Red Hat Enterprise Linux 9

配置防火墙和数据包过滤器

管理 firewalld 服务、nftables 框架和 XDP 数据包过滤功能

Last Updated: 2024-06-26

Red Hat Enterprise Linux 9 配置防火墙和数据包过滤器

管理 firewalld 服务、nftables 框架和 XDP 数据包过滤功能

法律通告

摘要

数据包过滤器（如防火墙）使用规则来控制传入、传出和转发网络流量。在 Red Hat Enterprise Linux (RHEL)中，您可以使用 firewalld 服务和 nftables 框架过滤网络流量并构建性能关键防火墙。您还可以使用内核的 Express Data Path (XDP)功能以非常高的速度处理或丢弃网络接口上的网络数据包。

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. 在顶部导航栏中点 **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。包括到文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 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打开的端口的所有流量。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**。

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**:使用 **firewalld** 实用程序进行简单防火墙用例。此工具易于使用，并涵盖了这些场景的典型用例。
- **nftables**:使用 **nftables** 实用程序设置复杂和高性能的防火墙，如为整个网络设置。
- **iptables**:Red Hat Enterprise Linux 上的 **iptables** 工具使用 **nf_tables** 内核 API 而不是 **legacy** 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。

1.2. 防火墙区域

您可以使用 **firewalld** 工具根据您在该网络中与接口和流量的信任级别将网络划分为不同的区域。连接只能是一个区域的一部分，但您可以将该区域用于许多网络连接。

firewalld 在区域方面遵循严格的原则：

1. 流量只进入一个区域。
2. 流量只从一个区域出去。
3. 区域定义了信任级别。
4. 默认情况下，允许内部区域流量（在同一区域中）。
5. 默认情况下，内部区域流量（从区域到区域）被拒绝。

原则 4 和 5 是原则 3 的结果。

可以通过区域选项 **--remove-forward** 配置原则 4 。可以通过添加新策略配置原则 5 。

NetworkManager 通知接口区的 **firewalld**。您可以使用以下工具将区域分配给接口：

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

RHEL web 控制台、**firewall-config** 和 **firewall-cmd** 只能编辑合适的 **NetworkManager** 配置文件。如果您使用 web 控制台、**firewall-cmd** 或 **firewall-config** 更改接口的区域，则请求被转发到 **NetworkManager**，且不会由 **firewalld** 处理。

/usr/lib/firewalld/zones/ 目录存储预定义的区域，且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

- 适用于：任何传入的网络连接都会被拒绝，并报 **IPv4** 的 icmp-host-prohibited 消息和 **IPv6** 的 icmp6-adm-prohibited 消息 。
- 接受：只从系统启动的网络连接。

dmz

- 适用于：DMZ 中可以公开访问，但可以有限访问您的内部网络的计算机。
- 接受：仅所选的传入连接。

drop

适用于：所有传入的网络数据包都会丢失，没有任何通知。

- 接受：仅传出的网络连接。

external

- 适用于：启用了伪装的外部网络，特别是对于路由器。不信任网络上的其他计算机的情况。
- 接受：仅所选的传入连接。

home

- 适用于：您主要信任网络上其他计算机的主环境。
- 接受：仅所选的传入连接。

internal

- 适用于：您主要信任网络上其他计算机的内部网络。
- 接受：仅所选的传入连接。

public

- 适用于：不信任网络上其他计算机的公共区域。
- 接受：仅所选的传入连接。

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工作环境。

- 接受：仅所选的传入连接。

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** 区域。您可以更改默认区域。

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

1.3. 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要采取的规则和操作。通常，策略包含用于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域的概念。每个区域都与一组特定的决定允许的流量的防火墙规则关联。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。

策略与入口区域和出口区域关联。入口区域是流量起源的地方（接收）。出口区域是流量离开的地方(发送)。

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中应用一致的配置。

1.4. 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，以保护系统免受安全威胁。

防火墙规则通常根据各种属性定义某些条件。属性可以是如下：

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

firewalld 工具将防火墙规则组织到区域(如**public**、**internal** 等)和策略中。每个区域都有自己的一组规则，其决定与特定区域关联的网络接口的流量自由度级别。

1.5. 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

1.6. 预定义的 FIREWALLD 服务

firewalld 服务是一组预定义的防火墙规则，其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，因为它会一次实现多个任务。例如，**firewalld** 一次可以执行以下任务：

- 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，在在 **/etc/firewalld/services/** 中没有相应的 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

1.7. 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。

1.7.1. 为特定区域自定义防火墙设置，以增强安全性

您可以通过修改防火墙设置，关联特定的网络接口或与特定防火墙区域的连接来增强网络安全性。通过为区域定义粒度规则和限制，您可以根据您想要的安全级别控制入站和出站流量。

例如，您可以取得以下好处：

- 敏感数据的保护
- 防止未授权访问
- 潜在网络威胁的缓解

先决条件

- **firewalld** 服务在运行。

流程

1. 列出可用的防火墙区域：

```
# 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。要查看所有区域的详情，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，要允许 **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
   ```

   区域的活动是由网络接口的存在或与其配置匹配的源地址范围确定的。默认区域对于未分类的流量处于活动状态，但如果没有流量匹配其规则，则并不是总是处于活动状态。

   b. 为所选区域分配一个网络接口：

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

   为区域分配一个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 连接配置文件，以永久更改防火墙配置。**firewalld** 和 NetworkManager 之间的这种集成确保一致的网络和防火墙设置。

验证

1. 显示您选择的区域的更新设置：

```
# 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）。

1.7.2. 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被保留并准备好使用。

通常，根据 NetworkManager 连接配置文件中的 **connection.zone** 设置，区域被分配给接口。另外，重启后，NetworkManager 管理"激活"这些区域的分配。

先决条件

- **firewalld** 服务在运行。

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。

1.7.3. 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
```

1.7.4. 使用 nmcli 为连接分配区域

您可以使用 **nmcli** 工具将 **firewalld** 区域添加到 **NetworkManager** 连接。

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
```

1.7.5. 在连接配置文件文件中，手动将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对应文件来分配一个 **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 对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，更新对应的文件：

- 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
```

1.7.6. 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。

先决条件

- **firewalld** 服务在运行。

流程

1. 创建一个新区：

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

2. 使新区域可用：

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

命令将最新的更改应用到防火墙配置，而不中断已运行的网络服务。

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
```

1.7.7. 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**:接受除特定规则不允许的所有传入的数据包。
- **REJECT**:拒绝所有传入的数据包，但特定规则允许的数据包除外。当 **firewalld** 拒绝数据包时，源机器会发出有关拒绝的信息。
- **DROP**:除非由特定规则允许，丢弃所有传入数据包。当 **firewalld** 丢弃数据包时，源机器不知道数据包丢弃的信息。
- **default**:与 **REJECT** 的行为类似，但在某些情况下有特殊含义。

先决条件

- **firewalld** 服务在运行。

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

1.8. 使用 FIREWALLD 控制网络流量

firewalld 软件包安装了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，而无需了解协议及它们使用的端口号。

1.8.1. 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。

先决条件

- **firewalld** 服务在运行。

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** 中所有预定义的服务：

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

命令显示默认区域的可用服务的列表。

3. 将服务添加到 **firewalld** 允许的服务的列表中：

```
# 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中。默认情况下，它将这些更改应用到默认区域的配置中。

验证

1. 列出所有永久的防火墙规则：

```
# 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:
```

命令显示完整的带有默认防火墙区域的永久防火墙规则(**公共**)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，命令会返回一个错误，并带有进一步详情：

```
# 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来验证设置。主配置文件为 **/etc/firewalld/firewalld.conf**。特定于区域的配置文件位于 **/etc/firewalld/zones/** 目录中，策略位于 **/etc/firewalld/policies/** 目录中。

1.8.2. 使用 GUI，使用预定义的服务控制流量

您可以使用图形用户界面，使用预定义的服务控制网络流量。防火墙配置应用程序提供命令行工具的可访问和用户友好的替代选择。

先决条件

- **firewall-config** 软件包已安装 。
- **firewalld** 服务在运行。

流程

1. 启用或禁用预定义或自定义服务：
   a. 启动 **firewall-config** 工具，并选择其服务要被配置的网络区域。
   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。

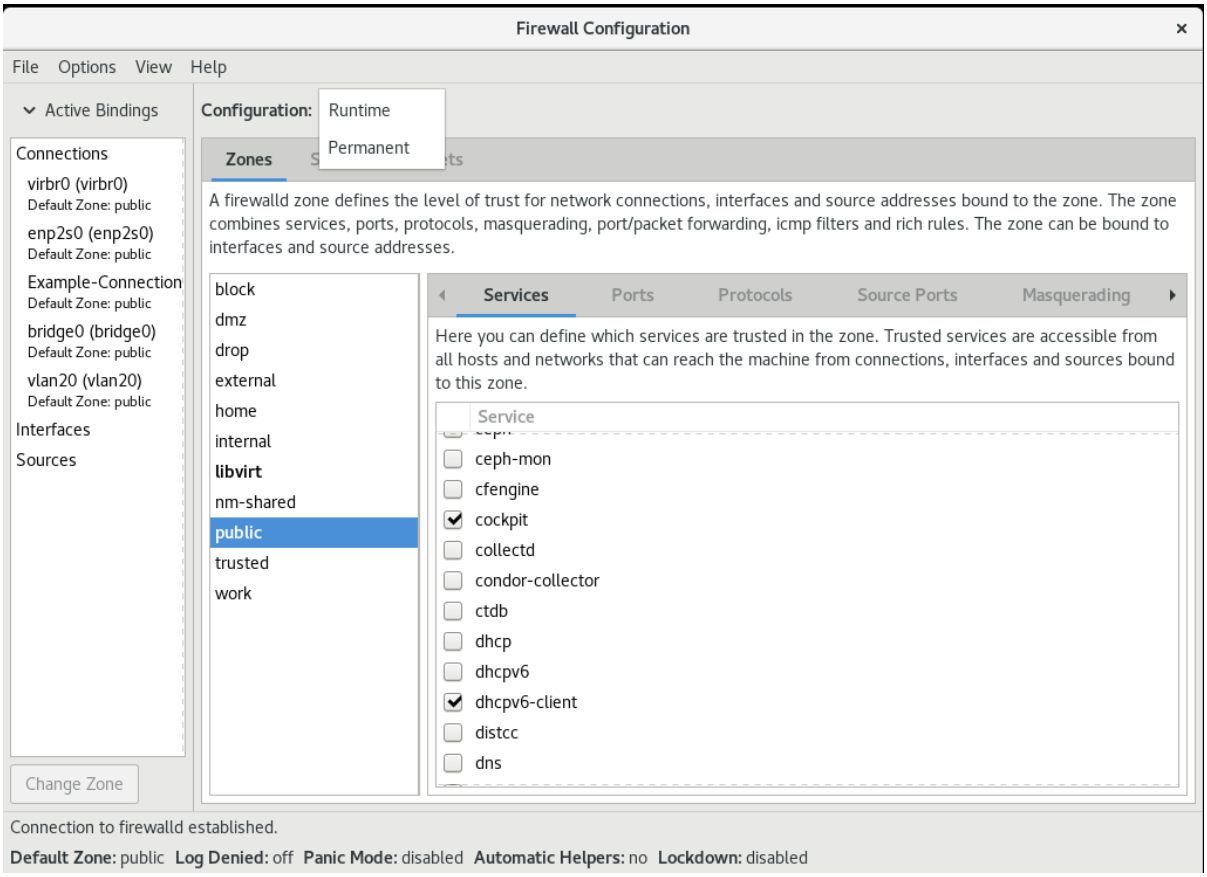
Ports、**Protocols**和 **Source Port** 选项卡支持添加、更改和删除所选服务的端口、协议和源端口。模块选项卡是用于配置 Netfilter 助手模块的。**Destination** 选项卡允许将流量限制到特定的目标地址和互联网协议(**IPv4** 或 **IPv6**)。

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。
  - 您还可以使用命令行，通过输入 **firewall-config** 命令来启动图形防火墙配置工具。
- 查看防火墙配置的列表：



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。

1.8.3. 配置 firewalld 以允许托管安全 Web 服务器

端口是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的逻辑服务。系统服务由侦听端口的守护进程表示，并等待任何进入到此端口的流量。

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。

先决条件

- **firewalld** 服务在运行。

流程

1. 检查当前活跃的防火墙区域：

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

2. 将 HTTPS 服务添加到合适的区域：

```
# 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. 检查端口是否在 **firewalld** 中打开：
   - 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
```

1.8.4. 关闭未使用或不必要的端口，以增强网络安全性

当不再需要开放端口时，您可以使用 **firewalld** 工具关闭它。

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的列表。您将无法看到作为服务打开的任何开放端口。对于这种情况，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，来为传入流量关闭它：

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

这个命令从区域中删除一个端口。如果没有指定区域，它将从默认区域中删除端口。

3. 使新设置具有持久性：

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

如果没有指定区域，这个命令将运行时更改应用到默认区域的永久配置中。

验证

1. 列出活跃区域，并选择要检查的区域：

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

2. 列出所选区域中当前打开的端口，以检查未使用或不必要的端口是否已关闭：

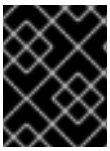
```
# firewall-cmd --zone=<zone_to_inspect> --list-ports
```

1.8.5. 通过 CLI 控制流量

您可以使用 **firewall-cmd** 命令来：

- 禁用网络流量
- 启用网络流量

因此，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
```

1.8.6. 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。

先决条件

- **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。

1.9. 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任何前往系统的数据，或者传递运行 **firewalld** 的主机传递。源通常指的是流量来自的 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域，以允许或禁止该流量可到达的服务。

按源地址匹配优先于按接口名称匹配。当您向区域添加源时，防火墙将对传入的流量优先选择基于源的规则，而不是基于接口的规则。这意味着，如果传入的流量匹配为特定区域指定的源地址，则与该源地址关联的区域将决定如何处理流量，无论它通过哪个接口到达。另一方面，基于接口的规则通常是与特定的基于源的规则不匹配的流量的回退。这些规则应用到源没有明确地与区域关联的流量。这允许您为没有特定的源定义区域的流量定义一个默认行为。

1.9.1. 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
```

1.9.2. 删除源

当您从区域中删除一个源时，起源于源的流量不再被通过为该源指定的规则来定向。相反，流量会退回到与其起源的接口关联的区域的规则和设置，或进入默认区域。

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
```

1.9.3. 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>
```

1.9.4. 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

1.10. 在区域间过滤转发的流量

firewalld 使您能够控制不同的 **firewalld** 区域之间的网络数据流。通过定义规则和策略，当流量在这些区域之间移动时，您可以管理它们是如何被允许或阻止的。

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，来允许访问本地托管的虚拟机，以连接主机。

1.10.1. 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、端口和富规则）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。

1.10.2. 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：

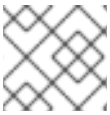
```
# firewall-cmd --permanent --policy mypolicy --set-priority -500
```

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。

低数字优先级值具有更高的优先级，被首先应用。

1.10.3. 使用策略对象过滤本地托管的容器和物理连接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 工具所需的选择性服务。

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，并在 Podman 上只允许必要的服务：

```
# 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 意味着您从 Podman 过滤到其他区域。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
  ...
```

1.10.4. 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
```

1.10.5. 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。

先决条件

- 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，并从本地主机传出的数据包。**--add-egress-zone=ANY** 选项是指移到任何区域的流量。

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 TCP 流量从路由器 IP 地址(192.0.2.1)上的端口 443 转发到 Web 服务器 IP 地址(192.51.100.20)上的端口 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被保留。

- 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 命令对于应用实时更改很有用，但配置在系统重启后不能保持不变。

验证

1. 连接到路由器的 IP 地址以及您已转发到 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，是否包含您需要的设置，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)** 手册页
- 使用 /etc/sysctl.d/ 中的配置文件调整内核参数

1.11. 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

1.11.1. 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，则您可以在路由器上设置 DNAT 规则，将传入的流量重定向到此服务器。

重定向

此类型是 DNAT 的一个特殊情况，其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。

1.11.2. 配置 IP 地址伪装

您可以在系统上启用 IP 伪装。在访问互联网时，IP 伪装会隐藏网关后面的单个机器。

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。

1.11.3. 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和端口。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。

先决条件

- **firewalld** 服务在运行。

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
```

之前的命令使用以下设置定义一个 DNAT 规则：

- **--zone=public** - 用来配置 DNAT 规则的防火墙区域。您可以将其调整为您需要的任何区域。
- **--add-forward-port** - 指示您要添加一个端口转发规则的选项。
- **port=80** - 外部目标端口。
- **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** - 内部服务器的目标端口。
- **--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域的 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

1.11.4. 重定向来自非标准端口的流量，以便使 Web 服务在标准端口上可访问

您可以使用重定向机制使内部运行在非标准端口上的 Web 服务可以访问，而无需用户在 URL 中指定端口。因此，URL 更简单，提供更好的浏览体验，而非标准端口仍然在内部使用或用于特定的要求。

先决条件

- **firewalld** 服务在运行。

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
   ```

   之前的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域。您可以将其调整为您需要的任何区域。
   - **--add-forward-port=port=*<non_standard_port>*** - 指示您要使用最初接收传入流量的源端口添加端口转发（重定向）规则的选项。
   - **proto=tcp** - 指示您重定向 TCP 流量的协议。
   - **toport=*<standard_port>*** - 目标端口，在源端口上收到后，传入的流量应被重定向到的端口。
   - **--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域的重定向规则：

```
# 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

1.12. 管理 ICMP 请求

Internet 控制消息协议 (ICMP)是一种支持协议，被各种网络设备用于测试、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，以及将此类信息滥用于各种欺诈活动。因此，**firewalld** 允许控制 **ICMP** 请求来保护您的网络信息。

1.12.1. 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哪些 ICMP 类型和代码到达您的系统。ICMP 类型和代码是 ICMP 消息的特定类别和子类别。

例如，ICMP 过滤在以下方面有帮助：

- 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 - 仅允许必要的 ICMP 类型，以优化网络性能，并防止大量的 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，以进行网络故障排除，并阻止表示潜在安全风险的 ICMP 类型。

先决条件

- **firewalld** 服务在运行。

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
     ```

     命令删除对 echo requests ICMP 类型的任何现有的阻止规则。

   - 阻止 ICMP 类型：

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

     命令确保重定向消息 ICMP 类型被防火墙阻止。

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

1.13. 使用 FIREWALLD 设置和控制 IP 集

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

例如，当您需要时，IP 集合在以下场景中很有价值：

- 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 集合。

1.13.1. 配置动态更新，以允许 IP 集合列表

您可以进行几乎实时更新，来灵活地允许 IP 集合中的特定 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，如安全威胁的检测或网络行为的更改。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。

先决条件

- **firewalld** 服务在运行。

流程

1. 创建一个带有有意义名称的 IP 集合：

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

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

2. 向 IP 集合添加动态更新：

```
# 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 集合，防火墙允许其传输网络流量。

3. 创建一个引用之前创建的 IP 集合的防火墙规则：

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

   如果没有此规则，IP 集合对网络流量没有任何影响。默认防火墙策略将优先。

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 集合：

```
# 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:
  ...
```

命令行输出的 **sources** 部分对允许或拒绝哪些流量源（主机名、接口、IP 集、子网等）访问特定的防火墙区域提供见解。在这种情况下，包含在 **allowlist** IP 集合中的 IP 地址被允许通过防火墙为 **public** 区域传输流量。

3. 探索 IP 集合的内容：

```
# 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

- 使用脚本或安全工具来获取您的威胁情报源，并以自动化方式相应地更新 **allowlist**。

其他资源

- **firewall-cmd (1)** 手册页

1.14. 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升序排序。当更多规则有相同的 **priority** 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。

1.14.1. 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，较低的数字值有较高的优先级。

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。

1.14.2. 设置丰富的规则的优先级

以下是如何创建一条富规则的示例，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
  }
}
```

1.15. 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。

1.15.1. 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. 要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
```

1.15.2. 锁定 allowlist 配置文件的概述

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/** 目录则符号链接到 **/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。

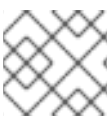
命令的 name 属性末尾的 * 表示所有以这个字符串开头的命令都匹配。如果没有 *，则包括参数的绝对命令必须匹配。

1.16. 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。

1.16.1. 区内转发与默认目标设置为 ACCEPT 的区域之间的区别

启用区域内转发后，单个 **firewalld** 区域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，则流量保留在同一区域内。



注意

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

firewalld 使用不同的区域来管理传入的流量和传出的流量。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，当区域的目标设置为 **default** 时，转发的流量通常默认丢弃。

要控制流量如何在区域内不同接口或源之间被转发，请确保您理解并相应地配置了区域的目标。

1.16.2. 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。此功能带来以下好处：

- 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
- 被网络中多个设备或用户访问和使用的共享资源（如打印机、数据库、网络连接的存储等）
- 高效的内部网络（如平稳的通信、降低的延迟、资源可访问性等）

您可以为单个 **firewalld** 区域启用此功能。

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. 确保要启用区域内转发之间的接口只分配给 **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. 登录到与启用了区域转发的主机的 **enp1s0** 接口位于同一网络上的主机。
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. 输入一些内容并按 **Enter**。验证文本是否被发送回来。

其他资源

- **firewalld.zones(5)** 手册页

1.17. 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。

1.17.1. 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

1.17.2. 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。

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。

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，但会保护有效的配置。

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

1.17.3. 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。

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。

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，但会保护有效的配置。

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

1.17.4. 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，您可以配置默认区域，以允许 HTTPS 服务的传入流量。

在 Ansible 控制节点上执行此步骤。

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。

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，但会保护有效的配置。

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

1.17.5. 使用 firewall RHEL 系统角色配置 firewalld DMZ 区

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。

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。

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，但会保护有效的配置。

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 2 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：

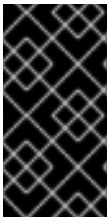
- 内置查找表而不是线性处理
- **IPv4** 和 **IPv6** 协议的单一框架
- 规则会以一个整体被应用，而不是分为抓取、更新和存储完整的规则集的步骤
- 支持在规则集(**nftrace**)和监控追踪事件（**nft**）中调试和追踪
- 更加一致和压缩的语法，没有特定协议的扩展
- 用于第三方应用程序的 Netlink API

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。

2.1. 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**。



重要

ipset 和 **iptables-nft** 软件包已在 Red Hat Enterprise Linux 9 中弃用。这包括 **nft-variants**（如 **iptables**、**ip6tables**、**arptables** 和 **ebtables** 工具）的弃用。如果您使用其中任何一个工具，例如，因为您从早期的 RHEL 版本升级，红帽建议迁移到 **nftables** 软件包提供的 **nft** 命令行工具。

2.1.1. 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**:使用 **firewalld** 实用程序进行简单防火墙用例。此工具易于使用，并涵盖了这些场景的典型用例。
- **nftables**:使用 **nftables** 实用程序设置复杂和高性能的防火墙，如为整个网络设置。
- **iptables**:Red Hat Enterprise Linux 上的 **iptables** 工具使用 **nf_tables** 内核 API 而不是 **legacy** 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。

2.1.2. 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**。

先决条件

- 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

2.1.3. 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。

先决条件

- 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**

2.1.4. 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
```

2.2. 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

当安装 **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。

2.2.1. 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
```

2.2.2. 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 **-f** 参数，**nft** 实用程序不会读取脚本并显示：**Error: syntax error, unexpected newline, expecting string**。

     ii. 可选：将脚本的所有者设置为 **root** ：

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

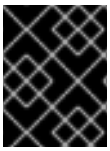
     iii. 使脚本可以被其所有者执行：

```
# 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

2.2.3. 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
...
```

2.2.4. 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

2.2.5. 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。

如果您只指定文件名，而没有绝对路径或相对路径，则 **nftables** 包括默认搜索路径中的文件，该路径被设为 Red Hat Enterprise Linux 上的 **/etc**。

例 2.1. 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 2.2. 包含目录中的所有 *.nft 文件

要包括所有存储在 **/etc/nftables/rulesets/** 目录中、以 ***.nft** 结尾的文件：

```
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

2.2.6. 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。

先决条件

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

流程

1. 编辑 **/etc/sysconfig/nftables.conf** 文件。
   - 如果您使用 **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** 服务以在不重启系统的情况下加载防火墙规则：

   ```
   # systemctl start nftables
   ```

3. 启用 **nftables** 服务。

```
# systemctl enable nftables
```

其他资源

- 支持的 nftables 脚本格式

2.3. 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。

2.3.1. 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。地址系列定义此表处理的数据包类型。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 ingress 的数据包。

如果要添加表，所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>
```

2.3.2. nftables 链的基础知识

表由链组成，链又是规则的容器。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** ：您可以将常规链用作 **jump** 目标来更好地组织规则。

如果要向表中添加基本链，所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，请将 \ 转义字符放在分号前面。

这两个示例都创建 **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钩子的概述：

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

链优先级
priority 参数指定数据包遍历具有相同 hook 值的链的顺序。您可以将此参数设为整数值，或使用标准优先级名称。

以下列表是标准优先级名称及其数字值的一个概述，以及您可以使用它们的哪个地址系列和钩子：

文本值	数字值	地址系列	钩子
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

文本值	数字值	地址系列	钩子
srcnat	**100**	**ip,ip6,inet**	**postrouting**
	300	**bridge**	**postrouting**
out	**100**	**bridge**	**output**

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

- **accept** （默认）
- **drop**

2.3.3. nftables 规则的基础知识

规则定义对通过包含此规则的链的数据包执行的操作。如果规则还包含匹配表达式，则 **nftables** 仅在所有之前的表达式都应用时才执行操作。

如果要在链中添加一条规则，所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 命令在链的末尾附加新规则。如果要在链的开头添加一条规则，请使用 **nft insert** 命令而不是 **nft add**。

2.3.4. 使用 nft 命令管理表、链和规则

要在命令行上或 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，且没有被优化。此流程只演示了如何使用 **nft** 命令来管理表、链和规则。

流程

1. 创建一个带有 **inet** 地址系列的名为 **nftables_svc** 的表，以便表可以处理 IPv4 和 IPv6 数据包：

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

2. 将处理传入网络流量的、名为 **INPUT** 的基本链添加到 **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** 链添加规则。例如，允许端口 22 和 443 上的传入 TCP 流量，并作为 **INPUT** 链的最后一条规则，拒绝其他传入的流量，并伴有互联网控制消息协议(ICMP)端口无法访问的消息：

```
# 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，则**nft** 会将按与运行命令相同的顺序将规则添加到链。

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. 在句柄为 3 的现有规则前面插入一条规则。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. 在句柄为 3 的现有规则后面附加一条规则。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. 再次显示带有 handle 的规则集。验证是否后添加的规则已添加到指定位置：

```
# 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 handle 为 6 的规则：

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

要删除规则，您必须指定 handle。

9. 显示规则集，并验证删除的规则是否不再存在：

```
# 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。

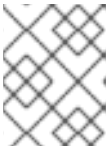
13. 显示规则集，并验证 **INPUT** 链是否已被删除：

```
# 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，而不是在单独的命令中手动删除所有规则、链和表。

其他资源

nft(8) 手册页

2.4. 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**)。

2.4.1. 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，则您可以在路由器上设置 DNAT 规则，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。

2.4.2. 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。

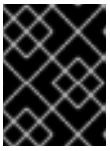
流程

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
```

2.4.3. 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。

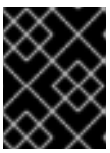
流程

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

2.4.4. 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无法从互联网直接访问的主机。

例如，有了 DNAT，路由器将发送给端口 **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。

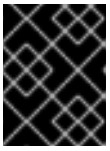
流程

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** 链添加一条规则，将传入的流量从路由器 **ens3** 接口上的端口 **80** 和 **443** 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

2.4.5. 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

2.4.6. 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 机制具有以下功能：

- 使用连接跟踪来绕过经典的数据包转发路径。
- 避免通过绕过经典数据包处理来重新查看路由表。
- 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** 表：

```
# nft add table inet <example-table>
```

2. 添加一个带有 **ingress** 钩子和 **filter** 作为为优先级类型的 **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. 添加一个具有 **established** 连接跟踪状态的规则，以卸载 **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

2.5. 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。

2.5.1. 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。

先决条件

- **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
  }
}
```

2.5.2. 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**)。

先决条件

- **example_chain** 链和 **example_table** 表存在。

流程

1. 创建一个空集。以下示例为 IPv4 地址创建了一个集合：
   - 要创建可存储多个独立 IPv4 地址的集合：

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

   - 要创建可存储 IPv4 地址范围的集合：

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

重要

要防止 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**。

2.5.3. 其他资源

- **nft(8)** 手册页中的 **Sets** 部分

2.6. 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。

2.6.1. 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。

2.6.2. 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
  }
}
```

2.6.3. 其他资源

- **nft(8)** 手册页中的 **Maps** 部分

2.7. 例如：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，该脚本可保护内部 LAN 中的网络客户端，以及 DMZ 中的 Web 服务器，不受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，描述了一个具有特定要求的场景。

防火墙脚本高度依赖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。

2.7.1. 网络条件

本例中的网络具有以下条件：

- 路由器连接到以下网络：
  - 通过接口 **enp1s0** 的互联网
  - 内部 LAN 通过接口 **enp7s0**
  - DMZ 通过 **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，从 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** 中的公共 IP 地址。
- DMZ 中的 Web 服务器使用 IP 地址 **198.51.100.5** 和 **2001:db8:b::5**。
- 路由器为 LAN 和 DMZ 中的主机充当缓存 DNS 服务器。

2.7.2. 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：
  - 查询运行在路由器上的缓存 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 中的路由器以及每个服务器。
- DMZ 中的 Web 服务器必须能够：
  - 查询运行在路由器上的缓存 DNS 服务器。
  - 访问互联网上的 HTTPS 服务器来下载更新。
- 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。

2.7.3. 配置将丢弃的数据包记录到文件

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。

先决条件

- **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** 服务会将丢弃的数据包记录到 **/var/log/nftables.log** 文件，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** 在下一次轮转操作过程中删除超过 30 天的轮转日志。

其他资源

- **rsyslog.conf(5)** 手册页
- **logrotate(8)** 手册页

2.7.4. 编写并激活 nftables 脚本

本例是运行在 RHEL 路由器上的一个 **nftables** 防火墙脚本，其保护内部 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 对防火墙脚本的安全要求 。

警告

此 **nftables** 防火墙脚本仅用于演示目的。不要在未经调整为适合您环境和安全要求的情况下使用它。

先决条件

- 网络已配置，如 网络条件 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 SSH 从 DMZ 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 ...
     ```

2.8. 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。

2.8.1. 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
   ```

2.8.2. 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私有 IP 地址主机上的服务。

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。

先决条件

- 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
```

2.9. 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。

2.9.1. 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。

先决条件

- **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。

2.9.2. 在一分钟内尝试超过十个进入的 TCP 连接的 IP 地址

您可以临时阻止在一分钟内建立十个 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. 在 **filter** 表中添加 **input** 链：

```
# nft add chain ip filter input { type filter hook input priority 0 \; }
```

3. 在 **filter** 表中添加名为 **denylist** 的集合：

```
# nft add set ip filter denylist { type ipv4_addr \; flags dynamic, timeout \; timeout 5m \; }
```

这个命令为 IPv4 地址创建动态设置。**timeout 5m** 参数定义了 **nftables** 在五分钟后自动删除条目，以防止集合被陈旧的条目填满。

4. 添加一条规则，该规则会将在一分钟内试图建立十多个新 TCP 连接的主机的源 IP 地址添加到 **denylist** 集合中：

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked add @denylist { ip saddr limit rate over 10/minute } drop
```

其他资源

- 在 nftables 中使用命名集

2.10. 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。

2.10.1. 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。

- 有关向现有规则添加计数器的流程的更多信息，请参阅 向现有规则添加计数器。

先决条件

- 您要添加该规则的链已存在。

流程

1. 向链中添加一条带有 **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
  }
}
```

2.10.2. 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。

- 有关使用计数器添加新规则的流程的更多信息，请参阅 使用计数器创建规则。

先决条件

- 您要添加计数器的规则已存在。

流程

1. 在链中显示规则及其句柄：

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
```

```
    tcp dport ssh accept # handle 4
  }
}
```

2. 通过使用 **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
  }
}
```

2.10.3. 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，用它来监控匹配此规则的数据包。

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。

2.11. 备份和恢复 NFTABLES 规则集

您可以将 **nftables** 规则备份到文件，并稍后恢复它们。此外，管理员可以使用带有规则的文件，来将规则传给其他服务器。

2.11.1. 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
    ```

2.11.2. 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
    ```

2.12. 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:nftables 是将来的选择

第 3 章 使用 XDP-FILTER 进行高性能流量过滤以防止 DDOS 攻击

与 **nftables** 等数据包过滤器相比，Express Data Path(XDP)在网络接口处处理和丢弃网络数据包。因此，XDP 在到达防火墙或其他应用程序前决定了软件包的下一步。因此，XDP 过滤器需要较少的资源，处理网络数据包的速度要比传统数据包过滤器快得多，从而防止分布式拒绝服务(DDoS)攻击。例如，在测试过程中，红帽在单核上每秒丢弃了 2,600万个网络数据包，这比同一硬件上的 **nftables** 的丢弃率要高得多。

xdp-filter 工具使用 XDP 允许或丢弃传入的网络数据包。您可以创建规则来过滤与特定对象或特定命令的流量：

- IP 地址
- MAC 地址
- 端口

请注意，即使 **xdp-filter** 具有非常高的数据包处理率，但它不具有与 **nftables** 相同的功能。将 **xdp-filter** 视为一个概念性工具，来演示使用 XDP 的数据包过滤。另外，您可以使用工具代码来更好地了解如何编写您自己的 XDP 应用程序。

重要

在 AMD 和 Intel 64 位以外的构架上，**xdp-filter** 工具仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

3.1. 丢弃匹配 XDP-FILTER 规则的网络数据包

您可以使用 **xdp-filter** 丢弃网络数据包：

- 到特定目的地端口
- 从一个指定的 IP 地址
- 从一个指定的 MAC 地址

xdp-filter 的 **allow** 策略定义所有流量都被允许，过滤器只丢弃与特定规则匹配的网络数据包。例如，如果您知道要丢弃的数据包的源 IP 地址，请使用这个方法。

先决条件

- **xdp-tools** 软件包已安装。
- 支持 XDP 程序的网络驱动程序。

流程

1. 加载 **xdp-filter** 来处理特定接口上传入的数据包，如 **enp1s0** ：

```
# xdp-filter load enp1s0
```

默认情况下，**xdp-filter** 使用 **allow** 策略，并且工具只丢弃与任何规则匹配的流量。

（可选）使用 **-f *feature*** 选项仅启用特定功能，如 **tcp**、**ipv4** 或 **ethernet**。仅加载所需的功能而不是全部功能，从而提高数据包处理的速度。要启用多个功能，使用逗号分隔它们。

如果该命令出错，则网络驱动程序不支持 XDP 程序。

2. 添加规则来丢弃与它们匹配的数据包。例如：

   - 要将传入数据包放到端口 **22**，请输入：

     ```
     # xdp-filter port 22
     ```

     这个命令添加一个匹配 TCP 和 UDP 流量的规则。要只匹配特定的协议，请使用 **-p *protocol*** 选项。

   - 要丢弃来自 **192.0.2.1** 的传入数据包，请输入：

     ```
     # xdp-filter ip 192.0.2.1 -m src
     ```

     请注意，**xdp-filter** 不支持 IP 范围。

   - 要丢弃来自 MAC 地址 **00:53:00:AA:07:BE** 的传入数据包，请输入：

     ```
     # xdp-filter ether 00:53:00:AA:07:BE -m src
     ```

验证

- 使用以下命令显示丢弃和允许的数据包统计信息：

  ```
  # xdp-filter status
  ```

其他资源

- **xdp-filter(8)** 手册页
- 如果您是开发人员，并对 **xdp-filter** 代码感兴趣，请从红帽客户门户网站下载并安装相应的源 RPM (SRPM)。

3.2. 丢弃所有与 XDP-FILTER 规则匹配的网络数据包

您可以使用 **xdp-filter** 来只允许网络数据包：

- 来自和到一个特定目的地端口
- 来自和到一个特定 IP 地址
- 来自和到特定的 MAC 地址

要做到这一点，请使用 **xdp-filter** 的 **deny** 策略，其定义该过滤器会丢弃所有的网络数据包，与特定规则匹配的除外。例如，如果您不知道要丢弃的数据包的源 IP 地址，请使用这个方法。

警告

如果您在接口上加载 **xdp-filter** 时将默认策略设为 **deny**，则内核会立即丢弃来自这个接口的所有数据包，直到您创建允许某些流量的规则。要避免从系统中锁定，在本地输入命令或者通过不同的网络接口连接到主机。

先决条件

- **xdp-tools** 软件包已安装。
- 您登录到本地主机，或使用您不计划过滤流量的网络接口。
- 支持 XDP 程序的网络驱动程序。

流程

1. 加载 **xdp-filter** 来处理特定接口上的数据包，如 **enp1s0** ：

```
# xdp-filter load enp1s0 -p deny
```

（可选）使用 **-f *feature*** 选项仅启用特定功能，如 **tcp**、**ipv4** 或 **ethernet**。仅加载所需的功能而不是全部功能，从而提高数据包处理的速度。要启用多个功能，使用逗号分隔它们。

如果该命令出错，则网络驱动程序不支持 XDP 程序。

2. 添加规则以允许匹配它们的数据包。例如：

- 要允许数据包发送端口 **22** ，请输入：

```
# xdp-filter port 22
```

这个命令添加一个匹配 TCP 和 UDP 流量的规则。要只匹配特定的协议，请将 **-p *protocol*** 选项传给命令。

- 要允许数据包发送到 **192.0.2.1** ，请输入：

```
# xdp-filter ip 192.0.2.1
```

请注意，**xdp-filter** 不支持 IP 范围。

- 要允许数据包发送到 MAC 地址 **00:53:00:AA:07:BE** ，请输入：

```
# xdp-filter ether 00:53:00:AA:07:BE
```

重要

xdp-filter 工具不支持有状态数据包检查。这要求您不使用 **-m *mode*** 选项设置模式，或者您添加显式规则来允许机器接收传入流量来作为对传出流量的答复。

验证

- 使用以下命令显示丢弃和允许的数据包统计信息：

```
# xdp-filter status
```

其他资源

- **xdp-filter(8)** 手册页。
- 如果您是开发人员，并且您对 **xdp-filter** 的代码感兴趣，请从红帽客户门户网站下载并安装相应的源 RPM (SRPM)。