在您选择的文本编辑器中打开一个新配置文件，例如：

```
$ vi <example-server.cfg>
```

   c. 编辑文件以包含必要的认证详情：

```
signing_key
encryption_key
key_agreement

tls_www_server

country = "US"
organization = "Example Organization"
cn = "server.example.com"

dns_name = "example.com"
dns_name = "server.example.com"
ip_address = "192.168.0.1"
ip_address = "::1"
ip_address = "127.0.0.1"
```

   d. 使用之前创建的私钥生成一个请求：

```
$ certtool --generate-request --load-privkey <example-server.key> --template <example-server.cfg> --outfile <example-server.crq>
```

   e. 生成证书并使用 CA 的私钥对其签名：

```
$ certtool --generate-certificate --load-request <example-server.crq> --load-ca-certificate <ca.crt> --load-ca-privkey <ca.key> --outfile <example-server.crt>
```

其他资源

- **certtool(1)** 和 **trust(1)** 手册页

2.7. 使用 GNUTLS 为 TLS 服务器证书创建私钥和 CSR

要获取证书，您必须首先为您的服务器创建私钥和证书签名请求(CSR)。

流程

1. 在服务器系统中生成私钥，例如：

```
$ certtool --generate-privkey --sec-param High --outfile <example-server.key>
```

2. 可选：使用您选择的文本编辑器准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_server.cnf>
signing_key
encryption_key
key_agreement

tls_www_server

country = "US"
organization = "Example Organization"
cn = "server.example.com"

dns_name = "example.com"
dns_name = "server.example.com"
ip_address = "192.168.0.1"
ip_address = "::1"
ip_address = "127.0.0.1"
```

3. 使用之前创建的私钥创建 CSR：

```
$ certtool --generate-request --template <example-server.cfg> --load-privkey <example-server.key> --outfile <example-server.crq>
```

如果省略 **--template** 选项，**certool** 工具会提示您输入额外的信息，例如：

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Generating a PKCS #10 certificate request...
Country name (2 chars): <US>
State or province name: <Washington>
Locality name: <Seattle>
Organization name: <Example Organization>
Organizational unit name:
Common name: <server.example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 以签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。请参阅 第 2.9 节 “使用私有 CA，使用 GnuTLS 为 CSR 发布证书” 了解更多信息。

验证

1. 从 CA 获取请求的证书后，检查证书的人类可读部分是否与您的要求匹配，例如：

```
$ certtool --certificate-info --infile <example-server.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

其他资源

- **certtool(1)** 手册页

2.8. 使用 GNUTLS 为 TLS 客户端证书创建私钥和 CSR

要获取证书，您必须首先为您的客户端创建私钥和证书签名请求(CSR)。

流程

1. 在客户端系统中生成私钥，例如：

```
$ certtool --generate-privkey --sec-param High --outfile <example-client.key>
```

2. 可选：使用您选择的文本编辑器准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_client.cnf>
signing_key
encryption_key

tls_www_client
```

```
cn = "client.example.com"
email = "client@example.com"
```

3. 使用之前创建的私钥创建 CSR：

```
$ certtool --generate-request --template <example-client.cfg> --load-privkey <example-client.key> --outfile <example-client.crq>
```

如果省略 **--template** 选项 **certtool** 工具会提示您输入额外的信息，例如：

```
Generating a PKCS #10 certificate request...
Country name (2 chars): <US>
State or province name: <Washington>
Locality name: <Seattle>
Organization name: <Example Organization>
Organizational unit name:
Common name: <server.example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 以签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。请参阅 第 2.9 节 “使用私有 CA，使用 GnuTLS 为 CSR 发布证书” 了解更多信息。

验证

1. 检查证书人类可读的部分是否与您的要求匹配，例如：

```
$ certtool --certificate-info --infile <example-client.crt>
Certificate:
…
        X509v3 Extended Key Usage:
            TLS Web Client Authentication
        X509v3 Subject Alternative Name:
            email:client@example.com
…
```

其他资源

- **certtool(1)** 手册页

2.9. 使用私有 CA，使用 GNUTLS 为 CSR 发布证书

要让系统建立 TLS 加密通信频道，证书颁发机构(CA)必须为它们提供有效的证书。如果您有私有 CA，您可以通过从系统签署证书签名请求(CSR)来创建请求的证书。

前提条件

- 您已配置了私有 CA。请参阅 第 2.6 节 “使用 GnuTLS 创建私有 CA” 了解更多信息。
- 您有一个包含 CSR 的文件。您可以在 第 2.7 节 “使用 GnuTLS 为 TLS 服务器证书创建私钥和 CSR” 中找到创建 CSR 的示例。

流程

1. 可选：使用您选择的文本编辑器准备 GnuTLS 配置文件，来为证书添加扩展，例如：

```
$ vi <server-extensions.cfg>
honor_crq_extensions
ocsp_uri = "http://ocsp.example.com"
```

2. 使用 **certtool** 工具创建基于 CSR 的证书，例如：

```
$ certtool --generate-certificate --load-request <example-server.crq> --load-ca-privkey
<ca.key> --load-ca-certificate <ca.crt> --template <server-extensions.cfg> --outfile
<example-server.crt>
```

其他资源

- **certtool(1)** 手册页

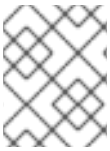
第 3 章 使用共享的系统证书

共享的系统证书存储使 NSS、GnuTLS、OpenSSL 和 Java 能够共享用于检索系统证书锚和块列表信息的默认源。默认情况下，信任存储包含 Mozilla CA 列表，包括正和负信任。系统允许更新核心 Mozilla CA 列表或选择其他证书列表。

3.1. 系统范围的信任存储

在 RHEL 中，整合的系统范围的信任存储位于 **/etc/pki/ca-trust/** 和 **/usr/share/pki/ca-trust-source/** 目录中。对 **/usr/share/pki/ca-trust-source/** 中信任设置的优先级的处理低于 **/etc/pki/ca-trust/** 中的设置。

证书文件根据它们所安装到的子目录处理。例如，信任定位符属于 **/usr/share/pki/ca-trust-source/anchors/** 或 **/etc/pki/ca-trust/source/anchors/** 目录。



注意

在分层加密系统中，信任锚是其他各方认为值得信任的权威实体。在 X.509 架构中，根证书是从中派生信任链的信任锚。要启用链验证，信任方必须首先能够访问信任锚。

其他资源

- **update-ca-trust(8)** 和 **trust(1)** 手册页

3.2. 添加新证书

要使用新的信任来源确认系统上的应用程序，请将相应的证书添加到系统范围的存储中，并使用 **update-ca-trust** 命令。

前提条件

- **ca-certificates** 软件包存在于系统中。

流程

1. 要在简单的 PEM 或 DER 文件格式中添加证书到系统中信任的 CA 列表中,请将证书文件复制到 **/usr/share/pki/ca-trust-source/anchors/** 或 **/etc/pki/ca-trust/source/anchors/** 目录中，例如：

```
# cp ~/certificate-trust-examples/Cert-trust-test-ca.pem /usr/share/pki/ca-trust-source/anchors/
```

2. 要更新系统范围的信任存储配置，请使用 **update-ca-trust** 命令：

```
# update-ca-trust
```

注意

虽然 Firefox 浏览器可以在不预先执行 **update-ca-trust** 的情况下使用一个添加的证书，但在每次 CA 更改后需要输入 **update-ca-trust** 命令。另请注意，浏览器（如 Firefox、Chromium 和 GNOME Web 缓存文件）,您可能需要清除浏览器的缓存或重新启动浏览器来加载当前的系统证书配置。

其他资源

- **update-ca-trust(8)** 和 **trust(1)** 手册页

3.3. 管理信任的系统证书

trust 命令提供了在共享的系统范围信任存储中管理证书的一种便捷方式。

- 要列出、提取、添加、删除或修改信任锚，请使用 **trust** 命令。要查看这个命令的内置帮助信息，请不要输入任何参数，或使用 **--help** 指令：

```
$ trust
usage: trust command <args>...

Common trust commands are:
  list            List trust or certificates
  extract         Extract certificates and trust
  extract-compat  Extract trust compatibility bundles
  anchor          Add, remove, change trust anchors
  dump            Dump trust objects in internal format

See 'trust <command> --help' for more information
```

- 要列出所有系统信任锚和证书，请使用 **trust list** 命令：

```
$ trust list
pkcs11:id=%d2%87%b4%e3%df%37%27%93%55%f6%56%ea%81%e5%36%cc%8c%1e%3
f%bd;type=cert
    type: certificate
    label: ACCVRAIZ1
    trust: anchor
    category: authority

pkcs11:id=%a6%b3%e1%2b%2b%49%b6%d7%73%a1%aa%94%f5%01%e7%73%65%4c%
ac%50;type=cert
    type: certificate
    label: ACEDICOM Root
    trust: anchor
    category: authority
...
```

- 要将信任锚存储在系统范围的信任存储中，请使用 **trust anchor** 子命令，并指定证书的路径。将 *<path.to/certificate.crt>* 替换为证书的路径及其文件名：

```
# trust anchor <path.to/certificate.crt>
```

- 要删除证书，请使用证书的路径或证书的 ID：

```
# trust anchor --remove <path.to/certificate.crt>
# trust anchor --remove "pkcs11:id=<%AA%BB%CC%DD%EE>;type=cert"
```

其他资源

- **trust** 命令的所有子命令都提供了详细的内置帮助，例如。

```
$ trust list --help
usage: trust list --filter=<what>

  --filter=<what>     filter of what to export
                      ca-anchors        certificate anchors
...
  --purpose=<usage>   limit to certificates usable for the purpose
                      server-auth       for authenticating servers
...
```

其他资源

- **update-ca-trust(8)** 和 **trust(1)** 手册页

第 4 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广，产生的安全性就越低。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。请确保以最严格的可用配置为目标，并且仅在出于兼容性原因需要时才放宽配置。

4.1. SSL 和 TLS 协议

安全套接字层(SSL)协议最初使由 Netscape 公司开发的，以提供一种在互联网上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之间，例如 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了。
SSL v3	不要使用。具有严重的安全漏洞。从 RHEL 8 开始从核心加密库中删除了。
TLS 1.0	不建议使用。已知的无法以保证互操作性方式缓解的问题，且不支持现代密码套件。在 RHEL 8 中，只在 **LEGACY** 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。在 RHEL 8 中，只在 **LEGACY** 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，所以可以更快。在所有系统范围的加密策略中也启用了 TLS 1.3。

其他资源

- IETF：传输层安全(TLS)协议版本 1.3。

4.2. 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大大简化了。**DEFAULT** 加密策略只允许 TLS 1.2 和 1.3。要允许您的系统使用早期版本的 TLS 来协商连接，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 **LEGACY** 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 实现尽可能使用安全算法，而不阻止来自或到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，在这些环境中，不支持安全算法或协议的旧客户端或服务器不应连接或不允许连接。

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** 。

警告

为 **LEGACY** 加密策略禁用的算法不符合红帽的 RHEL 8 安全愿景，其安全属性不可靠。考虑放弃使用这些算法，而不是重新启用它们。如果您确实决定重新启用它们（例如，为了与旧硬件的互操作性），请将它们视为不安全的，并应用额外的保护措施，例如将其网络交互隔离到单独的网络段。不要在公共网络中使用它们。

如果您决定不遵循 RHEL 系统范围的加密策略，或根据您的设置创建自定义的加密策略，请在自定义配置中对首选协议、密码套件和密钥长度使用以下建议：

4.2.1. 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，否则请允许您的系统使用至少 TLS 版本 1.2 来协商连接。

请注意，尽管 RHEL 8 支持 TLS 版本 1.3，但 RHEL 8 组件并不完全支持这个协议的所有功能。例如，Apache Web 服务器尚不完全支持可降低连接延迟的 0-RTT(Zero R Trip Time)功能。

4.2.2. 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。一直禁止 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，它们被有意地弱化了，因此很容易被破解。

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(完美)转发保密(PFS)的密码套件，这样可确保加密数据的机密性，以防服务器密钥被泄露。此规则排除了快速 RSA 密钥交换，但允许使用 ECDHE 和 DHE。在两者中，ECDHE 更快，因此是首选。

您还应该优先选择 AEAD 密码，如 AES-GCM，使用 CBC 模式密码，因为它们不容易受到 padding oracle 攻击的影响。此外，在很多情况下，在 CBC 模式下，AES-GCM 比 AES 快，特别是当硬件具有 AES 加密加速器时。

另请注意，在使用带有 ECDSA 证书的 ECDHE 密钥交换时，事务的速度甚至比纯 RSA 密钥交换要快。为了给旧客户端提供支持，您可以在服务器上安装两对证书和密钥：一对带有 ECDSA 密钥（用于新客户端），另一对带有 RSA 密钥（用于旧密钥）。

4.2.3. 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)** 手册页。

4.3. 在应用程序中强化 TLS 配置

在 RHEL 中，系统范围的加密策略 提供了一种方便的方法，来确保您的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来强化与 TLS 相关的配置，您可以使用本节中描述的加密配置选项，并以最少的需求量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来决定。

4.3.1. 将 Apache HTTP 服务器配置为使用 TLS

Apache HTTP 服务器 可以使用 **OpenSSL** 和 **NSS** 库来满足其 TLS 的需求。RHEL 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，该文件可用来修改 **Apache HTTP 服务器** 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** 文件中详细介绍。**/usr/share/httpd/manual/ssl/ssl_howto.html** 文件中描述了各种设置的示例。

修改 **/etc/httpd/conf.d/ssl.conf** 配置文件中的设置时，请确保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

使用这个指令来指定您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。

4.3.2. 将 Nginx HTTP 和代理服务器配置为使用 TLS

要在 **Nginx** 中启用 TLS 1.3 支持，请将 **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。

4.3.3. 将 Dovecot 邮件服务器配置为使用 TLS

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** 文件中找到一些基本配置指令的说明，该文件与 **Dovecot** 的标准安装一起安装。

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
: 使用这个指令指定您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
```

其他资源

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

第 5 章 使用 IPSEC 配置 VPN

在 RHEL 8 中，您可以使用 **IPsec** 协议配置虚拟专用网络(VPN)，Libreswan 应用程序支持该协议。

5.1. LIBRESWAN 作为 IPSEC VPN 的实现

在 RHEL 中，您可以使用 IPsec 协议配置虚拟专用网络(VPN)，Libreswan 应用程序支持该协议。Libreswan 是 Openswan 应用程序的延续，Openswan 文档中的许多示例可以通过 Libreswan 交换。

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。IPsec VPN 变体，它使用 Level 2 Tunneling Protocol(L2TP)，被称为 L2TP/IPsec VPN，它需要 **optional** 软件仓库提供的 **xl2tpd** 软件包。

libreswan 是一个开源用户空间 IKE 实现。IKE v1 和 v2 作为用户级别的守护进程实现。IKE 协议也加密。IPsec 协议由 Linux 内核实现，Libreswan 配置内核以添加和删除 VPN 隧道配置。

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

- 封装安全性 Payload(ESP)，其协议号为 50。
- 经过身份验证的标头(AH)，其协议号为 51。

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

IPsec 协议提供两种操作模式：

- 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 *手动键控*。您还可以使用 **ip xfrm** 命令来配置手动密钥，但为了安全起见，强烈建议您不要这样做。Libreswan 使用 Netlink 接口与 Linux 内核进行通信。内核执行数据包加密和解密。

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

重要

IKE/IPsec VPN（由 Libreswan 和 Linux 内核实现）是 RHEL 中推荐的唯一 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 RHEL 中，Libreswan 默认**遵循系统范围的加密策略**。这样可确保 Libreswan 将当前威胁模型包括（IKEv2）的安全设置用作默认协议。如需更多信息，请参阅 使用系统范围的加密策略。

Libreswan 没有使用术语"源（source）"和"目的地（destination）"或"服务器（server）"和"客户端（client）"，因为 IKE/IPsec 使用对等（peer to peer）协议。相反，它使用术语"左"和"右"来指端点（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"，对远程主机使用"右"。

leftid 和 **rightid** 选项充当身份验证过程中相应主机的标识。详情请查看 **ipsec.conf(5)** 手册页。

5.2. LIBRESWAN 中的身份验证方法

Libreswan 支持多种身份验证方法，每种方法适合不同的场景。

预共享密钥(PSK)

预共享密钥(PSK)是最简单的身份验证方法。出于安全考虑，请勿使用小于 64 个随机字符的 PSK。在 FIPS 模式中，PSK 必须符合最低强度要求，具体取决于所使用的完整性算法。您可以使用 **authby=secret** 连接来设置 PSK。

原始 RSA 密钥

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。每个主机都使用所有其他主机的公共 RSA 密钥手动配置，Libreswan 在每对主机之间建立 IPsec 隧道。对于大量主机，这个方法不能很好地扩展。

您可以使用 **ipsec newhostkey** 命令在主机上生成原始 RSA 密钥。您可以使用 **ipsec showhostkey** 命令列出生成的密钥。使用 CKA ID 密钥的连接配置需要 **leftrsasigkey=** 行。原始 RSA 密钥使用 **authby=rsasig** 连接选项。

X.509 证书

X.509 证书 通常用于大规模部署连接到通用 IPsec 网关的主机。中心 *证书颁发机构*(CA)为主机或用户签署 RSA 证书。此中心 CA 负责中继信任，包括单个主机或用户的撤销。

例如，您可以使用 **openssl** 命令和 NSS **certutil** 命令来生成 X.509 证书。因为 Libreswan 使用 **leftcert=** 配置选项中证书的昵称从 NSS 数据库读取用户证书，所以在创建证书时请提供昵称。

如果使用自定义 CA 证书，则必须将其导入到网络安全服务(NSS)数据库中。您可以使用 **ipsec import** 命令将 PKCS #12 格式的任何证书导入到 Libreswan NSS 数据库。

警告

Libreswan 需要互联网密钥交换(IKE)对等 ID 作为每个对等证书的主题替代名称(SAN)，如 RFC 4945 的 3.1 章节 所述。通过更改 **require-id-on-certificated=** 选项禁用此检查可能会导致系统容易受到中间人攻击。

使用 **authby=rsasig** 连接选项，根据使用带 SHA-1 和 SHA-2 的 RSA 的 X.509 证书进行身份验证。您可以进一步对它进行限制，对于 ECDSA 数字签名使用 SHA-2（将 **authby=** 设置为 **ecdsa**），以及基于 RSA Probabilistic Signature Scheme (RSASSA-PSS) 数据签名的验证使用 SHA-2（**authby=rsa-sha2**）。默认值为 **authby=rsasig,ecdsa**。

证书和 **authby=** 签名方法应匹配。这提高了互操作性，并在一个数字签名系统中保留身份验证。

NULL 身份验证

NULL 身份验证 用来在没有身份验证的情况下获得网状加密。它可防止被动攻击，但不能防止主动攻击。但是，因为 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规模的机会主义 IPsec。在此模型中，客户端对服务器进行身份验证，但服务器不对客户端进行身份验证。此模型类似于使用 TLS 的安全网站。使用 **authby=null** 进行 NULL 身份验证。

保护量子计算机

除了上述身份验证方法外，您还可以使用 *Post-quantum Pre-shared Key* (PPK)方法来防止量子计算机可能的攻击。单个客户端或客户端组可以通过指定与带外配置的预共享密钥对应的 PPK ID 来使用它们自己的 PPK。

使用带有预共享密钥的 IKEv1 防止量子攻击者。重新设计 IKEv2 不会原生提供这种保护。Libreswan 提供使用 *Post-quantum Pre-shared Key* (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**。如需要 PPK，请添加 **ppk=insist**。然后，可以给每个客户端一个带有 secret 值的 PPK ID，该 secret 值在带外进行通信（最好是量子安全的）。PPK 应该具有很强的随机性，而不是基于字典中的单词。PPK ID 和 PPK 数据保存在 **ipsec.secrets** 文件中，例如：

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

PPKS 选项指的是静态 PPK。这个实验性功能使用基于一次性平板的动态 PPK。在每个连接中，一次性平板的一个新部件用作 PPK。当使用时，文件中动态 PPK 的那部分被零覆盖，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)** 手册页。

警告

动态 PPK 的实现是作为不受支持的技术预览提供的。请谨慎使用。

5.3. 安装 LIBRESWAN

在通过 Libreswan IPsec/IKE 实现设置 VPN 之前，您必须安装相应的软件包，启动 **ipsec** 服务，并在防火墙中允许该服务。

前提条件

- **AppStream**存储库已启用。

流程

1. 安装 **libreswan** 软件包：

```
# yum install libreswan
```

2. 如果要重新安装 Libreswan，请删除其旧的数据库文件，并创建一个新的数据库：

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

3. 启动 **ipsec** 服务，并启用该服务，以便其在引导时自动启动：

```
# 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
```

5.4. 创建主机到主机的 VPN

您可以将 Libreswan 配置为使用原始 RSA 密钥身份验证的称为 *left* 和 *right* 的两个主机之间创建主机到主机的 IPsec VPN。

前提条件

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

流程

1. 在每台主机上生成原始 RSA 密钥对：

```
# ipsec newhostkey
```

2. 上一步返回生成的密钥的 **ckaid**。在 *左* 主机上使用 **ckaid** 和以下命令，例如：

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

上一命令的输出生成了配置所需的 **leftrsasigkey=** 行。在第二台主机（*右*）上执行相同的操作：

```
# 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. 加载连接：

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

6. 建立隧道：

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

7. 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
```

5.5. 配置站点到站点的 VPN

要创建站点到站点的 IPsec VPN，通过加入两个网络，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. 将带有主机到主机 VPN 配置的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
```

5.6. 配置远程访问 VPN

公路勇士是指拥有移动客户端和动态分配的 IP 地址的旅行用户。移动客户端使用 X.509 证书进行身份验证。

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 or deny client
    # pam-authorize=yes
    # load connection, do not initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

在移动客户端（即 road warrior 的设备）上，使用与之前配置稍有不同的配置：

```
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
```

5.7. 配置网格 VPN

网格 VPN 网络（也称为 *any-to-any* VPN ） 是一个所有节点都使用 IPsec 进行通信的网络。该配置可以对于无法使用 IPsec 的节点进行例外处理。可使用两种方式配置网格 VPN 网络：

- 需要 IPsec。
- 首选 IPsec，但允许回退到使用明文通信。

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

您对 *opportunistic IPsec* 使用任何常规的 IKEv2 验证方法，因为这些连接是常规的 Libreswan 配置，除了由 **right=%opportunisticgroup** 条目定义的 opportunistic IPsec 之外。常见的身份验证方法是主机使用常用共享认证机构(CA)根据 X.509 证书进行互相验证。作为标准流程的一部分，云部署通常为云中的每个节点发布证书。

重要

不要使用 PreSharedKey (PSK)身份验证，因为一个被泄露的主机也会导致组 PSK secret 被泄露。

您可以使用 NULL 身份验证在节点间部署加密，而无需认证，这只防止被动攻击者。

以下流程使用 X.509 证书。您可以使用任何类型的 CA 管理系统（如 Dogtag 证书系统）生成这些证书。dogtag 假设每个节点的证书以 PKCSautomationhub 格式(**.p12** 文件)提供，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"，为此，我们使用名称"节点"，这样引用友好名称的配置文件对所有节点都是相同的。

前提条件

- Libreswan 已安装，并在每个节点上启动了 **ipsec** 服务。
- 一个新的 NSS 数据库已初始化。

1. 如果您已经有一个旧的 NSS 数据库，请删除旧的数据库文件：

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

2. 您可以使用以下命令初始化新数据库：

```
# ipsec initnss
```

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 **IPsec 需要的**（专用）、**IPsec 可选的** (private-or-clear)和 **No IPsec** (clear)配置文件创建以下三个连接定义：

```
# cat /etc/ipsec.d/mesh.conf
conn clear
 auto=ondemand (1)
 type=passthrough
 authby=never
 left=%defaultroute
 right=%group

conn private
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=drop
 negotiationshunt=drop
 ikev2=insist
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert (2)
 rightid=%fromcert
 right=%opportunisticgroup

conn private-or-clear
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=passthrough
 negotiationshunt=passthrough
 # left
 left=%defaultroute
 leftcert=nodeXXXX (3)
 leftid=%fromcert
 leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup
```

(1) **auto** 变量有几个选项：

您可以使用带有 opportunistic IPsec 的 **ondemand** 连接选项来启动 IPsec 连接，或者用于显式配置不需要一直激活的连接。这个选项在内核中建立一个陷阱 XFRM 策略，使 IPsec 连接在收到与该策略匹配的第一个数据包时开始。

您可以使用以下选项有效地配置和管理 IPsec 连接，无论是使用 Opportunistic IPsec 还是明确配置的连接：

add 选项
: 加载连接配置，并为响应远程启动做好准备。但是，连接不会自动从本地端启动。您可以使用 **ipsec auto --up** 命令手动启动 IPsec 连接。

start 选项

加载连接配置，并为响应远程启动做好准备。此外，它会立即启动到远程对等点的连接。您可以将这个选项用于永久的和一直活跃的连接。

2 **leftid** 和 **rightid** 变量标识 IPsec 隧道连接的右侧和左侧频道。如果您配置了证书，您可以使用这些变量来获取本地 IP 地址或本地证书的主题 DN 的值。

3 **leftcert** 变量定义您要使用的 NSS 数据库的 nickname。

3. 将网络的 IP 地址添加到对应的类中。例如，如果所有节点都位于 **10.15.0.0/16** 网络中，则所有节点都必须使用 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. 要将一个不支持 IPsec 的主机（如 **10.15.1.2** ）定义到 clear 组，请使用：

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

您可以从每个新节点的模板中创建 **/etc/ipsec.d/policies** 目录中创建文件，也可以使用 Puppet 或 Ansible 来置备它们。

请注意，每个节点都有相同的异常列表或不同的流量预期。因此，两个节点可能无法通信，因为一个节点需要 IPsec，而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
```

验证

1. 使用 **ping** 命令打开 IPsec 隧道：

```
# ping <nodeYYY>
```

2. 显示带有导入认证的 NSS 数据库：

```
# certutil -L -d sql:/etc/ipsec.d

Certificate Nickname    Trust Attributes
                        SSL,S/MIME,JAR/XPI

west                     u,u,u
ca                      CT,,
```

3. 查看节点上打开了哪个隧道：

```
# ipsec trafficstatus
006 #2: "private#10.15.0.0/16"[1] ...<nodeYYY>, type=ESP, add_time=1691399301,
inBytes=512, outBytes=512, maxBytes=2^63B, id='C=US, ST=NC, O=Example
```

```
Organization, CN=east'
```

其他资源

- **IPsec.conf (5)** 手册页。
- 有关 **authby** 变量的更多信息，请参阅 6.2.Libreswan 中的身份验证方法。

5.8. 部署 FIPS 兼容 IPSEC VPN

您可以使用 Libreswan 部署 FIPS 兼容 IPsec VPN 解决方案。要做到这一点，您可以识别哪些加密算法可用，且 Libreswan 在 FIPS 模式中禁用。

前提条件

- **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，并启用该服务，以便其在引导时自动启动：

```
# 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. 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

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
```

其他资源

- 使用系统范围的加密策略。

5.9. 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。要提高安全性，您可以添加密码保护。

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 认证为该标准级别 1，并且该状态不需要对数据库进行密码保护。

前提条件

- **/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 **/etc/ipsec.d/nsspassword** 文件，其中包含您在上一步中设置的密码，例如：

```
# cat /etc/ipsec.d/nsspassword
NSS Certificate DB:_<password>_
```

nsspassword 文件使用以下语法：

```
<token_1>:<password1>
<token_2>:<password2>
```

默认的 NSS 软件令牌是 **NSS 证书 数据库**。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，在完成了 **nsspassword** 文件后，启动或重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

验证

1. 在其 NSS 数据库中添加非空密码后，检查 **ipsec** 服务是否运行：

```
# 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[6214]: Initializing NSS using read-write database "sql:/etc/ipsec.d"
pluto[6214]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate
DB" with length 20 passed to NSS
pluto[6214]: NSS crypto library initialized
...
```

其他资源

- **certutil(1)** 手册页。
- 合规性活动和 政府标准 知识库文章 FIPS 140-2 和 FIPS 140-3。

5.10. 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。有了这个功能，您可以在网络上建立 IPsec VPN，以防止通过 UDP 和封装安全负载(ESP)传输的流量。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需要永久阻止 UDP 时，才使用 TCP 作为主 VPN 协议。

前提条件

- 已配置了 远程访问 VPN。

流程

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

```
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 会被永久阻止，请在客户端的连接配置中使用以下选项：

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

其他资源

- IETF RFC 8229：IKE 和 IPsec 数据包的 TCP 封装 。

5.11. 配置自动检测和使用 ESP 硬件卸载来加速 IPSEC 连接

卸载硬件的封装安全负载(ESP)来加速以太网上的 IPsec 连接。默认情况下，Libreswan 会检测硬件是否支持这个功能，并因此启用 ESP 硬件卸载。如果这个功能被禁用或明确启用，您可以切回到自动检测。

先决条件

- 网卡支持 ESP 硬件卸载。
- 网络驱动程序支持 ESP 硬件卸载。
- IPsec 连接已配置且可以正常工作。

步骤

1. 编辑应使用 ESP 硬件卸载支持的自动检测连接的 **/etc/ipsec.d/** 目录中的 Libreswan 配置文件。
2. 确保 **nic-offload** 参数没有在连接的设置中设置。
3. 如果您删除了 **nic-offload**，请重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

验证

1. 显示 IPsec 连接使用的以太网设备的 **tx_ipsec** 和 **rx_ipsec** 计数器：

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

2. 通过 IPsec 隧道发送流量。例如，ping 远程 IP 地址：

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

3. 再次显示以太网设备的 **tx_ipsec** 和 **rx_ipsec** 计数器：

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

如果计数器值增加了，ESP 硬件卸载正常工作。

其他资源

- 使用 IPsec 配置 VPN

5.12. 在绑定中配置 ESP 硬件卸载以加快 IPSEC 连接

将封装安全负载(ESP)卸载到硬件可加速 IPsec 连接。如果出于故障转移原因而使用网络绑定，配置 ESP 硬件卸载的要求和流程与使用常规以太网设备的要求和流程不同。例如，在这种情况下，您可以对绑定启用卸载支持，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。
- 网络驱动程序支持对绑定设备的 ESP 硬件卸载。在 RHEL 中，只有 **ixgbe** 驱动程序支持此功能。
- 绑定已配置且可以正常工作。
- 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。

步骤

1. 对网络绑定启用 ESP 硬件卸载支持：

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

这个命令在对 **bond0** 连接启用 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. 显示活动端口的 **tx_ipsec** 和 **rx_ipsec** 计数器：

```
# ethtool -S 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. 再次显示活动端口的 **tx_ipsec** 和 **rx_ipsec** 计数器：

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

如果计数器值增加了，ESP 硬件卸载正常工作。

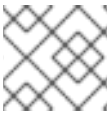
其他资源

- 配置网络绑定
- 使用 IPsec 配置 VPN

5.13. 使用 RHEL 系统角色配置带有 IPSEC 的 VPN 连接

使用 **vpn** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 系统上配置 VPN 连接。您可以使用它来设置主机到主机、网络到网络、VPN 远程访问服务器和网格配置。

对于主机到主机连接，角色使用默认参数在 **vpn_connections** 列表中的每一对主机之间设置 VPN 通道，包括根据需要生成密钥。另外，您还可以将其配置为在列出的所有主机之间创建 机会主义网格配置。该角色假定 **hosts** 下的主机名称与 Ansible 清单中使用的主机的名称相同，并且您可以使用这些名称来配置通道。



注意

vpn RHEL 系统角色目前仅支持 Libreswan （即 IPsec 实现），作为 VPN 供应商。

5.13.1. 使用 vpn RHEL 系统角色使用 IPsec 创建主机到主机的 VPN

您可以通过在控制节点上运行 Ansible playbook 来使用 **vpn** 系统角色配置主机到主机的连接，这将配置清单文件中列出的所有受管节点。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。

- 用于连接到受管节点的帐户具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
- name: Host to host VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed-node-01.example.com:
          managed-node-02.example.com:
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

此 playbook 通过使用系统角色自动生成的密钥，配置 **managed-node-01.example.com-to-managed-node-02.example.com**。因为 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设为 **true**，因此 **vpn** 角色使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

要配置从受管主机到清单文件中未列出的外部主机的连接，请将以下部分添加到主机的 **vpn_connections** 列表中：

```
    vpn_connections:
      - hosts:
          managed-node-01.example.com:
          <external_node>:
            hostname: <IP_address_or_hostname>
```

这将配置一个额外的连接： **managed-node-01.example.com-to-*<external_node>***

注意

连接仅在受管节点上配置，而不在外部节点上配置。

2. 可选：您可以使用 **vpn_connections** 中的其它部分为受管节点指定多个 VPN 连接，如 control plane 和 data plane ：

```
- name: Multiple VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 192.0.2.0 # IP for the control plane
          managed-node-02.example.com:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
```

```
        managed-node-01.example.com:
          hostname: 10.0.0.1 # IP for the data plane
        managed-node-02.example.com:
          hostname: 10.0.0.2
```

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 在受管节点上，确认连接已成功载入：

```
# ipsec status | grep <connection_name>
```

将 ***<connection_name>*** > 替换为来自此节点的连接的名称，如 **managed_node1-to-managed_node2**。

注意

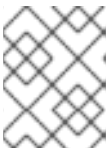
默认情况下，从每个系统的角度来看，角色为其创建的每个连接生成一个描述性名称。例如，当在 **managed_node1** 和 **managed_node2** 之间创建连接时，此连接在 **managed_node1** 上的描述性名称为 **managed_node1-to-managed_node2**，但在 **managed_node2** 上，连接的描述性名称为 **managed_node2-to-managed_node1**。

2. 在受管节点上，确认连接是否成功启动：

```
# ipsec trafficstatus | grep <connection_name>
```

3. 可选：如果连接没有成功加载，请输入以下命令来手动添加连接。这提供了更具体的信息，说明连接未能建立的原因：

```
# ipsec auto --add <connection_name>
```



注意

加载和启动连接过程中可能会出现的任何错误都会在 **/var/log/pluto.log** 文件中报告。由于这些日志很难解析，因此请手动添加连接以从标准输出中获取日志消息。

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.vpn/README.md** file
- **/usr/share/doc/rhel-system-roles/vpn/** directory

5.13.2. 使用 vpn RHEL 系统角色创建带有 IPsec 的 opportunistic mesh VPN 连接

您可以使用 **vpn** 系统角色来配置机会主义网格 VPN 连接，该连接通过在控制节点上运行 Ansible playbook 来使用证书进行身份验证，这将配置清单文件中列出的所有受管节点。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- **/etc/ipsec.d/** 目录中的 IPsec 网络安全服务(NSS)加密库包含必要的证书。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
- name: Mesh VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com, managed-node-03.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
            cidr: 192.0.2.7/32
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

通过在 playbook 中定义 **auth_method: cert** 参数来配置用证书进行身份验证。默认情况下，节点名称用作证书的昵称。在本例中，这是 **managed-node-01.example.com**。您可以使用清单中的 **cert_name** 属性来定义不同的证书名称。

在本例中，控制节点是您运行 Ansible playbook 的系统，与两个受管节点(192.0.2.0/24)共享相同的无类别域间路由(CIDR)数，并且 IP 地址 192.0.2.7。因此，控制节点属于为 CIDR 192.0.2.0/24 自动创建的私有策略。

为防止在操作期间出现 SSH 连接丢失，控制节点的清晰策略包含在策略列表中。请注意，在策略列表中还有一个项 CIDR 等于 default。这是因为此 playbook 覆盖了默认策略中的规则，以使其为私有，而非私有或清晰。

因为 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设为 **true**，因此 **vpn** 角色使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.vpn/README.md** file
- **/usr/share/doc/rhel-system-roles/vpn/** directory

5.14. 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，以下配置允许使用带有 AES 和 SHA-1 或 SHA-2的 IKEv1 连接，以及带有 AES-GCM 或 AES-CBC 的 IPsec(ESP) 连接：

```
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，请在 **/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。

其他资源

- 使用系统范围的加密策略。

5.15. 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符合一下任何一种情况，并应用相应的解决方案。

基本连接故障排除

VPN 连接的大多数问题都发生在新部署中，管理员使用不匹配的配置选项配置了端点。此外，正常工作的配置可能会突然停止工作，通常是由于新引入的不兼容的值。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，或者使用某些选项的不同默认值（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 端点或端点之间路由器上的防火墙将所有互联网密钥交换(IKE)数据包丢弃。

- 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，因此您只能使用 **tcpdump** 工具排除一小部分问题。如果防火墙丢弃了 IKE 或 IPsec 数据包，您可以尝试使用 **tcpdump** 工具来查找原因。但是，**tcpdump** 无法诊断 IPsec VPN 连接的其他问题。

- 捕获**eth0** 接口上的 VPN 协商以及所有加密数据：

```
# 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 中使用预共享密钥(PSK)时，如果双方没有放入相同的 PSK，则整个 IKE 信息将无法读取：

```
# 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

除防火墙阻止 IKE 或 IPsec 数据包外，网络问题的最常见原因与加密数据包的数据包大小增加有关。网络硬件对于大于最大传输单元(MTU)的数据包进行分片处理，例如 1500 字节。通常，片会丢失，数据包无法重新组装。当使用小数据包的 ping 测试可以正常工作，但其他流量失败时，这会导致间歇性故障。在这种情况下，您可以建立一个 SSH 会话，但是一使用它，终端就会冻结，例如，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中来减小 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场景中的问题，请在 **set-mss** 参数中直接指定较小的数值：

```
# 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，可能会意外地重新映射数据包。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 的系统有一个 NAT 规则：

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

如果地址为 10.0.2.33 的系统将数据包发送到 192.168.0.1，那么路由器会在应用 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， IPsec 不会加密此数据包。

要解决这个问题，请在路由器上插入目标 IPsec 子网范围不包含 NAT 的规则，例如：

```
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 bug 导致 IKE 用户空间和 IPsec 内核不同步时，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
...
```

上一命令输出中的任何非零值都表示有问题。如果您遇到这个问题，请开一个新的 支持问题单，并附上上一命令的输出与对应的 IKE 日志。

libreswan 日志

默认情况下，Libreswan 使用 **syslog** 协议的日志。您可以使用 **journalctl** 命令来查找与 IPsec 有关的日志条目。因为日志中相应的条目是由 **pluto** IKE 守护进程发送的，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，请将 **plutodebug=all** 选项添加到 **/etc/ipsec.conf** 文件的 **config setup** 部分来启用调试日志。

请注意，调试日志记录会生成大量的条目，**journald** 或 **syslogd** 服务的速率可能会抑制 **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**。

其他资源

- 使用日志文件 对问题进行故障排除。
- **tcpdump(8)** 和 **ipsec.conf(5)** 手册页。
- 使用和配置 firewalld

5.16. 其他资源

- **ipsec(8)**、**ipsec.conf(5)**、**ipsec.secrets(5)**、**ipsec_auto(8)** 和 **ipsec_rsasigkey(8)** 手册页。
- **/usr/share/doc/libreswan-*版本*/** 目录。
- Libreswan 项目 Wiki 。

- 所有 Libreswan 手册页 。
- NIST 特殊发布 800-77：IPsec VPN 指南 。

第 6 章 使用 MACSEC 加密同一物理网络中的第 2 层流量

您可以使用 MACsec 来保护两个设备（点到点）之间的通信。例如，您的分支办公室通过城际以太网与中心办公室连接，您可以在连接办公室的两个主机上配置 MACsec，以提高安全性。

介质访问控制安全(MACsec)是一种第 2 层的协议，用于保护以太网链路上的不同的流量类型，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**)以及
- 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的主机之间建立连接。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。您可以只在 MACsec 设备上设置 IP 配置，以便只使用加密连接与其他主机进行通信，也可以在父设备上设置 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，除非您只想在主机和交换机之间加密流量。在这种情况下，交换机还必须支持 MACsec。

换句话说，有 2 种常用的方法来配置 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。

6.1. 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。例如，您可以在通过以太网连接的两个主机之间创建 MACsec 连接。

流程

1. 在配置 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. 创建一个 16 字节的十六进制 CAK：

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

     b. 创建一个 32 字节的十六进制 CKN：

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

2. 在您要通过 MACsec 连接连接的两个主机上：

3. 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
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

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

4. 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'
```

5. 激活连接：

```
# nmcli connection up macsec0
```

验证

1. 验证流量是否加密：

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

2. 可选：显示未加密的流量：

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

3. 显示 MACsec 统计信息：

```
# ip macsec show
```

4. 显示每种保护类型的单独的计数器：仅完整性（关闭加密）和加密（打开加密）

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

6.2. 其他资源

- MACsec：加密网络流量的另一种解决方案 博客。

第 7 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，其提供一个带有 D-Bus 接口的、动态可定制的、基于主机的防火墙。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开放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** 选项将自定义的 **nftables** 规则传递到 **firewalld**。

7.1. 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** ：对简单的防火墙用例使用 **firewalld** 工具。此工具易于使用，并涵盖了这些场景的典型用例。
- **nftables** ：使用 **nftables** 工具来设置复杂和性能关键的防火墙，如用于整个网络。
- **iptables** ：Red Hat Enterprise Linux 上的 **iptables** 工具使用 **nf_tables** 内核 API 而不是 *传统的*后端。**nf_tables** API 提供了向后兼容性，以便使用 **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

要防止不同的与防火墙相关的服务(**firewalld**、**nftables** 或 **iptables**)相互影响，请在 RHEL 主机上仅运行其中一个服务，并禁用其他服务。

7.2. 防火墙区

您可以使用 **firewalld** 实用程序根据您在该网络中的接口和流量具有的信任级别将网络划分为不同的区。连接只能是一个区的一部分，但您可以对许多网络连接使用这个区域。

firewalld 遵循与区相关的严格的原则：

1. 流量入口只有一个区。
2. 流量只出站一个区。
3. 区域定义了信任级别。
4. 默认情况下，允许区域流量（在同一区中使用）。
5. 默认情况下，Interzone 流量（从区域到区）被拒绝。

原则 4 和 5 是原则 3 的后果。

原则 4 可以通过区选项 **--remove-forward** 进行配置。原则 5 可以通过添加新策略来进行配置。

NetworkManager 通知接口区的 **firewalld**。您可以使用以下工具为接口分配区：

- **NetworkManager**
- **firewall-config** 工具
- **firewall-cmd** 工具
- RHEL web 控制台

RHEL web 控制台、**firewall-config** 和 **firewall-cmd** 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** 更改接口区，则请求将转发到 **NetworkManager**，且不会由**firewalld** 处理。

/usr/lib/firewalld/zones/ 目录存储预定义的区域，您可以立即将它们应用到任何可用的网络接口。只有在修改后，这些文件才会被拷贝到 **/etc/firewalld/zones/** 目录中。预定义区的默认设置如下：

block

- 适合：任何传入的网络连接都会被拒绝，并带有 **IPv4** 的 icmp-host-prohibited 消息，对于 **IPv6** 的 icmp6-adm-prohibited 消息。
- 接受：只有从系统中启动的网络连接。

dmz

- 适用于：您的 DMZ 中的计算机可通过有限访问您的内部网络。
- accept: 只有所选传入的连接。

drop

适合：任何传入的网络数据包都会丢失，没有任何通知。

- 接受：只有传出的网络连接。

external

- 适用于：启用伪装的外部网络，特别是路由器。不信任网络中的其他计算机的情况。
- accept: 只有所选传入的连接。

home

- 适用于：您主要信任网络中其他计算机的主页环境。
- accept: 只有所选传入的连接。

internal

- 适用于：您主要信任网络中其他计算机的内部网络。
- accept: 只有所选传入的连接。

public

- 适用于：您不信任网络上其他计算机的公共区域。
- accept: 只有所选传入的连接。

trusted

- 接受：所有网络连接。

work

适用于：您主要信任网络上其他计算机的运行环境。

- accept: 只有所选传入的连接。

这些区中的一个被设置为 *default* 区。当接口连接被添加到 **NetworkManager** 中时，它们会被分配到默认区。安装时，**firewalld** 中的默认区是 **public** 区。您可以更改默认区。

注意

使网络区域名称自我解释，以帮助用户快速了解它们。

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。

其他资源

- **firewalld.zone(5)** 手册页

7.3. 防火墙策略

防火墙策略指定网络所需的安全状态。它们概述了对不同类型的流量采取的规则和操作。通常，策略包含以下类型流量的规则：

- 传入流量
- 传出流量
- 转发流量
- 特定服务和应用程序
- 网络地址转换(NAT)

防火墙策略使用防火墙区的概念。每个区都与一组特定的防火墙规则关联，决定允许的流量。策略以有状态、单向的方式应用防火墙规则。这意味着您只考虑流量的一个方向。由于 **firewalld** 的有状态过滤，流量返回路径被隐式允许。

策略与入口区和出口区关联。ingress 区域是流量源自的位置（接收）。出口区是流量离开的位置(sent)。

策略中定义的防火墙规则可以引用防火墙区，以便在多个网络接口之间应用一致的配置。

7.4. 防火墙规则

您可以使用防火墙规则来实现特定的配置，以允许或阻止网络流量。因此，您可以控制网络流量的流，以防止系统免受安全威胁。

防火墙规则通常根据各种属性定义某些条件。属性可以如下：

- 源 IP 地址
- 目标 IP 地址
- 传输协议(TCP、UDP、...)
- 端口
- 网络接口

firewalld 实用程序将防火墙规则组织到区域(**如公共、内部** 等)和策略。每个区域都有自己的一组规则，决定与特定区关联的网络接口的流量自由度级别。

7.5. 区配置文件

firewalld 区配置文件包含区的信息。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。

以下示例展示了允许 **TCP** 和 **UDP** 协议的一个服务(**SSH**)和一个端口范围的配置：

```
<?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>
```

其他资源

- **firewalld.zone** 手册页

7.6. 预定义的 FIREWALLD 服务

firewalld 服务是一组预定义的防火墙规则，用于定义对特定应用程序或网络服务的访问。每个服务都代表了以下元素的组合：

- 本地端口
- 网络协议
- 关联的防火墙规则
- 源端口和目的地
- 启用服务时自动载入的防火墙帮助程序模块

服务可以简化数据包过滤并节省时间，因为它会一次性实现多个任务。例如，**firewalld** 可以一次执行以下任务：

- 打开端口
- 定义网络协议
- 启用数据包转发

服务配置选项和通用文件信息在 **firewalld.service (5)** 手册页中进行了描述。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。

您可以使用以下方法配置 **firewalld** ：

- 使用工具：
  - **firewall-config** - 图形化工具
  - **firewall-cmd** - 命令行工具
  - **firewall-offline-cmd** - 命令行工具
- 编辑 **/etc/firewalld/services/** 目录中的 XML 文件。
  如果您没有添加或更改服务，**/etc/firewalld/services/** 中没有对应的 XML 文件。您可以使用 **/usr/lib/firewalld/services/** 中的文件作为模板。

其他资源

- **firewalld.service (5)** 手册页

7.7. 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。

7.7.1. 自定义特定区的防火墙设置以增强安全性

您可以通过修改防火墙设置并将特定网络接口或与特定防火墙区的连接来增强网络安全性。通过为区定义粒度规则和限制，您可以根据预期的安全级别控制入站和出站流量。

例如，您可以实现以下优点：

- 保护敏感数据
- 防止未授权访问
- 缓解潜在的网络威胁

前提条件

- **firewalld** 服务正在运行。

流程

1. 列出可用的防火墙区：

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

firewall-cmd --get-zones 命令显示系统上所有可用的区，但不显示特定区的详情。要查看所有区的详情，请使用 **firewall-cmd --list-all-zones** 命令。

2. 选择您要用于此配置的区域。

3. 修改所选区域的防火墙设置。例如，允许 **SSH** 服务并删除 **ftp** 服务：

```
# firewall-cmd --add-service=ssh --zone=<your_chosen_zone>
# firewall-cmd --remove-service=ftp --zone=<same_chosen_zone>
```

4. 为防火墙区分配网络接口：

   a. 列出可用的网络接口：

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

   区域的活动是通过存在与其配置匹配的网络接口或源地址范围来确定的。默认区域对于未分类的流量活跃，但如果没有流量匹配其规则，则始终处于活动状态。

   b. 为所选区分配一个网络接口：

   ```
   # firewall-cmd --zone=<your_chosen_zone> --change-interface=<interface_name> --permanent
   ```

   为区分配网络接口更适合将一致的防火墙设置应用到特定接口（物理或虚拟）上的所有流量。

   firewall-cmd 命令与 **--permanent** 选项一起使用时，通常涉及更新 NetworkManager 连接配置集以永久更改防火墙配置。**firewalld** 和 NetworkManager 之间的这种集成可确保一致的网络和防火墙设置。

验证

1. 显示您选择的区的更新设置：

```
# firewall-cmd --zone=<your_chosen_zone> --list-all
```

命令输出显示所有区设置，包括分配的服务、网络接口和网络连接（源）。

7.7.2. 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** 会加载默认区的设置，并使其处于活动状态。请注意，所有其他区域的设置都会被保留并可使用。

通常，区会根据 NetworkManager 连接配置文件中的 **connection.zone** 设置分配给接口。另外，重启 NetworkManager 后，管理这些区域的分配。

前提条件

- **firewalld** 服务正在运行。

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

按照此流程，设置是一个永久设置，即使没有 **--permanent** 选项。

7.7.3. 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
```

7.7.4. 使用 nmcli 为连接分配区域

您可以使用 **nmcli** 实用程序在 **NetworkManager** 连接中添加 **firewalld** 区域。

流程

1. 将区分配给 **NetworkManager** 连接配置文件：

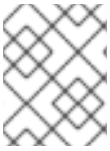
```
# nmcli connection modify profile connection.zone zone_name
```

2. 激活连接：

```
# nmcli connection up profile
```

7.7.5. 在连接配置文件文件中手动将一个区域分配给网络连接

如果您无法使用 **nmcli** 工具修改连接配置文件，您可以手动编辑配置文件的相应文件来分配一个 **firewalld** 区域。



注意

使用 **nmcli** 工具修改连接配置文件来分配 **firewalld** 区域效率更高。详情请参阅 将网络接口分配给区域。

流程

1. 确定连接配置文件的路径及其格式：

```
# nmcli -f NAME,FILENAME connection
NAME    FILENAME
enp1s0  /etc/NetworkManager/system-connections/enp1s0.nmconnection
enp7s0  /etc/sysconfig/network-scripts/ifcfg-enp7s0
```

NetworkManager 为不同的连接配置文件格式使用单独的目录和文件名称：

- **/etc/NetworkManager/system-connections/*<connection_name>*.nmconnection** 文件中的配置文件使用 keyfile 格式。
- **/etc/sysconfig/network-scripts/ifcfg-*<interface_name>*** 文件中的配置文件使用 ifcfg 格式。

2. 根据格式，更新相应的文件：

- 如果文件使用 keyfile 格式，请将 **zone=*<name>*** 附加到 **/etc/NetworkManager/system-connections/*<connection_name>*.nmconnection** 文件的 **[connection]** 部分：

```
[connection]
...
zone=internal
```

- 如果文件使用 ifcfg 格式，请将 **ZONE=*<name>*** 附加到 **/etc/sysconfig/network-scripts/ifcfg-*<interface_name>*** 文件中：

```
ZONE=internal
```

3. 重新加载连接配置文件：

```
# nmcli connection reload
```

4. 重新激活连接配置文件

```
# nmcli connection up <profile_name>
```

验证

- 显示接口的区域，例如：

```
# firewall-cmd --get-zone-of-interface enp1s0
internal
```

7.7.6. 在 ifcfg 文件中手动将区分配给网络连接

当连接由 NetworkManager 管理时，必须了解它使用的区。对于每个网络连接配置文件，可以指定区，根据计算机使用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

- 要为连接设置一个区，请编辑 **/etc/sysconfig/network-scripts/ifcfg-*connection_name*** 文件，并添加将区分配给这个连接的行：

```
ZONE=zone_name
```

7.7.7. 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。新区需要 **--permanent** 选项，否则命令无法工作。

前提条件

- **firewalld** 服务正在运行。

流程

1. 创建一个新区：

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

2. 使新区可用：

```
# firewall-cmd --reload
```

该命令会在不中断已在运行的网络服务的情况下对防火墙配置应用最新的更改。

验证

- 检查是否在您的永久设置中添加了新的区：

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

7.7.8. 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。此行为是通过设置区的目标来定义的。有四个选项：

- **ACCEPT** ：接受所有传入的数据包，除了特定规则禁止的。
- **REJECT** ：拒绝所有传入的数据包，除了特定规则允许的。当 **firewalld** 拒绝数据包时，会告知源机器有关拒绝的信息。
- **DROP** ：丢弃所有传入的数据包，除了特定规则允许的。当 **firewalld** 丢弃数据包时，不会告知源机器有关丢弃数据包的信息。
- **default** ：与 **REJECT** 的行为类似，但在某些情况下具有特殊含义。

前提条件

前提条件

- **firewalld** 服务正在运行。

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>
```

其他资源

- **firewall-cmd(1)** 手册页

7.8. 使用 FIREWALLD 控制网络流量

firewalld 软件包安装大量预定义的服务文件，您可以添加更多或自定义它们。然后，您可以使用这些服务定义为服务打开或关闭端口，而无需了解这些服务使用的协议和端口号。

7.8.1. 使用 CLI 控制预定义服务的流量

控制流量的最简单的方法是在 **firewalld** 中添加预定义的服务。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。

前提条件

- **firewalld** 服务正在运行。

流程

1. 检查 **firewalld** 中的服务是否没有被允许：

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

命令列出默认区域中启用的服务。

2. 列出 **firewalld** 中的所有预定义服务：

```
# 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 ...
```

命令显示默认区域的可用服务列表。

3. 将服务添加到 **firewalld** 允许的服务列表中：

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

命令将指定的服务添加到默认区域中。

4. 使新设置具有持久性：

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

命令将这些运行时更改应用到防火墙的永久配置。默认情况下，它会将这些更改应用到默认区的配置。

验证

1. 列出所有永久防火墙规则：

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

使用默认防火墙区域 的永久防火墙规则(**公共**)显示完整的配置。

2. 检查 **firewalld** 服务的永久配置的有效性。

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

如果永久配置无效，命令会返回带有更多详情的错误：

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

您还可以手动检查永久配置文件以验证设置。主配置文件为 **/etc/firewalld/firewalld.conf**。特定于区的配置文件位于 **/etc/firewalld/zones/** 目录中，策略位于 **/etc/firewalld/policies/** 目录中。

7.8.2. 使用 GUI 使用预定义服务控制流量

您可以使用图形用户界面控制预定义服务的网络流量。防火墙配置应用程序提供对命令行工具的可访问和用户友好的替代选择。

前提条件

- **firewall-config** 软件包已安装 。

- **firewalld** 服务正在运行。

流程

1. 启用或禁用预定义或自定义服务：
   a. 启动 **firewall-config** 工具，并选择要配置其服务的网络区。
   b. 选择 **Zones** 选项卡，然后选择下面的 **Services** 选项卡。
   c. 选中您要信任的每种服务类型的复选框，或者清除复选框以阻止所选区域中的服务。
2. 编辑服务：
   a. 启动 **firewall-config** 工具。
   b. 从标有 **Configuration** 的菜单中选择 **Permanent**。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。

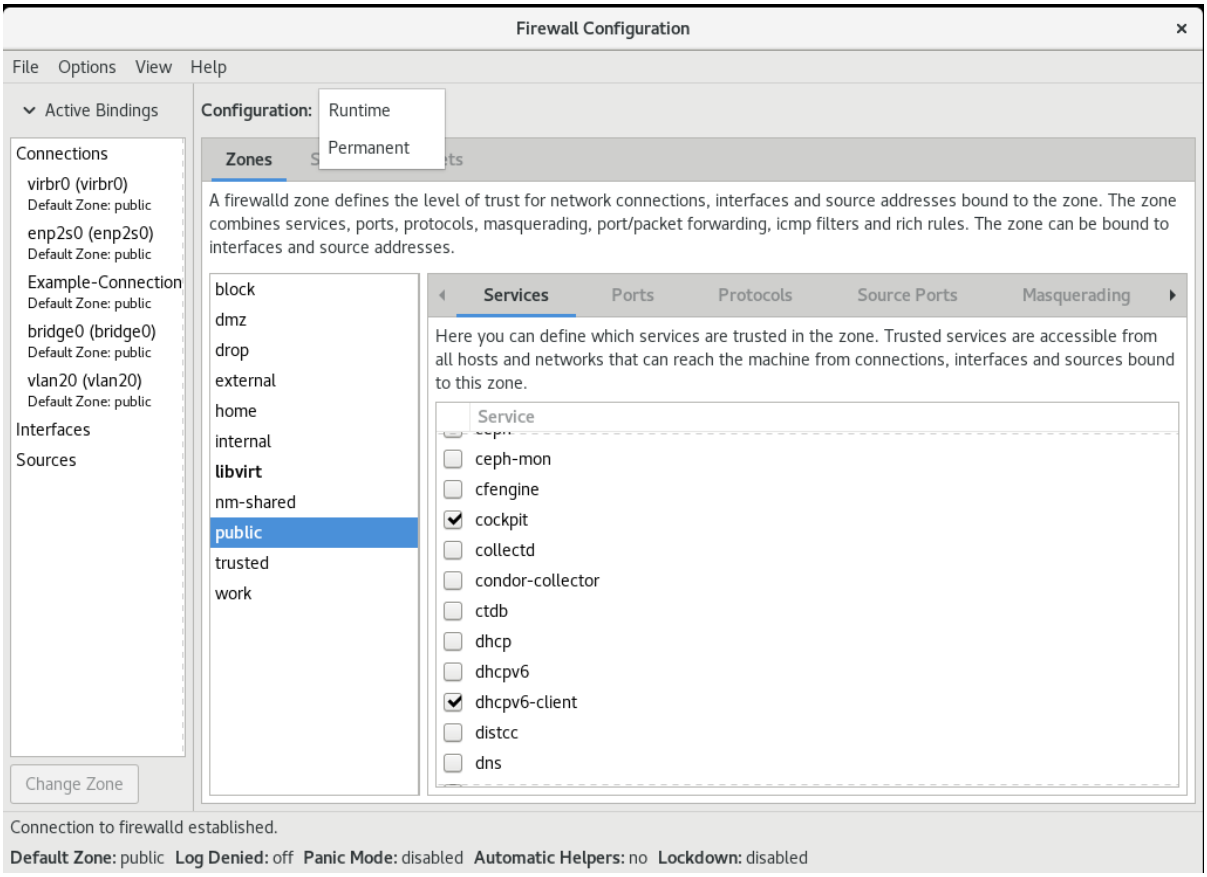
Ports、**Protocols**和 **Source Port** 选项卡支持添加、更改和删除所选服务的端口、协议和源端口。模块选项卡是用于配置 **Netfilter** 助手模块的。**Destination** 选项卡允许将流量限制到特定的目标地址和互联网协议(**IPv4** 或 **IPv6**)。

注意

在 **Runtime** 模式下无法更改服务设置。

验证

- 按 **Super** 键进入活动概览。
- 选择 Firewall Configuration 工具。
  - 您还可以通过输入 **firewall-config** 命令，使用命令行启动图形防火墙配置实用程序。
- 查看防火墙配置列表：



此时会打开 **Firewall Configuration** 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。

7.8.3. 配置 firewalld 以允许托管安全 Web 服务器

端口是逻辑服务，使操作系统能够接收和区分网络流量并将其转发到系统服务。系统服务由侦听端口的守护进程表示，并等待任何进入此端口的流量。

通常，系统服务侦听为它们保留的标准端口。例如，**httpd** 守护进程监听 80 端口。但是，系统管理员可以直接指定端口号，而不是服务名称。

您可以使用 **firewalld** 服务配置对托管数据的安全 Web 服务器的访问。

前提条件

- **firewalld** 服务正在运行。

流程

1. 检查当前活跃的防火墙区：

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

2. 在适当的区中添加 HTTPS 服务：

```
# firewall-cmd --zone=<zone_name> --add-service=https --permanent
```

3. 重新载入防火墙配置：

```
# firewall-cmd --reload
```

验证

1. 检查 **firewalld** 中是否打开端口：
   - 如果您通过指定端口号来打开端口，请输入：

     ```
     # firewall-cmd --zone=<zone_name> --list-all
     ```

   - 如果您通过指定服务定义来打开端口，请输入：

     ```
     # firewall-cmd --zone=<zone_name> --list-services
     ```

7.8.4. 关闭未使用的或不必要的端口以增强网络安全性

当不再需要开放端口时，您可以使用 **firewalld** 实用程序关闭它。

重要

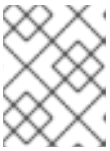
关闭所有不必要的端口，以减少潜在的攻击面，并最大程度降低未授权访问或利用漏洞的风险。

流程

1. 列出所有允许的端口：

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

   默认情况下，此命令列出了默认区域中启用的端口。

   注意

   此命令只为您提供作为端口打开的端口列表。您将无法看到作为服务打开的任何开放端口。在这种情况下，请考虑使用 **--list-all** 选项而不是 **--list-ports**。

2. 从允许的端口列表中删除端口，为传入的流量关闭它：

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

   这个命令从区中删除端口。如果没有指定区域，它将从 default 区域中删除端口。

3. 使新设置具有持久性：

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

   如果没有指定区域，这个命令会对默认区域的永久配置应用运行时更改。

验证

1. 列出活跃区并选择要检查的区：

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

2. 列出所选区中当前打开的端口，以检查未使用的或不必要的端口是否已关闭：

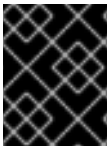
```
# firewall-cmd --zone=<zone_to_inspect> --list-ports
```

7.8.5. 通过 CLI 控制流量

您可以使用 **firewall-cmd** 命令：

- 禁用网络流量
- 启用网络流量

例如，您可以增强系统防御，确保数据隐私或优化网络资源。



重要

启用 panic 模式可停止所有网络流量。因此，只有当您具有对机器的物理访问权限或使用串行控制台登录时，才应使用它。

流程

1. 要立即禁用网络流量，请切换 panic 模式：

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

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
```

7.8.6. 使用 GUI 控制协议的流量

如果想使用某种协议允许流量通过防火墙，您可以使用 GUI。

前提条件

- **firewall-config** 软件包已安装

流程

1. 启动 **firewall-config** 工具，并选择要更改其设置的网络区。
2. 选择 **Protocols** 选项卡，然后点击右侧的 **Add** 按钮。此时会打开 协议 窗口。
3. 从列表中选择协议，或者选择 **Other Protocol** 复选框，并在字段中输入协议。

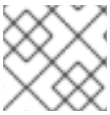
7.9. 根据源使用区管理传入流量

您可以使用区管理传入的流量，根据其源管理传入的流量。此上下文中的传入流量是适用于您的系统的任何数据，或者传递运行 **firewalld** 的主机。源通常指的是流量源自的 IP 地址或网络范围。因此，您可以对传入的流量进行排序，并将其分配给不同的区，以允许或禁止该流量可访问的服务。

通过源地址匹配优先于接口名称的匹配。当您向区添加源时，防火墙会根据基于接口的规则为传入的流量优先选择基于源的规则。这意味着，如果传入的流量与特定区指定的源地址匹配，与该源地址关联的区域将决定如何处理流量，而不考虑它到达的接口。另一方面，基于接口的规则通常是对与特定基于源的规则不匹配的流量的回退。这些规则应用到流量，其中源没有明确地与区关联。这可让您为没有特定源定义区的流量定义默认行为。

7.9.1. 添加源

要将传入的流量路由到特定区，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 **受信任** 区中 *192.168.2.15* 的所有传入的流量：

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
```

7.9.2. 删除源

当您从区中删除源时，源自源的流量不再被定向到该源指定的规则。相反，流量会返回与它源自的接口关联的区域的规则和设置，或转至默认区域。

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
```

7.9.3. 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>
```

7.9.4. 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。以下流程只允许来自 **192.0.2.0/24** 网络的 HTTP 流量，而阻止其他任何流量。

警告

当您配置此场景时，请使用具有 **default** 目标的区。使用目标设为 **ACCEPT** 的区存在安全风险，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 **internal** 区，来将来自源的流量通过区：

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

3. 在 **internal** 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 **internal** 区是否活跃，以及该区中服务是否被允许：

```
# 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
  ...
```

其他资源

- **firewalld.zones(5)** 手册页

7.10. 在区域间过滤转发的流量

firewalld 可让您控制不同 **firewalld** 区域之间的网络数据流。通过定义规则和策略，您可以管理在这些区域之间移动流量时如何允许或拒绝流量。

策略对象功能在 **firewalld** 中提供转发和输出过滤。您可以使用 **firewalld** 过滤不同区域之间的流量，以允许访问本地托管的虚拟机来连接主机。

7.10.1. 策略对象和区域之间的关系

策略对象允许用户将 firewalld 的原语（如服务、端口和丰富的规则）附加到策略。您可以将策略对象应用到以有状态和单向的方式在区域间传输的流量上。

```
# firewall-cmd --permanent --new-policy myOutputPolicy

# firewall-cmd --permanent --policy myOutputPolicy --add-ingress-zone HOST

# firewall-cmd --permanent --policy myOutputPolicy --add-egress-zone ANY
```

HOST 和 **ANY** 是 ingress 和 egress 区域列表中使用的符号区域。

- **HOST** 符号区域对于来自运行 firewalld 的主机的流量，或具有到运行 firewalld 的主机的流量允许策略。
- **ANY** 符号区对所有当前和将来的区域应用策略。**ANY** 符号区域充当所有区域的通配符。

7.10.2. 使用优先级对策略进行排序

多个策略可以应用到同一组流量，因此应使用优先级为可能应用的策略创建优先级顺序。

要设置优先级来对策略进行排序：

```
# firewall-cmd --permanent --policy mypolicy --set-priority -500
```

在上例中，*-500* 是较低的优先级值，但具有较高的优先级。因此，-500 将在 -100 之前执行。

较低数字优先级值具有更高的优先级，首先应用。

7.10.3. 使用策略对象过滤本地托管的容器和物理连接到主机的网络之间的流量

策略对象功能允许用户过滤 Podman 和 firewalld 区域之间的流量。

注意

红帽建议默认阻止所有流量，并打开 Podman 工具所需的可选择的服务。

流程

1. 创建一个新的防火墙策略：

```
# firewall-cmd --permanent --new-policy podmanToAny
```

2. 阻止从 Podman 到其它区域的所有流量，并只允许 Podman 上必要的服务：

```
# firewall-cmd --permanent --policy podmanToAny --set-target REJECT
# firewall-cmd --permanent --policy podmanToAny --add-service dhcp
# firewall-cmd --permanent --policy podmanToAny --add-service dns
# firewall-cmd --permanent --policy podmanToAny --add-service https
```

3. 创建一个新的 Podman 区域：

```
# firewall-cmd --permanent --new-zone=podman
```

4. 为策略定义 ingress 区域：

```
# firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

5. 为所有其他区域定义 egress 区域：

```
# firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

将 egress 区域设置为 ANY 意味着您可以从 Podman 过滤到其他区。如果要过滤到主机，则将 egress 区域设置为 HOST。

6. 重启 firewalld 服务：

```
# systemctl restart firewalld
```

验证

- 验证到其他区域的 Podman 防火墙策略：

```
# firewall-cmd --info-policy podmanToAny
podmanToAny (active)
```

```
...
target: REJECT
ingress-zones: podman
egress-zones: ANY
services: dhcp dns https
...
```

7.10.4. 设置策略对象的默认目标

您可以为策略指定 --set-target 选项。可用的目标如下：

- **ACCEPT** - 接受数据包
- **DROP** - 丢弃不需要的数据包
- **REJECT** - 拒绝不需要的数据包，并带有 ICMP 回复
- **CONTINUE**（默认）- 数据包将遵循以下策略和区域中的规则。

```
# firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

验证

- 验证有关策略的信息

```
# firewall-cmd --info-policy mypolicy
```

7.10.5. 使用 DNAT 将 HTTPS 流量转发到不同主机

如果您的 web 服务器使用私有 IP 地址在 DMZ 中运行，您可以配置目标网络地址转换 (DNAT) 以使互联网上的客户端能够连接到此 web 服务器。在本例中，Web 服务器的主机名解析为路由器的公共 IP 地址。当客户端建立到路由器上定义的端口的连接时，路由器会将数据包转发到内部 Web 服务器。

前提条件

- DNS 服务器将 Web 服务器的主机名解析为路由器的 IP 地址。
- 您知道以下设置：
  - 您要转发的专用 IP 地址和端口号
  - 要使用的 IP 协议
  - 要重定向数据包的 web 服务器的目标 IP 地址和端口

流程

1. 创建防火墙策略：

```
# firewall-cmd --permanent --new-policy <example_policy>
```

与区域不同，策略也允许数据包过滤输入、输出和转发流量。这很重要，因为将流量转发到本地的 web 服务器、容器或虚拟机上的端点需要此类功能。

2. 为入站和出站流量配置符号链接区域，以便路由器本身连接到其本地 IP 地址并转发此流量：

```
# firewall-cmd --permanent --policy=<example_policy> --add-ingress-zone=HOST
# firewall-cmd --permanent --policy=<example_policy> --add-egress-zone=ANY
```

--add-ingress-zone=HOST 选项是指本地生成的数据包，并从本地主机中传输。**--add-egress-zone=ANY** 选项引用移动到任何区域的流量。

3. 添加将流量转发到 Web 服务器的富规则：

```
# firewall-cmd --permanent --policy=<example_policy> --add-rich-rule='rule
family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp"
to-port="443" to-addr="192.51.100.20"'
```

富规则将路由器 IP 地址(192.0.2.1)上端口 443 的 TCP 流量转发到 Web 服务器 IP 地址(192.51.100.20)的端口 443。

4. 重新载入防火墙配置文件：

```
# firewall-cmd --reload
success
```

5. 在内核中激活 127.0.0.0/8 的路由：

- 对于持久性更改，请运行：

```
# echo "net.ipv4.conf.all.route_localnet=1" > /etc/sysctl.d/90-enable-route-
localnet.conf
```

命令永久配置 **route_localnet** 内核参数，并确保设置在系统重启后保留。

- 要在不重启系统的情况下立即应用设置，请运行：

```
# sysctl -p /etc/sysctl.d/90-enable-route-localnet.conf
```

sysctl 命令可用于应用实时更改，但配置在系统重启后不会保留。

验证

1. 连接到路由器的 IP 地址和您转发到 web 服务器的端口：

```
# curl https://192.0.2.1:443
```

2. 可选：验证 **net.ipv4.conf.all.route_localnet** 内核参数是否活跃：

```
# sysctl net.ipv4.conf.all.route_localnet
net.ipv4.conf.all.route_localnet = 1
```

3. 验证 **<example_policy>** 是否活跃，并包含您需要的设置，特别是源 IP 地址和端口，要使用的协议以及目标 IP 地址和端口：

```
# firewall-cmd --info-policy=<example_policy>
example_policy (active)
  priority: -1
```

```
  target: CONTINUE
  ingress-zones: HOST
  egress-zones: ANY
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
 rule family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp" to-
port="443" to-addr="192.51.100.20"
```

其他资源

- **firewall-cmd (1)**, **firewalld.policies (5)**, **firewalld.richlanguage (5)**, **sysctl (8)**, 和 **sysctl.conf (5)** man page
- 使用 /etc/sysctl.d/ 中的配置文件调整内核参数

7.11. 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
- 目标 NAT（DNAT）
- 重定向

7.11.1. 网络地址转换类型

这些是不同的网络地址转换（NAT）类型：

伪装

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(ISP)不会路由私有 IP 范围，如 **10.0.0.0/8**。如果您在网络中使用私有 IP 范围，并且用户可以访问互联网上的服务器，请将这些范围内的数据包的源 IP 地址映射到公共 IP 地址。
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。

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，那么您可以在路由器上设置一个 DNAT 规则，来将传入的流量重定向到此服务器。

重定向

这个类型是 DNAT 的一个特殊情况，用于将数据包重定向到本地计算机上的不同端口。例如，如果服务运行在与其标准端口不同的端口上，您可以将传入的流量从标准端口重定向到此特定端口。

7.11.2. 配置 IP 地址伪装

您可以在系统中启用 IP 伪装。在访问互联网时，IP 伪装会将单个机器隐藏在网关后面。

流程

1. 要检查是否启用了 IP 伪装（例如，对于 **external** 区），以 **root** 用户身份输入以下命令：

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

如果已启用，命令将会打印 **yes**，且退出状态为 **0**。否则，将打印 **no**，退出状态为 **1**。如果省略了 **zone**，则将使用默认区。

2. 要启用 IP 伪装，请以 **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，请将 **--permanent** 选项传给命令。

4. 要禁用 IP 伪装，请以 **root** 身份输入以下命令：

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

要使此设置永久生效，请将 **--permanent** 选项传给命令。

7.11.3. 使用 DNAT 转发传入的 HTTP 流量

您可以使用目标网络地址转换(DNAT)将传入的流量从一个目标地址和端口定向到另一个目标地址。通常，这对于将来自外部网络接口的请求重定向到特定的内部服务器或服务非常有用。

前提条件

- **firewalld** 服务正在运行。

流程

1. 使用以下内容创建 **/etc/sysctl.d/90-enable-IP-forwarding.conf** 文件：

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

此设置在内核中启用 IP 转发。它使内部 RHEL 服务器充当路由器，并将数据包从网络转发到网络。

2. 从 **/etc/sysctl.d/90-enable-IP-forwarding.conf** 文件中载入设置：

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. 转发传入的 HTTP 流量：

```
# firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toaddr=198.51.100.10:toport=8080 --permanent
```

以上命令使用以下设置定义 DNAT 规则：

- **--zone=public** - 配置 DNAT 规则的防火墙区。您可以将它调整为您需要的任何区域。
- **--add-forward-port** - 指示您要添加端口转发规则的选项。

- **port=80** - 外部目的地端口。
- **proto=tcp** - 指示您转发 TCP 流量的协议。
- **toaddr=198.51.100.10** - 目标 IP 地址。
- **ToPort=8080** - 内部服务器的目的地端口。
- **--permanent** - 使 DNAT 规则在重启后保持不变的选项。

4. 重新载入防火墙配置以应用更改：

```
# firewall-cmd --reload
```

验证

- 验证您使用的防火墙区的 DNAT 规则：

```
# firewall-cmd --list-forward-ports --zone=public
port=80:proto=tcp:toport=8080:toaddr=198.51.100.10
```

或者，查看对应的 XML 配置文件：

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to
not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="80" protocol="tcp" to-port="8080" to-addr="198.51.100.10"/>
  <forward/>
</zone>
```

其他资源

- 在运行时配置内核参数
- **firewall-cmd (1)** 手册页

7.11.4. 重定向来自非标准端口的流量，以便在标准端口上访问 Web 服务

您可以使用重定向机制使内部在非标准端口上运行的 Web 服务可以访问，而无需用户在 URL 中指定端口。因此，URL 更为简单，提供更好的浏览体验，而非标准端口仍然在内部或满足特定要求的情况下使用。

前提条件

- **firewalld** 服务正在运行。

流程

1. 使用以下内容创建 **/etc/sysctl.d/90-enable-IP-forwarding.conf** 文件：

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

此设置在内核中启用 IP 转发。

2. 从 **/etc/sysctl.d/90-enable-IP-forwarding.conf** 文件中载入设置：

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. 创建 NAT 重定向规则：

```
# firewall-cmd --zone=public --add-forward-
port=port=<standard_port>:proto=tcp:toport=<non_standard_port> --permanent
```

上一命令使用以下设置定义 NAT 重定向规则：

- **--zone=public** - 用于配置规则的防火墙区。您可以将它调整为您需要的任何区域。
- **--add-forward-port=port= <non_standard_** port> - 指示您使用最初接收传入流量的源端口添加端口转发（重定向）规则。
- **proto=tcp** - 指示您重定向 TCP 流量的协议。
- **ToPort=<*standard_port*>** - 目的地端口，在源端口上收到传入流量后，应重定向到该传入流量。
- **--permanent** - 使规则在重启后保持不变的选项。

4. 重新载入防火墙配置以应用更改：

```
# firewall-cmd --reload
```

验证

- 验证您使用的防火墙区的重定向规则：

```
# firewall-cmd --list-forward-ports
port=8080:proto=tcp:toport=80:toaddr=
```

或者，查看对应的 XML 配置文件：

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to
not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="8080" protocol="tcp" to-port="80"/>
  <forward/>
</zone>
```

其他资源

- 在运行时配置内核参数
- **firewall-cmd (1)** 手册页

7.12. 管理 ICMP 请求

Internet 控制消息协议 (ICMP)是一种支持协议，供各种网络设备用来测试、故障排除和诊断。**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。

您可以使用 **ICMP** 消息（特别是 **echo-request** 和 **echo-reply** ）来显示有关网络的信息，并为各种欺诈活动使用此类信息。因此，**firewalld** 允许控制 **ICMP** 请求来保护您的网络信息。

7.12.1. 配置 ICMP 过滤

您可以使用 ICMP 过滤来定义您希望防火墙允许或拒绝到达系统的 ICMP 类型和代码。ICMP 类型和代码是 ICMP 消息的特定类别和子类。

例如，在以下区域中的 ICMP 过滤可帮助：

- 安全增强 - 阻止潜在的 ICMP 类型和代码，以减少您的攻击面。
- 网络性能 - 仅计算必要的 ICMP 类型以优化网络性能，并防止受过度 ICMP 流量导致的潜在网络拥塞。
- 故障排除控制 - 维护基本的 ICMP 功能，用于表示潜在的安全风险的网络故障排除和阻止 ICMP 类型。

前提条件

- **firewalld** 服务正在运行。

流程

1. 列出可用的 ICMP 类型和代码：

```
# firewall-cmd --get-icmptypes
address-unreachable bad-header beyond-scope communication-prohibited destination-unreachable echo-reply echo-request failed-policy fragmentation-needed host-precedence-violation host-prohibited host-redirect host-unknown host-unreachable
...
```

   从该预定义列表中选择允许或阻止的 ICMP 类型和代码。

2. 通过以下方法过滤特定的 ICMP 类型：

   - 允许 ICMP 类型：

```
# firewall-cmd --zone=<target-zone> --remove-icmp-block=echo-request --permanent
```

     该命令删除对 echo requests ICMP 类型的任何现有的阻塞规则。

   - 阻塞 ICMP 类型：

```
# firewall-cmd --zone=<target-zone> --add-icmp-block=redirect --permanent
```

该命令确保防火墙阻止重定向消息 ICMP 类型。

3. 重新载入防火墙配置以应用更改：

```
# firewall-cmd --reload
```

验证

- 验证您的过滤规则是否生效：

```
# firewall-cmd --list-icmp-blocks
redirect
```

命令输出显示您允许或拒绝的 ICMP 类型和代码。

其他资源

- **firewall-cmd (1)** 手册页

7.13. 使用 FIREWALLD 设置和控制 IP 集

IP 集是一种 RHEL 功能，可将 IP 地址和网络分组到集合中，以实现更灵活、有效的防火墙规则管理。

例如，当您需要时 IP 集非常有价值：

- 处理大量 IP 地址列表
- 对大量 IP 地址列表实施动态更新
- 创建基于 IP 的自定义策略，以增强网络安全性和控制

警告

红帽建议使用 **firewall-cmd** 命令来创建和管理 IP 集。

7.13.1. 使用 IP 集为允许列表配置动态更新

您可以进行接近实时更新，来灵活地允许 IP 集中的特定 IP 地址或范围，即使在无法预计的情况下也是如此。这些更新可由各种事件触发，如检测安全威胁或更改网络行为。通常，此类解决方案利用自动化来减少手动工作，并通过快速响应情况来提高安全性。

前提条件

- **firewalld** 服务正在运行。

流程

1. 创建一个具有有意义的名称的 IP 集：

```
# firewall-cmd --permanent --new-ipset=allowlist --type=hash:ip
```

名为 **allowlist** 的新 IP 集包含您希望防火墙允许的 IP 地址。

2. 为 IP 集添加动态更新：

```
# firewall-cmd --permanent --ipset=allowlist --add-entry=198.51.100.10
```

此配置使用新添加的 IP 地址更新 **allowlist** IP 集，允许防火墙传递网络流量。

3. 创建引用之前创建的 IP 集的防火墙规则：

```
# firewall-cmd --permanent --zone=public --add-source=ipset:allowlist
```

如果没有此规则，IP 集不会影响网络流量。默认防火墙策略会预先显示。

4. 重新载入防火墙配置以应用更改：

```
# firewall-cmd --reload
```

验证

1. 列出所有 IP 集：

```
# firewall-cmd --get-ipsets
allowlist
```

2. 列出活跃的规则：

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s1
  sources: ipset:allowlist
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  ...
```

命令行参数的 **sources** 部分可让您了解流量来源（主机名、接口、IP 集、子网等）被允许或拒绝访问特定防火墙区。在这种情况下，**允许允许允许允许列表** IP 集中的 IP 地址通过 **公共区** 的防火墙传递流量。

3. 探索 IP 集的内容：

```
# cat /etc/firewalld/ipsets/allowlist.xml
<?xml version="1.0" encoding="utf-8"?>
<ipset type="hash:ip">
  <entry>198.51.100.10</entry>
</ipset>
```

后续步骤

- 使用脚本或安全实用程序来获取您的威胁智能源，并以自动化方式更新 **允许列表**。

其他资源

- **firewall-cmd (1)** 手册页

7.14. 丰富规则的优先级

默认情况下，富规则是根据其规则操作进行组织的。例如，**deny** 规则优先于 **allow** 规则。富规则中的 **priority** 参数可让管理员对富规则及其执行顺序进行精细的控制。在使用 **priority** 参数时，规则首先按优先级值排序。当更多规则 **具有相同的优先级** 时，其顺序由规则操作决定，如果操作也相同，则顺序可能未定义。

7.14.1. priority 参数如何将规则组织为不同的链

您可以将富规则中的 **priority** 参数设置为 **-32768** 和 **32767** 之间的任意数字，数字值具有更高的优先级。

firewalld 服务根据优先级值将规则组织到不同的链中：

- 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：根据操作，规则会被重定向到带有 **_log**、**_deny** 或 **_allow** 操作的链中。

在这些子链中，**firewalld** 根据其优先级值对规则进行排序。

7.14.2. 设置丰富的规则的优先级

以下是如何创建富规则的示例，该规则使用 **priority** 参数来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

- 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"'
```

这个命令还将日志条目数量限制为每分钟 **5** 条。

验证

- 显示命令在上一步中创建的 **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
  }
}
```

7.15. 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。

7.15.1. 使用 CLI 配置锁定

您可以使用命令行启用或禁用锁定功能。

流程

1. 查询是否启用了锁定：

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

2. 通过以下方法管理锁定配置：

   - 启用锁定：

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

   - 禁用锁定：

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

7.15.2. 锁定允许列表配置文件概述

默认允许列表配置文件包含 **NetworkManager** 上下文和 **libvirt** 的默认上下文。用户 ID 0 也位于列表中。

allowlist 配置文件存储在 **/etc/firewalld/** 目录中。

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
  <command name="/usr/bin/python3 -s /usr/bin/firewall-config"/>
   <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>
```

以下是一个允许列表配置文件的示例，它为名为 *user* ID 为 **815** 的用户允许 **firewall-cmd** 工具的所有命令：

```
<?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>
```

此示例展示了 **user id** 和 **user name**，但只需要其中一个选项。Python 是程序解释器，它位于命令行的前面。

在 Red Hat Enterprise Linux 中，所有工具都放在 **/usr/bin/** 目录中，**/bin/** 目录被符号链接到 **/usr/bin/** 目录。换句话说，虽然以 **root** 身份输入时 **firewall-cmd** 的路径可能解析为 **/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，那么除了必须添加到非 **root** 用户传统使用的 **/usr/bin/firewall-cmd** 外，该命令还必须添加到允许列表中。

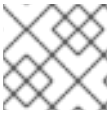
命令的 name 属性末尾的 * 表示所有以这个字符串开头的命令都匹配。如果没有 *，则包括参数的绝对命令必须匹配。

7.16. 启用 FIREWALLD 区域中不同接口或源之间的流量转发

区内转发是 **firewalld** 的一种功能，它允许 **firewalld** 区域内接口或源之间的流量转发。

7.16.1. 区内转发与默认目标设置为 ACCEPT 的区域之间的区别

启用区内转发后，单个 **firewalld** 区中的流量可以从一个接口或源流到另一个接口或源。区指定接口和源的信任级别。如果信任级别相同，流量会保持在同一区内。



注意

在 **firewalld** 的默认区域中启用区内转发，仅适用于添加到当前默认区的接口和源。

firewalld 使用不同的区来管理传入流量和传出流量。每个区域都有自己的一组规则和行为。例如，**trusted** 区域默认允许所有转发的流量。

其他区域可以有不同的默认行为。在标准区域中，当区的目标设置为默认时，转发的流量通常 **默认** 被丢弃。

要控制如何在区域内不同接口或源之间转发流量，请确保相应地理解并配置区的目标。

7.16.2. 使用区内转发来在以太网和 Wi-Fi 网络间转发流量

您可以使用区内转发来在同一 **firewalld** 区内的接口和源之间转发流量。此功能具有以下优点：

- 有线设备和无线设备间的无缝连接（您可以在连接到 **enp1s0** 的以太网网络和连接到 **wlp0s20**的 Wi-Fi 网络之间转发流量）
- 支持灵活的工作环境
- 可以被网络中的多个设备或用户访问和使用的共享资源（如打印机、数据库、网络连接存储等）
- 高效的内部网络（如平稳通信、降低延迟、资源可访问性等）

您可以为单独的 **firewalld** 区启用此功能。

流程

1. 在内核中启用数据包转发：

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2. 确保要启用区内转发的接口只分配给 **internal** 区：

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

3. 如果接口当前被分配给了不是 **internal** 的区，请重新分配它：

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

4. 将 **enp1s0** 和 **wlp0s20** 接口添加到 **internal** 区：

```
# firewall-cmd --zone=internal --add-interface=enp1s0 --add-interface=wlp0s20
```

5. 启用区域内部转发：

```
# firewall-cmd --zone=internal --add-forward
```

验证

以下验证步骤要求 **nmap-ncat** 软件包在两个主机上都已安装。

1. 登录到与您启用了区转发的主机的 **enp1s0** 接口位于同一网络的主机。
2. 使用 **ncat** 启动 echo 服务来测试连接：

```
# ncat -e /usr/bin/cat -l 12345
```

3. 登录到与 **wlp0s20** 接口在同一网络的主机。
4. 连接到在与 **enp1s0** 在同一网络的主机上运行的 echo 服务器：

```
# ncat <other_host> 12345
```

5. 输入内容并按 **Enter**。验证文本是否已发送回来。

其他资源

- **firewalld.zones(5)** 手册页

7.17. 使用 RHEL 系统角色配置 FIREWALLD

您可以使用 **firewall** RHEL 系统角色一次在多个客户端上配置 **firewalld** 服务的设置。这个解决方案：

- 提供具有有效输入设置的接口。
- 将所有预期的 **firewalld** 参数保存在一个地方。

在控制节点上运行 **firewall** 角色后，RHEL 系统角色立即将 **firewalld** 参数应用到受管节点，并使其在重启后持久保留。

7.17.1. firewall RHEL 系统角色简介

RHEL 系统角色是 Ansible 自动化工具的一组内容。此内容与 Ansible 自动化工具一起提供了一致的配置界面，来远程管理多个系统。

RHEL 系统角色中的 **rhel-system-roles.firewall** 角色是为自动化 **firewalld** 服务的配置而引入的。**rhel-system-roles** 软件包包含这个 RHEL 系统角色，以及参考文档。

要以自动化方式在一个或多个系统上应用 **firewalld** 参数，请在 playbook 中使用 **firewall** RHEL 系统角色变量。playbook 是一个或多个以基于文本的 YAML 格式编写的 play 的列表。

您可以使用清单文件来定义您希望 Ansible 来配置的一组系统。

使用 **firewall** 角色，您可以配置许多不同的 **firewalld** 参数，例如：

- 区。
- 应允许哪些数据包的服务。
- 授权、拒绝或丢弃访问端口的流量。
- 区的端口或端口范围的转发。

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件
- **/usr/share/doc/rhel-system-roles/firewall/** 目录
- 使用 playbook
- 如何构建清单

7.17.2. 使用 firewall RHEL 系统角色重置 firewalld 设置

使用 **firewall** RHEL 系统角色，您可以将 **firewalld** 设置重置为其默认状态。如果您将 **previous:replaced** 参数添加到变量列表中，RHEL 系统角色会删除所有现有用户定义的设置，并将 **firewalld** 重置为默认值。如果将 **previous:replaced** 参数与其他设置相结合，则 **firewall** 角色会在应用新设置前删除所有现有设置。

在 Ansible 控制节点上执行此步骤。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
    - name: Reset firewalld
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
```

```
    vars:
      firewall:
        - previous: replaced
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 在受管节点上以 **root** 用户身份运行这个命令，以检查所有区域：

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

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件
- **/usr/share/doc/rhel-system-roles/firewall/** 目录

7.17.3. 使用 firewall RHEL 系统角色，将 firewalld 中的传入流量从一个本地端口转发到不同的本地端口

使用 **firewall** 角色，您可以远程配置 **firewalld** 参数，使其对多个受管主机有效。

在 Ansible 控制节点上执行此步骤。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Forward incoming traffic on port 8080 to 443
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
```

```
    vars:
      firewall:
        - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 在受管主机上显示 **firewalld** 设置：

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

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件
- **/usr/share/doc/rhel-system-roles/firewall/** 目录

7.17.4. 使用 firewall RHEL 系统角色管理 firewalld 中的端口

您可以使用 **firewall** RHEL 系统角色为传入的流量在本地防火墙中打开或关闭端口，并使新配置在重启后保持不变。例如，您可以将默认区配置为允许 HTTPS 服务的传入流量。

在 Ansible 控制节点上执行此步骤。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Allow incoming HTTPS traffic to the local host
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
```

```
    firewall:
      - port: 443/tcp
        service: http
        state: enabled
        runtime: true
        permanent: true
```

permanent: true 选项可使新设置在重启后保持不变。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 在受管节点上，验证与 **HTTPS** 服务关联的 **443/tcp** 端口是否已打开：

```
# firewall-cmd --list-ports
443/tcp
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件
- **/usr/share/doc/rhel-system-roles/firewall/** 目录

7.17.5. 使用 firewall RHEL 系统角色配置 firewalld DMZ 区域

作为系统管理员，您可以使用 **firewall** RHEL 系统角色在 **enp1s0** 接口上配置一个 **dmz** 区域，以允许 **HTTPS** 流量到达区域。这样，您可以让外部用户访问您的 web 服务器。

在 Ansible 控制节点上执行此步骤。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
```

```
tasks:
  - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
    ansible.builtin.include_role:
      name: rhel-system-roles.firewall
    vars:
      firewall:
        - zone: dmz
          interface: enp1s0
          service: https
          state: enabled
          runtime: true
          permanent: true
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 在受管节点上，查看关于 **dmz** 区的详细信息：

```
# firewall-cmd --zone=dmz --list-all
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件
- **/usr/share/doc/rhel-system-roles/firewall/** 目录

第 8 章 NFTABLES 入门

nftables 框架对数据包进行分类，它是 **iptables**、**ip6tables**、**arptables**、**ebtables** 和 **ipset** 实用程序的后续者。与之前的数据包过滤工具相比，它在方便、特性和性能方面提供了大量改进，最重要的是：

- 内置查找表而不是线性处理
- **IPv4** 和 **IPv6** 协议的单一框架
- 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。**nft** 工具替换了之前数据包过滤框架中的所有工具。您可以使用 **libnftnl** 库通过 **libmnl** 库来处理 **nftables** Netlink API 的低级别交互。

要显示规则集变化的影响，请使用 **nft list ruleset** 命令。由于这些工具向 **nftables** 规则集添加表、链、规则、集合和其他对象，请注意 **nftables** 规则集的操作（如 **nft flush ruleset** 命令）可能会影响使用 **iptables** 命令安装的规则集。

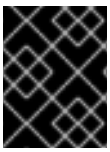
8.1. 从 IPTABLES 迁移到 NFTABLES

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

8.1.1. 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** ：对简单的防火墙用例使用 **firewalld** 工具。此工具易于使用，并涵盖了这些场景的典型用例。
- **nftables** ：使用 **nftables** 工具来设置复杂和性能关键的防火墙，如用于整个网络。
- **iptables** ：Red Hat Enterprise Linux 上的 **iptables** 工具使用 **nf_tables** 内核 API 而不是 **传统的**后端。**nf_tables** API 提供了向后兼容性，以便使用 **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

要防止不同的与防火墙相关的服务(**firewalld**、**nftables** 或 **iptables**)相互影响，请在 RHEL 主机上仅运行其中一个服务，并禁用其他服务。

8.1.2. 将 iptables 和 ip6tables 规则集转换为 nftables

使用 **iptables-restore-translate** 和 **ip6tables-restore-translate** 实用程序将 **iptables** 和 **ip6tables** 规则集转换为 **nftables**。

前提条件

- 已安装 **nftables** 和 **iptables** 软件包。

- 系统配置了 **iptables** 和 **ip6tables** 规则。

流程

1. 将 **iptables** 和 **ip6tables** 规则写入一个文件：

```
# iptables-save >/root/iptables.dump
# ip6tables-save >/root/ip6tables.dump
```

2. 将转储文件转换为 **nftables** 指令：

```
# iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-from-iptables.nft
# ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-from-ip6tables.nft
```

3. 检查，如果需要，手动更新生成的 **nftables** 规则。

4. 要启用 **nftables** 服务来加载生成的文件，请在 **/etc/sysconfig/nftables.conf** 文件中添加以下内容：

```
include "/etc/nftables/ruleset-migrated-from-iptables.nft"
include "/etc/nftables/ruleset-migrated-from-ip6tables.nft"
```

5. 停止并禁用 **iptables** 服务：

```
# systemctl disable --now iptables
```

如果您使用自定义脚本加载 **iptables** 规则，请确保脚本不再自动启动并重新引导以刷新所有表。

6. 启用并启动 **nftables** 服务：

```
# systemctl enable --now nftables
```

验证

- 显示 **nftables** 规则集：

```
# nft list ruleset
```

其他资源

- 系统引导时自动载入 nftables 规则

8.1.3. 将单个 iptables 和 ip6tables 规则转换为 nftables

Red Hat Enterprise Linux 提供了 **iptables-translate** 和 **ip6tables-translate** 工具来将 **iptables** 或 **ip6tables** 规则转换为与 **nftables** 相等的规则。

前提条件

- 已安装 **nftables** 软件包。

流程

- 使用 **iptables-translate** 或 **ip6tables-translate** 程序而不是 **iptables** 或 **ip6tables** 显示对应的 **nftables** 规则，例如：

```
# iptables-translate -A INPUT -s 192.0.2.0/24 -j ACCEPT
nft add rule ip filter INPUT ip saddr 192.0.2.0/24 counter accept
```

请注意，一些扩展可能缺少响应的转换支持。在这些情况下，实用程序会输出以 **#** 符号为前缀的未转换规则，例如：

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

其他资源

- **iptables-translate --help**

8.1.4. 常见的 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**

- 列出某个表和链：

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。只有当用户手动创建它们时它们才会存在。

列出 firewalld 生成的规则：

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

8.2. 编写和执行 NFTABLES 脚本

使用 **nftables** 框架的主要优点是脚本的执行是原子的。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，使用 **nftables** 脚本环境时，您可以：

- 添加评论
- 定义变量
- 包括其他规则集文件

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

8.2.1. 支持的 nftables 脚本格式

您可以使用以下格式在 **nftables** 脚本环境中编写脚本：

- 与 **nft list ruleset** 命令相同的格式显示规则集：

```
#!/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
  }
}
```

- 与 **nft** 命令的语法相同：

```
#!/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
```

8.2.2. 运行 nftables 脚本

您可以通过将脚本传递给 **nft** 实用程序或直接执行脚本来运行 **nftables** 脚本。

流程

- 要通过将其传给 **nft** 工具来运行 **nftables** 脚本，请输入：

```
# nft -f /etc/nftables/<example_firewall_script>.nft
```

- 要直接运行 **nftables** 脚本：

a. 在进行这个时间时：

i. 确保脚本以以下 shebang 序列开头：

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

重要

如果省略了 **-f** 参数，**nft** 工具不会读取脚本，并显示 **Error: syntax error, unexpected newline, expecting string**。

ii. 可选：将脚本的所有者设为 **root** ：

```
# chown root /etc/nftables/<example_firewall_script>.nft
```

iii. 使脚本可以被其所有者执行：

```
# chmod u+x /etc/nftables/<example_firewall_script>.nft
```

b. 运行脚本：

```
# /etc/nftables/<example_firewall_script>.nft
```

如果没有输出结果，系统将成功执行该脚本。

重要

即使 **nft** 成功地执行了脚本，在脚本中错误放置的规则、缺失的参数或其他问题都可能会导致防火墙的行为不符合预期。

其他资源

- **chown (1)** 手册页
- **chmod (1)** 手册页
- 系统引导时自动载入 nftables 规则

8.2.3. 使用 nftables 脚本中的注释

nftables 脚本环境将 **#** 字符右侧的所有内容解释为行尾。

注释可在行首或命令旁边开始：

```
...
```

```
# Flush the rule set
flush ruleset

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

8.2.4. 使用 nftables 脚本中的变量

要在 **nftables** 脚本中定义一个变量，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
```

您可以通过输入 **$** 符号后再输入变量名称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 **$** 符号后跟变量名称来在脚本中使用变量：

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

注意

当您在规则中使用大括号时具有特殊的语义，因为它们表示变量代表一个集合。

其他资源

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

8.2.5. 在 nftables 脚本中包含文件

在 **nftables** 脚本环境中，您可以使用 **include** 语句包含其他脚本。

如果您只指定没有绝对或相对路径的文件名，**nftables** 包含了默认搜索路径（设置为 Red Hat Enterprise Linux 上的 **/etc** ）。

例 8.1. 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 8.2. 包含目录中的所有 *.nft 文件

要包括以 ***.nft** 结尾的所有文件，它们存储在 **/etc/nftables/rulesets/** 目录中：

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

请注意，**include** 语句不匹配以点开头的文件。

其他资源

- **nft(8)** 手册页中的 **Include files** 部分

8.2.6. 系统引导时自动载入 nftables 规则

nftables systemd 服务加载包含在 **/etc/sysconfig/nftables.conf** 文件中的防火墙脚本。

前提条件

- **nftables** 脚本存储在 **/etc/nftables/** 目录中。

流程

1. 编辑 **/etc/sysconfig/nftables.conf** 文件。
   - 如果您使用 **nftables** 软件包的安装修改了在 **/etc/nftables/** 中创建的 ***.nft** 脚本，请取消对这些脚本的 **include** 语句的注释。
   - 如果您编写了新脚本，请添加 **include** 语句以包含这些脚本。例如，要在 **nftables** 服务启动时加载 **/etc/nftables/*example*.nft** 脚本，请添加：

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

2. 可选：启动 **nftables** 服务来加载防火墙规则，而无需重启系统：

   ```
   # systemctl start nftables
   ```

3. 启用 **nftables** 服务。

   ```
   # systemctl enable nftables
   ```

其他资源

- 支持的 nftables 脚本格式

8.3. 创建和管理 NFTABLES 表、链和规则

您可以显示 **nftables** 规则集并管理它们。

8.3.1. nftables 表的基础知识

nftables 中的表是一个包含链、规则、集合和其他对象集合的名字空间。

每个表都必须分配一个地址系列。地址系列定义了此表进程的数据包类型。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** ：仅匹配 IPv6 数据包。
- **inet** ：匹配 IPv4 和 IPv6 数据包。
- **arp**：匹配 IPv4 地址解析协议(ARP)数据包。
- **bridge** ：匹配通过网桥设备的数据包。
- **netdev** ：匹配来自入口的数据包。

如果要添加表，要使用的格式取决于您的防火墙脚本：

- 在原生语法的脚本中，使用：

```
table <table_address_family> <table_name> {
}
```

- 在 shell 脚本中，使用：

```
nft add table <table_address_family> <table_name>
```

8.3.2. nftables 链的基础知识

表由链组成，后者是规则的容器。存在以下两种规则类型：

- **基本链** ：您可以使用基本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 **跳板** 目标来更好地组织规则。

如果要在表中添加基本链，要使用的格式取决于您的防火墙脚本：

- 在原生语法的脚本中，使用：

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority>
    policy <policy> ;
  }
}
```

- 在 shell 脚本中，使用：

```
nft add chain <table_address_family> <table_name> <chain_name> { type <type> hook <hook> priority <priority> \; policy <policy> \; }
```

为了避免 shell 将分号解释为命令的结尾，请在分号的前面放置 \ 转义字符。

这两个示例都创建了 **基本链**。要创建 **常规链**，请不要在大括号中设置任何参数。

链类型
以下是链类型以及您可以使用的地址系列和 hook 概述：

类型	地址系列	Hook	描述
filter	all	all	标准链类型
nat	**ip,ip6,inet**	**PREROUTING、输入、输出、postrouting**	这个类型的链根据连接跟踪条目执行原生地址转换。只有第一个数据包遍历此链类型。
route	**ip,ip6**	**output**	如果 IP 标头的相关部分已更改，则遍历此链类型的数据包会导致新的路由查找。

链优先级
priority 参数指定数据包使用相同的 hook 值遍历链的顺序。您可以将此参数设置为整数值，或使用标准优先级名称。

以下列表概述了标准优先级名称及其数字值，您可以使用哪个地址系列和 hook：

文本值	数字值	地址系列	Hook
raw	**-300**	**ip,ip6,inet**	all
mangle	**-150**	**ip,ip6,inet**	all
dstnat	**-100**	**ip,ip6,inet**	**prerouting**
	-300	**bridge**	**prerouting**
filter	**0**	**ip,ip6,inet,arp,netdev**	all
	-200	**bridge**	all
安全	**50**	**ip,ip6,inet**	all
srcnat	**100**	**ip,ip6,inet**	**postrouting**
	300	**bridge**	**postrouting**
out	**100**	**bridge**	**output**

链策略
链策略定义 **nftables** 是否应该接受或丢弃数据包（如果此链中的规则没有指定任何操作）。您可以在链中设置以下策略之一：

- **accept**（默认）

- **drop**

8.3.3. nftables 规则的基础知识

规则定义要在传递包含此规则的链的数据包上执行操作。如果规则还包含匹配的表达式，**nftables** 仅在所有之前的表达式都应用时才执行操作。

如果要在链中添加规则，要使用的格式取决于您的防火墙脚本：

- 在原生语法的脚本中，使用：

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority> ; policy <policy> ;
      <rule>
  }
}
```

- 在 shell 脚本中，使用：

```
nft add rule <table_address_family> <table_name> <chain_name> <rule>
```

此 shell 命令在链末尾附加新规则。如果要在链开始时添加规则，请使用 **nft insert** 命令而不是 **nft add**。

8.3.4. 使用 nft 命令管理表、链和规则

要在命令行或 shell 脚本中管理 **nftables** 防火墙，请使用 **nft** 工具。

重要

此流程中的命令不代表典型的工作流，且不会被优化。此流程演示了如何使用 **nft** 命令管理表、链和规则。

流程

1. 使用 **inet** 地址系列创建一个名为 **nftables_svc** 的表，以便表可以处理 IPv4 和 IPv6 数据包：

```
# nft add table inet nftables_svc
```

2. 将名为 **INPUT** 的基本链（用于处理传入的网络流量）添加到 **inet nftables_svc** 表中：

```
# nft add chain inet nftables_svc INPUT { type filter hook input priority filter \; policy accept \; }
```

为了避免 shell 将分号解释为命令的结尾，请使用 \ 字符转义分号。

3. 向 **INPUT** 链中添加规则。例如，允许端口 22 和 443 上的传入 TCP 流量，以及作为 **INPUT** 链的最后一个规则，拒绝其他带有互联网控制消息协议(ICMP)端口无法访问的消息的流量：

```
# nft add rule inet nftables_svc INPUT tcp dport 22 accept
# nft add rule inet nftables_svc INPUT tcp dport 443 accept
# nft add rule inet nftables_svc INPUT reject with icmpx type port-unreachable
```

如果您输入 **nft add rule** 命令，**nft** 会将规则按与运行命令相同的顺序添加规则。

4. 显示包括句柄的当前规则集：

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

5. 在带有句柄 3 的现有规则前面插入规则。例如，要插入一个允许端口 636 上 TCP 流量的规则，请输入：

```
# nft insert rule inet nftables_svc INPUT position 3 tcp dport 636 accept
```

6. 使用句柄 3 在现有规则后面附加规则。例如，要插入一个允许端口 80 上 TCP 流量的规则，请输入：

```
# nft add rule inet nftables_svc INPUT position 3 tcp dport 80 accept
```

7. 再次使用 handle 显示规则集。验证后续添加的规则是否已添加到指定位置：

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # 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
    reject # handle 4
  }
}
```

8. 删除带有句柄 6 的规则：

```
# nft delete rule inet nftables_svc INPUT handle 6
```

要删除规则，您必须指定句柄。

9. 显示规则集，并验证删除的规则不再存在：

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # 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
```

```
    reject # handle 4
  }
}
```

10. 从 **INPUT** 链中删除所有剩余的规则：

```
# nft flush chain inet nftables_svc INPUT
```

11. 显示规则集，并验证 **INPUT** 链是否为空：

```
# nft list table inet nftables_svc
table inet nftables_svc {
  chain INPUT {
    type filter hook input priority filter; policy accept
  }
}
```

12. 删除 **INPUT** 链：

```
# nft delete chain inet nftables_svc INPUT
```

您还可以使用此命令删除仍然包含规则的链。

13. 显示规则集，并验证 **INPUT** 链已被删除：

```
# nft list table inet nftables_svc
table inet nftables_svc {
}
```

14. 删除 **nftables_svc** 表：

```
# nft delete table inet nftables_svc
```

您还可以使用此命令删除仍然包含链的表。

注意

要删除整个规则集，请使用 **nft flush ruleset** 命令，而不是手动删除独立命令中的所有规则、链和表。

其他资源

nft(8) man page

8.4. 使用 NFTABLES 配置 NAT

有了 **nftables**，您就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

重要

您只能在 **iifname** 和 **oifname** 参数中使用实际的接口名称，不支持替代名称(**altname**)。

8.4.1. NAT 类型

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(ISP)不会路由私有 IP 范围，如 **10.0.0.0/8**。如果您在网络中使用私有 IP 范围，并且用户可以访问互联网上的服务器，请将这些范围内的数据包的源 IP 地址映射到公共 IP 地址。
伪装和 SNAT 相互类似。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，那么您可以在路由器上设置一个 DNAT 规则，来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，如果服务运行在与其标准端口不同的端口上，您可以将传入的流量从标准端口重定向到此特定端口。

8.4.2. 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，如果接口被分配了一个新 IP，**nftables** 会在替换源 IP 时自动使用新的 IP。

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. 向表中添加 **prerouting** 和 **postrouting** 链：

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

重要

即使您没有向 **prerouting** 链中添加规则，**nftables** 框架也会要求此链与传入的数据包回复匹配。

请注意，您必须将 **--** 选项传递给 **nft** 命令，以防止 shell 将负优先级值解释为 **nft** 命令的选项。

3. 向 **postrouting** 链中添加一条规则，来匹配 **ens3** 接口上传出的数据包：

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

8.4.3. 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。然后，路由器会替换传出数据包的源 IP。

流程

1. 创建一个表：

```
# nft add table nat
```

2. 向表中添加 **prerouting** 和 **postrouting** 链：

```
# 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，**nftables** 框架也会要求此链与传出数据包回复相匹配。

请注意，您必须将 **--** 选项传递给 **nft** 命令，以防止 shell 将负优先级值解释为 **nft** 命令的选项。

3. 向 **postrouting** 链中添加一条规则，该规则将使用 **192.0.2.1** 替换通过**ens3** 的传出数据包的源 IP：

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

其他资源

- 将特定本地端口上传入的数据包转发到不同主机

8.4.4. 使用 nftables 配置目标 NAT

目标 NAT (DNAT)可让您将路由器上的流量重定向到从互联网无法访问的主机。

例如，对于 DNAT，路由器将发送到端口 **80** 和 **443** 的传入流量重定向到 IP 地址为 **192.0.2.1** 的 Web 服务器。

流程

1. 创建一个表：

```
# nft add table nat
```

2. 向表中添加 **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，**nftables** 框架也会要求此链与传出数据包回复相匹配。

请注意，您必须将 **--** 选项传递给 **nft** 命令，以防止 shell 将负优先级值解释为 **nft** 命令的选项。

3. 向 **prerouting** 链中添加一条规则，该规则将路由器的 **ens3** 接口上端口 **80** 和 **443** 的传入流量重定向到 IP 地址为 **192.0.2.1** 的 web 服务器：

```
# 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，将从 Web 服务器返回的数据包的源地址改为发送者：

   a. 如果 **ens3** 接口使用动态 IP 地址，请添加一条伪装规则：

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

   b. 如果 **ens3** 接口使用静态 IP 地址，请添加一条 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
```

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
```

其他资源

- NAT 类型

8.4.5. 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。

例如，您可以将发送到本地主机端口 **22** 的流量重定向到端口 **2222**。

流程

1. 创建一个表：

```
# nft add table nat
```

2. 向表中添加 **prerouting** 链：

```
# 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，以防止 shell 将负优先级值解释为 **nft** 命令的选项。

3. 向 **prerouting** 链中添加一条规则，其将端口 **22** 上的传入流量重定向到端口 **2222** ：

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

其他资源

- NAT 类型

8.4.6. 使用 nftables 配置 flowtable

nftables 工具使用 **netfilter** 框架为网络流量提供网络地址转换(NAT)，并提供基于快速路径功能的 **flowtable** 机制来加快数据包转发。

flowtable 机制有以下功能：

- 使用连接跟踪来绕过典型的数据包转发路径。
- 避免通过绕过经典数据包处理来重新访问路由表。
- 只适用于 TCP 和 UDP 协议。
- 硬件独立软件快速路径。

流程

1. 添加 **inet** 系列的一个 **example-table** ：

```
# nft add table inet <example-table>
```

2. 添加一个带有 **ingress** 钩子和 **filter** 作为优先级类型的 **example-flowtable** flowtable ：

```
# nft add flowtable inet <example-table> <example-flowtable> { hook ingress priority filter \; devices = { enp1s0, enp7s0 } \; }
```

3. 将 **example-forwardchain** 流添加到数据包处理表中的 flowtable ：

```
# nft add chain inet <example-table> <example-forwardchain> { type filter hook forward priority filter \; }
```

此命令添加了一个带有 **forward** 钩子和 **filter** 优先级的 **filter** 类型的 flowtable 。

4. 添加一个 **established** 连接跟踪状态的规则，来卸载 **example-flowtable** 流：

```
# nft add rule inet <example-table> <example-forwardchain> ct state established flow add @<example-flowtable>
```

验证

- 验证 **example-table** 的属性：

```
# nft list table inet <example-table>
table inet example-table {
    flowtable example-flowtable {
```

```
        hook ingress priority filter
            devices = { enp1s0, enp7s0 }
    }

    chain example-forwardchain {
type filter hook forward priority filter; policy accept;
ct state established flow add @example-flowtable
    }
}
```

其他资源

- **nft(8)** 手册页

8.5. 使用 NFTABLES 命令中的集合

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。

8.5.1. 在 nftables 中使用匿名集合

匿名集合包含用逗号分开的值，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 地址以及任何其他匹配标准。

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，使用命名集合，如 在 nftables 中使用命名集合 中所述。

前提条件

- **inet** 系列中的 **example_chain** 链和 **example_table** 表存在。

流程

1. 例如，要向 **example_table** 中的 **example_chain** 添加一条规则，其允许传入流量到端口 **22**、**80** 和 **443** ：

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

2. 可选： 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
  }
}
```

8.5.2. 在 nftables 中使用命名集

nftables 框架支持可变命名集合。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 IPv6 地址或范围的集合的 **ipv6_addr**，如 **2001:db8:1::1** 或 **2001:db8:1::1/64**。
- 包含介质访问控制(MAC)地址列表的集合的 **ether_addr** ，如 **52:54:00:6b:66:42**。
- 包含互联网协议类型列表的集合的 **inet_proto**，如 **tcp**。
- 包含互联网服务列表的集合的 **inet_service**，如 **ssh**。
- 包含数据包标记列表的集合的 **mark**。数据包标记可以是任意正 32 位整数值(**0** 到 **2147483647**)。

前提条件

- **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

     要防止 shell 将分号解释为命令结尾，您必须使用反斜杠转义分号。

2. 可选：创建使用集合的规则。例如，以下命令向 **example_table** 中的 **example_chain** 中添加一条规则，该规则将丢弃 **example_set** 中来自 IPv4 地址的所有数据包。

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

   由于 **example_set** 仍为空，所以该规则目前不起作用。

3. 向 **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，您可以使用无类别域间路由(CIDR)表示法，如上例中的 **192.0.2.0/24**。

8.5.3. 其他资源

- **nft(8)** 手册页中的 **Sets** 部分

8.6. 在 NFTABLES 命令中使用 VERDICT 映射

判决映射（也称为字典），使 **nft** 能够通过将匹配条件映射到某个操作来根据数据包信息执行操作。

8.6.1. 在 nftables 中使用匿名映射

匿名映射是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，如果要修改映射，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。

例如，您可以使用匿名映射将 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. 创建新表：

```
# 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. 向统计此链中流量的 **tcp_packets** 中添加一条规则：

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

4. 在 **example_table** 中创建 **udp_packets** 链

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

5. 向统计此链中流量的 **udp_packets** 中添加一条规则：

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

6. 为传入的流量创建一个链。例如，要在过滤传入的流量的 **example_table** 中创建一个名为 **incoming_traffic** 的链：

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

7. 添加一条带有到 **incoming_traffic** 匿名映射的规则 ：

```
# 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，请显示 **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** 链中的计数器显示两者接收的数据包和字节数。

8.6.2. 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。匿名映射的另一个好处在于，您可以更新命名映射而不比替换使用它的规则。

在创建命名映射时，您必须指定元素的类型：

- 匹配部分包含 IPv4 地址的映射的 **ipv4_addr**，如 **192.0.2.1**。
- 匹配部分包含 IPv6 地址的映射的 **ipv6_addr**，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**。
- 匹配部分包含互联网协议类型的映射的 **inet_proto**，如 **tcp**。
- 匹配部分包含互联网服务名称端口号的映射的 **inet_service**，如 **ssh** 或 **22**。
- 匹配部分包含数据包的映射的 **mark** 。数据包标记可以是任意正 32 位整数值(**0** 到 **2147483647**)。
- 匹配部分包含计数器值的映射的 **counter**。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的 **quota**。配额值可以是任意正 64 位整数值。

例如，您可以根据其源 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场景，而 IP 地址和操作被动态存储在映射中。

流程

1. 创建表。例如，要创建一个处理 IPv4 数据包的、名为 **example_table** 的表：

```
# 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

要防止 shell 将分号解释为命令结尾，您必须使用反斜杠转义分号。

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，以下命令向 **example_table** 中的 **example_chain** 添加了一条规则，该规则将操作应用到在 **example_map** 中定义的 IPv4 地址：

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. 向 **example_map** 添加 IPv4 地址和相应的操作：

```
# 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。与以上创建的规则相结合，防火墙接受来自 **192.0.2.1** 的数据包，丢弃来自 **192.0.2.2** 的数据包。

6. 可选：通过添加另一个 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. 可选：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
  }
}
```

8.6.3. 其他资源

- **nft(8)** 手册页中的 **Maps** 部分

8.7. 示例：使用 NFTABLES 脚本保护 LAN 和 DMZ

使用 RHEL 路由器上的 **nftables** 框架来编写和安装防火墙脚本，保护内部 LAN 中的网络客户端和 DMZ 中的 Web 服务器免受来自互联网和其他网络的未授权访问。

重要

这个示例仅用于演示目的，并描述了具有特定要求的场景。

防火墙脚本高度依赖于网络基础架构和安全要求。当您为自己的环境编写脚本时，请使用此示例了解 **nftables** 防火墙的概念。

8.7.1. 网络状况

本例中的网络具有以下条件：

- 路由器连接到以下网络：
  - 互联网通过接口 **enp1s0**
  - 内部 LAN 通过接口 **enp7s0**
  - DMZ through **enp8s0**
- 路由器的互联网接口分配了静态 IPv4 地址(**203.0.113.1**)和 IPv6 地址(**2001:db8:a::1**)。
- 内部 LAN 中的客户端仅使用范围 **10.0.0.0/24** 中的专用 IPv4 地址。因此，来 LAN 到互联网的流量需要源网络地址转换(SNAT)。
- 内部 LAN 中的管理员 PC 使用 IP 地址 **10.0.0.100** 和 **10.0.0.200**。
- DMZ 使用范围 **198.51.100.0/24** 和 **2001:db8:b::/56** 的公共 IP 地址。
- DMZ 中的 Web 服务器使用 IP 地址 **198.51.100.5** 和 **2001:db8:b::5**。
- 路由器充当 LAN 和 DMZ 中的主机的缓存 DNS 服务器。

8.7.2. 防火墙脚本的安全要求

以下是示例网络中 **nftables** 防火墙的要求：

- 路由器必须能够：
  - 递归解析 DNS 查询。
  - 在回环接口上执行所有连接。
- 内部 LAN 中的客户端必须能够：
  - 查询在路由器上运行的缓存 DNS 服务器。
  - 访问 DMZ 中的 HTTPS 服务器。
  - 访问互联网上的任何 HTTPS 服务器。
- 管理员的 PC 必须能够使用 SSH 访问 DMZ 中的路由器和每个服务器。
- DMZ 中的 Web 服务器必须能够：

- 查询在路由器上运行的缓存 DNS 服务器。
- 访问互联网上的 HTTPS 服务器以下载更新。

- 互联网上的主机必须能够：
  - 访问 DMZ 中的 HTTPS 服务器。
- 另外，存在以下安全要求：
  - 应丢弃未明确允许的连接尝试。
  - 应记录丢弃的数据包。

8.7.3. 配置丢弃的数据包记录到文件中

默认情况下，**systemd** 会将内核消息（如 丢弃的数据包）记录到日志中。另外，您可以配置 **rsyslog** 服务，来将此类条目记录到单独的文件中。为确保日志文件不会无限增长，请配置轮转策略。

前提条件

- **rsyslog** 软件包已安装。
- **rsyslog** 服务正在运行。

流程

1. 使用以下内容创建 **/etc/rsyslog.d/nftables.conf** 文件：

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

使用这个配置，**rsyslog** 服务会将数据包丢弃到 **/var/log/nftables.log** 文件中，而不是 **/var/log/messages**。

2. 重启 **rsyslog** 服务：

```
# systemctl restart rsyslog
```

3. 使用以下内容创建 **/etc/logrotate.d/nftables** 文件，以便在大小超过 10 MB 时轮转 **/var/log/nftables.log** ：

```
/var/log/nftables.log {
  size +10M
  maxage 30
  sharedscripts
  postrotate
    /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
  endscript
}
```

maxage 30 设置定义了 **logrotate** 在下一次轮转操作期间删除超过 30 天的轮转日志。

其他资源

- **rsyslog.conf (5)** 手册页
- **logrotate(8)** man page

8.7.4. 编写并激活 nftables 脚本

本例是在 RHEL 路由器上运行的 **nftables** 防火墙脚本，可保护内部 LAN 中的客户端以及 DMZ 中的 Web 服务器。有关示例中使用的防火墙的网络和要求的详情，请参阅 网络条件 和安全要求。

警告

此 **nftables** 防火墙脚本仅用于演示目的。在符合您的环境和安全要求的情况下不要使用它。

前提条件

- 网络已配置，如 网络状况 中所述。

流程

1. 使用以下内容创建 **/etc/nftables/firewall.nft** 脚本：

```
# Remove all rules
flush ruleset


# Table for both IPv4 and IPv6 rules
table inet nftables_svc {

  # Define variables for the interface name
  define INET_DEV = enp1s0
  define LAN_DEV  = enp7s0
  define DMZ_DEV  = enp8s0


  # Set with the IPv4 addresses of admin PCs
  set admin_pc_ipv4 {
    type ipv4_addr
    elements = { 10.0.0.100, 10.0.0.200 }
  }


  # Chain for incoming trafic. Default policy: drop
  chain INPUT {
    type filter hook input priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }
```

```
    # Accept incoming traffic on loopback interface
    iifname lo accept

    # Allow request from LAN and DMZ to local DNS server
    iifname { $LAN_DEV, $DMZ_DEV } meta l4proto { tcp, udp } th dport 53 accept

    # Allow admins PCs to access the router using SSH
    iifname $LAN_DEV ip saddr @admin_pc_ipv4 tcp dport 22 accept

    # Last action: Log blocked packets
    # (packets that were not accepted in previous rules in this chain)
    log prefix "nft drop IN : "
  }


  # Chain for outgoing traffic. Default policy: drop
  chain OUTPUT {
    type filter hook output priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # Accept outgoing traffic on loopback interface
    oifname lo accept

    # Allow local DNS server to recursively resolve queries
    oifname $INET_DEV meta l4proto { tcp, udp } th dport 53 accept

    # Last action: Log blocked packets
    log prefix "nft drop OUT: "
  }


  # Chain for forwarding traffic. Default policy: drop
  chain FORWARD {
    type filter hook forward priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # IPv4 access from LAN and internet to the HTTPS server in the DMZ
    iifname { $LAN_DEV, $INET_DEV } oifname $DMZ_DEV ip daddr 198.51.100.5 tcp dport
443 accept

    # IPv6 access from internet to the HTTPS server in the DMZ
    iifname $INET_DEV oifname $DMZ_DEV ip6 daddr 2001:db8:b::5 tcp dport 443 accept

    # Access from LAN and DMZ to HTTPS servers on the internet
    iifname { $LAN_DEV, $DMZ_DEV } oifname $INET_DEV tcp dport 443 accept

    # Last action: Log blocked packets
    log prefix "nft drop FWD: "
  }
```

```
    # Postrouting chain to handle SNAT
    chain postrouting {
      type nat hook postrouting priority srcnat; policy accept;

      # SNAT for IPv4 traffic from LAN to internet
      iifname $LAN_DEV oifname $INET_DEV snat ip to 203.0.113.1
    }
  }
```

2. 在 **/etc/sysconfig/nftables.conf** 文件中包括 **/etc/nftables/firewall.nft** 脚本：

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

3. 启用 IPv4 转发：

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

4. 启用并启动 **nftables** 服务：

```
# systemctl enable --now nftables
```

验证

1. 可选：验证 **nftables** 规则集：

```
# nft list ruleset
...
```

2. 尝试执行防火墙阻止的访问。例如，尝试使用 DMZ 中的 SSH 访问路由器：

```
# ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. 根据您的日志记录设置，搜索：

- 阻塞的数据包的 **systemd** 日志：

```
# journalctl -k -g "nft drop"
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

- 阻塞的数据包的 **/var/log/nftables.log** 文件：

```
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

8.8. 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一条端口转发规则，将防火墙上端口 **80** 和 **443** 上的传入数据包转发到 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。

8.8.1. 将传入的数据包转发到不同的本地端口

您可以使用 **nftables** 来转发数据包。例如，您可以将端口 **8022** 上的传入 IPv4 数据包转发到本地系统上的端口 **22**。

流程

1. 创建一个名为 **nat**、具有 **ip** 地址系列的表：

```
# nft add table ip nat
```

2. 向表中添加 **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，以防止 shell 将负优先级值解析为 **nft** 命令的选项。

3. 向 **prerouting** 链中添加一条规则，将端口 **8022** 上的传入数据包重定向到本地端口 **22** ：

```
# nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

8.8.2. 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。这可让互联网上的用户访问运行在具有私有 IP 地址的主机上的服务。

例如，您可以将本地端口 **443** 上的传入 IPv4 数据包转发到 IP 地址为 **192.0.2.1** 的远程系统上的同一端口号。

前提条件

- 您以 **root** 用户身份登录应该转发数据包的系统上。

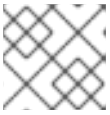
流程

1. 创建一个名为 **nat**、具有 **ip** 地址系列的表：

```
# nft add table ip nat
```

2. 向表中添加 **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，以防止 shell 将负优先级值解析为 **nft** 命令的选项。

3. 向 **prerouting** 链添加一条规则，该规则将端口 **443** 上的传入数据包重定向到 **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. 向 **postrouting** 链中添加一条规则来伪装出站流量：

```
# nft add rule ip nat postrouting 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
```

8.9. 使用 NFTABLES 来限制连接数量

您可以使用 **nftables** 来限制连接数或限制到建立给定数量连接的块 IP 地址，以防止它们使用太多的系统资源。

8.9.1. 使用 nftables 限制连接数量

nft 工具的 **ct count** 参数使管理员能够限制连接数。

前提条件

- **example_table** 中的基础**example_chain** 存在。

流程

1. 为 IPv4 地址创建动态集合：

```
# nft add set inet example_table example_meter { type ipv4_addr\; flags dynamic \;}
```

2. 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
```

3. 可选：显示上一步中创建的集合：

```
# nft list set inet example_table example_meter
table inet 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  }
  }
}
```

elements 条目显示当前与该规则匹配的地址。在本例中，**elements** 列出了与 SSH 端口有活动连接的 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。

8.9.2. 在一分钟内尝试超过十个进入的 TCP 连接的 IP 地址

您可以在一分钟内临时阻止建立十个 IPv4 TCP 连接的主机。

流程

1. 创建具有 **ip** 地址系列的 **filter** 表：

```
# nft add table ip filter
```

2. 向 **filter** 表中添加 **input** 链：

```
# nft add chain ip filter input { type filter hook input priority 0 \; }
```

3. 添加一条规则，其丢弃来自源地址的所有数据包，并尝试在一分钟内建立十个 TCP 连接：

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked meter ratemeter { ip saddr timeout 5m limit rate over 10/minute } drop
```

timeout 5m 参数定义 **nftables** 在五分钟后自动删除条目，以防止计量被过时的条目填满。

验证

- 要显示 meter 的内容，请输入：

```
# nft list meter ip filter ratemeter
table ip filter {
  meter ratemeter {
    type ipv4_addr
    size 65535
    flags dynamic,timeout
    elements = { 192.0.2.1 limit rate over 10/minute timeout 5m expires 4m58s224ms }
  }
}
```

8.10. 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，以及数据包是否匹配。

8.10.1. 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。

- 有关在现有规则中添加计数器的流程的更多信息，请参阅 **配置和管理网络** 中的 向现有规则中添加计数器

前提条件

- 您要添加该规则的链已存在。

流程

1. 向链中添加一条带有 **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
  }
}
```

8.10.2. 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。

- 有关添加带有计数器的新规则的流程的更多信息，请参阅 **配置和管理网络** 中的 创建带有计数器的规则

前提条件

- 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. 通过使用 **counter** 参数替换规则来添加计数器。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
  }
}
```

8.10.3. 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，使管理员能够显示与某一规则匹配的数据包。您可以为规则启用追踪，使用它来监控与此规则匹配的数据包。

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. 通过使用 **meta nftrace set 1** 参数替换规则来添加追踪功能。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** 命令来显示追踪。以下示例过滤了命令的输出，来只显示包含 **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

根据启用了追踪的规则数量以及匹配流量的数量，**nft monitor** 命令可以显示很多输出。使用 **grep** 或其他工具来过滤输出。

8.11. 备份和恢复 NFTABLES 规则集

您可以将 **nftables** 规则备份到文件，稍后恢复它们。此外，管理员可以使用规则的文件将规则传输至其他服务器。

8.11.1. 将 nftables 规则集备份到文件

您可以使用 **nft** 工具将 **nftables** 规则集备份到文件。

流程

- 要备份 **nftables** 规则：
  - 以 **nft list ruleset** 格式生成的格式：

```
# nft list ruleset > file.nft
```

  - JSON 格式：

```
# nft -j list ruleset > file.json
```

8.11.2. 从文件中恢复 nftables 规则集

您可以从文件恢复 **nftables** 规则集。

流程

- 要恢复 **nftables** 规则：
  - 如果要恢复的文件是 **nft list ruleset** 生成的格式，或直接包含 **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
```

8.12. 其他资源

- 在 Red Hat Enterprise Linux 8 中使用 nftables
- iptables 之后是什么？当然，它的继任者是：nftables
- Firewalld：未来是 nftables

第 9 章 保护网络服务

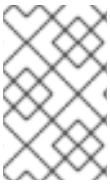
Red Hat Enterprise Linux 8 支持许多不同类型的网络服务器。这些服务器提供的网络服务可能会暴露系统的安全性，从而可能会面临各种类型的攻击，如拒绝服务攻击(DoS)、分布式拒绝服务攻击(DDoS)、脚本漏洞攻击和缓冲区溢出攻击。

为增加系统的安全性，务必要监控您使用的活跃网络服务。例如，当网络服务在计算机上运行时，其守护进程会侦听网络端口的连接，这可能会降低系统的安全性。要限制暴露会受到攻击的网络，应关闭所有未使用的服务。

9.1. 保护 RPCBIND 服务

rpcbind 服务是用于远程过程调用(RPC)服务的动态端口分配守护进程，如网络信息服务(NIS)和网络文件共享(NFS)。由于其身份验证机制较弱，并可为其控制的服务分配大量端口，因此保护 **rpcbind** 服务非常重要。

您可以通过限制访问所有网络并使用服务器上的防火墙规则来定义特定例外来保护 **rpcbind**。



注意

- **NFSv2** 和 **NFSv3** 服务器中需要 **rpcbind** 服务。
- **NFSv4** 上不需要 **rpcbind** 服务。

前提条件

- 已安装 **rpcbind** 软件包。
- **firewalld** 软件包已安装，且服务正在运行。

流程

1. 添加防火墙规则，例如：
   - 限制 TCP 连接，并只接受来自 **192.168.0.0/24** 主机 **111** 端口的包：

     ```
     # firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24" invert="True" drop'
     ```

   - 限制 TCP 连接，并只接受来自本地主机 **111** 端口的包：

     ```
     # firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept'
     ```

   - 限制 UDP 连接，并只接受来自 **192.168.0.0/24** 主机 **111** 端口的包：

     ```
     # firewall-cmd --permanent --add-rich-rule='rule family="ipv4" port port="111" protocol="udp" source address="192.168.0.0/24" invert="True" drop'
     ```

     要使防火墙设置永久化，请在添加防火墙规则时使用 **--permanent** 选项。

2. 重新载入防火墙以应用新规则：

   ```
   # firewall-cmd --reload
   ```

-

验证步骤

- 列出防火墙规则：

```
# firewall-cmd --list-rich-rule
rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24"
invert="True" drop
rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept
rule family="ipv4" port port="111" protocol="udp" source address="192.168.0.0/24"
invert="True" drop
```

其他资源

- 有关 **只使用 NFSv4 的服务器** 的详情，请参考 配置只使用 NFSv4 的服务器。
- 使用和配置 firewalld

9.2. 保护 RPC.MOUNTD 服务

rpc.mountd 守护进程实现 NFS 挂载协议的服务器端。NFS 挂载协议用于 NFS 版本 2(RFC 1904)和 NFS 版本 3(RFC 1813)。

您可以通过对服务器添加防火墙规则来保护 **rpc.mountd** 服务。您可以限制对所有网络的访问，并使用防火墙规则定义特定的异常。

前提条件

- 已安装 **rpc.mountd** 软件包。
- **firewalld** 软件包已安装，且服务正在运行。

流程

1. 在服务器中添加防火墙规则，例如：
   - 接受来自 **192.168.0.0/24** 主机的**mountd** 连接：

```
# firewall-cmd --add-rich-rule 'rule family="ipv4" service name="mountd" source
address="192.168.0.0/24" invert="True" drop'
```

   - 接受来自本地主机的 **mountd** 连接：

```
# firewall-cmd --permanent --add-rich-rule 'rule family="ipv4" source address="127.0.0.1"
service name="mountd" accept'
```

   要使防火墙设置永久化，请在添加防火墙规则时使用 **--permanent** 选项。

2. 重新载入防火墙以应用新规则：

```
# firewall-cmd --reload
```

验证步骤

- 列出防火墙规则：

```
# firewall-cmd --list-rich-rule
rule family="ipv4" service name="mountd" source address="192.168.0.0/24" invert="True"
drop
rule family="ipv4" source address="127.0.0.1" service name="mountd" accept
```

其他资源

- 使用和配置 firewalld

9.3. 保护 NFS 服务

您可以使用 Kerberos 验证并加密所有文件系统操作来保护网络文件系统 4(NFSv4)。在将 NFSv4 与网络地址转换(NAT)或防火墙搭配使用时，您可以通过修改 **/etc/default/nfs** 文件来关闭委托。委托（Delegation）是服务器将文件管理委派给客户端的一种技术。相反，NFSv2 和 NFSv3 不使用 Kerberos 锁定和挂载文件。

NFS 服务在所有 NFS 版本中使用 TCP 发送流量。该服务支持 Kerberos 用户和组身份验证，作为 **RPCSEC_GSS** 内核模块的一部分。

NFS 允许远程主机通过网络挂载文件系统，并与这些文件系统进行交互，就像它们被挂载到本地一样。您可以在集中服务器中合并资源，并在共享文件系统时额外自定义 **/etc/nfsmount.conf** 文件中的 NFS 挂载选项。

9.3.1. 保护 NFS 服务器的导出选项

NFS 服务器决定有关将哪些文件系统导出到 **/etc/exports** 文件中的目录和主机的列表结构。

警告

/etc/exports 文件语法中的额外空格可能会导致配置中的主要更改。

在以下示例中，**/tmp/nfs/** 目录与 **bob.example.com** 主机共享，并且具有读取和写入权限。

```
/tmp/nfs/     bob.example.com(rw)
```

以下示例与前一个相同，但对 **bob.example.com** 主机共享具有只读权限的相同的目录，由于主机名后面有一个空格字符，因此可以对 *world* 共享具有读写权限的目录。

```
/tmp/nfs/     bob.example.com (rw)
```

您可以通过输入 **showmount -e < *hostname> 命令来检查*** 系统上的共享目录。

您可以在 **/etc/exports** 文件中使用以下导出选项：

警告

要导出整个文件系统，因为导出文件系统的子目录不安全。攻击者可能会访问部分导出的文件系统中的未导出部分。

ro

将 NFS 卷导出为只读。

rw

允许在 NFS 卷上读取和写入请求。请小心使用这个选项，因为允许写入访问会增加攻击的风险。如果您的场景需要使用 **rw** 选项挂载目录，请确保所有用户都无法写入，以减少可能的风险。

root_squash

将来自 **uid/gid** 0 的请求映射到匿名 **uid/gid**。这不适用于其它可能比较敏感的 **uid** 或 **gid**，如 **bin** 用户或 **staff** 组。

no_root_squash

关闭 root 压缩。默认情况下，NFS 共享将 **root** 用户改为 **nobody** 用户，这是一个非特权用户帐户。这会将所有 **root** 创建的文件的所有者改为 **nobody**，这样可以防止上传设置了 **setuid** 位的程序。如果使用 **no_root_squash** 选项，则远程 root 用户可以更改共享文件系统上的任何文件，并将感染特洛伊木马的应用程序留给其他用户。

secure

将导出限制为保留端口。默认情况下，服务器只允许客户端通信通过保留的端口。但是在网络上，任何人都可以容易地成为客户端上的 **root** 用户，因此，对于服务器来说，假设通过保留端口的通信都具有特权是不安全的。因此，对保留端口的限制具有有限的值；最好根据 Kerberos、防火墙和对特定客户端的导出限制来决定。

另外，在导出 NFS 服务器时请考虑以下最佳实践：

- 导出主目录存在风险，因为某些应用以纯文本或弱加密格式存储密码。您可以通过检查并改进应用程序代码来降低风险。
- 有些用户未对 SSH 密钥设置密码，这再次给主目录带来风险。您可以通过强制使用密码或使用 Kerberos 来降低这些风险。
- 将 NFS 导出仅限制为所需的客户端。在 NFS 服务器上使用 **showmount -e** 命令来检查服务器正在导出什么。不要导出不需要的任何内容。
- 不要允许不必要的用户登录到服务器，以减少攻击风险。您可以定期检查谁可以访问服务器，以及可以访问服务器的什么数据。

其他资源

- 使用红帽身份管理时在 IdM 中使用自动挂载
- **exports(5)** 和 **nfs(5)** 手册页

9.3.2. 保护 NFS 客户端的挂载选项

您可以将以下选项传递给 **mount** 命令，以增加基于 NFS 的客户端的安全性：

nosuid

使用 **nosuid** 选项禁用 **set-user-identifier** 或 **set-group-identifier** 位。这样可防止远程用户通过运行 **setuid** 程序获得更高的特权，并可使用此选项与 **setuid** 选项相反。

noexec

使用 **noexec** 选项禁用客户端上的所有可执行文件。使用此选项可防止用户意外执行位于共享文件系统中的文件。

nodev

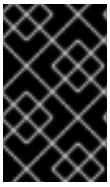
使用 **nodev** 选项防止客户端将设备文件作为硬件设备处理。

resvport

使用 **resvport** 选项将通信限制到保留端口，您可以使用特权源端口与服务器通信。保留的端口是为特权用户和进程保留的，如 **root** 用户。

秒

使用 NFS 服务器上的 **sec** 选项，选择 RPCGSS security 类别来访问挂载点上的文件。有效的安全类别包括 **none**, **sys**, **krb5**, **krb5i**, 和 **krb5p**。



重要

krb5-libs 软件包提供的 MIT Kerberos 库不支持新部署中的数据加密标准(DES)算法。因为安全和兼容性的原因，在 Kerberos 库中默认禁用 DES。出于兼容性的原因，请使用更新、更安全的算法而不是 DES。

其他资源

- 常用的 NFS 挂载选项

9.3.3. 使用防火墙保护 NFS

要保护 NFS 服务器上的防火墙，请仅开放所需的端口。不要将 NFS 连接端口号用于任何其他服务。

前提条件

- 已安装 **nfs-utils** 软件包。
- **firewalld** 软件包已安装并运行。

流程

- 在 NFSv4 上，防火墙必须打开 TCP 端口 **2049**。
- 在 NFSv3 上，使用 **2049** 打开四个额外端口：

1. **rpcbind** 服务动态分配 NFS 端口，这可能会在创建防火墙规则时导致问题。要简化这个过程，使用 **/etc/nfs.conf** 文件指定要使用哪些端口：

   a. 在 **[mountd]** 部分为 **mountd** (**rpc.mountd**) 设置 TCP 和 UDP 端口，格式为 **port=*<value>***。

   b. 在 **[statd]** 部分为 **statd** (**rpc.statd**) 设置 TCP 和 UDP 端口，格式为 **port=*<value>***。

2. 在 **/etc/nfs.conf** 文件中为 NFS 锁定管理器(**nlockmgr**)设置 TCP 和 UDP 端口：

a. 在 **[lockd]** 部分为 **nlockmgr** (**rpc.statd**) 设置 TCP 端口，格式为 **port=value**。或者，也可以使用 **/etc/modprobe.d/lockd.conf** 文件中的 **nlm_tcpport** 选项。

b. 在 **[lockd]** 部分为 **nlockmgr** (**rpc.statd**) 设置 UDP 端口，格式为 **udp-port=value**。或者，您可以使用 **/etc/modprobe.d/lockd.conf** 文件中的 **nlm_udpport** 选项。

验证步骤

- 列出 NFS 服务器中的活跃端口和 RPC 程序：

```
$ rpcinfo -p
```

其他资源

- 使用红帽身份管理时在 IdM 中使用自动挂载
- **exports(5)** 和 **nfs(5)** 手册页

9.4. 保护 FTP 服务

您可以使用文件传输协议(FTP)来通过网络传输文件。因为服务器的所有 FTP 事务（包括用户身份验证）均未加密，因此请确保它被安全地配置。

RHEL 8 提供两个 FTP 服务器：

红帽内容加速器(tux)

具有 FTP 功能的内核空间 Web 服务器。

非常安全的 FTP 守护进程(vsftpd)

FTP 服务的独立、面向安全的实现。

以下安全指南是用于设置 **vsftpd** FTP 服务：

9.4.1. 保护 FTP 的问候横幅

当用户连接到 FTP 服务时，FTP 会显示一个问候标语，它默认包含版本信息。攻击者可以利用此信息来识别系统中的弱点。您可以通过更改默认的横幅来隐藏此信息。

您可以通过编辑 **/etc/banners/ftp.msg** 文件来定义自定义横幅。它可以直接包含一个单行消息，或引用一个单独的文件，其中包含多行消息。

流程

- 要定义单行消息，请在 **/etc/vsftpd/vsftpd.conf** 文件中添加以下选项：

```
ftpd_banner=Hello, all activity on ftp.example.com is logged.
```

- 在单独的文件中定义信息：
  - 创建一个 **.msg** 文件，其中包含横幅消息，如 **/etc/banners/*ftp*.msg** ：

```
######### Hello, all activity on ftp.example.com is logged. #########
```

为了简化多个横幅的管理，请将所有横幅放在 **/etc/banners/** 目录中。

- 将横幅文件的路径添加到 **/etc/vsftpd/vsftpd.conf** 文件中的 **banner_file** 选项中：

```
banner_file=/etc/banners/ftp.msg
```

验证

- 显示修改后的横幅：

```
$ ftp localhost
Trying ::1…
Connected to localhost (::1).
Hello, all activity on ftp.example.com is logged.
```

9.4.2. 防止匿名访问并在 FTP 中上传

默认情况下，安装 **vsftpd** 软件包会为匿名用户创建 **/var/ftp/** 目录和一个目录树，用户对这些目录有只读权限。由于匿名用户可以访问数据，因此请勿将敏感数据存储在这些目录中。

要增加系统的安全性，您可以将 FTP 服务器配置为允许匿名用户将文件上传到特定目录并阻止匿名用户读取数据。在以下流程中，匿名用户可以将文件上传到 **root** 用户拥有的目录中，但不能更改该它。

流程

- 在 **/var/ftp/pub/** 目录中创建只写的目录：

```
# mkdir /var/ftp/pub/upload
# chmod 730 /var/ftp/pub/upload
# ls -ld /var/ftp/pub/upload
drwx-wx---. 2 root ftp 4096 Nov 14 22:57 /var/ftp/pub/upload
```

- 在 **/etc/vsftpd/vsftpd.conf** 文件中添加以下行：

```
anon_upload_enable=YES
anonymous_enable=YES
```

- 可选：如果您的系统启用了 SELinux 并使用强制（enforcing）模式，请启用 SELinux 布尔值属性 **allow_ftpd_anon_write** 和 **allow_ftpd_full_access**。

警告

允许匿名用户在目录中读取和写入，可能会导致服务器成为盗取软件的存储库。

9.4.3. 为 FTP 保护用户帐户

FTP 通过不安全的网络以未加密的方式传输用户名和密码以进行身份验证。您可以通过拒绝系统用户从其用户帐户访问服务器来提高 FTP 安全性。

请根据您的配置执行以下几个步骤。

流程

- 通过在 **/etc/vsftpd/vsftpd.conf** 文件中添加以下行来禁用 **vsftpd** 服务器中的所有用户帐户：

```
local_enable=NO
```

- 要禁用特定帐户或特定帐户组（如 **root** 用户和带有 **sudo** 权限的组）对 FTP 的访问，您可以将用户名添加到 **/etc/pam.d/vsftpd** PAM 配置文件。
- 通过在 **/etc/vsftpd/ftpusers** 文件中添加用户名来禁用用户帐户。

9.4.4. 其他资源

- **ftpd_selinux(8)** 手册页

9.5. 保护 HTTP 服务器

9.5.1. httpd.conf 中的安全增强

您可以通过在 **/etc/httpd/conf/httpd.conf** 文件中配置安全选项来提高 Apache HTTP 服务器的安全性。

在将脚本加入到生产环境前，需要验证它们是否可以正常工作。

确保只有 **root** 用户对包含脚本或通用网关接口(CGI)的任何目录具有写入权限。要将目录所有权改为具有写入权限的 **root**，请输入以下命令：

```
# chown root <directory_name>
# chmod 755 <directory_name>
```

在 **/etc/httpd/conf/httpd.conf** 文件中，您可以配置以下选项：

FollowSymLinks

这个指令默认为启用，并遵循目录中的符号链接。

索引

这个指令被默认启用。禁用这个指令，以防止访问者浏览服务器上的文件。

UserDir

这个指令默认为禁用，因为它可以确认系统中存在用户帐户。要激活用户目录浏览 **/root/** 以外的所有用户目录，使用 **UserDir enabled** 和 **UserDir disabled** root 指令。要将用户添加到禁用帐户列表中，请在 **UserDir disabled** 行中添加以空格分隔的用户列表。

ServerTokens

这个指令控制向客户端发送的服务器响应标头字段。您可以使用以下参数来自定义信息：

ServerTokens Full

提供所有可用信息，如 Web 服务器版本号、服务器操作系统详情、已安装的 Apache 模块，例如：

```
Apache/2.4.37 (Red Hat Enterprise Linux) MyMod/1.2
```

ServerTokens Full-Release

提供与发行版本相关的所有可用信息，例如：

```
Apache/2.4.37 (Red Hat Enterprise Linux) (Release 41.module+el8.5.0+11772+c8e0c271)
```

ServerTokens Prod / ServerTokens ProductOnly

提供 Web 服务器名称，例如：

```
Apache
```

ServerTokens Major

提供 Web 服务器主版本，例如：

```
Apache/2
```

ServerTokens Minor

提供 Web 服务器次版本，例如：

```
Apache/2.4
```

ServerTokens Min / ServerTokens Minimal

提供 Web 服务器最小发行版本，例如：

```
Apache/2.4.37
```

ServerTokens OS

提供 Web 服务器发行版本和操作系统，例如：

```
Apache/2.4.37 (Red Hat Enterprise Linux)
```

使用 **ServerTokens Prod** 选项降低攻击者获取您系统的任何宝贵信息的风险。

重要

不要删除 **IncludesNoExec** 指令。默认情况下，Server Side Includes (SSI) 模块无法执行命令。更改这个设置可让攻击者在系统中输入命令。

删除 httpd 模块

您可以删除 **httpd** 模块来限制 HTTP 服务器的功能。编辑 **/etc/httpd/conf.modules.d/** 或 **/etc/httpd/conf.d/** 目录中的配置文件。例如，要删除代理模块：

```
echo '# All proxy modules disabled' > /etc/httpd/conf.modules.d/00-proxy.conf
```

其他资源

- Apache HTTP 服务器
- 为 Apache HTTP 服务器自定义 SELinux 策略

9.5.2. 保护 Nginx 服务器配置

Nginx 是一个高性能 HTTP 和代理服务器。您可以使用以下配置选项强化 Nginx 配置。

流程

- 要禁用版本字符串，修改 **server_tokens** 配置选项：

```
server_tokens off;
```

这个选项将停止显示其他详细信息，如服务器版本号。此配置仅显示 Nginx 提供的所有请求中的服务器名称，例如：

```
$ curl -sI http://localhost | grep Server
Server: nginx
```

- 添加额外的安全标头，以缓解特定 **/etc/nginx/** conf 文件中的某些已知 Web 应用程序漏洞：
  - 例如，**X-Frame-Options** 标头选项拒绝您的域之外的任何页面来帧由 Nginx 提供的任何内容，缓解了攻击：

```
add_header X-Frame-Options "SAMEORIGIN";
```

  - 例如，**x-content-type** 标头在某些较旧的浏览器中可以防止 MIME-type sniffing:

```
add_header X-Content-Type-Options nosniff;
```

  - 例如，**X-XSS-Protection** 标头启用跨站点脚本过滤(XSS)过滤，这可防止浏览器渲染由 Nginx 中包含的潜在的恶意内容：

```
add_header X-XSS-Protection "1; mode=block";
```

- 您可以限制公开的服务，并限制它们对访问者执行的操作，例如：

```
limit_except GET {
   allow 192.168.1.0/32;
   deny  all;
}
```

该片段将限制对除 **GET** 和 **HEAD** 外的所有方法的访问。

- 您可以禁用 HTTP 方法，例如：

```
# Allow GET, PUT, POST; return "405 Method Not Allowed" for all others.
if ( $request_method !~ ^(GET|PUT|POST)$ ) {
   return 405;
}
```

- 您可以配置 SSL 来保护 Nginx web 服务器提供的数据，请考虑仅通过 HTTPS 提供它。另外，您可以使用 Mozilla SSL 配置生成器生成在 Nginx 服务器中启用 SSL 的安全配置配置文件。生成的配置确保了，所有已知存在安全漏洞的协议（例如, SSLv2 和 SSLv3），加密程序和哈希算法（例如 3DES 和 MD5) 都已禁用。您还可以使用 SSL 服务器测试来验证您的配置是否满足现代安全要求。

其他资源

- Mozilla SSL 配置生成器
- SSL 服务器测试

9.6. 通过限制对经过身份验证的用户的访问来保护 POSTGRESQL

PostgreSQL 是一个对象相关数据库管理系统(DBMS)。在 Red Hat Enterprise Linux 中，PostgreSQL 由 **postgresql-server** 软件包提供。

您可以通过配置客户端身份验证来降低攻击的风险。**pg_hba.conf** 配置文件存储在数据库集群的数据目录中，控制客户端身份验证。按照以下步骤为基于主机的验证配置 PostgreSQL。

流程

1. 安装 PostgreSQL：

```
# yum install postgresql-server
```

2. 使用以下选项之一初始化数据库存储区域：

   a. 使用 **initdb** 工具：

   ```
   $ initdb -D /home/postgresql/db1/
   ```

   带有 **-D** 选项的 **initdb** 命令会创建您指定的目录（如果尚未存在），例如 ***/home/postgresql/db1/***。然后，此目录包含数据库中存储的所有数据以及客户端身份验证配置文件。

   b. 使用 **postgresql-setup** 脚本：

   ```
   $ postgresql-setup --initdb
   ```

   默认情况下，该脚本使用 **/var/lib/pgsql/data/** 目录。此脚本可以帮助具有基本数据库集群管理的系统管理员。

3. 要允许任何经过身份验证的用户使用其用户名访问任何数据库，请在 **pg_hba.conf** 文件中修改以下行：

```
local   all             all                                     trust
```

   当使用创建数据库用户和没有本地用户的层次应用程序时，这可能会造成问题。如果您不想显式控制系统中所有用户名，请从 **pg_hba.conf** 文件中删除 **local** 行条目。

4. 重启数据库以应用更改：

```
# systemctl restart postgresql
```

   上一命令更新数据库，同时还验证配置文件的语法。

9.7. 保护 MEMCACHED 服务

Memcached 是一个开源、高性能分布式内存对象缓存系统。它可以通过降低数据库负载来提高动态 Web 应用程序的性能。

Memcached 是一个内存键值存储，用于任意数据（如字符串和对象）的小块，来自于数据库调用、API 调用或页面渲染的结果。Memcached 允许将内存从未充分利用的区域分配给需要更多内存的应用程序。

2018 年，发现了向公共互联网公开的 Memcached 服务器漏洞 DDoS 扩展攻击。这些攻击利用了使用 UDP 协议进行传输的 Memcached 通信。这个攻击非常有效，因为其具有非常高的放大比率，具有几百字节大小的请求会产生带有几兆字节甚至几百兆字节的响应。

在大多数情况下，**memcached** 服务不需要向公共互联网公开。如果公开，其本身可能会带有安全问题。远程攻击者可能会泄漏或修改存储在 Memcached 中的信息。

9.7.1. 针对 DDoS 强化 Memcached

要降低安全风险，请根据您的配置执行以下步骤。

流程

- 在 LAN 中配置防火墙。如果您的 Memcached 服务器应只在本地网络访问，请不要将外部流量路由到 **memcached** 服务使用的端口。例如，从允许的端口列表中删除默认端口 **11211** ：

```
# firewall-cmd --remove-port=11211/udp
# firewall-cmd --runtime-to-permanent
```

- 如果您在与应用程序相同的机器上使用单一 Memcached 服务器，请设置 **memcached** 来仅侦听 localhost 流量。修改 **/etc/sysconfig/memcached** 文件中的 **OPTIONS** 值：

```
OPTIONS="-l 127.0.0.1,::1"
```

- 启用简单验证和安全层(SASL)身份验证：

  1. 修改或添加 **/etc/sasl2/memcached.conf** 文件：

     ```
     sasldb_path: /path.to/memcached.sasldb
     ```

  2. 在 SASL 数据库中添加帐户：

     ```
     # saslpasswd2 -a memcached -c cacheuser -f /path.to/memcached.sasldb
     ```

  3. 确保 **memcached** 用户和组可以访问数据库：

     ```
     # chown memcached:memcached /path.to/memcached.sasldb
     ```

  4. 通过在 **/etc/sysconfig/memcached** 文件中的 **OPTIONS** 参数中添加 **-S** 值在 Memcached 中启用 SASL 支持：

     ```
     OPTIONS="-S"
     ```

  5. 重启 Memcached 服务器以应用更改：

     ```
     # systemctl restart memcached
     ```

6. 将 SASL 数据库中创建的用户名和密码添加到应用程序的 Memcached 客户端配置中。

- 使用 TLS 加密 Memcached 客户端和服务器间的通信：

1. 通过在 **/etc/sysconfig/memcached** 文件中的 **OPTIONS** 参数中添加 **-Z** 值来启用 Memcached 客户端和服务器间的加密通信：

```
OPTIONS="-Z"
```

2. 使用 **-o ssl_chain_cert** 选项，以 PEM 格式添加证书链文件路径。

3. 使用 **-o ssl_key** 选项添加私钥文件路径。