Red Hat Enterprise Linux 9

配置防火墙和数据包过滤器

指南

Last Updated: 2022-07-31

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

指南

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

法律通告

摘要

目录

让开源更具包容性

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请让我们了解如何改进文档。

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

第 1 章 使用和配置 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. FIREWALLD入门

本节提供有关 **firewalld** 的信息。

1.1.1. 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** 工具来设置复杂和性能关键的防火墙，如整个网络。
- **iptables**:Red Hat Enterprise Linux 上的 **iptables** 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

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

1.1.2. Zones

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

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

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

后三个只能编辑适当的 **NetworkManager** 配置文件。如果您使用 web 控制台、 **firewall-cmd** 或 **firewall-config** 修改了接口区，那么请求会被转发到 **NetworkManager**，并且不会由 **firewalld** 来处理。

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

block

任何传入的网络连接都会被拒绝，对于 **IPv4** 会显示 icmp-host-prohibited 消息，对于 **IPv6** 会显示 icmp6-adm-prohibited 消息。只有从系统启动的网络连接才能进行。

dmz

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

drop

所有传入的网络数据包都会丢失，没有任何通知。只有外发网络连接也是可行的。

external

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

home

用于家用，因为您可以信任其他计算机。只接受所选的入站连接。

internal

当您主要信任网络中的其他计算机时，供内部网络使用。只接受所选的入站连接。

public

可用于您不信任网络中其他计算机的公共区域。只接受所选的入站连接。

trusted

所有网络连接都被接受。

work

可用于您主要信任网络中其他计算机的工作。只接受所选的入站连接。

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

注意

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

其他资源

- **firewalld.zone(5)** 手册页。

1.1.3. 预定义的服务

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

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

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

或者，您可以编辑 **/etc/firewalld/services/** 目录中的 XML 文件。如果用户未添加或更改服务，则在 **/etc/firewalld/services/** 中没有相应的 XML 文件。如果要添加或更改服务，**/usr/lib/firewalld/services/** 目录中的文件可作用作模板。

其他资源

- **firewalld.service(5)**手册页

1.1.4. 启动 firewalld

步骤

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

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

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

```
# systemctl enable firewalld
```

1.1.5. 停止 firewalld

步骤

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

```
# systemctl stop firewalld
```

2. 要防止 **firewalld** 在系统启动时自动启动：

```
# systemctl disable firewalld
```

3. 要确保访问 **firewalld D-Bus**接口时未启动firewalld，并且其他服务需要 **firewalld** 时也未启动 firewalld ：

```
# systemctl mask firewalld
```

1.1.6. 验证永久 firewalld 配置

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

先决条件

- **firewalld** 服务在运行。

步骤

1. 验证 **firewalld** 服务的永久配置：

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

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

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

1.2. 查看 FIREWALLD的当前状态和设置

本节涵盖了有关查看 **firewalld** 的当前状态、允许的服务以及当前设置的信息。

1.2.1. 查看 firewalld 的当前状态

默认情况下，防火墙服务 **firewalld** 会在系统上安装。使用 **firewalld** CLI 接口来检查该服务是否正在运行。

步骤

1. 查看服务的状态：

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

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

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

1.2.2. 使用 GUI 查看允许的服务

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

您可以使用命令行启动图形防火墙配置工具。

先决条件

- 已安装 **firewall-config** 软件包。

步骤

- 使用命令行启动图形防火墙配置工具：

```
$ firewall-config
```

防火墙配置 窗口打开。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。

1.2.3. 使用 CLI 查看 firewalld 设置

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

Firewalld 使用区来管理流量。如果没有用 **--zone** 选项来指定区，该命令将在分配给活跃网络接口和连接的默认区中有效。

步骤

- 要列出默认区的所有相关信息：

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

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

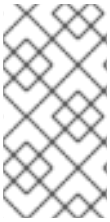
```
# firewall-cmd --list-all --zone=home
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh mdns samba-client dhcpv6-client
... [trimmed for clarity]
```

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

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

- 查看当前区中允许哪些服务：

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



注意

使用 CLI 工具列出某个子部分的设置有时会比较困难。例如，您允许 **SSH** 服务，**firewalld** 为该服务开放必要的端口(22)。之后，如果您列出允许的服务，列表将显示 **SSH** 服务，但如果列出开放的端口，则不会显示任何内容。因此，建议您使用 **--list-all** 选项来确保您收到完整的信息。

1.3. 使用 FIREWALLD 控制网络流量

本节涵盖了使用 **firewalld** 来控制网络流量的信息。

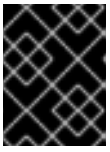
1.3.1. 使用 CLI 禁用紧急事件的所有流量

在紧急情况下，如系统攻击，可以禁用所有网络流量并关闭攻击者。

流程

1. 要立即禁用网络流量，请切换 panic 模式：

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



重要

启用 panic 模式可停止所有网络流量。因此，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。

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
```

1.3.2. 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向 **firewalld** 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。

流程

1. 检查该服务是否还未被允许：

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

2. 列出所有预定义的服务：

```
# firewall-cmd --get-services
```

```
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc
bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp dhcpv6
dhcpv6-client dns docker-registry ...
[trimmed for clarity]
```

3. 在允许的服务中添加服务：

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

4. 使新设置持久：

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

1.3.3. 通过 GUI，使用预定义服务控制流量

这个步骤描述了如何使用图形用户界面控制预定义服务的网络流量。

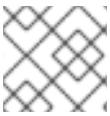
先决条件

- 已安装 **firewall-config** 软件包

步骤

1. 启用或禁用预定义或自定义服务：
   a. 启动 **firewall-config** 工具并选择要配置的服务的网络区。
   b. 选择 **Zones** 选项卡，然后选下面的 **Services** 选项卡。
   c. 选择需要在所选区中信任的服务类型的复选框，或者取消选择要阻断的服务复选框。
2. 编辑服务：
   a. 启动 **firewall-config** 工具。
   b. 从标为 **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** 选项卡可为所选的服务启用、更改和删除端口、协议和源端口。模块标签是用来配置 **Netfilter** helper 模块。**Destination** 选项卡允许将流量限制到特定的目标地址和Internet 协议(**IPv4** 或 **IPv6**)。

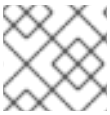


注意

在**Runtime** 模式下无法更改服务设置。

1.3.4. 添加新服务

可以使用图形化的 firewall-config 工具、**firewall-cmd** 和 **firewall-offline-cmd** 来添加和删除服务。或者，您可以编辑 **/etc/firewalld/services/** 中的 XML 文件。如果用户未添加或更改服务，则在 **/etc/firewalld/services/** 中没有相应的 XML 文件。如果要添加或更改服务，则文件 **/usr/lib/firewalld/services/** 可用作模板。



注意

服务名称必须是字母数字，此外只能包含 _ （下划线）和 - （短划线）字符。

步骤

要在终端中添加新服务，请使用 **firewall-cmd** 或在 **firewalld** 未激活的情况下，使用**firewall-offline-cmd** 。

1. 运行以下命令以添加新和空服务：

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

2. 要使用本地文件添加新服务，请使用以下命令：

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

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

3. 更改服务设置后，服务的更新副本放在 **/etc/firewalld/services/** 中。
   作为 **root** 用户，您可以输入以下命令来手动复制服务：

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

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

1.3.5. 使用 GUI 打开端口

要允许流量通过防火墙到达某个端口，您可以在 GUI 中打开端口。

先决条件

- 已安装 **firewall-config** 软件包

步骤

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

1.3.6. 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

- 已安装 **firewall-config** 软件包

步骤

1. 启动 **firewall-config** 工具并选择要更改的网络区。
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。

1.3.7. 使用 GUI 打开源端口

要允许来自某个端口的流量通过防火墙，您可以使用 GUI。

先决条件

- 已安装 **firewall-config** 软件包

步骤

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

1.4. 使用 CLI 控制端口

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

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

1.4.1. 打开端口

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

流程

要获得当前区的打开端口列表：

1. 列出所有允许的端口：

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

2. 在允许的端口中添加一个端口，以便为入站流量打开这个端口：

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

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

3. 使新设置具有持久性：

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

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

1.4.2. 关闭端口

当打开的端口不再需要时，在 **firewalld** 中关闭此端口。强烈建议您尽快关闭所有不必要的端口，因为端口处于打开状态会存在安全隐患。

流程

要关闭某个端口，请将其从允许的端口列表中删除：

1. 列出所有允许的端口：

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

警告

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

2. 从允许的端口中删除端口，以便对传入的流量关闭：

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

3. 使新设置具有持久性：

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

1.5. 使用系统角色配置端口

您可以使用 Red Hat Enterprise Linux(RHEL) **firewalld** 系统角色为传入的流量打开或关闭本地防火墙中的端口，并在重新引导后保持新配置。这个示例描述了如何配置 default 区以允许 **HTTPS** 服务的传入流量。

在 Ansible 控制节点上运行此步骤。

先决条件

- 可以访问一个或多个 *受管节点*，它们是您要使用 **firewalld** 系统角色来配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
- **ansible-core** 和 **rhel-system-roles** 软件包在控制节点上安装。
- 如果您在运行 playbook 时使用了与 **root** 不同的远程用户，则此用户在受管节点上具有合适的 **sudo** 权限。
- 主机使用 NetworkManager 配置网络。

流程

1. 如果您要在其上执行 playbook 中指令的主机还没有被列入清单，请将此主机的 IP 或名称添加到 **/etc/ansible/hosts** Ansible 清单文件中：

```
node.example.com
```

2. 使用以下内容创建 **~/adding-and-removing-ports.yml** playbook：

```
---
- name: Allow incoming HTTPS traffic to the local host
  hosts: node.example.com
  become: true

  tasks:
    - include_role:
        name: linux-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新引导后仍然有效。

3. 运行 playbook：

- 要以 **root** 用户身份连接到受管主机，请输入：

```
# ansible-playbook -u root ~/adding-and-removing-ports.yml
```

- 以用户身份连接到受管主机，请输入：

```
# ansible-playbook -u user_name --ask-become-pass ~/adding-and-removing-ports.yml
```

--ask-become-pass 选项确保 **ansible-playbook** 命令提示输入 **-u *user_name*** 选项中定义的用户的 **sudo** 密码。

如果没有指定 **-u *user_name*** 选项，**ansible-playbook** 以当前登录到控制节点的用户身份连接到受管主机。

验证

1. 连接到受管节点：

```
$ ssh user_name@node.example.com
```

2. 验证与 **HTTPS** 服务关联的 **443/tcp** 端口是否打开：

```
$ sudo firewall-cmd --list-ports
443/tcp
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md**
- **ansible-playbook(1)** 手册页

1.6. 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。

1.6.1. 列出区域

这个步骤描述了如何使用命令行列出区。

流程

1. 查看系统中有哪些可用区：

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

   firewall-cmd --get-zones 命令显示系统上所有可用的区，但不显示特定区的任何详情。

2. 查看所有区的详细信息：

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

3. 查看特定区的详细信息：

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

1.6.2. 更改特定区的 firewalld 设置

使用 cli 控制预定义服务的流量 和 使用cli控制端口 解释了如何在当前工作区范围内添加服务或修改端口。有时，需要在不同区内设置规则。

步骤

- 要在不同的区中工作，请使用 **--zone=*zone-name*** 选项。例如，允许在区 *public* 中使用 **SSH** 服务：

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

1.6.3. 更改默认区

系统管理员在其配置文件中为网络接口分配区域。如果接口没有被分配给指定区，它将被分配给默认区。每次重启 **firewalld** 服务后，**firewalld** 加载默认区的设置，使其处于活动状态。

步骤

设置默认区：

1. 显示当前的默认区：

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

2. 设置新的默认区：

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

注意

遵循此流程后，该设置是永久设置，即使没有 **--permanent** 选项。

1.6.4. 将网络接口分配给区

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

流程

要将区分配给特定的接口：

1. 列出活跃区以及分配给它们的接口：

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

2. 为不同的区分配接口：

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

1.6.5. 使用 nmcli 为连接分配区域

这个流程描述了如何使用 **nmcli** 工具将 **firewalld** 区添加到 **NetworkManager** 连接中。

步骤

1. 将区分配到 **NetworkManager** 连接配置文件：

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

2. 激活连接：

```
# nmcli connection up profile
```

1.6.6. 在 ifcfg 文件中手动将区分配给网络连接

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

步骤

- 要为连接设置区，请编辑 **/etc/sysconfig/network-scripts/ifcfg-*connection_name*** 文件，并添加一行，将区分配给这个连接：

```
ZONE=zone_name
```

1.6.7. 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 命令不起作用。

步骤

1. 创建一个新区：

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

2. 检查是否在您的永久设置中添加了新的区：

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

3. 使新设置具有持久性：

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

1.6.8. 区配置文件

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

firewalld 区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* 的长度目前限制为 17 个字符。区配置文件位于 **/usr/lib/firewalld/zones/** 和 **/etc/firewalld/zones/** 目录中。

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

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

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

其他资源

- **firewalld.zone** 手册页

1.6.9. 使用区目标设定传入流量的默认行为

对于每个区，您可以设置一种处理尚未进一步指定的传入流量的默认行为。此行为是通过设置区目标来定义的。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。详情请查看 **firewall-cmd(1)** man page 中的 **Options to Adapt and Query Zones and Policies** 部分。

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

1.7. 根据源使用区管理传入流量

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

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

1.7.1. 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：

步骤

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
```

1.7.2. 删除源

从区中删除源会关闭来自它的网络流量。

流程

1. 列出所需区的允许源：

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

2. 从区永久删除源：

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

3. 使新设置具有持久性：

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

1.7.3. 添加源端口

要启用基于源端口的流量排序，请使用 **--add-source-port** 选项来指定源端口。您还可以将其与 **--add-source** 选项结合使用，将流量限制在某个 IP 地址或 IP 范围。

步骤

- 添加源端口：

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

1.7.4. 删除源端口

通过删除源端口，您可以根据原始端口禁用对流量排序。

流程

- 要删除源端口：

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

1.7.5. 使用区和源来允许一个服务只适用于一个特定的域

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

警告

配置此场景时，请使用具有**默认**目标的区。使用将目标设为 **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。

步骤

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，以将来自源的流量路由到区：

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

3. 将**http** 服务添加到 **internal** 区中：

```
# 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** 区是否处于活跃状态，以及该区中是否允许服务：

```
# 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

1.8. 在区域间过滤转发的流量

使用策略对象，用户可以对策略中需要类似权限的不同身份进行分组。您可以根据流量的方向应用策略。

策略对象功能在 firewalld 中提供转发和输出过滤。以下描述了使用 firewalld 来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。

1.8.1. 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）附加到策略上。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。

1.8.2. 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。较高的优先级值优先于较低的优先级值。

以下规则适用于策略优先级：

- 具有负优先级的策略在区域中的规则之前应用。
- 具有正优先级的策略在区域中的规则之后应用。
- 优先级 0 被保留，因此不能使用。

1.8.3. 使用策略对象来过滤本地托管容器与主机物理连接的网络之间的流量

策略对象功能允许用户过滤其容器和虚拟机流量。

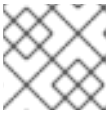
步骤

1. 创建新策略。

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

2. 阻止所有流量。

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



注意

红帽建议您默认阻止到主机的所有流量，然后有选择地打开主机所需的服务。

3. 定义与策略一起使用的 ingress 区域。

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

4. 定义与策略一起使用的 egress 区域。

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

验证

- 验证关于策略的信息。

```
# firewall-cmd --info-policy podmanToHost
```

1.8.4. 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
```

1.9. 使用 FIREWALLD 配置 NAT

使用 **firewalld**，您可以配置以下网络地址转换(NAT)类型：

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

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

这些是不同的网络地址转换（NAT）类型：

伪装和源 NAT（SNAT）

使用以上 NAT 类型之一更改数据包的源 IP 地址。例如，互联网服务提供商不会路由私有 IP 范围，如 **10.0.0.0/8**。如果您在网络中使用私有 IP 范围，并且用户应该能够访问 Internet 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 都非常相似。不同之处是：

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

目标 NAT（DNAT）

使用此 NAT 类型重写传入数据包的目标地址和端口。例如，如果您的 Web 服务器使用私有 IP 范围内的 IP 地址，那么无法直接从互联网访问它，您可以在路由器上设置 DNAT 规则，以便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。

1.9.2. 配置 IP 地址伪装

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

步骤

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**，且退出状态为 **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. 要使此设置具有持久性，请将 **--permanent** 选项传递给命令。
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** 选项传递给命令。

1.10. 端口转发

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

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

注意

您无法通过从配置了本地转发的主机重定向的端口访问服务。

1.10.1. 添加一个端口来重定向

使用 **firewalld**，您可以设置端口重定向，以便到达您系统上某个端口的任何传入的流量都被传送到您选择的其他内部端口或另一台计算机上的外部端口。

先决条件

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

流程

1. 将端口重新指向另一个端口：

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

2. 将端口重定向到不同 IP 地址的另一个端口：

   a. 添加要转发的端口：

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

   b. 启用伪装：

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

1.10.2. 将 TCP 端口 80 重定向到同一台机器中的 88 端口

按照以下步骤将 TCP 端口 80 重定向到端口 88。

流程

1. 将端口 80 重定向到 TCP 流量的端口 88:

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

2. 使新设置具有持久性：

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

3. 检查是否重定向了端口：

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

1.10.3. 删除重定向的端口

这个步骤描述了如何删除重定向的端口。

流程

1. 要删除重定向的端口：

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

2. 要删除重定向到不同地址的转发端口：

   a. 删除转发的端口：

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

   b. 禁用伪装：

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

1.10.4. 在同一台机器上将 TCP 端口 80 转发到端口 88

这个步骤描述了如何删除端口重定向。

流程

1. 列出重定向的端口：

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

2. 从防火墙中删除重定向的端口：

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

3. 使新设置具有持久性：

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

1.11. 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发送错误消息和表示连接问题的操作信息，例如，请求的服务不可用。**ICMP** 与 TCP 和 UDP 等传输协议不同，因为它不用于在系统之间交换数据。

不幸的是，可以使用 **ICMP** 消息（特别是 **echo-request** 和 **echo-reply** ）来揭示关于您网络的信息，并将这些信息滥用于各种欺诈活动。因此，**firewalld** 允许阻止 **ICMP** 请求，来保护您的网络信息。

1.11.1. 列出和阻塞 ICMP 请求

列出 ICMP 请求

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

- 要列出所有可用的 **ICMP** 类型：

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

- IPv4、IPv6 或这两种协议都可以使用 **ICMP** 请求。要查看 **ICMP** 请求使用了哪种协议：

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

- 如果请求当前被阻止了，则 **ICMP** 请求的状态显示为 **yes**，如果没有被阻止，则显示为 **no**。查看 **ICMP** 请求当前是否被阻断了：

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

阻止或取消阻止 ICMP 请求

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

- 要查看 **ICMP** 请求当前是否被阻断了：

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

- 要阻止 **ICMP** 请求：

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

- 要删除 **ICMP** 请求的块：

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

在不提供任何信息的情况下阻塞 ICMP 请求

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

- 要阻止和丢弃所有 **ICMP** 请求：
- 将区的目标设为 **DROP** ：

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

现在，除您明确允许的流量外，所有流量（包括 **ICMP** 请求）都将被丢弃。

阻止和丢弃某些 **ICMP** 请求，而允许其他的请求：

1. 将区的目标设为 **DROP** ：

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

2. 添加 ICMP block inversion 以一次阻止所有 **ICMP** 请求：

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

3. 为您要允许的 **ICMP** 请求添加 ICMP 块：

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

4. 使新设置具有持久性：

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

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

将块 inversion 恢复到完全 permissive 设置：

1. 将区的目标设置为 **default** 或 **ACCEPT**:

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

2. 删除 **ICMP** 请求的所有添加的块：

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

3. 删除 **ICMP** block inversion：

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

4. 使新设置具有持久性：

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

1.11.2. 使用 GUI 配置 ICMP 过滤器

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

- 若要启用反向 **ICMP Filter**，可点击右侧的 **Invert Filter** 复选框。现在只接受标记为 **ICMP** 的类型，所有其他的均被拒绝。在使用 DROP 目标的区域里它们会被丢弃。

1.12. 使用 FIREWALLD 设置和控制 IP 集

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

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

1.12.1. 使用 CLI 配置 IP 设置选项

IP 集可以在 **firewalld** 区中用作源，也可以用作富规则中的源。在 Red Hat Enterprise Linux 中，首选的方法是使用在直接规则中使用通过 **firewalld** 创建的 IP 集。

- 要列出 permanent 环境中 **firewalld** 已知的 IP 集，请以 **root** 用户身份运行以下命令：

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

- 要添加新的 IP 集，请以 **root** 用户身份使用 permanent 环境来运行以下命令：

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

  上述命令为 **IPv4** 创建了一个名为 *test*， 类型为 **hash:net** 的新的 IP 集。要创建用于 **IPv6** 的 IP 集，请添加 **--option=family=inet6** 选项。要使新设置在运行时环境中有效，请重新加载 **firewalld**。

- 使用以下命令，以 **root** 用户身份列出新的 IP 集：

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

- 要获取有关 IP 集的更多信息，请以 **root** 用户身份运行以下命令：

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

  请注意，IP 集目前没有任何条目。

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

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

  前面的命令将 IP 地址 *192.168.0.1* 添加到 IP 集合中。

- 要获取 IP 集中的当前条目列表，请以 **root** 用户身份运行以下命令：

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

- 生成包含 IP 地址列表的文件，例如：

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

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

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

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

- 要查看 IP 集的扩展条目列表，请以 **root** 用户身份运行以下命令：

```
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

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

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

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

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

源中的 **ipset:** 前缀显示 **firewalld** 的源是一个 IP 集，而不是 IP 地址或地址范围。

IP 集的创建和删除只限于 permanent 环境，所有其他 IP 集选项也可以用在运行时环境中，而不需要 **--permanent** 选项。

警告

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

1.13. 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。

1.13.1. 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，值越小优先级越高。

firewalld 服务会根据其优先级的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，规则将重定向到带有 **_log**、**_deny** 或 **_allow** 的链中。

在这些子链中，**firewalld** 会根据其优先级的值对规则进行排序。

1.13.2. 设置丰富的规则的优先级

该流程描述了如何创建一个富规则的示例，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

1. 添加一个带有非常低优先级的丰富规则来记录未由其他规则匹配的所有流量：

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

   命令还将日志条目的数量限制为每分钟 **5** 个。

2. 另外，还可显示上一步中命令创建的 **nftables** 规则：

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

1.14. 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。

1.14.1. 使用 CLI 配置锁定

这个流程描述了如何使用命令行来启用或禁用锁定。

- 要查询是否启用了锁定，请以 **root** 用户身份运行以下命令：

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

  如果启用了锁定，该命令将打印 **yes**，且退出状态为 **0**。否则，将打印 **no**，且退出状态为 **1**。

- 要启用锁定，请以 **root** 用户身份输入以下命令：

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

- 要禁用锁定，请以 **root** 用户身份使用以下命令：

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

1.14.2. 使用 CLI 配置锁定允许列表选项

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

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

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

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

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

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

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

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

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

- 要从允许列表中删除命令 *command*，请以 **root** 用户身份输入以下命令：

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

- 要查询命令 *command* 是否在允许列表中，请以 **root** 用户身份输入以下命令：

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

如果为真，该命令将打印 **yes**，且退出状态为 **0**。否则，将打印 **no**，且退出状态为 **1**。

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

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

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

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

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

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

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

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

如果为真，则打印 **yes**，且退出状态为 **0**，否则，打印 **no**，且退出状态为 **1**。

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

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

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

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

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

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

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

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

如果为真，则打印 **yes**，且退出状态为 **0**，否则，打印 **no**，且退出状态为 **1**。

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

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

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

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

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

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

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

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

如果为真，则打印 **yes**，且退出状态为 **0**，否则，打印 **no**，且退出状态为 **1**。

1.14.3. 使用配置文件配置锁定的 allowlist 选项

默认的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>
   <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示例，为 **firewall-cmd** 工具启用所有命令，对于名为 *user* 的用户，其用户 ID 为 **815** ：

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

此示例显示了**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。您还可以使用特定的命令，例如：

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

在该示例中，只允许 **--lockdown-on** 命令。

在 Red Hat Enterprise Linux 中，所有工具都放在 **/usr/bin/** 目录中，**/bin/** 目录被符号链接到 **/usr/bin/** 目录。换句话说，尽管以 **root** 身份输入的 **firewall-cmd** 的路径可能会被解析为 **/bin/firewall-cmd**，但现在 **/usr/bin/firewall-cmd** 可以使用。所有新脚本都应该使用新位置。但请注意，如果以 **root** 身份运行的脚本被写为使用 **/bin/firewall-cmd** 路径，那么除了通常只用于非**root** 用户的 **/usr/bin/firewall-cmd** 路径外，还必须在允许列表中添加该命令的路径。

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。

1.15. 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。

1.15.1. 区域内部转发与默认目标设置为 ACCEPT 的区域之间的区别

启用区内部转发时，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。区域指定接口和源的信任级别。如果信任级别相同，则接口或源之间的通信是可能的。

请注意，如果您在 **firewalld** 的默认区域中启用了区域内部转发，则它只适用于添加到当前默认区域的接口和源。

firewalld 的 **trusted** 区域使用设为 **ACCEPT** 的默认目标。这个区域接受所有转发的流量，但不支持区域内转发。

对于其他默认目标值，默认情况下会丢弃转发的流量，这适用于除可信区域之外的所有标准的区域。

1.15.2. 使用区域内部转发来在以太网和 Wi-Fi 网络间转发流量

您可以使用区域内部转发来转发同一 **firewalld** 区域内接口和源之间转发流量。例如，使用此功能来转发连接到 **enp1s0** 以太网和连接到 **wlp0s20** Wi-Fi 网络之间的流量。

步骤

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. 确保要在其之间启用区域内部转发的接口没有被分配给与 **internal** 区域不同的区域：

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

3. 如果接口当前分配给了 **internal** 以外的区域，请对其重新分配：

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

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

```
# 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域转发的主机的 **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** 接口位于同一网络的主机。

4. 连接到运行在与 **enp1s0** 在同一网络的主机上的 echo 服务器：

```
# ncat <other host> 12345
```

5. 输入一些内容，并按 **Enter**，然后验证文本是否发送回来。

其他资源

- **firewalld.zones(5)** 手册页

1.16. 在 ANSIBLE 中使用 RHEL 系统角色配置 FIREWALLD 设置

您可以使用 Ansible 防火墙系统角色一次性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。
- 保留所有预期的 **firewalld** 参数。

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

重要

请注意，通过 RHEL 频道提供的 RHEL 系统角色可在默认 **AppStream** 软件仓库中作为 RPM 软件包提供给 RHEL 客户。RHEL 系统角色还可以通过 Ansible Automation Hub 为客户提供 Ansible 订阅的集合。

1.16.1. 防火墙 RHEL 系统角色简介

RHEL 系统角色是 Ansible 自动化实用程序的一组内容。此内容与 Ansible 自动化实用程序相结合，提供了一致的配置界面，用于远程管理多个系统。

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

要以自动化的方式在一个或多个系统上应用 **firewalld** 参数，请在 playbook 中使用 **firewall** 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 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（zone）。
- 应允许数据包的服务。
- 授予、拒绝或丢弃对端口的流量访问。
- 为区转发端口或端口范围。

其他资源

- **README.md** 和 **README.html** 文件位于 **/usr/share/doc/rhel-system-roles/firewall/** 目录中
- 使用 playbook

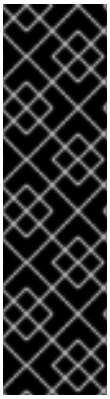
- 如何构建清单

1.16.2. 将传入的流量从一个本地端口转发到不同的本地端口

使用 **rhel-system-roles.firewall** 角色，您可以远程配置 **firewalld** 参数，同时对多个受管主机的影响具有持久性。

先决条件

- RHEL 订阅授权，您将在控制节点上安装 **ansible-core** 和 **rhel-system-roles** 软件包。
- 受管主机清单存在于控制计算机上，Ansible 能够连接它们。
- 有权限在受管主机上运行 Ansible playbook。
- 如果您运行 playbook 时使用了与 **root** 不同的远程用户，则此用户对受管主机具有适当的 **sudo** 权限。
- 清单文件列出 playbook 应执行操作的主机。此流程中的 playbook 在组 **testinservers** 的主机上运行。



重要

RHEL 8.0 - 8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**; 连接器，如 **docker** 和 **podman**; 以及插件和模块的整个环境。有关如何获取并安装 Ansible Engine 的信息，请参阅如何下载和安装 Red Hat Ansible Engine? 。

RHEL 8.6 和更新的版本中引入了 Ansible Core（以 **ansible-core** RPM 提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。AppStream 存储库提供 **ansible-core**，它的范围有限。如需更多信息，请参阅 RHEL 9 AppStream 中包含的 ansible-core 软件包的范围。

步骤

1. 创建 **~/port_forwarding.yml** 文件并添加以下内容：

```
---
- name: Forward incoming traffic on port 8080 to 443
  hosts: testingservers

  tasks:
    - include_role:
        name: rhel-system-roles.firewall

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

此文件代表一个 playbook，通常包含了一组有特定顺序的任务（也称为 *play* ）列表。这些任何会根据 **inventory** 文件中选择的特定管理主机进行。在这种情况下，该 playbook 将针对受管主机的 **testingservers** 组运行。

Play 中的 **hosts** 键指定对其运行 play 的主机。您可以将这个键的值作为受管主机的单独名称，或作为 **inventory** 文件中定义的主机组提供。

tasks 部分包含 **include_role** 键，它指定了哪些系统角色将配置 **vars** 部分中提到的参数和值。

vars 部分包含一个名为 **firewall** 的角色变量。此变量是字典值列表，并指定应用于受管主机上的 **firewalld** 的参数。example 角色将进入端口 8080 的流量转发到端口 443。设置将立即生效，并将在重启后保留。

2. （可选）验证 playbook 中的语法是否正确：

```
# ansible-playbook --syntax-check ~/port_forwarding.yml

playbook: port_forwarding.yml
```

本例演示了对 playbook 的成功验证。

3. 执行 playbook：

```
# ansible-playbook ~/port_forwarding.yml
```

验证

- 在受管主机上：
  - 重启主机以验证 **firewalld** 设置是否在重启后是否仍存在：

```
# reboot
```

  - 显示 **firewalld** 设置：

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

其他资源

- RHEL 系统角色入门
- **README.html** 和 **README.md** 文件在 **/usr/share/doc/rhel-system-roles/firewall/** 目录中
- 构建您的清单
- 配置 Ansible
- 使用 Playbook
- 使用变量
- 角色

1.16.3. 使用系统角色配置端口

您可以使用 Red Hat Enterprise Linux(RHEL) **firewalld** 系统角色为传入的流量打开或关闭本地防火墙中的端口，并在重新引导后保持新配置。这个示例描述了如何配置 default 区以允许 **HTTPS** 服务的传入流量。

在 Ansible 控制节点上运行此步骤。

先决条件

- 可以访问一个或多个 *受管节点*，它们是您要使用 **firewalld** 系统角色来配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
- **ansible-core** 和 **rhel-system-roles** 软件包在控制节点上安装。
- 如果您在运行 playbook 时使用了与 **root** 不同的远程用户，则此用户在受管节点上具有合适的 **sudo** 权限。
- 主机使用 NetworkManager 配置网络。

步骤

1. 如果您要在其上执行 playbook 中指令的主机还没有被列入清单，请将此主机的 IP 或名称添加到 **/etc/ansible/hosts** Ansible 清单文件中：

```
node.example.com
```

2. 使用以下内容创建 **~/adding-and-removing-ports.yml** playbook：

```
---
- name: Allow incoming HTTPS traffic to the local host
  hosts: node.example.com
  become: true

  tasks:
    - include_role:
        name: linux-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新引导后仍然有效。

3. 运行 playbook：

- 要以 **root** 用户身份连接到受管主机，请输入：

```
# ansible-playbook -u root ~/adding-and-removing-ports.yml
```

- 以用户身份连接到受管主机，请输入：

```
# ansible-playbook -u user_name --ask-become-pass ~/adding-and-removing-ports.yml
```

--ask-become-pass 选项确保 **ansible-playbook** 命令提示输入 **-u *user_name*** 选项中定义的用户的 **sudo** 密码。

如果没有指定 **-u *user_name*** 选项，**ansible-playbook** 以当前登录到控制节点的用户身份连接到受管主机。

验证

1. 连接到受管节点：

```
$ ssh user_name@node.example.com
```

2. 验证与 **HTTPS** 服务关联的 **443/tcp** 端口是否打开：

```
$ sudo firewall-cmd --list-ports
443/tcp
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md**
- **ansible-playbook(1)** 手册页

1.16.4. 使用 firewalld RHEL 系统角色配置 DMZ firewalld 区域

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

先决条件

- 对一个或多个 *受管节点* 的访问和权限，受管节点是您要使用 VPN 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
- 列出受管节点的清单文件。
- **ansible-core** 和 **rhel-system-roles** 软件包在控制节点上安装。
- 如果您在运行 playbook 时使用了与 **root** 不同的远程用户，则此用户在受管节点上具有合适的 **sudo** 权限。
- 受管节点使用 **NetworkManager** 配置网络。

步骤

1. 使用以下内容创建 **~/configuring-a-dmz-using-the-firewall-system-role.yml** playbook：

```
---
- name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
  hosts: node.example.com
  become: true

  tasks:
    - include_role:
        name: linux-system-roles.firewall

      vars:
```

```
    firewall:
      - zone: dmz
        interface: enp1s0
        service: https
        state: enabled
        runtime: true
        permanent: true
```

2. 运行 playbook：

   - 要以 **root** 用户身份连接到受管主机，请输入：

     ```
     $ ansible-playbook -u root ~/configuring-a-dmz-using-the-firewall-system-role.yml
     ```

   - 以用户身份连接到受管主机，请输入：

     ```
     $ ansible-playbook -u user_name --ask-become-pass ~/configuring-a-dmz-using-the-firewall-system-role.yml
     ```

     --ask-become-pass 选项确保 **ansible-playbook** 命令提示输入 **-u *user_name*** 选项中定义的用户的 **sudo** 密码。

   如果没有指定 **-u *user_name*** 选项，**ansible-playbook** 以当前登录到控制节点的用户身份连接到受管主机。

验证

- 在受管节点上，查看 **dmz** 区域的详细信息：

```
# 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:
```

1.17. 其他资源

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

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

第 2 章 NFTABLES 入门

nftables 框架提供了数据包分类功能。最显著的功能是：

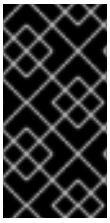
- 内置查找表而不是线性处理
- **IPv4** 和 **IPv6** 使用同一个协议框架
- 规则会以一个整体被应用，而不是分为抓取、更新和存储完整的规则集的步骤
- 支持在规则集(**nftrace**)和监控追踪事件（**nft**）中调试和追踪
- 更加一致和压缩的语法，没有特定协议的扩展
- 用于第三方应用程序的 Netlink API

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

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

2.1. 从 IPTABLES 迁移到 NFTABLES

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



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** 工具来设置复杂和性能关键的防火墙，如整个网络。
- **iptables**:Red Hat Enterprise Linux 上的 **iptables** 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

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

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。

步骤

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 和 （如果需要），请手动更新生成的 **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，以将 **iptables** 或 **ip6tables** 规则转换为 **nftables** 的对等规则。

先决条件

- 已安装 **nftables** 软件包。

步骤

- 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

以下是常见的 **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。

Example:**列出 firewalld 生成的规则**

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

2.1.5. 其他资源

- iptables：这两个变体及其与 nftables 的关系

2.2. 编写和执行 NFTABLES 脚本

nftables 框架提供了一个原生脚本环境，与使用shell脚本来维护防火墙规则相比，它带来了一个主要好处：执行脚本是原子的。这意味着，系统会应用整个脚本，或者在出现错误时防止执行。这样可保证防火墙始终处于一致状态。

另外，**nftables** 脚本环境使管理员能够：

- 添加评论
- 定义变量
- 包含其他规则集文件

本节介绍了如何使用这些功能，以及如何创建和执行 **nftables** 脚本。

安装 **nftables** 软件包时，Red Hat Enterprise Linux 会在 **/etc/nftables/** 目录中自动创建 ***.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

nftables 脚本环境支持以下格式的脚本：

- 您可以以与 **nft list ruleset** 命令相同的格式来编写脚本，显示规则集：

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

# Flush the rule set
flush ruleset

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

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

- 你可以对命令使用与 **nft** 命令相同的语法：

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

# Flush the rule set
flush ruleset

# Create a table
add table inet example_table
```

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

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

2.2.2. 运行 nftables 脚本

您可以通过将其传给 **nft** 工具或直接执行脚本来运行 **nftables** 脚本。

先决条件

- 本节的流程假设您在 **/etc/nftables/example_firewall.nft** 文件中存储了 **nftables** 脚本。

步骤

- 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.nft
```

- 要直接运行 **nftables** 脚本：
  a. 只需要执行一次的步骤：
     i. 确保脚本以以下 shebang 序列开头：

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

重要

如果省略 **-f** 参数，**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.nft
```

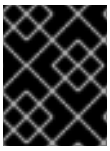
     iii. 使脚本可以被其所有者执行：

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

  b. 运行脚本：

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

如果没有输出结果，系统将成功执行该脚本。



重要

即使 **nft** 成功执行了脚本，在脚本中错误放置的规则、缺少参数或其他问题都可能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都视为注释。

例 2.1. nftables 脚本中的注释

注释可在一行的开始，也可以在命令后：

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

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

2.2.4. 使用 nftables 脚本中的变量

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

只有一个值的变量

以下示例定义了一个名为 **INET_DEV** 的变量，其值为 **enp1s0** ：

```
define INET_DEV = enp1s0
```

您可以在脚本中使用变量，方法是在 **$** 符号后跟变量名：

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

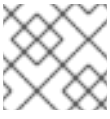
包含匿名集合的变量

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 }
```

您可以在脚本中使用变量，方法是在 **$** 符号后跟变量名：

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



注意

请注意，在规则中使用大括号时具有特殊的意义，因为它们表示变量代表一个集合。

其他资源

- 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

nftables 脚本环境可让管理员通过使用 **include** 语句来包含其他脚本。

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 2.2. 包含默认搜索目录中的文件

从默认搜索目录中包含一个文件：

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

例 2.3. 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

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。这部分论述了如何在系统引导时载入防火墙规则。

先决条件

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

步骤

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

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

2. （可选）启动 **nftables** 服务来载入防火墙规则，而不用重启系统：

```
# systemctl start nftables
```

3. 启用 **nftables** 服务。

```
# systemctl enable nftables
```

其他资源

- 支持的 nftables 脚本格式

2.3. 创建和管理 NFTABLES 表、链和规则

本节介绍了如何显示 **nftables** 规则集以及如何管理它们。

2.3.1. 标准链优先级值和文本名称

当创建链时，您可以将 **priority** 设为整数值或标准名称，来指定具有相同 **hook** 值链的顺序。

名称和值是根据 **xtables** 在注册其默认链时使用的优先级来定义的。

注意

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

例 2.4. 使用文本值设定优先级

以下命令使用标准优先级值 **50**， 在 **example_table** 中创建一个名为 **example_chain** 的链：

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

因为优先级是一个标准值，所以您可以使用文本值：

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

表 2.1. 标准优先级名称、系列和 hook 兼容性列表

名称	值	系列	Hook
raw	-300	**ip ip6、inet**	all
mangle	-150	**ip ip6、inet**	all
dstnat	-100	**ip ip6、inet**	prerouting
filter	0	**ip、ip6、inet、arp、netdev**	all

名称	值	系列	Hook
安全	50	**ip ip6、inet**	all
srcnat	100	**ip ip6、inet**	postrouting

所有系列都使用相同的值，但 **bridge** 系列使用以下值：

表 2.2. 网桥系列的标准优先级名称和 hook 兼容性

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

其他资源

- **nft(8)** 手册页中的 **Chains** 部分

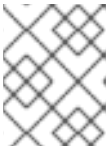
2.3.2. 显示 nftables 规则集

nftables 的规则集包含表、链和规则。本节介绍如何显示规则集。

流程

- 要显示规则集，请输入：

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



注意

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

2.3.3. 创建 nftables 表

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

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

- **ip**:仅匹配 IPv4 数据包。如果没有指定地址系列，这是默认设置。
- **ip6** ：仅匹配 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。

步骤

1. 使用 **nft add table** 命令来创建新表。例如，要创建一个名为 **example_table** 、用来处理 IPv4 和 IPv6 数据包的表：

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

2. 另外，还可列出规则集中的所有表：

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

其他资源

- **nft(8)** 手册页中的 **Address families** 部分
- **nft(8)** 手册页中的 **Tables** 部分

2.3.4. 创建 nftables 链

chains 是规则的容器。存在以下两种规则类型：

- 基本链：您可以使用基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 **jump** 目标，并更好地组织规则。

这个步骤描述了如何在现有表中添加基本链。

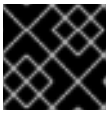
先决条件

- 已存在您要添加新链的表。

步骤

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

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



重要

为避免 shell 将分号解析为命令的结尾，请在分号前加上 \ 转义字符。

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

2. 另外，还可以显示所有链：

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

其他资源

- **nft(8)** 手册页中的 **Address families** 部分
- **nft(8)** 手册页中的 **Chains** 部分

2.3.5. 将规则附加到 nftables 链的末尾

本节介绍了如何将规则附加到现有 **nftables** 链的末尾。

先决条件

- 您要添加该规则的链已存在。

步骤

1. 要添加新的规则，请使用 **nft add rule** 命令。例如，要在 **example_table** 的 **example_chain** 中添加一条允许端口 22 上 TCP 流量的规则：

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

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

2. 另外，还可在 **example_table** 中显示所有的链及其规则：

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

其他资源

- **nft(8)** 手册页中的 **Address families** 部分
- **nft(8)** 手册页中的 **Rules** 部分

2.3.6. 在 nftables 链的开头插入一条规则

本节介绍了如何在现有 **nftables** 链的开头插入一条规则。

先决条件

- 您要添加该规则的链已存在。

步骤

1. 要插入新规则，请使用 **nft insert rule** 命令。例如，要在 **example_table** 的 **example_chain** 中插入一条允许端口 22 上 TCP 流量的规则：

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

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

2. 另外，还可在 **example_table** 中显示所有的链及其规则：

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

其他资源

- **nft(8)** 手册页中的 **Address families** 部分
- **nft(8)** 手册页中的 **Rules** 部分

2.3.7. 在 nftables 链的特定位置插入一条规则

本节介绍了如何在 **nftables** 链中现有规则的前和后插入规则。这样，您可以将新规则放在正确的位置上。

先决条件

- 您要添加规则的链存在。

步骤

1. 使用 **nft -a list ruleset** 命令显示 **example_table** 中的所有的链及其规则，包括它们的句柄：

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

使用 **-a** 显示句柄。您需要此信息才能在后续步骤中定位新规则。

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

   - 要在句柄 **3** 前插入一条允许端口 **636** 上TCP 流量的规则，请输入：

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

   - 要在句柄 **3** 后添加一条允许端口 **80** 上 TCP 流量的规则，请输入：

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

3. 另外，还可在 **example_table** 中显示所有的链及其规则：

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

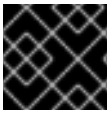
其他资源

- **nft(8)** 手册页中的 **Address families** 部分
- **nft(8)** 手册页中的 **Rules** 部分

2.4. 使用 NFTABLES 配置 NAT

使用 **nftables**，您可以配置以下网络地址转换(NAT)类型：

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



重要

您只能在 **iifname** 和 **oifname** 参数中使用实际接口名称，不支持其他名称(**altname**)。

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

这些是不同的网络地址转换（NAT）类型：

伪装和源 NAT（SNAT）

使用以上 NAT 类型之一更改数据包的源 IP 地址。例如，互联网服务提供商不会路由私有 IP 范围，如 **10.0.0.0/8**。如果您在网络中使用私有 IP 范围，并且用户应该能够访问 Internet 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 都非常相似。不同之处是：

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

目标 NAT（DNAT）

使用此 NAT 类型重写传入数据包的目标地址和端口。例如，如果您的 Web 服务器使用私有 IP 范围内的 IP 地址，那么无法直接从互联网访问它，您可以在路由器上设置 DNAT 规则，以便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，如果接口被分配了新的 IP，**nftables** 会在替换源 IP 时自动使用新的 IP。

以下流程描述了如何将通过 **ens3** 接口的离开主机的数据包的源 IP 替换为 **ens3** 上设置的 IP。

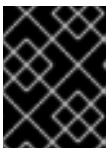
步骤

1. 创建一个表：

```
# nft add table nat
```

2. 将 **prerouting** 和 **postrouting** 链添加到表中：

```
# 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 **prerouting** 添加规则，**nftables** 框架也要求此链与传入的数据包回复匹配。

请注意，您必须将 **--** 选项传给 **nft** 命令，以避免 shell 将负优先级的值解析为 **nft** 命令的一个选项。

3. 在 **postrouting** 链中添加一条与 **ens3** 接口上的传出数据包匹配的规则：

```
# 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。

以下流程描述了如何将通过 **ens3** 接口的离开路由器的数据包的源 IP 替换为 **192.0.2.1**。

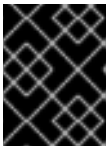
步骤

1. 创建一个表：

```
# nft add table nat
```

2. 将 **prerouting** 和 **postrouting** 链添加到表中：

```
# 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** 框架也要求此链与传出的数据包回复匹配。

请注意，您必须将 **--** 选项传给 **nft** 命令，以避免 shell 将负优先级的值解析为 **nft** 命令的一个选项。

3. 在 **postrouting** 链中添加一条规则，其将通过 **ens3** 传出的数据包的源 IP 替换为 **192.0.2.1** ：

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