Red Hat Enterprise Linux 9

방화벽 및 패킷 필터 구성

firewalld 서비스, nftables 프레임워크 및 XDP 패킷 필터링 기능 관리

Last Updated: 2025-07-11

Red Hat Enterprise Linux 9 방화벽 및 패킷 필터 구성

firewalld 서비스, nftables 프레임워크 및 XDP 패킷 필터링 기능 관리

법적 공지

초록

방화벽과 같은 패킷 필터는 규칙을 사용하여 들어오고 나가는 네트워크 트래픽을 제어합니다. RHEL(Red Hat Enterprise Linux)에서는 firewalld 서비스 및 nftables 프레임워크를 사용하여 네트워크 트래픽을 필터링하고 성능에 중요한 방화벽을 빌드할 수 있습니다. 커널의 XDP(Express Data Path) 기능을 사용하여 매우 높은 속도로 네트워크 인터페이스에서 네트워크 패킷을 처리하거나 삭제할 수도 있습니다.

차례

RED HAT 문서에 관한 피드백 제공 3

1장. FIREWALLD 사용 및 구성 4
1.1. FIREWALLD, NFTABLES 또는 IPTABLES를 사용하는 경우 4
1.2. 방화벽 영역 4
1.3. 방화벽 정책 6
1.4. 방화벽 규칙 7
1.5. 영역 구성 파일 7
1.6. 사전 정의된 FIREWALLD 서비스 7
1.7. FIREWALLD 영역 작업 8
1.8. FIREWALLD를 사용하여 네트워크 트래픽 제어 16
1.9. 영역을 사용하여 소스에 따라 들어오는 트래픽 관리 25
1.10. 영역 간 전달된 트래픽 필터링 27
1.11. FIREWALLD를 사용하여 NAT 구성 32
1.12. ICMP 요청 관리 36
1.13. FIREWALLD를 사용하여 IP 세트 설정 및 제어 37
1.14. 풍부한 규칙 우선 순위 지정 39
1.15. 방화벽 잠금 구성 40
1.16. FIREWALLD 영역 내의 다양한 인터페이스 또는 소스 간 트래픽 전달 활성화 41
1.17. RHEL 시스템 역할을 사용하여 FIREWALLD 구성 43

2장. NFTABLES 시작하기 48
2.1. NFTABLES 테이블, 체인 및 규칙 생성 및 관리 48
2.2. IPTABLES에서 NFTABLES로 마이그레이션 53
2.3. NFTABLES 스크립트 작성 및 실행 58
2.4. NFTABLES를 사용하여 NAT 구성 62
2.5. NFTABLES 명령에서 세트 사용 67
2.6. NFTABLES 명령에서 VERDICT 맵 사용 69
2.7. 예제: NFTABLES 스크립트를 사용하여 LAN 및 DMZ 보호 72
2.8. NFTABLES를 사용하여 포트 전달 구성 77
2.9. NFTABLES를 사용하여 연결 양 제한 79
2.10. NFTABLES 규칙 디버깅 81
2.11. NFTABLES 규칙 세트 백업 및 복원 84
2.12. 추가 리소스 85

3장. DDOS 공격을 방지하기 위해 고성능 트래픽 필터링에 XDP-FILTER 사용 87
3.1. XDP-FILTER 규칙과 일치하는 네트워크 패킷 삭제 87
3.2. XDP-FILTER 규칙과 일치하는 항목을 제외한 모든 네트워크 패킷 삭제 89

RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

Jira를 통해 피드백 제출 (등록 필요)

1. Jira 웹 사이트에 로그인합니다.
2. 상단 탐색 모음에서 **생성** 을 클릭합니다.
3. **Summary** (요약) 필드에 설명 제목을 입력합니다.
4. **Description** (설명) 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

1장. FIREWALLD 사용 및 구성

방화벽 은 원치 않는 트래픽에서 시스템을 보호하는 방법입니다. 사용자는 *방화벽 규칙* 집합을 정의하여 호스트 시스템에서 들어오는 네트워크 트래픽을 제어할 수 있습니다. 이러한 규칙은 들어오는 트래픽을 정렬하고 이를 차단하거나 허용하는 데 사용됩니다.

firewalld 는 D-Bus 인터페이스를 사용하여 동적 사용자 지정 가능한 호스트 기반 방화벽을 제공하는 방화벽 서비스 데몬입니다. 동적이므로 규칙이 변경될 때마다 방화벽 데몬을 다시 시작할 필요 없이 규칙을 생성, 변경 및 삭제할 수 있습니다.

firewalld 는 트래픽 관리를 간소화하는 영역 및 서비스 개념을 사용합니다. 영역은 사전 정의된 규칙 세트입니다. 네트워크 인터페이스 및 소스를 영역에 할당할 수 있습니다. 허용되는 트래픽은 컴퓨터가 연결된 네트워크에 따라 다르며 이 네트워크에는 보안 수준이 할당됩니다. 방화벽 서비스는 특정 서비스에 대한 들어오는 트래픽을 허용하고 영역 내에서 적용하기 위해 필요한 모든 설정을 포함하는 사전 정의된 규칙입니다.

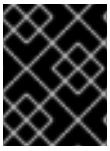
서비스는 네트워크 통신에 하나 이상의 포트 또는 주소를 사용합니다. 방화벽 필터는 포트를 기반으로 합니다. 서비스에 대한 네트워크 트래픽을 허용하려면 해당 포트가 열려 있어야 합니다. **firewalld** 는 명시적으로 open으로 설정되지 않은 포트에서 모든 트래픽을 차단합니다. trusted와 같은 일부 영역은 기본적으로 모든 트래픽을 허용합니다.

nftables 백엔드가 포함된 **firewalld** 는 **--direct** 옵션을 사용하여 사용자 정의 **nftables** 규칙 전달을 지원하지 않습니다.

1.1. FIREWALLD, NFTABLES 또는 IPTABLES를 사용하는 경우

다음은 다음 유틸리티 중 하나를 사용해야 하는 시나리오에 대한 간략한 개요입니다.

- **firewalld**: 간단한 방화벽 사용 사례에 **firewalld** 유틸리티를 사용합니다. 이 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.
- **nftables**: **nftables** 유틸리티를 사용하여 전체 네트워크에 대해 과 같이 복잡하고 성능이 중요한 방화벽을 설정합니다.
- **iptables**: Red Hat Enterprise Linux의 **iptables** 유틸리티는 **레거시** 백엔드 대신 **nf_tables** 커널 API를 사용합니다. **nf_tables** API는 이전 버전과의 호환성을 제공하므로 **iptables** 명령을 사용하는 스크립트가 여전히 Red Hat Enterprise Linux에서 작동합니다. 새 방화벽 스크립트의 경우 Red Hat은 **nftables** 를 사용하도록 권장합니다.



중요

다른 방화벽 관련 서비스(**firewalld**,**nftables** 또는 **iptables**)가 서로 영향을 미치지 않도록 하려면 RHEL 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

1.2. 방화벽 영역

firewalld 유틸리티를 사용하여 해당 네트워크 내의 인터페이스 및 트래픽과 함께 있는 신뢰 수준에 따라 네트워크를 다른 영역으로 분리할 수 있습니다. 연결은 하나의 영역의 일부일 수 있지만 많은 네트워크 연결에 해당 영역을 사용할 수 있습니다.

firewalld 는 영역과 관련하여 엄격한 원칙을 따릅니다.

1. 트래픽 수신은 하나의 영역만 포함됩니다.
2. 트래픽은 하나의 영역만 송신합니다.

3. 영역은 신뢰 수준을 정의합니다.

4. 기본적으로 Intrazone 트래픽(동일한 영역 내)이 허용됩니다.

5. 영역 간 트래픽은 기본적으로 거부됩니다.

규칙 4와 5는 원칙 3의 결과입니다.

원칙 4는 영역 옵션 **--remove-forward** 를 통해 구성할 수 있습니다. 원칙 5는 새로운 정책을 추가하여 구성할 수 있습니다.

NetworkManager 는 인터페이스의 영역을 **firewalld** 에 알립니다. 다음 유틸리티를 사용하여 인터페이스에 영역을 할당할 수 있습니다.

- **NetworkManager**
- **firewall-config** 유틸리티
- **firewall-cmd** 유틸리티
- RHEL 웹 콘솔

RHEL 웹 콘솔, **firewall-config** 및 **firewall-cmd** 는 적절한 **NetworkManager** 구성 파일만 편집할 수 있습니다. 웹 콘솔, **firewall-cmd** 또는 **firewall-config** 를 사용하여 인터페이스 영역을 변경하면 요청이 **NetworkManager** 로 전달되고**firewalld** 에서 처리되지 않습니다.

/usr/lib/firewalld/zones/ 디렉터리는 사전 정의된 영역을 저장하고 사용 가능한 네트워크 인터페이스에 즉시 적용할 수 있습니다. 이러한 파일은 수정된 경우에만 **/etc/firewalld/zones/** 디렉토리에 복사됩니다. 사전 정의된 영역의 기본 설정은 다음과 같습니다.

블록

- 적합한 대상: 들어오는 네트워크 연결은 **IPv4** 에 대한 icmp-host-prohibited 메시지와 icmp6-adm-adm-prohibited **IPv6** 로 거부됩니다.
- 허용: 시스템 내에서 시작된 네트워크 연결만 수행합니다.

dmz

- 적합한 대상: DMZ의 컴퓨터는 내부 네트워크에 대한 액세스 제한으로 공개적으로 액세스할 수 있습니다.
- 허용: 선택한 연결만 제공됩니다.

drop

적합한 대상: 들어오는 모든 네트워크 패킷은 알림 없이 삭제됩니다.

- 허용: 나가는 네트워크 연결만 가능합니다.

external

- 적합한 대상: 특히 라우터에 대해 마스커레이딩이 활성화된 외부 네트워크입니다. 네트워크에서 다른 컴퓨터를 신뢰하지 않는 경우입니다.
- 허용: 선택한 연결만 제공됩니다.

홈

- 적합한 대상: 네트워크상의 다른 컴퓨터를 주로 신뢰하는 홈 환경.
- 허용: 선택한 연결만 제공됩니다.

internal

- 적합한 대상: 네트워크에 있는 다른 컴퓨터를 주로 신뢰하는 내부 네트워크입니다.
- 허용: 선택한 연결만 제공됩니다.

public

- 적합한 대상: 네트워크에서 다른 컴퓨터를 신뢰하지 않는 공용 영역입니다.
- 허용: 선택한 연결만 제공됩니다.

trusted

- 허용: 모든 네트워크 연결

작업

적합한 대상: 네트워크에 있는 다른 컴퓨터를 주로 신뢰하는 작업 환경.

- 허용: 선택한 연결만 제공됩니다.

이러한 영역 중 하나가 *기본* 영역으로 설정됩니다. 인터페이스 연결이 **NetworkManager** 에 추가되면 기본 영역에 할당됩니다. 설치 시 **firewalld** 의 기본 영역은 **퍼블릭** 영역입니다. 기본 영역을 변경할 수 있습니다.

참고

네트워크 영역 이름을 자체 설명하여 사용자가 신속하게 이해할 수 있도록 합니다.

보안 문제를 방지하려면 기본 영역 구성을 검토하고 요구 사항 및 위험 평가에 따라 불필요한 서비스를 비활성화합니다.

추가 리소스

- 시스템의 **firewalld.zone(5)** 도움말 페이지

1.3. 방화벽 정책

방화벽 정책은 원하는 네트워크 보안 상태를 지정합니다. 다양한 유형의 트래픽에 대해 수행할 규칙과 작업을 간략하게 설명합니다. 일반적으로 정책에는 다음 유형의 트래픽에 대한 규칙이 포함됩니다.

- 들어오는 트래픽
- 나가는 트래픽
- 전송 트래픽
- 특정 서비스 및 애플리케이션
- NAT(네트워크 주소 변환)

방화벽 정책은 방화벽 영역의 개념을 사용합니다. 각 영역은 허용되는 트래픽을 결정하는 특정 방화벽 규칙 세트와 연결됩니다. 정책은 상태 저장되지 않은 방식으로 방화벽 규칙을 적용합니다. 즉, 트래픽의 한 방향만 고려합니다. **firewalld** 의 상태 저장 필터링으로 인해 트래픽 반환 경로는 암시적으로 허용됩니다.

정책은 Ingress 영역 및 송신 영역과 연결됩니다. Ingress 영역은 트래픽이 시작된 위치(received)입니다. 송신 영역은 트래픽이 떠나는 위치입니다(sent).

정책에 정의된 방화벽 규칙은 방화벽 영역을 참조하여 여러 네트워크 인터페이스에 일관된 구성을 적용할 수 있습니다.

1.4. 방화벽 규칙

방화벽 규칙을 사용하여 네트워크 트래픽을 허용하거나 차단하는 특정 구성을 구현할 수 있습니다. 따라서 네트워크 트래픽 흐름을 제어하여 시스템을 보안 위협으로부터 보호할 수 있습니다.

방화벽 규칙은 일반적으로 다양한 속성을 기반으로 특정 기준을 정의합니다. 속성은 다음과 같습니다.

- 소스 IP 주소
- 대상 IP 주소
- 전송 프로토콜 (TCP, UDP, ...)
- 포트
- 네트워크 인터페이스

firewalld 유틸리티는 방화벽 규칙을 영역(예: **공용**,**내부** 및 기타) 및 정책으로 구성합니다. 각 영역에는 특정 영역과 연결된 네트워크 인터페이스에 대한 트래픽 자유 수준을 결정하는 자체 규칙 세트가 있습니다.

1.5. 영역 구성 파일

firewalld 영역 구성 파일에는 영역에 대한 정보가 포함되어 있습니다. 이는 XML 파일 형식의 영역 설명, 서비스, 포트, 프로토콜, icmp-blocks, masquerade, forward-ports 및 풍부한 언어 규칙입니다. 파일 이름은 ***zone-name* 의 길이가 현재 17 chars로 제한되는 *zone-name*.xml** 이어야 합니다. 영역 구성 파일은 **/usr/lib/firewalld/zones/** 및 **/etc/firewalld/zones/** 디렉터리에 있습니다.

다음 예제에서는 **TCP** 및 **UDP** 프로토콜 모두에 대해 하나의 서비스(**SSH**)와 하나의 포트 범위를 허용하는 구성을 보여줍니다.

```
<?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>
```

추가 리소스

- **firewalld.zone** 설명서 페이지

1.6. 사전 정의된 FIREWALLD 서비스

firewalld 서비스는 특정 애플리케이션 또는 네트워크 서비스에 대한 액세스를 정의하는 사전 정의된 방화벽 규칙 세트입니다. 각 서비스는 다음 요소의 조합을 나타냅니다.

- 로컬 포트
- 네트워크 프로토콜
- 연결된 방화벽 규칙
- 소스 포트 및 대상
- 서비스가 활성화된 경우 자동으로 로드되는 방화벽 도우미 모듈

서비스는 패킷 필터링을 단순화하고 한 번에 여러 작업을 수행하기 때문에 시간을 절약합니다. 예를 들어 **firewalld** 는 다음 작업을 한 번에 수행할 수 있습니다.

- 포트 열기
- 네트워크 프로토콜 정의
- 패킷 전달 활성화

서비스 구성 옵션 및 일반 파일 정보는 시스템의 **firewalld.service(5)** 도움말 페이지에 설명되어 있습니다. 서비스는 개별 XML 구성 파일을 통해 지정됩니다. ***service-name*.xml .xml 형식으로 이름이 지정** 됩니다. 프로토콜 이름은 **firewalld** 에서 서비스 또는 애플리케이션 이름보다 우선합니다.

다음과 같은 방법으로 **firewalld** 를 구성할 수 있습니다.

- 유틸리티 사용:
  - **firewall-config** - 그래픽 유틸리티
  - **firewall-cmd** - 명령줄 유틸리티
  - **firewall-offline-cmd** - 명령줄 유틸리티
- **/etc/firewalld/services/** 디렉토리에서 XML 파일을 편집합니다.
  서비스를 추가하거나 변경하지 않으면 **/etc/firewalld/services/** 에 해당 XML 파일이 존재하지 않습니다. **/usr/lib/firewalld/services/** 의 파일을 템플릿으로 사용할 수 있습니다.

추가 리소스

- 시스템의 **firewalld.service(5)** 도움말 페이지

1.7. FIREWALLD 영역 작업

zones는 들어오는 트래픽을 보다 투명하게 관리하기 위한 개념을 나타냅니다. 영역은 네트워킹 인터페이스에 연결되거나 다양한 소스 주소가 할당됩니다. 각 영역에 대한 방화벽 규칙을 독립적으로 관리하여 복잡한 방화벽 설정을 정의하고 트래픽에 적용할 수 있습니다.

1.7.1. 보안을 강화하기 위해 특정 영역에 대한 방화벽 설정 사용자 정의

방화벽 설정을 수정하고 특정 네트워크 인터페이스 또는 특정 방화벽 영역과 연결하여 네트워크 보안을 강화할 수 있습니다. 영역에 대한 세분화된 규칙 및 제한을 정의하면 원하는 보안 수준에 따라 인바운드 및 아웃바운드 트래픽을 제어할 수 있습니다.

예를 들어 다음과 같은 이점을 얻을 수 있습니다.

- 민감한 데이터 보호
- 무단 액세스 방지
- 잠재적인 네트워크 위협 완화

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 사용 가능한 방화벽 영역을 나열합니다.

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

firewall-cmd --get-zones 명령은 시스템에서 사용할 수 있는 모든 영역을 표시하지만 특정 영역에 대한 세부 정보는 표시하지 않습니다. 모든 영역에 대한 자세한 정보를 보려면 **firewall-cmd --list-all-zones** 명령을 사용합니다.

2. 이 구성에 사용할 영역을 선택합니다.
3. 선택한 영역에 대한 방화벽 설정을 수정합니다. 예를 들어 **SSH** 서비스를 허용하고 **ftp** 서비스를 제거하려면 다음을 수행합니다.

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

4. 방화벽 영역에 네트워크 인터페이스를 할당합니다.
   a. 사용 가능한 네트워크 인터페이스를 나열합니다.

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

영역의 작업은 해당 구성과 일치하는 네트워크 인터페이스 또는 소스 주소 범위가 있는지에 따라 결정됩니다. 기본 영역은 분류되지 않은 트래픽에 대해 활성 상태이지만 트래픽이 규칙과 일치하지 않는 경우 항상 활성 상태인 것은 아닙니다.

   b. 선택한 영역에 네트워크 인터페이스를 할당합니다.

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

영역에 네트워크 인터페이스를 할당하는 것은 특정 인터페이스(물리적 또는 가상)의 모든 트래픽에 일관된 방화벽 설정을 적용하는 데 더 적합합니다.

firewall-cmd 명령을 **--permanent** 옵션과 함께 사용하는 경우 종종 NetworkManager 연결 프로필을 업데이트하여 방화벽 구성을 영구적으로 변경해야 합니다. **firewalld** 와 NetworkManager 간의 통합은 일관된 네트워크 및 방화벽 설정을 보장합니다.

검증

1. 선택한 영역에 대한 업데이트된 설정을 표시합니다.

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

명령 출력은 할당된 서비스, 네트워크 인터페이스 및 네트워크 연결(소스)을 포함한 모든 영역 설정을 표시합니다.

1.7.2. 기본 영역 변경

시스템 관리자는 해당 구성 파일의 네트워킹 인터페이스에 영역을 할당합니다. 인터페이스가 특정 영역에 할당되지 않은 경우 기본 영역에 할당됩니다. **firewalld** 서비스를 다시 시작할 때마다 **firewalld** 는 기본 영역에 대한 설정을 로드하여 활성화합니다. 다른 모든 영역에 대한 설정은 유지되며 사용할 준비가 되어 있습니다.

일반적으로 영역은 NetworkManager 연결 프로필의 **connection.zone** 설정에 따라 NetworkManager에 의해 인터페이스에 할당됩니다. 또한 재부팅 후 NetworkManager는 해당 영역의 "활성화" 할당을 관리합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

기본 영역을 설정하려면 다음을 수행합니다.

1. 현재 기본 영역을 표시합니다.

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

2. 새 기본 영역을 설정합니다.

```
# firewall-cmd --set-default-zone <zone_name>
```

참고

이 절차에서는 **--permanent** 옵션이 없어도 영구적인 설정입니다.

1.7.3. 영역에 네트워크 인터페이스 할당

다른 영역에 대해 다양한 규칙 세트를 정의한 다음 사용 중인 인터페이스의 영역을 변경하여 설정을 빠르게 변경할 수 있습니다. 여러 인터페이스를 통해 제공되는 트래픽을 구분하기 위해 각각 특정 영역을 설정할 수 있습니다.

절차

영역을 특정 인터페이스에 할당하려면 다음을 수행합니다.

1. 활성 영역 및 할당된 인터페이스를 나열합니다.

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

2. 인터페이스를 다른 영역에 할당합니다.

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

1.7.4. nmcli를 사용하여 영역에 연결 할당

nmcli 유틸리티를 사용하여 **NetworkManager** 연결에 **firewalld** 영역을 추가할 수 있습니다.

절차

1. **NetworkManager** 연결 프로필에 영역을 할당합니다.

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

2. 연결을 활성화합니다.

```
# nmcli connection up profile
```

1.7.5. 연결 프로필 파일에서 네트워크 연결에 수동으로 영역 할당

nmcli 유틸리티를 사용하여 연결 프로필을 수정할 수 없는 경우 프로필의 해당 파일을 수동으로 편집하여 **firewalld** 영역을 할당할 수 있습니다.

참고

firewalld 영역을 할당하도록 **nmcli** 유틸리티를 사용하여 연결 프로필을 수정하는 것이 더 효율적입니다. 자세한 내용은 영역에 네트워크 인터페이스 할당을 참조하십시오.

절차

1. 연결 프로필의 경로와 해당 형식을 결정합니다.

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

NetworkManager는 서로 다른 연결 프로필 형식에 대해 별도의 디렉터리 및 파일 이름을 사용합니다.

- **/etc/NetworkManager/system-connections/ <*connection_name*> .nmconnection** 파일의 프로필은 keyfile 형식을 사용합니다.
- **/etc/sysconfig/network-scripts/ifcfg- <*interface_name*>** 파일의 프로필은 ifcfg 형식을 사용합니다.

2. 형식에 따라 해당 파일을 업데이트합니다.

- 파일이 키 파일 형식을 사용하는 경우 **zone= <*name*>**을 **/etc/NetworkManager/system-connections/ <connection_*name*> *.nmconnection* 파일의 [connection] 섹션에 추가합니다.**

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

- 파일에서 ifcfg 형식을 사용하는 경우 **ZONE= <*name*>**을 **/etc/sysconfig/network-scripts/ifcfg- <*interface_name*>** 파일에 추가합니다.

```
ZONE=internal
```

3. 연결 프로필을 다시 로드합니다.

```
# nmcli connection reload
```

4. 연결 프로필 다시 활성화

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

검증

- 인터페이스 영역을 표시합니다. 예를 들면 다음과 같습니다.

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

1.7.6. 새 영역 생성

사용자 지정 영역을 사용하려면 새 영역을 만들고 사전 정의된 영역처럼 사용합니다. 새 영역에는 **--permanent** 옵션이 필요합니다. 그렇지 않으면 명령이 작동하지 않습니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 새 영역을 생성합니다.

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

2. 새 영역을 사용할 수 있도록 설정합니다.

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

명령은 이미 실행 중인 네트워크 서비스를 중단하지 않고 최근 방화벽 구성에 변경 사항을 적용합니다.

검증

- 새 영역이 영구 설정에 추가되었는지 확인합니다.

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

1.7.7. 웹 콘솔을 사용하여 영역 활성화

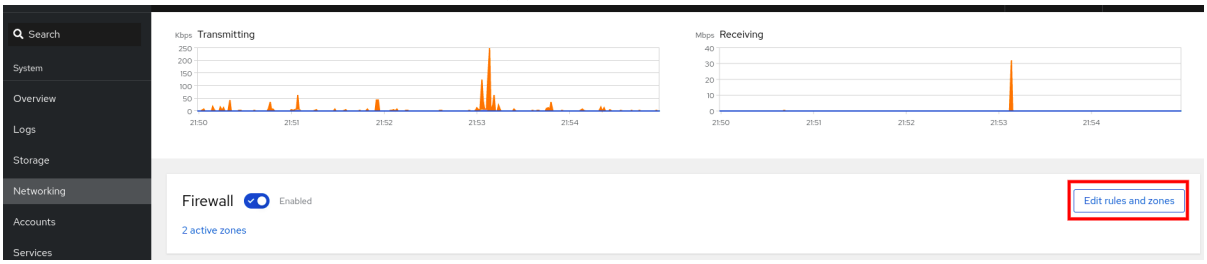
RHEL 웹 콘솔을 통해 특정 인터페이스 또는 IP 주소 범위에 사전 정의된 기존 방화벽 영역을 적용할 수 있습니다.

사전 요구 사항

- RHEL 9 웹 콘솔을 설치했습니다.
  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.

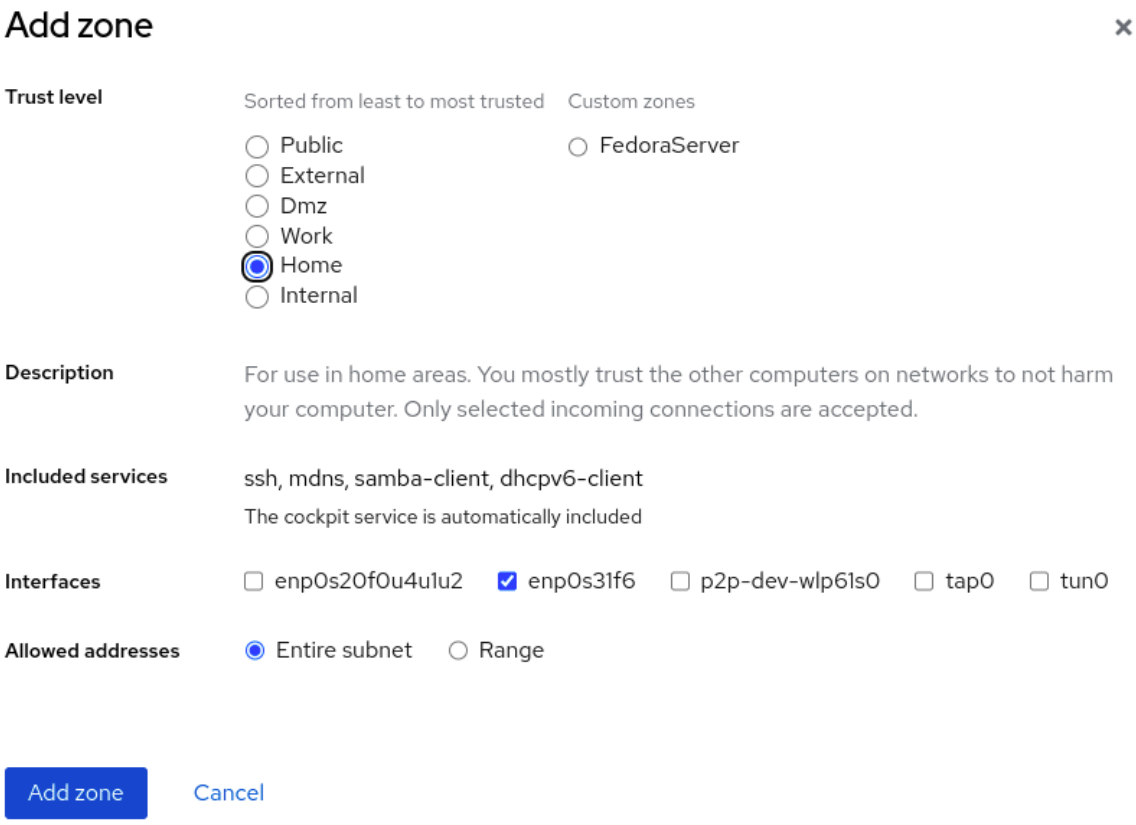
절차

1. RHEL 9 웹 콘솔에 로그인합니다.
   자세한 내용은 웹 콘솔에 로그인 을 참조하십시오.
2. **네트워킹** 을 클릭합니다.
3. **Edit rules and zones** 버튼을 클릭합니다.



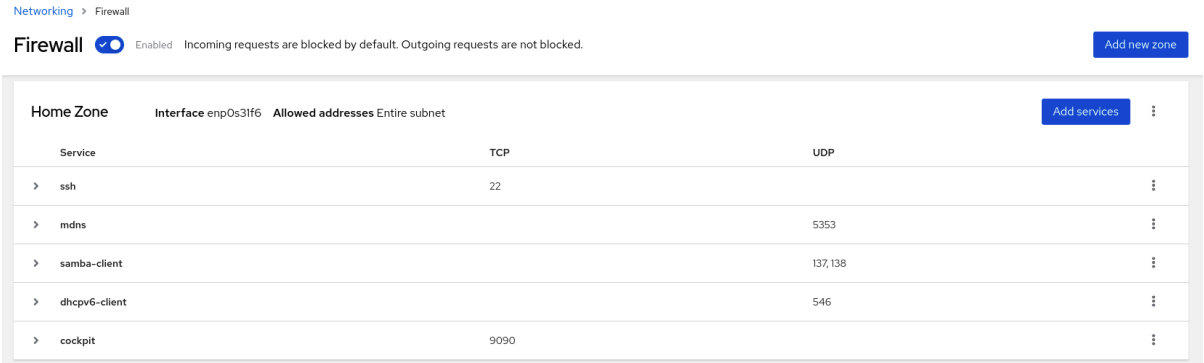
   Edit rules and zones 버튼이 표시되지 않으면 관리자 권한으로 웹 콘솔에 로그인합니다.

4. **방화벽** 섹션에서 **새 영역 추가** 를 클릭합니다.
5. **영역 추가** 대화 상자의 **신뢰 수준** 옵션에서 영역을 선택합니다.
   웹 콘솔은 **firewalld** 서비스에 사전 정의된 모든 영역을 표시합니다.
6. **인터페이스** 부분에서 선택한 영역이 적용되는 인터페이스 또는 인터페이스를 선택합니다.
7. **허용된 주소** 부분에서 영역이 적용되는지 여부를 선택할 수 있습니다.
   - 전체 서브넷
   - 또는 다음 형식의 IP 주소 범위:
     - 192.168.1.0
     - 192.168.1.0/24
     - 192.168.1.0/24, 192.168.1.0
8. **영역 추가** 버튼을 클릭합니다.



검증

- **방화벽** 섹션에서 구성을 확인합니다.



1.7.8. 웹 콘솔을 사용하여 영역 비활성화

웹 콘솔을 사용하여 방화벽 구성에서 방화벽 영역을 비활성화할 수 있습니다.

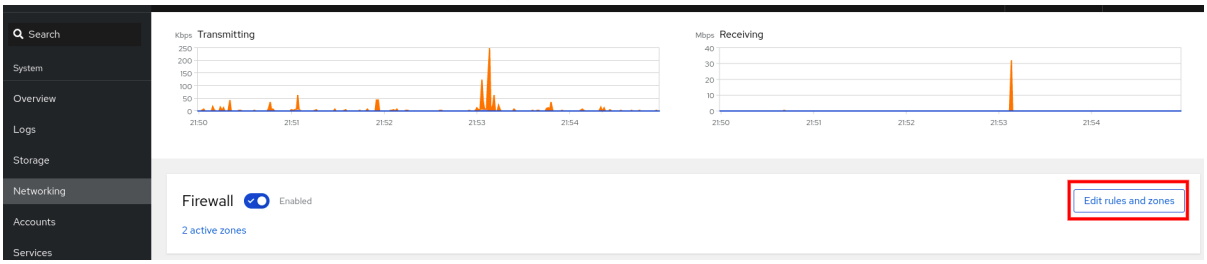
사전 요구 사항

- RHEL 9 웹 콘솔을 설치했습니다.
  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.

절차

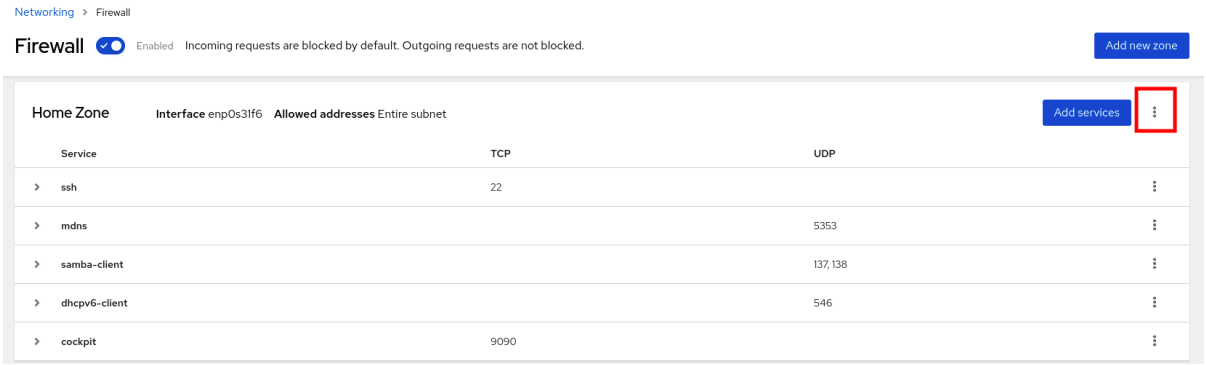
1. RHEL 9 웹 콘솔에 로그인합니다.
   자세한 내용은 웹 콘솔에 로그인 을 참조하십시오.
2. **네트워킹** 을 클릭합니다.

3. **Edit rules and zones** 버튼을 클릭합니다.



Edit rules and zones 버튼이 표시되지 않으면 관리자 권한으로 웹 콘솔에 로그인합니다.

4. 제거하려는 영역에서 **옵션** 아이콘을 클릭합니다.



5. **삭제**를 클릭합니다.

이제 영역이 비활성화되어 인터페이스에 영역에 구성된 열린 서비스 및 포트가 포함되지 않습니다.

1.7.9. 영역 대상을 사용하여 들어오는 트래픽에 대한 기본 동작 설정

모든 영역에서 더 이상 지정되지 않은 들어오는 트래픽을 처리하는 기본 동작을 설정할 수 있습니다. 이러한 동작은 영역의 대상을 설정하여 정의됩니다. 네 가지 옵션이 있습니다.

- **ACCEPT**: 특정 규칙에 의해 허용되지 않는 패킷을 제외하고 들어오는 모든 패킷을 허용합니다.
- **REJECT**: 특정 규칙에서 허용하는 패킷을 제외한 들어오는 모든 패킷을 거부합니다. **firewalld** 가 패킷을 거부하면 소스 시스템에 거부에 대한 정보가 표시됩니다.
- **DROP**: 특정 규칙에서 허용하는 패킷을 제외하고 들어오는 모든 패킷을 삭제합니다. **firewalld** 가 패킷을 삭제하면 소스 시스템에 패킷 삭제에 대한 정보가 표시되지 않습니다.
- **기본값**: **REJECT** 와 유사한 동작이지만 특정 시나리오에서 특별한 의미가 있습니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

영역의 대상을 설정하려면 다음을 수행합니다.

1. 기본 대상을 보려면 특정 영역에 대한 정보를 나열합니다.

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

2. 영역에 새 대상을 설정합니다.

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

추가 리소스

- 시스템의 **firewall-cmd(1)** 도움말 페이지

1.8. FIREWALLD를 사용하여 네트워크 트래픽 제어

firewalld 패키지는 많은 수의 사전 정의된 서비스 파일을 설치하고 더 많은 서비스를 추가하거나 사용자 지정할 수 있습니다. 그런 다음 이러한 서비스 정의를 사용하여 사용하는 프로토콜과 포트 번호를 몰라도 서비스에 대한 포트를 열거나 닫을 수 있습니다.

1.8.1. CLI를 사용하여 사전 정의된 서비스로 트래픽 제어

트래픽을 제어하는 가장 간단한 방법은 사전 정의된 서비스를 **firewalld** 에 추가하는 것입니다. 이렇게 하면 필요한 모든 포트를 열고 *서비스 정의 파일에* 따라 다른 설정을 수정합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. **firewalld** 의 서비스가 아직 허용되지 않았는지 확인합니다.

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

명령은 기본 영역에서 활성화된 서비스를 나열합니다.

2. **firewalld** 에서 사전 정의된 모든 서비스를 나열합니다.

```
# 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 ...
```

명령은 기본 영역에 사용 가능한 서비스 목록을 표시합니다.

3. **firewalld** 에서 허용하는 서비스 목록에 서비스를 추가합니다.

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

명령은 지정된 서비스를 기본 영역에 추가합니다.

4. 새 설정을 영구적으로 만듭니다.

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

명령은 이러한 런타임 변경 사항을 방화벽의 영구 구성에 적용합니다. 기본적으로 이러한 변경 사항은 기본 영역의 구성에 적용됩니다.

검증

1. 모든 영구 방화벽 규칙을 나열합니다.

```
# 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:
```

명령은 기본 방화벽 영역(**공용**)의 영구 방화벽 규칙을 사용하여 전체 구성을 표시합니다.

2. **firewalld** 서비스의 영구 구성의 유효성을 확인합니다.

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

영구 구성이 유효하지 않으면 명령에서 추가 세부 정보와 함께 오류를 반환합니다.

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

영구 구성 파일을 수동으로 검사하여 설정을 확인할 수도 있습니다. 기본 설정 파일은 **/etc/firewalld/firewalld.conf** 입니다. 영역별 구성 파일은 **/etc/firewalld/zones/** 디렉터리에 있으며 정책은 **/etc/firewalld/policies/** 디렉터리에 있습니다.

1.8.2. GUI를 사용하여 사전 정의된 서비스로 트래픽 제어

그래픽 사용자 인터페이스를 사용하여 사전 정의된 서비스로 네트워크 트래픽을 제어할 수 있습니다. 방화벽 구성 애플리케이션은 명령줄 유틸리티에 대한 액세스 가능하고 사용자에게 친숙한 대안을 제공합니다.

사전 요구 사항

- **firewall-config** 패키지를 설치했습니다.
- **firewalld** 서비스가 실행 중입니다.

절차

1. 사전 정의 또는 사용자 지정 서비스를 활성화하거나 비활성화하려면 다음을 수행합니다.
   a. **firewall-config** 유틸리티를 시작하고 서비스를 구성할 네트워크 영역을 선택합니다.
   b. **영역** 탭을 선택한 다음 아래의 **서비스** 탭을 선택합니다.

c. 신뢰할 각 서비스 유형에 대한 확인란을 선택하거나 선택한 영역에서 서비스를 차단하는 확인란을 지웁니다.

2. 서비스를 편집하려면 다음을 수행합니다.

a. **firewall-config** 유틸리티를 시작합니다.

b. **Configuration** (구성)이라는 레이블이 지정된 메뉴에서 **Permanent** (영구)를 선택합니다. **서비스** 창 하단에 추가 아이콘 및 메뉴 버튼이 나타납니다.

c. 구성할 서비스를 선택합니다.

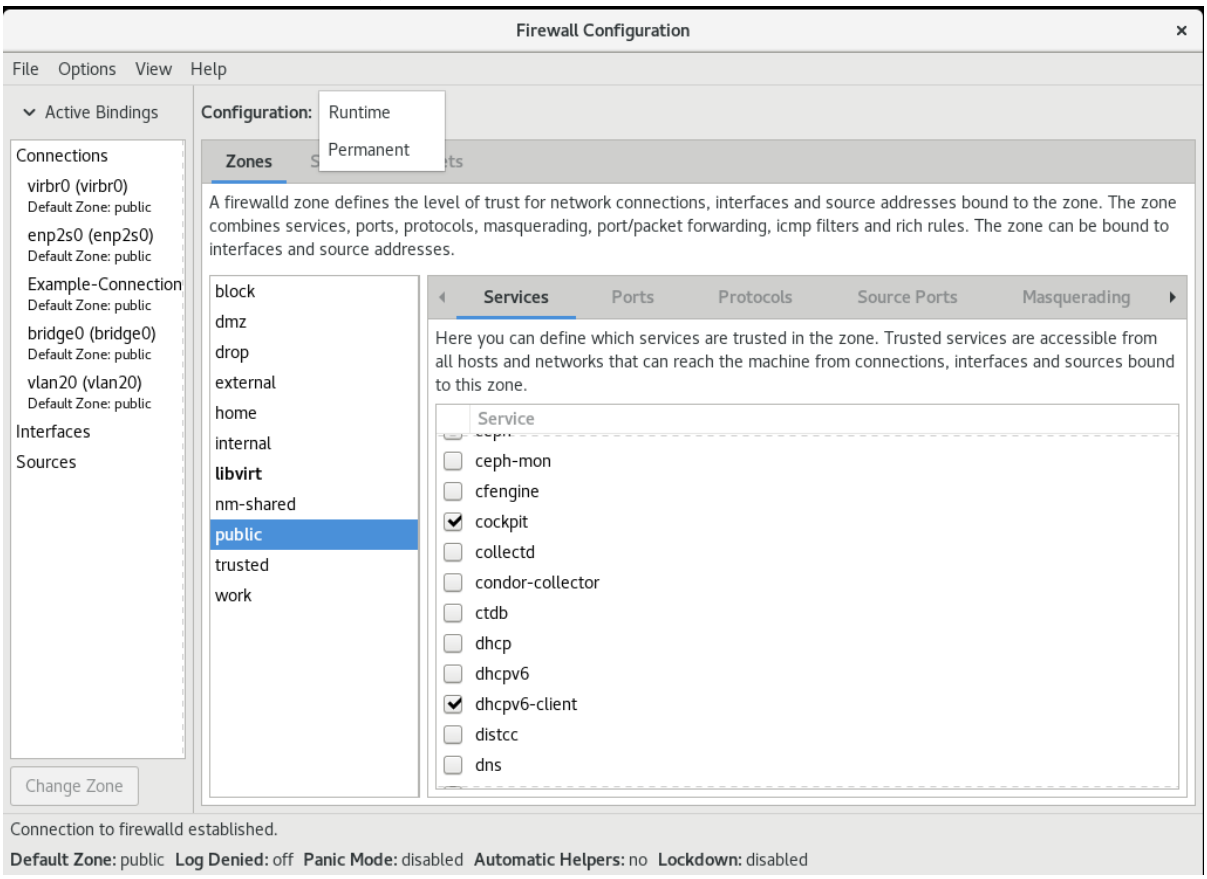
Ports,**Protocols**, **Source Port** 탭을 사용하면 선택한 서비스에 대한 포트, 프로토콜 및 소스 포트를 추가, 변경 및 제거할 수 있습니다. 모듈 탭은 **Netfilter** 도우미 모듈을 구성하는 데 사용됩니다. **대상** 탭에서는 트래픽을 특정 대상 주소 및 인터넷 프로토콜(**IPv4** 또는 **IPv6**)으로 제한할 수 있습니다.

참고

런타임 모드에서는 서비스 설정을 변경할 수 없습니다.

검증

- **Super** 키를 눌러 활동 개요를 입력합니다.
- Firewall Configuration 유틸리티를 선택합니다.
  - **firewall-config** 명령을 입력하여 명령줄을 사용하여 그래픽 방화벽 구성 유틸리티를 시작할 수도 있습니다.
- 방화벽 구성 목록을 확인합니다.

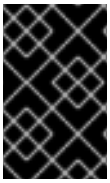


방화벽 구성 창이 열립니다. 이 명령은 일반 사용자로 실행할 수 있지만 때때로 관리자 암호를 입력하라는 메시지가 표시됩니다.

1.8.3. 웹 콘솔을 사용하여 방화벽에서 서비스 활성화

기본적으로 서비스는 기본 방화벽 영역에 추가됩니다. 더 많은 네트워크 인터페이스에서 방화벽 영역을 사용하는 경우 먼저 영역을 선택한 다음 포트로 서비스를 추가해야 합니다.

RHEL 9 웹 콘솔에는 사전 정의된 **firewalld** 서비스가 표시되고 활성 방화벽 영역에 추가할 수 있습니다.



중요

RHEL 9 웹 콘솔은 **firewalld** 서비스를 구성합니다.

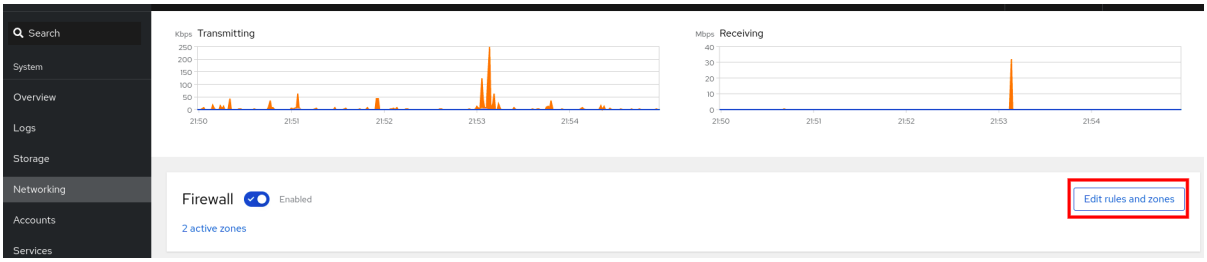
웹 콘솔은 웹 콘솔에 나열되지 않은 일반 **firewalld** 규칙을 허용하지 않습니다.

사전 요구 사항

- RHEL 9 웹 콘솔을 설치했습니다.
  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.

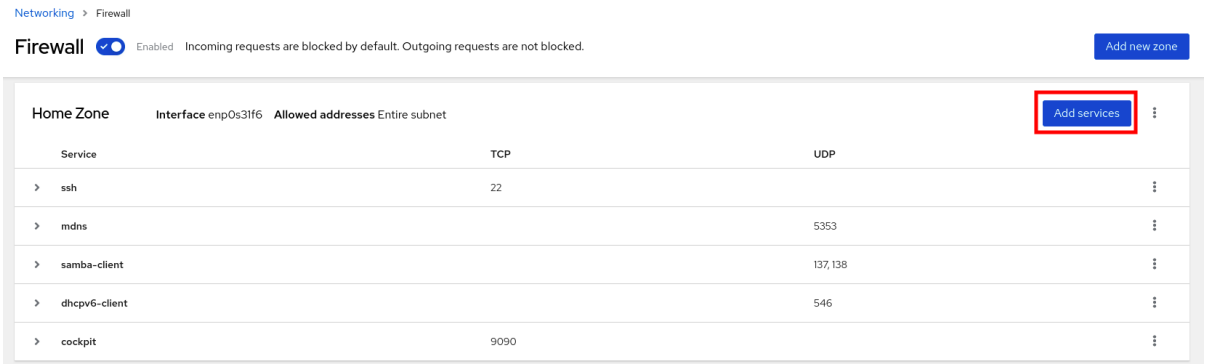
절차

1. RHEL 9 웹 콘솔에 로그인합니다.
   자세한 내용은 웹 콘솔에 로그인 을 참조하십시오.
2. **네트워킹** 을 클릭합니다.
3. **Edit rules and zones** 버튼을 클릭합니다.

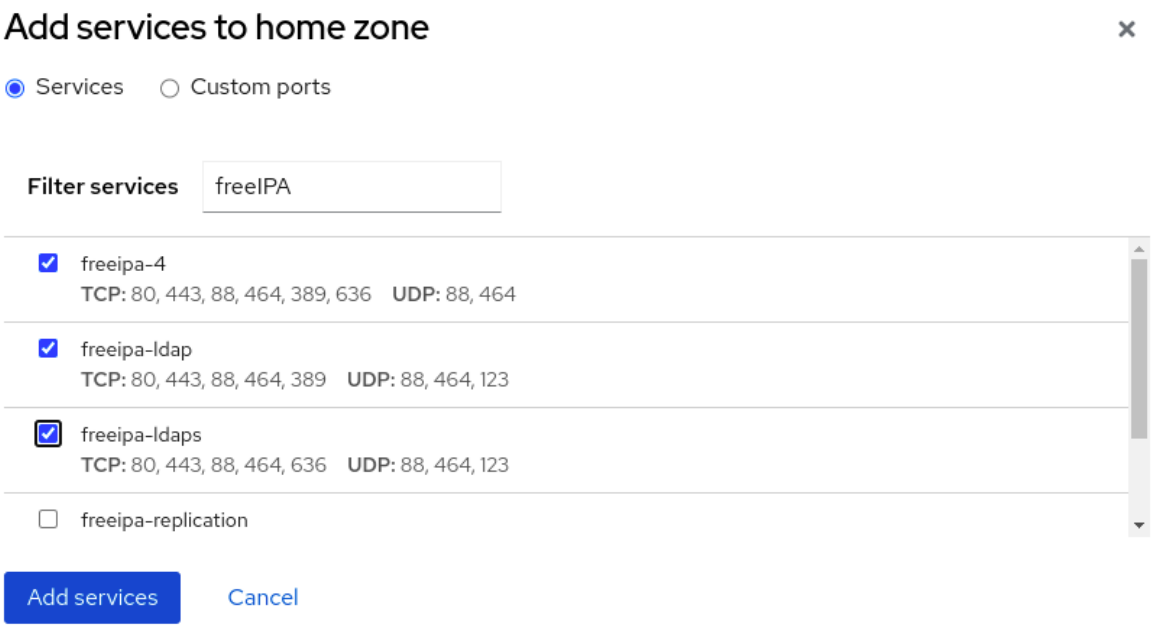


Edit rules and zones 버튼이 표시되지 않으면 관리자 권한으로 웹 콘솔에 로그인합니다.

4. **방화벽** 섹션에서 서비스를 추가할 영역을 선택하고 **서비스 추가** 를 클릭합니다.



5. **서비스 추가** 대화 상자에서 방화벽에서 활성화할 서비스를 찾습니다.
6. 시나리오에 따라 서비스를 활성화합니다.



7. **서비스 추가**를 클릭합니다.

이때 RHEL 9 웹 콘솔에 서비스가 영역의 서비스 목록에 표시됩니다.

1.8.4. 웹 콘솔을 사용하여 사용자 정의 포트 구성

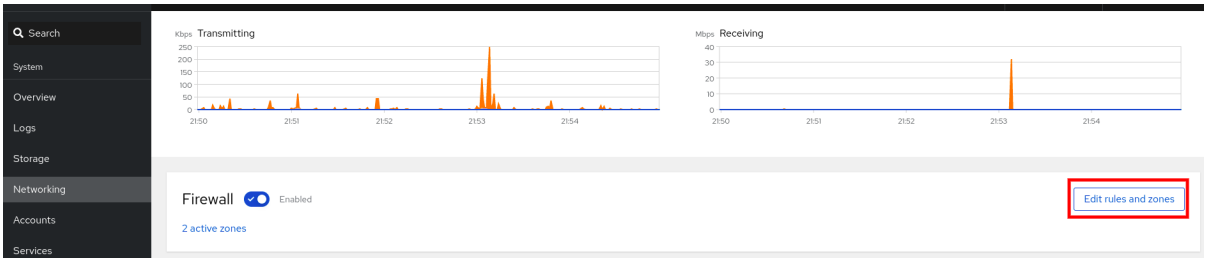
RHEL 웹 콘솔을 통해 서비스에 대한 사용자 지정 포트를 구성할 수 있습니다.

사전 요구 사항

- **RHEL 9 웹 콘솔을 설치했습니다.**
  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.
- **firewalld 서비스가 실행 중입니다.**

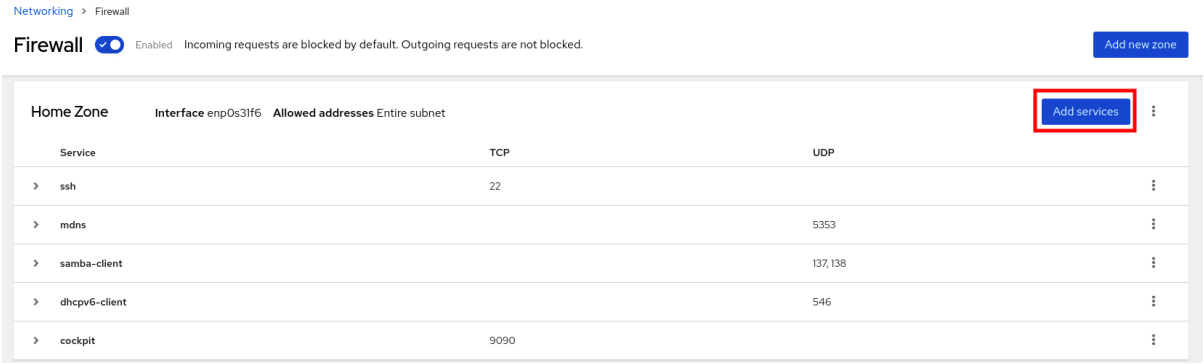
절차

1. **RHEL 9 웹 콘솔에 로그인합니다.**
   자세한 내용은 웹 콘솔에 로그인 을 참조하십시오.
2. **네트워킹 을 클릭합니다.**
3. **Edit rules and zones 버튼을 클릭합니다.**



Edit rules and zones 버튼이 표시되지 않으면 관리 권한으로 웹 콘솔에 로그인합니다.

4. **방화벽 섹션에서 사용자 지정 포트를 구성할 영역을 선택하고서비스 추가 를 클릭합니다.**

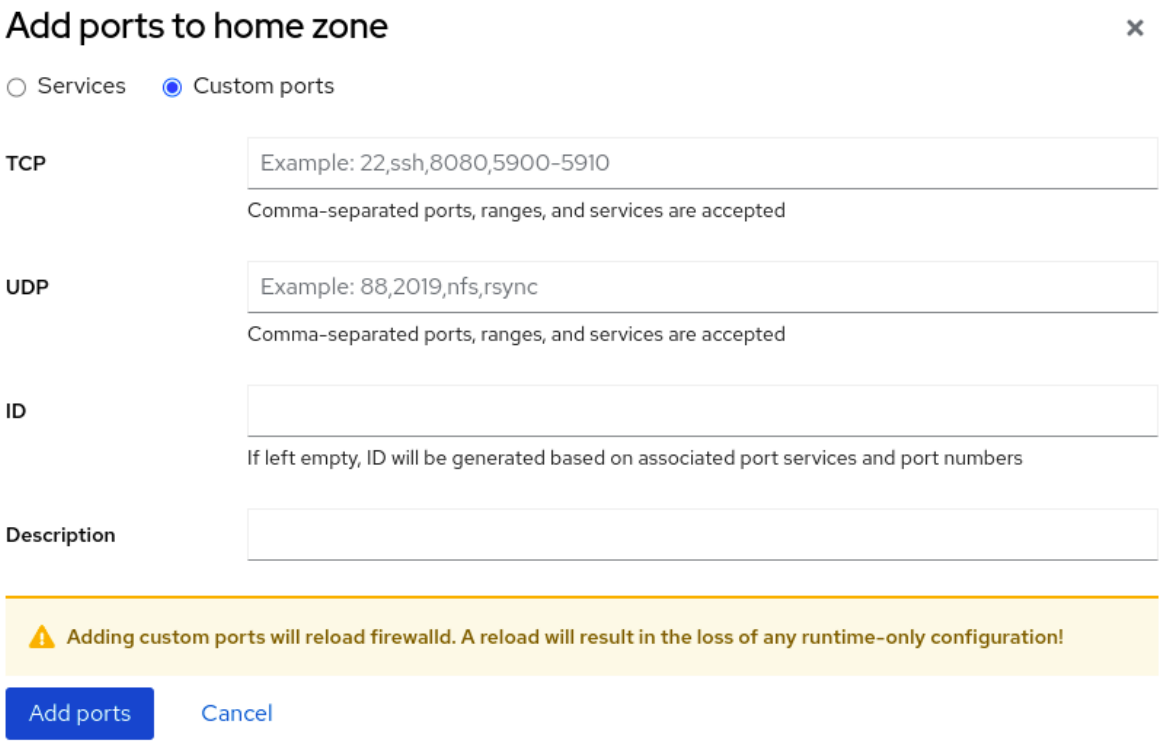


5. 서비스 추가 대화 상자에서 사용자 지정 포트 라디오 버튼을 클릭합니다.
6. TCP 및 UDP 필드에서 예제에 따라 포트를 추가합니다. 다음 형식으로 포트를 추가할 수 있습니다.
   - 22와 같은 포트 번호
   - 5900-5910과 같은 포트 번호 범위
   - nfs, rsync와 같은 별칭

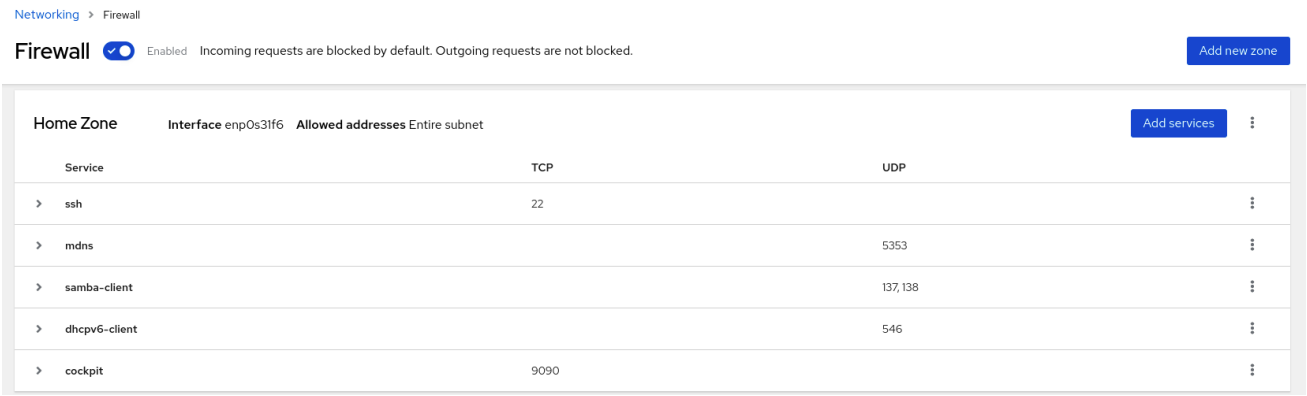
참고

각 필드에 여러 값을 추가할 수 있습니다. 값은 쉼표 없이 쉼표로 구분해야 합니다. 예를 들면 다음과 같습니다. 8080,8081,http

7. TCP filed에 포트 번호를 추가한 후,UDP 가 제출되었거나 둘 다되면Name 필드에서 서비스 이름을 확인합니다.
   Name 필드에는 이 포트가 예약된 서비스 이름이 표시됩니다. 이 포트를 자유롭게 사용할 수 있고 이 포트에서 서버가 통신할 필요가 없는 경우 이름을 다시 작성할 수 있습니다.
8. 이름 필드에 정의된 포트를 포함한 서비스의 이름을 추가합니다.
9. 포트 추가 버튼을 클릭합니다.



설정을 확인하려면 방화벽 페이지로 이동하여 영역의 서비스 목록에서 서비스를 찾습니다.



1.8.5. 보안 웹 서버 호스팅을 허용하도록 firewalld 구성

포트는 운영 체제가 네트워크 트래픽을 수신 및 구분하고 시스템 서비스로 전달할 수 있는 논리 서비스입니다. 시스템 서비스는 포트에서 수신 대기하고 이 포트로 들어오는 모든 트래픽을 대기하는 데몬으로 표시됩니다.

일반적으로 시스템 서비스는 해당 서비스를 위해 예약된 표준 포트에서 수신 대기합니다. 예를 들어 **httpd** 데몬은 포트 80에서 수신 대기합니다. 그러나 시스템 관리자는 서비스 이름 대신 포트 번호를 직접 지정할 수 있습니다.

firewalld 서비스를 사용하여 데이터를 호스팅하기 위해 보안 웹 서버에 대한 액세스를 구성할 수 있습니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 현재 활성화된 방화벽 영역을 확인합니다.

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

2. 적절한 영역에 HTTPS 서비스를 추가합니다.

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

3. 방화벽 구성을 다시 로드합니다.

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

검증

1. **firewalld** 에서 포트가 열려 있는지 확인합니다.
   - 포트 번호를 지정하여 포트를 연 경우 다음을 입력합니다.

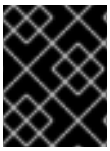
```
# firewall-cmd --zone=<zone_name> --list-all
```

   - 서비스 정의를 지정하여 포트를 연 경우 다음을 입력합니다.

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

1.8.6. 네트워크 보안을 강화하기 위해 사용되지 않거나 불필요한 포트 종료

열려 있는 포트가 더 이상 필요하지 않으면 **firewalld** 유틸리티를 사용하여 이를 종료할 수 있습니다.



중요

불필요한 모든 포트를 닫고 잠재적인 공격 면적을 줄이고 무단 액세스 또는 취약점 악용 위험을 최소화합니다.

절차

1. 허용되는 모든 포트를 나열합니다.

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

기본적으로 이 명령은 기본 영역에서 활성화된 포트를 나열합니다.

참고

이 명령은 포트로 열려 있는 포트 목록만 제공합니다. 서비스로 열려 있는 열려 있는 포트는 볼 수 없습니다. 이 경우 **--list-ports** 대신 **--list-all** 옵션을 사용하는 것이 좋습니다.

2. 허용되는 포트 목록에서 포트를 제거하여 들어오는 트래픽에 대해 종료합니다.

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

이 명령은 영역에서 포트를 제거합니다. 영역을 지정하지 않으면 기본 영역에서 포트가 제거됩니다.

3. 새 설정을 영구적으로 만듭니다.

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

영역을 지정하지 않으면 이 명령은 런타임 변경 사항을 기본 영역의 영구 구성에 적용합니다.

검증

1. 활성 영역을 나열하고 검사할 영역을 선택합니다.

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

2. 선택한 영역에서 현재 열려 있는 포트를 나열하여 사용되지 않거나 불필요한 포트가 닫혀 있는지 확인합니다.

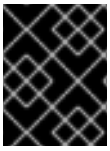
```
# firewall-cmd --zone=<zone_to_inspect> --list-ports
```

1.8.7. CLI를 통한 트래픽 제어

firewall-cmd 명령을 사용하여 다음을 수행할 수 있습니다.

- 네트워킹 트래픽 비활성화
- 네트워킹 트래픽 활성화

예를 들어 시스템 방어 기능을 강화하거나 데이터 개인 정보를 보장하거나 네트워크 리소스를 최적화할 수 있습니다.



중요

panic 모드를 활성화하면 모든 네트워킹 트래픽이 중지됩니다. 이러한 이유로 시스템에 대한 물리적 액세스 권한이 있거나 직렬 콘솔을 사용하여 로그인한 경우에만 사용해야 합니다.

절차

1. 네트워킹 트래픽을 즉시 비활성화하려면 다음에서 패닉 모드를 전환합니다.

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

2. panic 모드를 전환하면 방화벽을 영구 설정으로 되돌립니다. 패닉 모드를 전환하려면 다음을 입력합니다.

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

검증

- panic 모드가 켜거나 꺼졌는지 확인하려면 다음을 사용합니다.

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

1.8.8. GUI를 사용하여 프로토콜로 트래픽 제어

특정 프로토콜을 사용하여 방화벽을 통한 트래픽을 허용하려면 GUI를 사용할 수 있습니다.

사전 요구 사항

- **firewall-config** 패키지 설치

절차

1. firewall-config 도구를 시작하고 변경할 설정이 있는 네트워크 영역을 선택합니다.
2. 프로토콜 탭을 선택하고 오른쪽에 있는추가 버튼을 클릭합니다. 프로토콜 창이 열립니다.
3. 목록에서 프로토콜을 선택하거나 기타 프로토콜 확인란을 선택하고 필드에 프로토콜을 입력합니다.

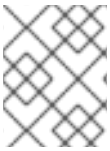
1.9. 영역을 사용하여 소스에 따라 들어오는 트래픽 관리

영역을 사용하여 소스에 따라 들어오는 트래픽을 관리할 수 있습니다. 이 컨텍스트에서 들어오는 트래픽은 시스템을 대상으로 하거나 **firewalld** 를 실행하는 호스트를 통과하는 모든 데이터입니다. 소스는 일반적으로 트래픽이 시작된 IP 주소 또는 네트워크 범위를 나타냅니다. 결과적으로 들어오는 트래픽을 정렬하고 다른 영역에 할당하여 해당 트래픽으로 연결할 수 있는 서비스를 허용하거나 허용하지 않을 수 있습니다.

소스 주소의 일치는 인터페이스 이름별 일치보다 우선합니다. 영역에 소스를 추가하면 방화벽은 인터페이스 기반 규칙을 통한 들어오는 트래픽에 대한 소스 기반 규칙의 우선 순위를 지정합니다. 즉, 들어오는 트래픽이 특정 영역에 지정된 소스 주소와 일치하는 경우 해당 소스 주소와 연결된 영역에 도달하는 인터페이스에 관계없이 트래픽이 처리되는 방법이 결정됩니다. 반면 인터페이스 기반 규칙은 일반적으로 특정 소스 기반 규칙과 일치하지 않는 트래픽에 대한 폴백입니다. 이러한 규칙은 소스가 영역과 명시적으로 연결되어 있지 않은 트래픽에 적용됩니다. 이를 통해 특정 소스 정의 영역이 없는 트래픽에 대한 기본 동작을 정의할 수 있습니다.

1.9.1. 소스 추가

들어오는 트래픽을 특정 영역으로 라우팅하려면 소스를 해당 영역에 추가합니다. 소스는 CIDR(Classless inter-domain routing) 표기법의 IP 주소 또는 IP 마스크일 수 있습니다.



참고

겹치는 네트워크 범위를 사용하여 여러 영역을 추가하는 경우 영역 이름으로 영숫자로 정렬되며 첫 번째 영역만 고려합니다.

- 현재 영역에서 소스를 설정하려면 다음을 수행합니다.

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

- 특정 영역의 소스 IP 주소를 설정하려면 다음을 수행합니다.

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

다음 절차에서는 **trusted** 영역에서의 모든 수신 트래픽을 허용합니다.

절차

1. 사용 가능한 모든 영역을 나열합니다.

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

2. 영구 모드의 신뢰할 수 있는 영역에 소스 IP를 추가합니다.

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

3. 새 설정을 영구적으로 만듭니다.

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

1.9.2. 소스 제거

영역에서 소스를 제거하면 소스에서 시작된 트래픽은 더 이상 해당 소스에 지정된 규칙을 통해 전달되지 않습니다. 대신 트래픽이 시작된 인터페이스와 연결된 영역의 규칙 및 설정으로 대체되거나 기본 영역으로 이동합니다.

절차

1. 필수 영역에 허용되는 소스를 나열합니다.

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

2. 영역에서 소스를 영구적으로 제거합니다.

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

3. 새 설정을 영구적으로 만듭니다.

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

1.9.3. 소스 포트 제거

소스 포트를 제거하면 원본 포트에 따라 트래픽 정렬을 비활성화합니다.

절차

- 소스 포트를 제거하려면 다음을 수행합니다.

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

1.9.4. 영역 및 소스를 사용하여 특정 도메인에만 서비스 허용

특정 네트워크의 트래픽이 시스템에서 서비스를 사용하도록 허용하려면 영역 및 소스를 사용합니다. 다음 절차에서는 192.0.2.0/24 네트워크의 HTTP 트래픽만 허용하고 다른 모든 트래픽은 차단됩니다.

주의

이 시나리오를 구성할 때 기본 대상이 있는 영역을 사용합니다. 대상이ACCEPT 로 설정된 영역을 사용하는 것은 192.0.2.0/24 의 트래픽의 경우 모든 네트워크 연결이 허용되기 때문에 보안 위험입니다.

절차

1. 사용 가능한 모든 영역을 나열합니다.

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

2. IP 범위를 내부 영역에 추가하여 소스에서 영역을 통해 발생하는 트래픽을 라우팅합니다.

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

3. 내부 영역에 http 서비스를 추가합니다.

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

4. 새 설정을 영구적으로 만듭니다.

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

검증

- 내부 영역이 활성화되어 있고 서비스가 허용되는지 확인합니다.

```
# 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
  ...
```

추가 리소스

- 시스템의 firewalld.zones(5) 도움말 페이지

1.10. 영역 간 전달된 트래픽 필터링

firewalld 를 사용하면 서로 다른firewalld 영역 간의 네트워크 데이터 흐름을 제어할 수 있습니다. 규칙과 정책을 정의하면 이러한 영역 간에 이동할 때 트래픽이 허용되거나 차단되는 방법을 관리할 수 있습니다.

정책 오브젝트 기능은 **firewalld** 에서 전달 및 출력 필터링 기능을 제공합니다.**firewalld** 를 사용하여 다른 영역 간 트래픽을 필터링하여 로컬 호스트 VM에 대한 액세스를 통해 호스트를 연결할 수 있습니다.

1.10.1. 정책 오브젝트와 영역 간의 관계

정책 오브젝트를 사용하면 사용자가 서비스, 포트 및 리치 규칙과 같은 firewalld 기본 기능을 정책에 연결할 수 있습니다. 정책 오브젝트를 상태 저장 및 단방향 방식으로 영역 간에 전달하는 트래픽에 적용할 수 있습니다.

```
# 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 및 **ALL** 은 수신 및 송신 영역 목록에 사용되는 심볼릭 영역입니다.

- **HOST**(호스트) 심볼릭 영역을 사용하면 firewalld를 실행하는 호스트의 대상이거나 에서 시작된 트래픽에 대한 정책을 사용할 수 있습니다.
- 모든 심볼 릭 영역은 모든 현재 및 향후 영역에 정책을 적용합니다모든 심볼릭 영역은 모든 영역에 대해 와일드카드 역할을 합니다.

1.10.2. 우선순위를 사용하여 정책 정렬

여러 정책이 동일한 트래픽 집합에 적용할 수 있으므로 적용할 수 있는 정책에 대한 우선 순위 순서를 생성하려면 우선 순위를 사용해야 합니다.

정책을 정렬하는 우선 순위를 설정하려면 다음을 수행합니다.

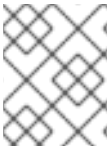
```
# firewall-cmd --permanent --policy mypolicy --set-priority -500
```

위의 예에서 *-500* 은 우선 순위가 낮지만 우선 순위가 높습니다. 따라서 -500은 -100 전에 실행됩니다.

낮은 숫자 우선순위 값은 우선순위가 높고 먼저 적용됩니다.

1.10.3. 정책 오브젝트를 사용하여 로컬 호스트 컨테이너와 호스트에 물리적으로 연결된 네트워크 간의 트래픽을 필터링

정책 오브젝트 기능을 사용하면 사용자가 Podman과 firewalld 영역 간의 트래픽을 필터링할 수 있습니다.



참고

Red Hat은 기본적으로 모든 트래픽을 차단하고 Podman 유틸리티에 필요한 선택적 서비스를 여는 것이 좋습니다.

절차

1. 새 방화벽 정책을 생성합니다.

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

2. Podman에서 다른 영역으로의 모든 트래픽을 차단하고 Podman에서 필요한 서비스만 허용합니다.

```
# 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. 새 Podman 영역을 생성합니다.

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

4. 정책의 수신 영역을 정의합니다.

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

5. 다른 모든 영역에 대한 송신 영역을 정의합니다.

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

송신 영역을 ANY로 설정하면 Podman에서 다른 영역으로 필터링합니다. 호스트에 필터링하려면 송신 영역을 HOST로 설정합니다.

6. firewalld 서비스를 다시 시작합니다.

```
# systemctl restart firewalld
```

검증

- Podman 방화벽 정책을 다른 영역에 확인합니다.

```
# firewall-cmd --info-policy podmanToAny
podmanToAny (active)
  ...
  target: REJECT
  ingress-zones: podman
  egress-zones: ANY
  services: dhcp dns https
  ...
```

1.10.4. 정책 오브젝트의 기본 대상 설정

정책에 대해 --set-target 옵션을 지정할 수 있습니다. 다음 대상을 사용할 수 있습니다.

- **ACCEPT** - 패킷을 수락
- **DROP** - 원하지 않는 패킷을 삭제합니다.
- **REJECT** - ICMP 응답을 사용하여 원하지 않는 패킷을 거부
- **CONTINUE** (기본값) - 패킷에는 다음 정책 및 영역의 규칙이 적용됩니다.

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

검증

- 정책에 대한 정보 확인

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

1.10.5. DNAT를 사용하여 HTTPS 트래픽을 다른 호스트로 전달

웹 서버가 개인 IP 주소가 있는 DMZ에서 실행되는 경우 인터넷의 클라이언트가 이 웹 서버에 연결할 수 있도록 대상 네트워크 주소 변환(DNAT)을 구성할 수 있습니다. 이 경우 웹 서버의 호스트 이름이 라우터의 공용 IP 주소로 확인됩니다. 클라이언트에서 라우터의 정의된 포트에 대한 연결을 설정하면 라우터가 패킷을 내부 웹 서버로 전달합니다.

사전 요구 사항

- DNS 서버는 웹 서버의 호스트 이름을 라우터의 IP 주소로 확인합니다.
- 다음 설정을 알고 있습니다.
  - 전달할 개인 IP 주소 및 포트 번호입니다.
  - 사용할 IP 프로토콜
  - 패킷을 리디렉션할 웹 서버의 대상 IP 주소 및 포트

절차

1. 방화벽 정책을 생성합니다.

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

영역과 달리 정책은 입력, 출력 및 전달된 트래픽에 대한 패킷 필터링을 허용합니다. 로컬에서 웹 서버, 컨테이너 또는 가상 머신에서 끝점으로 트래픽을 전달하려면 이러한 기능이 필요하므로 중요합니다.

2. 라우터 자체에서 로컬 IP 주소에 연결하고 이 트래픽을 전달할 수 있도록 수신 및 송신 트래픽에 대한 심볼릭 영역을 구성합니다.

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

--add-ingress-zone=HOST 옵션은 로컬에서 생성되어 로컬 호스트에서 전송된 패킷을 나타냅니다. **--add-egress-zone=ANY** 옵션은 모든 영역으로 이동하는 트래픽을 나타냅니다.

3. 트래픽을 웹 서버로 전달하는 리치 규칙을 추가합니다.

```
# 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"'
```

리치 규칙은 라우터의 IP 주소에 있는 포트 443에서 웹 서버(192.51.100.20)의 IP 주소의 포트 443으로 TCP 트래픽을 전달합니다.

4. 방화벽 구성 파일을 다시 로드합니다.

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

5. 커널에서 127.0.0.0/8의 라우팅을 활성화합니다.

   - 영구 변경 사항의 경우 다음을 실행합니다.

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

     이 명령은 **route_localnet** 커널 매개 변수를 영구적으로 구성하고 시스템이 재부팅된 후 설정이 보존되도록 합니다.

   - 시스템 재부팅없이 설정을 즉시 적용하려면 다음을 실행합니다.

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

     sysctl 명령은 변경 사항을 적용하는 데 유용하지만 시스템 재부팅 시 구성은 유지되지 않습니다.

검증

1. 라우터의 IP 주소 및 웹 서버로 전달된 포트에 연결합니다.

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

2. 선택 사항: **net.ipv4.conf.all.route_localnet** 커널 매개변수가 활성화되어 있는지 확인합니다.

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

3. 가 **<example_policy>** 활성화되어 있고 필요한 설정, 특히 소스 IP 주소와 포트, 사용할 프로토콜, 대상 IP 주소와 포트가 포함되어 있는지 확인합니다.

   ```
   # 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"
   ```

추가 리소스

- **firewall-cmd(1), firewalld.policies(5), firewalld.rich language(5), sysctl(8), sysctl.conf(5)** 도움말 페이지
- /etc/sysctl.d/의 설정 파일을 사용하여 커널 매개변수 조정

1.11. FIREWALLD를 사용하여 NAT 구성

firewalld 를 사용하면 다음 NAT(네트워크 주소 변환) 유형을 구성할 수 있습니다.

- 마스커레이딩
- 대상 NAT(DNAT)
- redirect

1.11.1. 네트워크 주소 변환 유형

다음은 다양한 NAT(네트워크 주소 변환) 유형입니다.

마스커레이딩

이러한 NAT 유형 중 하나를 사용하여 패킷의 소스 IP 주소를 변경합니다. 예를 들어, 인터넷 서비스 공급자(ISP)는 **10.0.0.0/8** 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하고 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 이러한 범위의 패킷의 소스 IP 주소를 공용 IP 주소에 매핑합니다.
마스커레이딩은 발신 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 발신 인터페이스가 동적 IP 주소를 사용하는 경우 masquerading을 사용합니다.

대상 NAT(DNAT)

이 NAT 유형을 사용하여 수신 패킷의 대상 주소와 포트를 다시 작성합니다. 예를 들어 웹 서버가 개인 IP 범위의 IP 주소를 사용하므로 인터넷에서 직접 액세스할 수 없는 경우 라우터에 DNAT 규칙을 설정하여 수신 트래픽을 이 서버로 리디렉션할 수 있습니다.

redirect

이 유형은 패킷을 로컬 시스템의 다른 포트로 리디렉션하는 특수한 DNAT의 경우입니다. 예를 들어 서비스가 표준 포트와 다른 포트에서 실행되는 경우 표준 포트에서 들어오는 트래픽을 이 특정 포트로 리디렉션할 수 있습니다.

1.11.2. IP 주소 마스커레이딩 구성

시스템에서 IP 마스커레이딩을 활성화할 수 있습니다. IP 마스커레이딩은 인터넷에 액세스할 때 게이트웨이 뒤에 있는 개별 머신을 숨깁니다.

절차

1. IP 마스커레이딩이 활성화되어 있는지 확인하려면 (예: 외부 영역의 경우) 다음 명령을 **root** 로 입력합니다.

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

이 명령은 활성화된 경우 종료 상태 **0** 으로 **yes** 를 출력합니다. 종료 상태 **1** 이 없으면 **no** 가 출력됩니다. **zone** 이 생략되면 기본 영역이 사용됩니다.

2. IP 마스커레이딩을 활성화하려면 **root** 로 다음 명령을 입력합니다.

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

3. 이 설정을 영구적으로 설정하려면 명령에 **--permanent** 옵션을 전달합니다.

4. IP 마스커레이딩을 비활성화하려면 **root** 로 다음 명령을 입력합니다.

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

이 설정을 영구적으로 만들려면 **--permanent** 옵션을 명령에 전달합니다.

1.11.3. DNAT를 사용하여 들어오는 HTTP 트래픽 전달

대상 네트워크 주소 변환(DNAT)을 사용하여 들어오는 트래픽을 하나의 대상 주소 및 포트에서 다른 대상 주소로 보낼 수 있습니다. 일반적으로 외부 네트워크 인터페이스에서 특정 내부 서버 또는 서비스로 들어오는 요청을 리디렉션하는 데 유용합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/sysctl.d/90-enable-IP-forwarding.conf** 파일을 생성합니다.

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

이 설정은 커널에서 IP 전달을 활성화합니다. 내부 RHEL 서버가 라우터 역할을 하고 네트워크에서 네트워크로 패킷을 전달합니다.

2. **/etc/sysctl.d/90-enable-IP-forwarding.conf** 파일에서 설정을 로드합니다.

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

3. 들어오는 HTTP 트래픽을 전달합니다.

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

이전 명령은 다음 설정으로 DNAT 규칙을 정의합니다.

- **--zone=public** - DNAT 규칙을 구성하는 방화벽 영역입니다. 필요한 모든 영역에 맞게 조정할 수 있습니다.
- **--add-forward-port** - 포트 전달 규칙을 추가 중임을 나타내는 옵션입니다.
- **port=80** - 외부 대상 포트입니다.
- **proto=tcp** - TCP 트래픽을 전달함을 나타내는 프로토콜입니다.
- **toaddr=198.51.100.10** - 대상 IP 주소입니다.
- **toport=8080** - 내부 서버의 대상 포트입니다.
- **--permanent** - 재부팅 시 DNAT 규칙을 유지할 수 있는 옵션입니다.

4. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

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

검증

- 사용한 방화벽 영역에 대한 DNAT 규칙을 확인합니다.

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

또는 해당 XML 구성 파일을 확인합니다.

```
# 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>
```

추가 리소스

- 런타임에 커널 매개변수 구성
- firewall-cmd(1) 매뉴얼 페이지

1.11.4. 비표준 포트에서 트래픽을 리디렉션하여 표준 포트에서 웹 서비스에 액세스하도록 설정

리디렉션 메커니즘을 사용하여 사용자가 URL에 포트를 지정할 필요 없이 내부적으로 비표준 포트에서 실행되는 웹 서비스를 만들 수 있습니다. 결과적으로 URL은 더 간단하며 더 나은 검색 환경을 제공하는 반면 비표준 포트는 여전히 내부적으로 또는 특정 요구 사항에 사용됩니다.

사전 요구 사항

- firewalld 서비스가 실행 중입니다.

절차

1. 다음 콘텐츠를 사용하여 /etc/sysctl.d/90-enable-IP-forwarding.conf 파일을 생성합니다.

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

이 설정은 커널에서 IP 전달을 활성화합니다.

2. /etc/sysctl.d/90-enable-IP-forwarding.conf 파일에서 설정을 로드합니다.

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

3. NAT 리디렉션 규칙을 생성합니다.

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

이전 명령은 다음 설정으로 NAT 리디렉션 규칙을 정의합니다.

- **--zone=public** - 규칙을 구성하는 방화벽 영역입니다. 필요한 모든 영역에 맞게 조정할 수 있습니다.
- **--add-forward-port=port= <*non_standard_port*>** - 들어오는 트래픽을 처음 수신하는 소스 포트를 사용하여 포트 전달(리렉션) 규칙을 추가 중임을 나타내는 옵션입니다.
- **proto=tcp** - TCP 트래픽을 리디렉션함을 나타내는 프로토콜입니다.
- **toport=<*standard_port*>** - 소스 포트에서 수신한 후 들어오는 트래픽을 리디렉션해야 하는 대상 포트입니다.
- **--permanent** - 다시 부팅 시 규칙을 유지할 수 있는 옵션입니다.

4. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

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

검증

- 다음을 사용한 방화벽 영역의 리디렉션 규칙을 확인합니다.

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

또는 해당 XML 구성 파일을 확인합니다.

```
# 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>
```

추가 리소스

- 런타임에 커널 매개변수 구성
- **firewall-cmd(1)** 매뉴얼 페이지

1.12. ICMP 요청 관리

ICMP(Internet Control Message Protocol)는 다양한 네트워크 장치에서 테스트, 문제 해결 및 진단을 위해 사용하는 지원 프로토콜입니다. **ICMP** 는 시스템 간에 데이터를 교환하는 데 사용되지 않으므로 TCP 및 UDP와 같은 전송 프로토콜과 다릅니다.

ICMP 메시지, 특히 **echo-request** 및 **echo-reply** 를 사용하여 네트워크에 대한 정보를 공개하고 다양한 종류의 사기 활동에 대해 이러한 정보를 오용할 수 있습니다. 따라서 **firewalld** 는 네트워크 정보를 보호하기 위해 **ICMP** 요청을 제어할 수 있습니다.

1.12.1. ICMP 필터링 구성

ICMP 필터링을 사용하여 방화벽에서 시스템에 도달할 수 있도록 허용하거나 거부할 ICMP 유형 및 코드를 정의할 수 있습니다. ICMP 유형 및 코드는 ICMP 메시지의 특정 카테고리 및 하위 범주입니다.

예를 들어 ICMP 필터링은 다음 영역에서 도움이 됩니다.

- 보안 개선 - 잠재적으로 유해한 ICMP 유형 및 코드를 차단하여 공격 면적을 줄입니다.
- 네트워크 성능 - 네트워크 성능을 최적화하고 과도한 ICMP 트래픽으로 인한 잠재적인 네트워크 혼잡을 방지하는 데 필요한 ICMP 유형만 허용합니다.
- 문제 해결 제어 - 잠재적인 보안 위험을 나타내는 네트워크 문제 해결 및 차단 ICMP 유형에 대한 필수 ICMP 기능을 유지 관리합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 사용 가능한 ICMP 유형 및 코드를 나열합니다.

```
# 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
...
```

   사전 정의된 목록에서 허용 또는 차단할 ICMP 유형 및 코드를 선택합니다.

2. 특정 ICMP 유형을 다음과 같이 필터링합니다.

   - ICMP 유형 허용:

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

     명령은 echo requests ICMP 유형의 기존 차단 규칙을 제거합니다.

   - ICMP 유형 차단:

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

이 명령을 사용하면 리디렉션 메시지 ICMP 유형이 방화벽에 의해 차단됩니다.

3. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

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

검증

- 필터링 규칙이 적용되었는지 확인합니다.

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

명령 출력에는 허용 또는 차단한 ICMP 유형 및 코드가 표시됩니다.

추가 리소스

- **firewall-cmd(1)** 매뉴얼 페이지

1.13. FIREWALLD를 사용하여 IP 세트 설정 및 제어

IP 세트는 보다 유연하고 효율적인 방화벽 규칙 관리를 위해 IP 주소 및 네트워크를 세트로 그룹화하는 RHEL 기능입니다.

예를 들어 필요한 경우 IP 세트는 시나리오에서 중요합니다.

- 대규모 IP 주소 목록 처리
- 이러한 대규모 IP 주소 목록에 대한 동적 업데이트 구현
- 사용자 지정 IP 기반 정책을 생성하여 네트워크 보안 및 제어 강화

주의

Red Hat은 **firewall-cmd** 명령을 사용하여 IP 세트를 생성하고 관리하는 것이 좋습니다.

1.13.1. IP 세트를 사용하여 허용 목록에 대한 동적 업데이트 구성

예측할 수 없는 조건에서도 IP 세트의 특정 IP 주소 또는 범위를 유연하게 허용하도록 거의 실시간 업데이트를 수행할 수 있습니다. 이러한 업데이트는 보안 위협 탐지 또는 네트워크 동작 변경과 같은 다양한 이벤트에 의해 트리거될 수 있습니다. 일반적으로 이러한 솔루션은 자동화를 활용하여 수동 작업을 줄이고 상황에 신속하게 대응하여 보안을 개선합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 의미 있는 이름으로 IP 세트를 생성합니다.

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

allowlist 라는 새 IP 세트에는 방화벽에서 허용할 IP 주소가 포함되어 있습니다.

2. IP 세트에 동적 업데이트를 추가합니다.

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

이 구성은 방화벽에서 네트워크 트래픽을 전달할 수 있는 새로 추가된 IP 주소로 허용 목록 IP 세트를 업데이트합니다.

3. 이전에 생성한 IP 세트를 참조하는 방화벽 규칙을 생성합니다.

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

이 규칙이 없으면 IP 세트가 네트워크 트래픽에 영향을 미치지 않습니다. 기본 방화벽 정책이 우선합니다.

4. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

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

검증

1. 모든 IP 세트를 나열합니다.

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

2. 활성 규칙을 나열합니다.

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

명령줄 출력의 소스 섹션에서는 특정 방화벽 영역에 대한 액세스 허용 또는 거부되는 트래픽(호스트, 인터페이스, IP 세트, 서브넷 등)에 대한 인사이트를 제공합니다. 이 경우 허용 목록 IP 세트에 포함된 IP 주소는 공용 영역의 방화벽을 통해 트래픽을 전달할 수 있습니다.

3. IP 세트의 내용을 살펴봅니다.

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

```
  <entry>198.51.100.10</entry>
</ipset>
```

다음 단계

- 스크립트 또는 보안 유틸리티를 사용하여 위협 정보 피드를 가져오고 이에 따라 허용 목록을 자동화된 방식으로 업데이트합니다.

추가 리소스

- **firewall-cmd(1)** 매뉴얼 페이지

1.14. 풍부한 규칙 우선 순위 지정

기본적으로 풍부한 규칙은 규칙 작업을 기반으로 구성됩니다. 예를 들어 거부 규칙은 허용 규칙보다 우선합니다. 풍부한 규칙의 우선순위 매개 변수를 통해 관리자는 풍부한 규칙과 실행 순서를 세밀하게 제어할 수 있습니다. **priority** 매개변수를 사용하는 경우 규칙은 우선 순위 값으로 오름차순으로 정렬됩니다. 더 많은 규칙에 동일한 우선 순위가 있는 경우 규칙 작업에 따라 순서가 결정되며, 작업이 동일한 경우 순서가 정의되지 않을 수 있습니다.

1.14.1. priority 매개변수가 규칙을 다른 체인으로 구성하는 방법

리치 규칙의 **priority** 매개변수를 **-32768** 과 **32767** 사이의 임의의 숫자로 설정할 수 있으며 더 낮은 숫자 값은 우선 순위가 높습니다.

firewalld 서비스는 우선 순위 값을 기반으로 다른 체인으로 규칙을 구성합니다.

- 우선순위가 0보다 낮습니다. 규칙은 **_pre** 접미사가 있는 체인으로 리디렉션됩니다.
- 우선순위가 0보다 높습니다. 규칙은 **_post** 접미사가 있는 체인으로 리디렉션됩니다.
- 우선 순위 0: 작업에 따라 규칙은 **_log**,**_deny** 또는 **_allow** 작업을 사용하여 체인으로 리디렉션됩니다.

이러한 하위 체인 내에서 **firewalld** 는 우선순위 값을 기반으로 규칙을 정렬합니다.

1.14.2. 풍부한 규칙의 우선 순위 설정

다음은 다른 규칙에서 허용되거나 거부되지 않는 모든 트래픽을 기록하기 위해 **priority** 매개변수를 사용하는 리치 규칙을 생성하는 예입니다. 이 규칙을 사용하여 예기치 않은 트래픽의 플래그를 지정할 수 있습니다.

절차

- 다른 규칙과 일치하지 않는 모든 트래픽을 기록하기 위해 우선 순위가 매우 낮은 풍부한 규칙을 추가합니다.

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

  이 명령은 로그 항목 수를 분당 **5** 개로 제한합니다.

검증

- 이전 단계에서 생성된 명령을 사용하여 **nftables** 규칙을 표시합니다.

```
# 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. 방화벽 잠금 구성

로컬 애플리케이션 또는 서비스는 **root** 로 실행되는 경우 방화벽 구성을 변경할 수 있습니다(예:libvirt). 이 기능을 사용하면 관리자가 방화벽 구성을 잠글 수 있으므로, 애플리케이션이나 잠금에 추가된 애플리케이션만 잠금 해제하면 방화벽 변경 사항을 요청할 수 있습니다. 잠금 설정이 기본적으로 비활성화되어 있습니다. 활성화된 경우 로컬 애플리케이션 또는 서비스에 의해 방화벽에 원하지 않는 구성 변경이 없는지 확인할 수 있습니다.

1.15.1. CLI를 사용하여 잠금 구성

명령줄을 사용하여 잠금 기능을 활성화하거나 비활성화할 수 있습니다.

절차

1. 잠금이 활성화되었는지 여부를 쿼리하려면 다음을 수행합니다.

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

2. 다음 중 하나를 통해 잠금 구성을 관리합니다.

   - 잠금 활성화:

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

   - 잠금 비활성화:

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

1.15.2. 잠금 허용 목록 구성 파일 개요

기본 allowlist 구성 파일에는 **NetworkManager** 컨텍스트 및 **libvirt** 의 기본 컨텍스트가 포함되어 있습니다. 사용자 ID 0도 목록에 있습니다.

허용 목록 구성 파일은 **/etc/firewalld/** 디렉터리에 저장됩니다.

```
<?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>
```

다음은 사용자 ID가 **815** 인 사용자에 대해 **firewall-cmd** 유틸리티에 대한 모든 명령을 활성화하는 allowlist 설정 파일의 예입니다.

-

```
<?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>
```

이 예에서는 사용자 ID와 사용자 이름 를 모두 표시하지만 하나의 옵션만 필요합니다. Python은 인터프리터이며 명령행에 추가됩니다.

Red Hat Enterprise Linux에서 모든 유틸리티는 /usr /bin/ 디렉토리에 있으며 /bin/ 디렉토리는/usr/bin/ 디렉토리에 sym-linked입니다. 즉, root 로 입력할 때 firewall-cmd 의 경로가/bin/firewall-cmd 로 확인될 수 있지만/usr/bin/firewall-cmd 를 사용할 수 있습니다. 모든 새 스크립트는 새 위치를 사용해야 합니다. 그러나 root 로 실행되는 스크립트가/bin/firewall-cmd 경로를 사용하도록 작성된 경우 전통적으로 바root 사용자에게만 사용되는 /usr/bin/firewall-cmd 경로와 더불어 allow 목록에 해당 명령 경로를 추가해야 합니다.

*명령의 name 속성 끝에 있는 * 는 이 문자열로 시작하는 모든 명령이 일치함을 의미합니다.* 가 없으면 인수를 포함한 absolute 명령이 일치해야 합니다.*

1.16. FIREWALLD 영역 내의 다양한 인터페이스 또는 소스 간 트래픽 전달 활성화

내부 영역 전달은 firewalld 영역 내의 인터페이스 또는 소스 간 트래픽 전달을 활성화하는firewalld 기능입니다.

1.16.1. 기본 타겟이 ACCEPT로 설정된 인트라 영역 전달과 영역의 차이점

영역 내 전달이 활성화된 경우 단일 firewalld 영역 내의 트래픽이 하나의 인터페이스 또는 소스에서 다른 인터페이스 또는 소스로 전달될 수 있습니다. 영역은 인터페이스 및 소스의 신뢰 수준을 지정합니다. 신뢰 수준이 동일한 경우 트래픽은 동일한 영역 내에 유지됩니다.

참고

firewalld 의 기본 영역에서 영역 내 전달을 활성화하면 현재 기본 영역에 추가된 인터페이스와 소스에만 적용됩니다.

firewalld 는 다른 영역을 사용하여 들어오고 나가는 트래픽을 관리합니다. 각 영역에는 고유한 규칙과 동작 세트가 있습니다. 예를 들어 신뢰할 수 있는 영역은 기본적으로 전달된 모든 트래픽을 허용합니다.

다른 영역에는 기본 동작이 다를 수 있습니다. 표준 영역에서 영역의 대상이 기본값으로 설정된 경우 전달된 트래픽은 일반적으로 기본적으로 삭제됩니다.

영역 내의 다양한 인터페이스 또는 소스 간에 트래픽이 전달되는 방법을 제어하려면 해당 영역의 대상을 적절하게 이해하고 구성해야 합니다.

1.16.2. 영역 내 전달을 사용하여 이더넷 및 Wi-Fi 네트워크 간의 트래픽 전달

intra-zone 전달을 사용하여 동일한 firewalld 영역 내의 인터페이스와 소스 간에 트래픽을 전달할 수 있습니다. 이 기능은 다음과 같은 이점을 제공합니다.

- *유선 및 무선 장치 간의 원활한 연결(W lp 0s 20에 연결된 이더넷 네트워크와 Wi-Fi 네트워크 간에 트래픽을 전달할 수 있음)*

- *유연한 작업 환경 지원*
- *프린터, 데이터베이스, 네트워크 연결 스토리지 등 여러 장치 또는 네트워크에서 액세스하고 사용하는 공유 리소스*
- *효율적인 내부 네트워킹(예: 원활한 통신, 대기 시간 감소, 리소스 접근성 등)*

*개별 **firewalld** 영역에 대해 이 기능을 활성화할 수 있습니다.*

절차

1. *커널에서 패킷 전달을 활성화합니다.*

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

2. *영역 내 전달을 활성화할 인터페이스가 내부 영역에만 할당되도록 합니다.*

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

3. *현재 인터페이스가 내부 가 아닌 영역에 할당되면 인터페이스를 다시 할당합니다.*

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

4. ***enp1s0** 및 **wlp0s20** 인터페이스를 내부 영역에 추가합니다.*

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

5. *지역 내 전달을 활성화합니다.*

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

검증

*다음 확인에서는 **nmap-ncat** 패키지가 두 호스트 모두에 설치되어 있어야 합니다.*

1. *영역 전달을 활성화한 호스트의 **enp1s0** 인터페이스와 동일한 네트워크에 있는 호스트에 로그인합니다.*
2. ***ncat** 으로 echo 서비스를 시작하여 연결을 테스트합니다.*

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

3. ***wlp0s20** 인터페이스와 동일한 네트워크에 있는 호스트에 로그인합니다.*
4. ***enp1s0** 과 동일한 네트워크에 있는 호스트에서 실행 중인 에코 서버에 연결합니다.*

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

5. *어떤 것을 입력하고 **Enter** 키를 누릅니다. 텍스트가 다시 전송되었는지 확인합니다.*

추가 리소스

- 시스템의 **firewalld.zones(5)** 도움말 페이지

1.17. RHEL 시스템 역할을 사용하여 FIREWALLD 구성

RHEL 시스템 역할은 Ansible 자동화 유틸리티의 콘텐츠 집합입니다. 이 콘텐츠는 Ansible 자동화 유틸리티와 함께 여러 시스템을 한 번에 원격으로 관리할 수 있는 일관된 구성 인터페이스를 제공합니다.

rhel-system-roles 패키지에는 **rhel-system-roles.firewall** RHEL 시스템 역할이 포함되어 있습니다. 이 역할은 **firewalld** 서비스의 자동 구성을 위해 도입되었습니다.

방화벽 RHEL 시스템 역할을 사용하면 다양한**firewalld** 매개변수를 구성할 수 있습니다. 예를 들면 다음과 같습니다.

- 영역
- 패킷을 허용해야 하는 서비스
- 포트에 대한 트래픽 액세스 권한 부여, 거부 또는 삭제
- 영역의 포트 또는 포트 범위 전달

1.17.1. 방화벽 RHEL 시스템 역할을 사용하여 firewalld 설정 재설정

시간이 지남에 따라 방화벽 구성을 업데이트하면 의도하지 않은 보안 위험이 발생할 수 있습니다. 방화벽 RHEL 시스템 역할을 사용하면 **firewalld** 설정을 자동으로 기본 상태로 재설정할 수 있습니다. 이렇게 하면 의도하지 않거나 안전하지 않은 방화벽 규칙을 효율적으로 제거하고 관리를 단순화할 수 있습니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- 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
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

이전: 교체

기존 사용자 정의 설정을 모두 제거하고 **firewalld** 설정을 기본값으로 재설정합니다. 이 전:**replaced** 매개변수를 다른 설정과 결합하면**firewall** 역할은 새 설정을 적용하기 전에 기존 설정을 모두 제거합니다.
플레이북에 사용되는 모든 변수에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일을 참조하십시오.

2. 플레이북 구문을 확인합니다.

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

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

검증

- 제어 노드에서 이 명령을 실행하여 관리 노드의 모든 방화벽 구성이 기본값으로 재설정되었는지 원격으로 확인합니다.

```
# ansible managed-node-01.example.com -m ansible.builtin.command -a 'firewall-cmd --list-all-zones'
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일
- **/usr/share/doc/rhel-system-roles/firewall/** 디렉터리

1.17.2. 방화벽 RHEL 시스템 역할을 사용하여 하나의 로컬 포트에서 다른 로컬 포트로 firewalld 에서 들어오는 트래픽 전달

방화벽 RHEL 시스템 역할을 사용하여 하나의 로컬 포트에서 다른 로컬 포트로 들어오는 트래픽 전달을 원격으로 구성할 수 있습니다.

예를 들어 여러 서비스가 동일한 시스템에 공존하고 동일한 기본 포트가 필요한 환경이 있는 경우 포트 충돌이 발생할 수 있습니다. 이러한 충돌로 인해 서비스가 중단되고 다운타임이 발생할 수 있습니다. 방화벽 RHEL 시스템 역할을 사용하면 트래픽을 대체 포트로 효율적으로 전달하여 구성을 수정하지 않고도 서비스를 동시에 실행할 수 있습니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- 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
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

forward_port: 8080/tcp;443

TCP 프로토콜을 사용하여 로컬 포트 8080으로 들어오는 트래픽은 포트 443으로 전달됩니다.

runtime: true

*런타임 구성에서 변경 사항을 활성화합니다. 기본값은 **true** 로 설정됩니다.*
*플레이북에 사용되는 모든 변수에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일을 참조하십시오.*

2. *플레이북 구문을 확인합니다.*

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. *플레이북을 실행합니다.*

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

검증

- *제어 노드에서 다음 명령을 실행하여 관리 노드에서 forwarded-ports를 원격으로 확인합니다.*

```
# ansible managed-node-01.example.com -m ansible.builtin.command -a 'firewall-cmd
--list-forward-ports'
managed-node-01.example.com | CHANGED | rc=0 >>
port=8080:proto=tcp:toport=443:toaddr=
```

추가 리소스

- ***/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일*
- ***/usr/share/doc/rhel-system-roles/firewall/** 디렉터리*

1.17.3. 방화벽 RHEL 시스템 역할을 사용하여 firewalld DMZ 영역 구성

*시스템 관리자는 방화벽 RHEL 시스템 역할을 사용하여enp1s0 인터페이스에서 **dmz** 영역을 구성하여 **HTTPS** 트래픽을 영역에 허용할 수 있습니다. 이렇게 하면 외부 사용자가 웹 서버에 액세스할 수 있습니다.*

사전 요구 사항

- *제어 노드와 관리형 노드가 준비되어 있습니다.*
- *관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.*
- *관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.*

절차

1. *다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.*

```
---
- 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
```

*플레이북에 사용되는 모든 변수에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일을 참조하십시오.*

2. *플레이북 구문을 확인합니다.*

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. *플레이북을 실행합니다.*

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

검증

- *제어 노드에서 다음 명령을 실행하여 관리 노드의 **dmz** 영역에 대한 정보를 원격으로 확인합니다.*

```
# ansible managed-node-01.example.com -m ansible.builtin.command -a 'firewall-cmd
--zone=dmz --list-all'
managed-node-01.example.com | CHANGED | rc=0 >>
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:
```

추가 리소스

- ***/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일*
- ***/usr/share/doc/rhel-system-roles/firewall/** 디렉터리*

2 장. NFTABLES 시작하기

nftables 프레임워크는 패킷을 분류하고 **iptables**, **ip6tables**, **arptables**, **ebtables**, **ipset** 유틸리티의 후속 조치입니다. 이전의 패킷 필터링 툴에 비해 편의성, 기능 및 성능이 크게 향상되었으며 주요 개선 사항은 다음과 같습니다.

- 선형 처리 대신 기본 제공 조회 테이블
- **IPv4** 및 **IPv6** 프로토콜 모두를 위한 단일 프레임워크
- 모든 규칙은 전체 규칙 세트를 가져오기, 업데이트 및 저장하는 대신 원자적으로 적용됩니다.
- 규칙 세트(**nftrace**) 및 모니터링 추적 이벤트(**nft** 툴 내)에서 디버깅 및 추적 지원
- 프로토콜별 확장 없이 보다 일관되고 컴팩트한 구문
- 타사 애플리케이션을 위한 Netlink API

nftables 프레임워크는 테이블을 사용하여 체인을 저장합니다. 체인에는 작업을 수행하기 위한 개별 규칙이 포함되어 있습니다. **nft** 유틸리티는 이전 패킷 필터링 프레임워크의 모든 도구를 대체합니다.**libnftnl** 라이브러리를 사용하여 **libmnl** 라이브러리를 통해 **nftables** Netlink API와 하위 수준의 상호 작용을 수행할 수 있습니다.

규칙 세트 변경의 효과를 표시하려면 **nft list ruleset** 명령을 사용합니다. 이러한 유틸리티는 테이블, 체인, 규칙, 세트 및 기타 오브젝트를 **nftables** 규칙 세트에 추가하므로 **nft flush ruleset** 명령과 같은 **nftables** 규칙 세트 작업이 **iptables** 명령을 사용하여 설치된 규칙 세트에 영향을 미칠 수 있습니다.

2.1. NFTABLES 테이블, 체인 및 규칙 생성 및 관리

nftables 규칙 세트를 표시하고 관리할 수 있습니다.

2.1.1. nftables 테이블 기본

nftables 의 테이블은 체인, 규칙, 세트 및 기타 오브젝트 컬렉션을 포함하는 네임스페이스입니다.

각 테이블에는 주소 제품군이 할당되어 있어야 합니다. 주소 family는 이 테이블이 처리하는 패킷 유형을 정의합니다. 테이블을 만들 때 다음 주소 제품군 중 하나를 설정할 수 있습니다.

- **ip**: IPv4 패킷만 일치시킵니다. 주소 제품군을 지정하지 않는 경우 기본값은 기본값입니다.
- **ip6**: IPv6 패킷만 일치시킵니다.
- **inet**: IPv4 및 IPv6 패킷과 일치합니다.
- **ARP**: IPv4 주소 확인 프로토콜(ARP) 패킷과 일치합니다.
- **bridge**: 브리지 장치를 통과하는 패킷과 일치합니다.
- **netdev**: 수신의 패킷과 일치합니다.

테이블을 추가하려면 사용할 형식은 방화벽 스크립트에 따라 다릅니다.

- 기본 구문 스크립트의 경우 다음을 사용합니다.

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

- *쉘 스크립트에서는 다음을 사용합니다.*

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

2.1.2. nftables 체인의 기본 사항

테이블은 체인으로 구성되며, 이 체인은 규칙용 컨테이너입니다. 다음 두 가지 규칙 유형이 있습니다.

- *기본 체인: 기본 체인을 네트워킹 스택의 패킷 진입점으로 사용할 수 있습니다.*
- *일반 체인: 규칙을 더 잘 구성하기 위해 일반 체인을이동 대상으로 사용할 수 있습니다.*

테이블에 기본 체인을 추가하려면 사용할 형식은 방화벽 스크립트에 따라 다릅니다.

- *기본 구문 스크립트의 경우 다음을 사용합니다.*

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

- *쉘 스크립트에서는 다음을 사용합니다.*

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

쉘이 명령 끝으로 해석되지 않도록 하려면 \ 이스케이프 문자 앞에 \ 이스케이프 문자를 배치합니다.

두 예에서는 기본 체인을 생성합니다. 일반 체인 을 생성하려면 중괄호에서 매개 변수를 설정하지 마십시오.

체인 유형
다음은 체인 유형 및 제품군과 후크를 사용할 수 있는 개요입니다.

유형	주소 제품군	후크	설명
filter	all	all	표준 체인 유형
nat	**ip**, **ip6**, **inet**	**PREROUTING, 입력,출력,후드**	이 유형의 체인은 연결 추적 항목을 기반으로 기본 주소 변환을 수행합니다. 첫 번째 패킷만 이 체인 유형을 통과합니다.
Route	**ip**, **ip6**	**출력**	이 체인 유형을 트래버스하는 수락된 패킷은 IP 헤더의 관련 부분이 변경된 경우 새로운 경로 조회를 유발합니다.

체인 우선순위
priority 매개변수는 패킷이 동일한 후크 값을 사용하는 체인을 트래버스하는 순서를 지정합니다. 이 매개변수를 정수 값으로 설정하거나 표준 우선순위 이름을 사용할 수 있습니다.

다음 매트릭스는 표준 우선 순위 이름과 해당 숫자 값에 대한 개요와 함께 사용할 수 있는 제품군과 후크를 처리합니다.

텍스트 값	숫자 값	주소 제품군	후크
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
보안	**50**	**ip**, **ip6**, **inet**	all
srcnat	**100**	**ip**, **ip6**, **inet**	**POSTROUTING**
	300	**Bridge**	**POSTROUTING**
out	**100**	**Bridge**	출력

체인 정책
*체인 정책은 이 체인의 규칙이 작업을 지정하지 않는 경우 **nftables** 가 패킷을 수락하거나 삭제해야 하는지 여부를 정의합니다. 체인에서 다음 정책 중 하나를 설정할 수 있습니다.*

- *수락 (기본값)*
- ***drop***

2.1.3. nftables *규칙의 기본 사항*

*규칙은 이 규칙을 포함하는 체인을 전달하는 패킷에서 수행할 작업을 정의합니다. 규칙에 일치하는 표현식도 포함된 경우 **nftables** 는 모든 이전 표현식이 적용되는 경우에만 작업을 수행합니다.*

체인에 규칙을 추가하려면 사용할 형식은 방화벽 스크립트에 따라 다릅니다.

- *기본 구문 스크립트의 경우 다음을 사용합니다.*

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

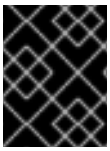
- *쉘 스크립트에서는 다음을 사용합니다.*

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

이 셸 명령은 체인 끝에 새 규칙을 추가합니다. 체인 시작 부분에 규칙을 추가하려면 **nft add** 대신 **nft insert** 명령을 사용하십시오.

2.1.4. nft 명령을 사용하여 테이블, 체인 및 규칙 관리

명령줄 또는 셸 스크립트에서 **nftables** 방화벽을 관리하려면 **nft** 유틸리티를 사용합니다.



중요

이 절차의 명령은 일반적인 워크플로우를 나타내지 않으며 최적화되지 않습니다. 이 절차에서는 일반적으로 **nft** 명령을 사용하여 테이블, 체인 및 규칙을 관리하는 방법을 보여줍니다.

절차

1. 테이블이 IPv4 및 IPv6 패킷을 모두 처리할 수 있도록 **inet** 주소 Family를 사용하여 **nftables_svc** 라는 테이블을 만듭니다.

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

2. 들어오는 네트워크 트래픽을 처리하는 **INPUT** 라는 기본 체인을 **inet nftables_svc** 테이블에 추가합니다.

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

셸이 명령 끝으로 해석되는 것을 방지하기 위해 \ 문자를 사용하여 이름이 0으로 이스케이프합니다.

3. **INPUT** 체인에 규칙을 추가합니다. 예를 들어 포트 22 및 443에서 들어오는 TCP 트래픽을 허용하고 **INPUT** 체인의 마지막 규칙으로 IMP(Internet Control Message Protocol) 포트 연결할 수 없는 메시지가 있는 다른 들어오는 트래픽을 거부합니다.

```
# 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
```

다음과 같이 **nft add rule** 명령을 입력하면 **nft** 는 명령을 실행할 때와 동일한 순서로 규칙을 체인에 추가합니다.

4. 프로세스를 포함한 현재 규칙 세트를 표시합니다.

```
# 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. *handle 3이 있는 기존 규칙 앞에 규칙을 삽입합니다. 예를 들어 포트 636에서 TCP 트래픽을 허용하는 규칙을 삽입하려면 다음을 입력합니다.*

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

6. *handle 3이 있는 기존 규칙 뒤에 규칙을 추가합니다. 예를 들어 포트 80에서 TCP 트래픽을 허용하는 규칙을 삽입하려면 다음을 입력합니다.*

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

7. *handles를 사용하여 규칙 세트를 다시 표시합니다. 나중에 추가된 규칙이 지정된 위치에 추가되었는지 확인합니다.*

```
# 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. *handle 6을 사용하여 규칙을 제거합니다.*

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

규칙을 제거하려면 handle을 지정해야 합니다.

9. *규칙 세트를 표시하고 제거된 규칙이 더 이상 존재하지 않는지 확인합니다.*

```
# 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. ***INPUT** 체인에서 나머지 모든 규칙을 제거합니다.*

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

11. *규칙 세트를 표시하고 **INPUT** 체인이 비어 있는지 확인합니다.*

```
# nft list table inet nftables_svc
table inet nftables_svc {
  chain INPUT {
```

```
    type filter hook input priority filter; policy accept
  }
}
```

12. **INPUT** 체인을 삭제합니다.

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

이 명령을 사용하여 여전히 규칙이 포함된 체인을 삭제할 수도 있습니다.

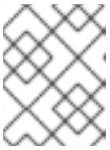
13. 규칙 세트를 표시하고 **INPUT** 체인이 삭제되었는지 확인합니다.

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

14. **nftables_svc** 테이블을 삭제합니다.

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

이 명령을 사용하여 체인이 여전히 포함된 테이블을 삭제할 수도 있습니다.



참고

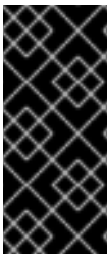
전체 규칙 세트를 삭제하려면 별도의 명령으로 모든 규칙, 체인 및 테이블을 수동으로 삭제하는 대신 **nft flush ruleset** 명령을 사용합니다.

추가 리소스

시스템의 **nft(8)** 도움말 페이지

2.2. IPTABLES 에서 NFTABLES 로 마이그레이션

방화벽 구성에서 **iptables** 규칙을 계속 사용하는 경우 **iptables** 규칙을 **nftables** 로 마이그레이션할 수 있습니다.



중요

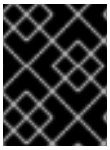
Red Hat Enterprise Linux 9에서는 **ipset** 및 **iptables-nft** 패키지가 더 이상 사용되지 않습니다. 여기에는 **iptables**, **ip6tables**, **arptables**, **ebtables** 유틸리티와 같은 **nft-variants** 의 사용 중단이 포함됩니다. 예를 들어 이전 RHEL 버전에서 업그레이드했기 때문에 이러한 툴을 사용하는 경우 **nftables** 패키지에서 제공하는 **nft** 명령줄 도구로 마이그레이션할 것을 권장합니다.

2.2.1. firewalld, nftables 또는 iptables 를 사용하는 경우

다음은 다음 유틸리티 중 하나를 사용해야 하는 시나리오에 대한 간략한 개요입니다.

- **firewalld**: 간단한 방화벽 사용 사례에 **firewalld** 유틸리티를 사용합니다. 이 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.
- **nftables**: **nftables** 유틸리티를 사용하여 전체 네트워크에 대해 과 같이 복잡하고 성능이 중요한 방화벽을 설정합니다.

- **iptables**: Red Hat Enterprise Linux의 **iptables** 유틸리티는 레거시 백엔드 대신 **nf_tables** 커널 API를 사용합니다. **nf_tables** API는 이전 버전과의 호환성을 제공하므로 **iptables** 명령을 사용하는 스크립트가 여전히 Red Hat Enterprise Linux에서 작동합니다. 새 방화벽 스크립트의 경우 Red Hat은 **nftables** 를 사용하도록 권장합니다.



중요

다른 방화벽 관련 서비스(**firewalld**, **nftables** 또는 **iptables**)가 서로 영향을 미치지 않도록 하려면 RHEL 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

2.2.2. nftables 프레임워크의 개념

iptables 프레임워크와 비교하여 **nftables** 는 보다 현대적이고 효율적이며 유연한 대안을 제공합니다. **iptables** 에 대한 고급 기능과 개선 사항을 제공하는 몇 가지 개념과 기능이 있습니다. 이러한 개선 사항을 통해 규칙 관리가 간소화되고 성능이 향상되어 **nftables** 를 복잡하고 고성능 네트워킹 환경의 최신 대안으로 만들 수 있습니다.

nftables 프레임워크에는 다음 구성 요소가 포함됩니다.

테이블 및 네임스페이스

nftables 에서 테이블은 관련 방화벽 체인, 세트, 흐름 테이블 및 기타 오브젝트를 함께 그룹화하는 조직 단위 또는 네임스페이스를 나타냅니다. **nftables** 에서는 테이블이 방화벽 규칙 및 관련 구성 요소를 구성하는 보다 유연한 방법을 제공합니다. **iptables** 에서는 테이블이 특정 목적에 따라 보다 효율적으로 정의되었습니다.

테이블 제품군

nftables 의 각 테이블은 특정 제품군(**ip**, **ip6**, **inet**, **arp**, **bridge** 또는 **netdev**)과 연결되어 있습니다. 이 연결은 테이블이 처리할 수 있는 패킷을 결정합니다. 예를 들어 **ip** 제품군의 테이블은 IPv4 패킷만 처리합니다. 반면 **inet** 은 테이블 가정의 특별한 경우입니다. IPv4 및 IPv6 패킷을 모두 처리할 수 있으므로 프로토콜 간에 통합된 접근 방식을 제공합니다. 특수 테이블 제품군의 또 다른 경우는 **netdev** 입니다. 네트워크 장치에 직접 적용되는 규칙에 사용되어 장치 수준에서 필터링이 가능하기 때문입니다.

기본 체인

nftables 의 기본 체인은 패킷 처리 파이프라인에서 구성 가능한 진입점으로, 사용자가 다음을 지정할 수 있습니다.

- 체인의 유형 (예: "filter"
- 패킷 처리 경로의 후크 지점(예: "input", "output", "forward")
- 체인의 우선순위

이러한 유연성을 통해 규칙이 네트워크 스택을 통과할 때 패킷에 적용되는 시기와 방법을 정확하게 제어할 수 있습니다. 체인의 특수 사례는 패킷 헤더를 기반으로 커널에 의해 이루어진 라우팅 결정에 영향을 미치는 데 사용되는 경로 체인입니다.

규칙 처리를 위한 가상 머신

nftables 프레임워크는 내부 가상 시스템을 사용하여 규칙을 처리합니다. 이 가상 머신은 어셈블리 언어 작업과 유사한 명령을 실행합니다(기록으로 데이터를 로드하고 비교 수행 등). 이러한 메커니즘은 매우 유연하고 효율적인 규칙 처리를 허용합니다.

nftables 의 개선 사항은 해당 가상 머신에 대한 새로운 지침으로 도입될 수 있습니다. 일반적으로 새 커널 모듈과 **libnftnl** 라이브러리 및 **nft** 명령줄 유틸리티를 업데이트해야 합니다.

또는 커널 수정 없이도 기존 지침을 혁신적으로 결합하여 새로운 기능을 도입할 수 있습니다. **nftables** 규칙의 구문은 기본 가상 시스템의 유연성을 반영합니다. 예를 들어 **meta** 마크가 **tcp dport** 맵을 설정하는 규칙

{ 22: 1, 80: 2 } 는 TCP 대상 포트가 22인 경우 패킷의 방화벽 표시를 1로 설정하고 포트가 80인 경우 2로 설정합니다. 이것은 복잡한 논리를 간결하게 표현 할 수있는 방법을 보여줍니다.

학습 및 개선 사항

__nftables__ 프레임워크는 IP 주소, 포트, 기타 데이터 유형 및 가장 중요한 조합에서 __iptables__ 에서 사용되는 __ipset__ 유틸리티의 기능을 통합하고 확장합니다. 이러한 통합을 통해 __nftables__ 내에서 직접 대규모의 동적 데이터 세트를 쉽게 관리할 수 있습니다. 다음으로 __nftables__ 는 모든 데이터 유형에 대한 여러 값 또는 범위를 기반으로 일치하는 패킷을 기본적으로 지원하므로 복잡한 필터링 요구 사항을 처리하는 기능이 향상됩니다. __nftables__ 를 사용하면 패킷 내의 모든 필드를 조작할 수 있습니다.

__nftables__ 에서 세트는 이름이 지정되거나 익명일 수 있습니다. 명명된 세트는 여러 규칙에서 참조하고 동적으로 수정할 수 있습니다. 익명 세트는 규칙 내에서 인라인으로 정의되며 변경할 수 없습니다. 세트에는 다양한 유형의 조합(예: IP 주소 및 포트 번호 쌍)이 포함된 요소가 포함될 수 있습니다. 이 기능을 사용하면 복잡한 기준과 일치하는 유연성이 향상됩니다. 세트를 관리하기 위해 커널은 특정 요구 사항(성능, 메모리 효율성 등)에 따라 가장 적절한 백엔드를 선택할 수 있습니다. 세트는 키-값 쌍이 있는 맵으로 기능할 수도 있습니다. value 부분은 데이터 포인트(패치 헤더에 쓸 값) 또는 이동할 정성 또는 체인으로 사용할 수 있습니다. 이를 통해 "verdict maps"라는 복잡하고 동적인 규칙 동작을 사용할 수 있습니다.

유연한 규칙 형식

__nftables__ 규칙의 구조는 간단합니다. 조건 및 작업은 왼쪽에서 오른쪽으로 순차적으로 적용됩니다. 이 직관적인 형식을 사용하면 규칙을 만들고 문제를 해결할 수 있습니다.

규칙의 조건은 논리적으로(AND 연산자를 사용하여) 함께 연결되므로 규칙이 일치하도록 모든 조건을 "true"로 평가해야 합니다. 조건이 하나라도 실패하면 평가가 다음 규칙으로 이동합니다.

__nftables__ 의 작업은 삭제 또는 수락 과 같은 최종 작업이 될 수 있으므로 패킷에 대한 추가 규칙 처리를 중지합니다. 카운터 로그 메타 마크 세트 __0x3__ 과 같은 터미널이 아닌 작업은 특정 작업(패킷, 로깅, 마크 설정 등)을 수행하지만 후속 규칙을 평가할 수 있습니다.

추가 리소스

- *__nft(8)__ 매뉴얼 페이지*
- *iptables의 뒤에는 무엇이 있습니까? 후속 조치: nftables*
- *Firewalld: 미래는 nftables입니다.*

2.2.3. 더 이상 사용되지 않는 iptables 프레임워크의 개념

적극적으로 유지 관리되는 __nftables__ 프레임워크와 유사하게 더 이상 사용되지 않는 __iptables__ 프레임워크를 사용하면 다양한 패킷 필터링 작업, 로깅 및 감사, NAT 관련 구성 작업 등을 수행할 수 있습니다.

__iptables__ 프레임워크는 각 테이블이 특정 목적을 위해 설계된 여러 테이블로 구성되어 있습니다.

filter

기본 테이블은 일반 패킷 필터링을 보장합니다.

nat

NAT(Network Address Translation)의 경우 패킷의 소스 및 대상 주소 변경을 포함합니다.

mangle

특정 패킷 변경을 위해 고급 라우팅 결정에 대해 패킷 헤더를 수정할 수 있습니다.

Raw

연결 추적 전에 수행해야 하는 구성의 경우

이러한 테이블은 별도의 커널 모듈로 구현되며, 각 테이블은 **INPUT**,**OUTPUT**, **FORWARD** 와 같은 고정된 내장 체인 세트를 제공합니다. 체인은 패킷이 평가되는 일련의 규칙입니다. 이러한 체인은 커널의 패킷 처리 흐름의 특정 지점에 연결됩니다. 체인은 여러 테이블 간에 이름이 동일하지만 실행 순서는 해당 후크 우선 순위에 따라 결정됩니다. 우선순위는 커널에서 내부적으로 관리되므로 규칙이 올바른 순서로 적용되는지 확인합니다.

원래 **iptables** 는 IPv4 트래픽을 처리하도록 설계되었습니다. 그러나 IPv6 프로토콜을 도입하면 비슷한 기능(**iptables**)을 제공하고 사용자가 IPv6 패킷에 대한 방화벽 규칙을 생성하고 관리할 수 있도록 **ip6tables** 유틸리티를 도입해야 합니다. 동일한 논리를 통해 **arptables** 유틸리티는 ARP(Address Resolution Protocol)를 처리하기 위해 생성되었으며 **ebtables** 유틸리티는 이더넷 브리징 프레임을 처리하기 위해 개발되었습니다. 이러한 툴은 **iptables** 의 패킷 필터링 기능을 다양한 네트워크 프로토콜에 적용하고 포괄적인 네트워크 범위를 제공할 수 있도록 합니다.

iptables 의 기능을 개선하기 위해 확장 기능을 개발하기 시작했습니다. 기능 확장은 일반적으로 사용자 공간 동적 공유 오브젝트(DSO)와 페어링되는 커널 모듈로 구현됩니다. 확장 기능으로 방화벽 규칙에 더 정교한 작업을 수행할 수 있는 "matches" 및 "대상"이 도입되었습니다. 확장 기능을 사용하면 복잡한 일치 및 대상을 활성화할 수 있습니다. 예를 들어 특정 계층 4 프로토콜 헤더 값을 일치시키거나 조작하고, rate-limiting을 수행하고 할당량을 적용할 수 있습니다. 일부 확장 기능은 기본 **iptables** 구문의 제한 사항을 해결하도록 설계되었습니다(예: "multiport" 일치 확장). 이 확장을 사용하면 단일 규칙이 일치되지 않은 여러 포트를 일치시켜 규칙 정의를 단순화하고 필요한 개별 규칙 수를 줄일 수 있습니다.

ipset 은 **iptables** 에 대한 특별한 종류의 기능 확장입니다.**iptables** 와 함께 사용하여 IP 주소, 포트 번호 및 패킷과 일치시킬 수 있는 기타 네트워크 관련 요소 컬렉션을 생성하는 커널 수준 데이터 구조입니다. 이러한 세트는 방화벽 규칙을 작성하고, 작성하고, 관리하는 프로세스를 크게 간소화, 최적화 및 가속화합니다.

추가 리소스

- **iptables(8)** 도움말 페이지

2.2.4. iptables 및 ip6tables 규칙 세트를 nftables로 변환

iptables-restore-translate 및 **ip6tables-restore-translate** 유틸리티를 사용하여 **iptables** 및 **ip6tables** 규칙 세트를 **nftables** 로 변환합니다.

사전 요구 사항

- **nftables** 및 **iptables** 패키지가 설치됩니다.
- 시스템에는 **iptables** 및 **ip6tables** 규칙이 구성되어 있습니다.

절차

1. **iptables** 및 **ip6tables** 규칙을 파일에 작성합니다.

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

2. 덤프 파일을 **nftables** 명령으로 변환합니다.

```
# 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. 및 필요한 경우 생성된 **nftables** 규칙을 수동으로 업데이트합니다.

4. 생성된 파일을 로드할 **nftables** 서비스를 활성화하려면 **/etc/sysconfig/nftables.conf** 파일에 다음을 추가합니다.

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

5. **iptables** 서비스를 중지하고 비활성화합니다.

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

사용자 지정 스크립트를 사용하여 **iptables** 규칙을 로드한 경우 스크립트가 더 이상 자동으로 시작되지 않고 재부팅하여 모든 테이블을 플러시해야 합니다.

6. **nftables** 서비스를 활성화하고 시작합니다.

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

검증

- **nftables** 규칙 세트를 표시합니다.

```
# nft list ruleset
```

추가 리소스

- 시스템 부팅 시 nftables 규칙 자동 로딩

2.2.5. 단일 iptables 및 ip6tables 규칙을 nftables로 변환

Red Hat Enterprise Linux는 **iptables-translate** 및 **ip6tables-translate** 유틸리티를 제공하여 **iptables** 또는 **ip6tables** 규칙을 **nftables** 에 해당하는 규칙으로 변환합니다.

사전 요구 사항

- **nftables** 패키지가 설치되어 있어야 합니다.

절차

- **iptables** 또는 **ip6tables** 대신 **iptables-translate** 또는 **ip6tables-translate** 유틸리티를 사용하여 해당 **nftables** 규칙을 표시합니다. 예를 들면 다음과 같습니다.

```
# 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
```

일부 확장 기능에는 해당 지원이 누락되어 있는 경우도 있습니다. 이 경우 유틸리티는 **#** 기호가 앞에 오는 untranslated 규칙을 출력합니다. 예를 들면 다음과 같습니다.

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

추가 리소스

- **iptables-translate --help**

2.2.6. 공통 iptables 및 nftables 명령 비교

다음은 일반적인 **iptables** 및 **nftables** 명령을 비교한 것입니다.

- 모든 규칙 나열:

iptables	nftables
iptables-save	**nft list ruleset**

- 특정 테이블 및 체인 나열:

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 ipat PREROUTING**

nft 명령은 테이블 및 체인을 미리 만들지 않습니다. 사용자가 수동으로 생성한 경우에만 존재합니다.

firewalld를 통해 생성된 규칙 나열:

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

2.3. NFTABLES 스크립트 작성 및 실행

nftables 프레임워크를 사용할 때의 주요 이점은 스크립트 실행이 atomic이라는 것입니다. 즉, 시스템이 전체 스크립트를 적용하거나 오류가 발생하면 실행을 방지합니다. 이렇게 하면 방화벽이 항상 일관된 상태가 됩니다.

또한 **nftables** 스크립트 환경에서 다음을 수행할 수 있습니다.

- 댓글 추가
- 변수 정의
- 기타 규칙 세트 파일 포함

nftables 패키지를 설치하면 Red Hat Enterprise Linux가 **/etc/nftables/** 디렉터리에 ***.nft** 스크립트를 자동으로 생성합니다. 이러한 스크립트에는 서로 다른 용도로 테이블 및 빈 체인을 생성하는 명령이 포함되어 있습니다.

2.3.1. 지원되는 nftables 스크립트 형식

다음 형식으로 **nftables** 스크립팅 환경에서 스크립트를 작성할 수 있습니다.

- **nft list ruleset** 명령과 동일한 형식으로 규칙 세트를 표시합니다.

```
#!/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
  }
}
```

- **nft** 명령의 구문은 다음과 같습니다.

```
#!/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
```

2.3.2. nftables 스크립트 실행

nftables 스크립트를 **nft** 유틸리티에 전달하거나 직접 스크립트를 실행하여 실행할 수 있습니다.

절차

- *nft* 유틸리티에 전달하여 **nftables** 스크립트를 실행하려면 다음 을 입력합니다.

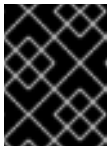
```
# nft -f /etc/nftables/<example_firewall_script>.nft
```

- **nftables** 스크립트를 직접 실행하려면 다음을 수행합니다.

  a. 이 작업을 수행하는 단일 시간 동안 다음을 수행합니다.

     i. 스크립트가 다음 hebang 시퀀스로 시작하는지 확인합니다.

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



중요

-f 매개변수를 생략하면 **nft** 유틸리티에서 스크립트를 읽지 않고 표시됩니다. 오류: 구문 오류, 예기치 않은 줄 바꿈, 문자열.

ii. 선택 사항: 스크립트의 소유자를 **root** 로 설정합니다.

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

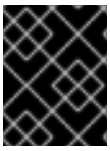
iii. 소유자를 위해 스크립트를 실행 가능하게 합니다.

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

b. 스크립트를 실행합니다.

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

출력이 표시되지 않으면 시스템에서 스크립트를 성공적으로 실행했습니다.



중요

nft 가 스크립트를 성공적으로 실행하고, 규칙을 잘못 배치했거나, 스크립트의 누락된 매개 변수 또는 기타 문제로 인해 방화벽이 예상대로 작동하지 않을 수 있습니다.

추가 리소스

- **chown(1)** 및 **chmod(1)** 매뉴얼 페이지
- 시스템 부팅 시 nftables 규칙 자동 로딩

2.3.3. nftables 스크립트에서 주석 사용

nftables 스크립팅 환경은 모든 것을**#** 문자 오른쪽에 주석으로 해석합니다.

주석은 행 시작 시 시작되거나 명령 옆에 있을 수 있습니다.

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

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

2.3.4. nftables 스크립트의 변수 사용

nftables 스크립트에서 변수를 정의하려면**define** 키워드를 사용합니다. 단일 값과 익명 집합을 변수에 저장할 수 있습니다. 더 복잡한 시나리오의 경우 set 또는 verdict 맵을 사용합니다.

단일 값이 있는 변수

다음 예제에서는 값이 **enp1s0** 인 **INET_DEV** 변수를 정의합니다.

```
define INET_DEV = enp1s0
```

$ 기호 다음에 변수 이름을 입력하여 스크립트에서 변수를 사용할 수 있습니다.

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

anonymous set를 포함하는 변수

다음 예제에서는 익명 집합을 포함하는 변수를 정의합니다.

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

$ 기호와 변수 이름을 작성하여 스크립트에서 변수를 사용할 수 있습니다.

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

참고

중괄호에는 변수가 집합을 나타내는 것을 나타내기 때문에 규칙에서 사용할 때 특수한 의미가 있습니다.

추가 리소스

- nftables 명령에서 세트 사용
- nftables 명령에서 검증 맵 사용

2.3.5. nftables 스크립트에 파일 포함

nftables 스크립팅 환경에서는 **include** 문을 사용하여 다른 스크립트를포함할 수 있습니다.

절대 또는 상대 경로 없이 파일 이름만 지정하는 경우 **nftables** 는 기본 검색 경로의 파일을 포함합니다. 이는 Red Hat Enterprise Linux의 **/etc** 로 설정됩니다.

예 2.1. 기본 검색 디렉터리의 파일 포함

기본 검색 디렉터리의 파일을 포함하려면 다음을 수행합니다.

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

예 2.2. 디렉터리의 모든 *.nft 파일 포함

/etc/nftables/rulesets/ 디렉토리에 저장된 ***.nft** 로 끝나는 모든 파일을 포함하려면 다음을 실행합니다.

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

include 문은 점으로 시작하는 파일과 일치하지 않습니다.

추가 리소스

- 시스템의 **nft(8)** 도움말 페이지의 파일 포함 섹션

2.3.6. 시스템 부팅 시 nftables 규칙 자동 로딩

nftables systemd 서비스는 **/etc/sysconfig/nftables.conf** 파일에 포함된 방화벽 스크립트를 로드합니다.

사전 요구 사항

- **nftables** 스크립트는 **/etc/nftables/** 디렉터리에 저장됩니다.

절차

1. **/etc/sysconfig/nftables.conf** 파일을 편집합니다.
   - **nftables** 패키지 설치로 **/etc/nftables/** 에 생성된 ***.nft** 스크립트를 수정한 경우 해당 스크립트에 대한 **include** 문의 주석 처리를 해제합니다.
   - 새 스크립트를 작성한 경우 **include** 문을 추가하여 이러한 스크립트를 포함합니다. 예를 들어 **nftables** 서비스가 시작될 때 **/etc/nftables/예제.nft** 스크립트를 로드하려면 다음을 추가합니다.

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

2. 선택 사항: 시스템을 재부팅하지 않고 **nftables** 서비스를 시작하여 방화벽 규칙을 로드합니다.

   ```
   # systemctl start nftables
   ```

3. **nftables** 서비스를 활성화합니다.

   ```
   # systemctl enable nftables
   ```

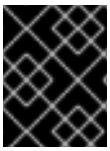
추가 리소스

- 지원되는 nftables 스크립트 형식

2.4. NFTABLES를 사용하여 NAT 구성

nftables 를 사용하면 다음 NAT(네트워크 주소 변환) 유형을 구성할 수 있습니다.

- 마스커레이딩
- 소스 NAT(SNAT)
- 대상 NAT(DNAT)
- redirect



중요

iifname 및 **oifname** 매개변수에서 실제 인터페이스 이름만 사용할 수 있으며 대체 이름(**ltname**)은 지원되지 않습니다.

2.4.1. NAT 유형

다음은 다양한 NAT(네트워크 주소 변환) 유형입니다.

마스커레이딩 및 소스 NAT (SNAT)

이러한 NAT 유형 중 하나를 사용하여 패킷의 소스 IP 주소를 변경합니다. 예를 들어, 인터넷 서비스 공급자(ISP)는 **10.0.0.0/8** 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하고 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 이러한 범위의 패킷의 소스 IP 주소를 공용 IP 주소에 매핑합니다.
마스커레이딩과 SNAT는 서로 매우 비슷합니다. 차이점은 다음과 같습니다.

- 마스커레이딩은 발신 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 발신 인터페이스가 동적 IP 주소를 사용하는 경우 masquerading을 사용합니다.
- SNAT는 패킷의 소스 IP 주소를 지정된 IP로 설정하고 발신 인터페이스의 IP를 동적으로 검색하지 않습니다. 따라서 SNAT는 마스커레이딩보다 빠릅니다. 발신 인터페이스가 고정 IP 주소를 사용하는 경우 SNAT를 사용합니다.

대상 NAT(DNAT)

이 NAT 유형을 사용하여 수신 패킷의 대상 주소와 포트를 다시 작성합니다. 예를 들어 웹 서버가 개인 IP 범위의 IP 주소를 사용하므로 인터넷에서 직접 액세스할 수 없는 경우 라우터에 DNAT 규칙을 설정하여 수신 트래픽을 이 서버로 리디렉션할 수 있습니다.

redirect

이 유형은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 DNAT의 특별한 경우입니다. 예를 들어 서비스가 표준 포트와 다른 포트에서 실행되는 경우 표준 포트에서 들어오는 트래픽을 이 특정 포트로 리디렉션할 수 있습니다.

2.4.2. nftables를 사용하여 마스커레이딩 구성

마스커레이딩을 사용하면 라우터에서 인터페이스를 통해 전송되는 패킷의 소스 IP를 인터페이스의 IP 주소로 동적으로 변경할 수 있습니다. 즉, 인터페이스에 새 IP가 할당되면 **nftables** 는 소스 IP를 교체할 때 새 IP를 자동으로 사용합니다.

ens3 인터페이스를 통해 호스트를 떠나는 패킷의 소스 IP를**ens3** 의 IP 세트로 교체합니다.

절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. 사전 설정 및 사 후 체인 을 테이블에 추가합니다.

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

중요

선점 체인에 규칙을 추가하지 않더라도**nftables** 프레임 워크에는 이 체인이 들어오는 패킷 응답을 일치해야 합니다.

셸에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록**--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3** 인터페이스에서 나가는 패킷과 일치하는**postrouting** 체인에 규칙을 추가합니다.

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

2.4.3. nftables 를 사용하여 소스 NAT 구성

라우터에서 SNAT(Source NAT)를 사용하면 인터페이스를 통해 전송된 패킷의 IP를 특정 IP 주소로 변경할 수 있습니다. 그런 다음 라우터는 발신 패킷의 소스 IP를 대체합니다.

절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. 사전 설정 및 사 후 체인 을 테이블에 추가합니다.

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

중요

사 후 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크를 사용하려면 이 체인이 발신 패킷 응답과 일치해야 합니다.

쉘에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3** 을 통해 나가는 패킷의 소스 IP를 **192.0.2.1** 로 대체하는후 라우팅 체인에 규칙을 추가합니다.

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

추가 리소스

- 특정 로컬 포트에서 들어오는 패킷을 다른 호스트로 전달

2.4.4. nftables 를 사용하여 대상 NAT 구성

대상 NAT(DNAT)를 사용하면 라우터의 트래픽을 인터넷에서 직접 액세스할 수 없는 호스트로 리디렉션할 수 있습니다.

예를 들어, DNAT를 사용하면 라우터가 포트 **80** 및 **443** 으로 전송된 들어오는 트래픽을 IP 주소 **192.0.2.1** 이 있는 웹 서버로 리디렉션합니다.

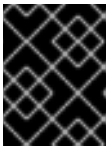
절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. 사전 설정 및 사 후 체인 을 테이블에 추가합니다.

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



중요

사 후 체인에 규칙을 추가하지 않더라도**nftables** 프레임워크를 사용하려면 이 체인이 발신 패킷 응답과 일치해야 합니다.

셸에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록**--** 옵션을**nft** 명령에 전달해야 합니다.

3. 라우터의 **ens3** 인터페이스에서 IP 주소**192.0.2.1** 을 사용하여 웹 서버로 들어오는 트래픽을 포트 **80** 및**443** 으로 리디렉션하는이전 체인에 규칙을 추가합니다.

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

4. 환경에 따라 SNAT 또는 마스커레이딩 규칙을 추가하여 웹 서버에서 보낸 사람에게 반환되는 패킷의 소스 주소를 변경합니다.

   a. **ens3** 인터페이스에서 동적 IP 주소를 사용하는 경우 masquerading 규칙을 추가합니다.

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

   b. **ens3** 인터페이스에서 고정 IP 주소를 사용하는 경우 SNAT 규칙을 추가합니다. 예를 들어**ens3** 에서 **198.51.100.1** IP 주소를 사용하는 경우:

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

5. 패킷 전달 활성화:

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

추가 리소스

- NAT 유형

2.4.5. nftables를 사용하여 리디렉션 구성

리디렉션 기능은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 대상 네트워크 주소 변환(DNAT) 특수한 경우입니다.

예를 들어 로컬 호스트의 포트 **22** 로 전송된 수신 및 전달 트래픽을 포트**2222** 로 리디렉션할 수 있습니다.

절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. 테이블에 사전 제한 체인 을 추가합니다.

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

셸에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록**--** 옵션을**nft** 명령에 전달해야 합니다.

3. 포트 **22** 에서 들어오는 트래픽을 포트 **2222** 로 리디렉션하는 사전 할당 체인에 규칙을 추가합니다.

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

추가 리소스

- NAT 유형

2.4.6. nftables 를 사용하여 flowtable 구성

nftables 유틸리티는 **netfilter** 프레임워크를 사용하여 네트워크 트래픽에 대해 NAT(네트워크 주소 변환)를 제공하고 패킷 전달을 가속화하기 위한 fastpath 기능 기반 흐름 가능 메커니즘을 제공합니다.

흐름 메커니즘에는 다음과 같은 기능이 있습니다.

- 연결 추적을 사용하여 클래식 패킷 전달 경로를 바이패스합니다.
- 클래식 패킷 처리를 우회하여 라우팅 테이블을 다시 방문하지 않도록 합니다.
- TCP 및 UDP 프로토콜에서만 작동합니다.
- 하드웨어 독립 소프트웨어 빠른 경로.

절차

1. **inet** family의 예제 테이블 추가:

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

2. **Ingress** 후크를 사용하여 **example-flowtable** flowtable을 추가하고 우선순위 유형으로 **filter** 를 추가합니다.

```
# nft add flowtable inet <example-table> <example-flowtable> { hook ingress priority
filter \; devices = { enp1s0, enp7s0 } \; }
```

3. 패킷 처리 테이블의 flowtable에 **example-forwardchain** flow를 추가합니다.

```
# nft add chain inet <example-table> <example-forwardchain> { type filter hook
forward priority filter \; }
```

이 명령은 전달 후크 및 필터 우선 순위로 필터 유형의 흐름을 추가합니다.

4. 설정된 연결 추적 상태가 포함된 규칙을 추가하여 **example-flowtable** 흐름을 오프로드합니다.

```
# nft add rule inet <example-table> <example-forwardchain> ct state established flow
add @<example-flowtable>
```

검증

- **example-table** 의 속성을 확인합니다.

```
# 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
    }
}
```

추가 리소스

- 시스템의 **nft(8)** 도움말 페이지

2.5. NFTABLES 명령에서 세트 사용

nftables 프레임워크는 기본적으로 세트를 지원합니다. 예를 들어 규칙이 여러 IP 주소, 포트 번호, 인터페이스 또는 기타 일치 기준과 일치해야 하는 경우 세트를 사용할 수 있습니다.

2.5.1. nftables 에서 익명 세트 사용

익명 세트에는 규칙에서 직접 사용하는 **{ 22, 80, 443 }** 과 같이 중괄호로 묶은 쉼표로 구분된 값이 포함되어 있습니다. IP 주소 및 기타 일치 기준에도 익명 세트를 사용할 수 있습니다.

익명 세트의 단점은 세트를 변경하려면 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 nftables에서 명명된 집합을 사용하는 데 설명된 대로 명명된 집합을 사용합니다.

사전 요구 사항

- **inet** 제품군의 **example_chain** 체인과 **example_table** 테이블이 있습니다.

절차

1. 예를 들어 포트 **22**,**80** 및 **443** 으로 들어오는 트래픽을 허용하는**example_table** 의 **example_chain** 에 규칙을 추가하려면 다음을 수행합니다.

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

2. 선택 사항: **example_table** 로 모든 체인과 해당 규칙을 표시하십시오.

```
# 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. nftables 에서 명명된 세트 사용

nftables 프레임워크는 set라는 변경 가능한 집합을 지원합니다. 명명된 집합은 테이블 내의 여러 규칙에 사용할 수 있는 요소 목록 또는 범위입니다.A named set is a list or range of elements that you can use in

multiple rules within a table. 익명 세트보다 다른 이점은 집합을 사용하는 규칙을 교체하지 않고도 명명된 집합을 업데이트할 수 있다는 것입니다.

명명된 집합을 만들 때는 세트에 포함된 요소 유형을 지정해야 합니다.When you create a named set, you must specify the type of elements the set contains. 다음 유형을 설정할 수 있습니다.

- **192.0.2.1** 또는 192.0.2.0/24와 같은 IPv4 주소 또는 범위가 포함된 세트의**ipv4_ addr** .
- **2001:db8:1::1** 또는 **2001:db8:1::1/64** 와 같은 IPv6 주소 또는 범위가 포함된 세트의**ipv6_addr**.
- **52:54:00:6b:66:42** 와 같은 MAC(Media Access Control) 주소 목록이 포함된 세트의**ether_addr**.
- **inet_proto** 는 **tcp** 와 같은 인터넷 프로토콜 유형 목록이 포함된 세트의 경우입니다.
- **ssh** 와 같은 인터넷 서비스 목록이 포함된 세트의**inet_service**.
- 패킷 표시 목록이 포함된 집합에 대한 **Mark** 입니다. 패킷 마크는 모든 양의 32 비트 정수 값 **(0 ~ 2147483647)** 일 수 있습니다.

사전 요구 사항

- **example_chain** 체인과 **example_table** 테이블이 있습니다.

절차

1. 빈 세트를 생성합니다. 다음 예제에서는 IPv4 주소에 대한 세트를 생성합니다.
   - 여러 개의 개별 IPv4 주소를 저장할 수 있는 세트를 만들려면 다음을 수행합니다.

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

   - IPv4 주소 범위를 저장할 수 있는 세트를 만들려면 다음을 수행합니다.

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

   중요

   쉘이 명령 끝부분을 해석하지 못하도록 하려면 백슬래시를 사용하여 host를 이스케이프해야 합니다.

2. 선택 사항: 세트를 사용하는 규칙을 만듭니다. 예를 들어 다음 명령은 **example_table** 의 **example_chain** 에 **example_set** 의 IPv4 주소에서 모든 패킷을 삭제하는 규칙을 추가합니다.

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

   example_set 는 여전히 비어 있기 때문에 현재 규칙이 적용되지 않습니다.

3. **example_set** 에 IPv4 주소를 추가합니다.
   - 개별 IPv4 주소를 저장하는 세트를 생성하는 경우 다음을 입력합니다.

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

   - IPv4 범위를 저장하는 세트를 생성하는 경우 다음을 입력합니다.

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

IP 주소 범위를 지정하면 위 예제에서 **192.0.2.0/24** 와 같은 CIDR(Classless Inter-Domain Routing) 표기법을 사용할 수도 있습니다.

2.5.3. 추가 리소스

- 시스템의 **nft(8)** 도움말 페이지의 설정 섹션

2.6. NFTABLES 명령에서 VERDICT 맵 사용

사전이라고도 하는 verdict 맵을 사용하면 **nft** 가 일치 기준을 작업에 매핑하여 패킷 정보에 따라 작업을 수행할 수 있습니다.

2.6.1. nftables 에서 익명 맵 사용

익명 맵은 **{ match_criteria : action }** 문입니다. 문에는 쉼표로 구분된 여러 매핑이 포함될 수 있습니다.

익명 맵의 단점은 맵을 변경하려면 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 nftables에서 명명된 맵을 사용하는 데 설명된 대로 명명된 맵을 사용합니다.

예를 들어 익명 맵을 사용하여 IPv4 및 IPv6 프로토콜의 TCP 및 UDP 패킷을 서로 다른 체인으로 라우팅하여 들어오는 TCP 및 UDP 패킷을 별도로 계산할 수 있습니다.

절차

1. 새 테이블을 만듭니다.

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

2. **example_table** 에 **tcp_packets** 체인을 생성합니다.

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

3. 이 체인에서 트래픽을 계산하는 **tcp_packets** 에 규칙을 추가합니다.

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

4. **example_table**에 **udp_packets** 체인 생성

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

5. 이 체인에서 트래픽을 계산하는 **udp_packets** 에 규칙을 추가합니다.

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

6. 들어오는 트래픽에 대한 체인을 만듭니다. 예를 들어, **example_table** 에서 들어오는 트래픽을 필터링하는 **incoming_traffic** 이라는 체인을 생성하려면 다음을 수행합니다.

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

7. anonymous 맵을 사용하여 **incoming_traffic** 에 규칙을 추가합니다.

```
# nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump
tcp_packets, udp : jump udp_packets }
```

익명 맵은 패킷을 구분하여 프로토콜을 기반으로 다양한 카운터 체인으로 전송합니다.

8. 트래픽 카운터를 나열하려면 **example_table** 을 표시합니다.

```
# 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 및 **udp_packets** 체인의 카운터는 수신된 패킷과 바이트 수를 모두 표시합니다.

2.6.2. nftables 에서 명명된 맵 사용

nftables 프레임워크는 이름이 지정된 맵을 지원합니다. 이러한 맵을 테이블 내에서 여러 규칙에 사용할 수 있습니다. 익명 맵의 또 다른 장점은 해당 맵을 사용하는 규칙을 교체하지 않고도 명명된 맵을 업데이트할 수 있다는 것입니다.

명명된 맵을 생성할 때 요소 유형을 지정해야 합니다.

- 일치하는 부분이 **192.0.2.1** 과 같은 IPv4 주소가 포함된 맵의 **ipv4_addr**.
- 일치하는 부분이 **2001:db8:1::1** 과 같은 IPv6 주소가 포함된 맵의 **ipv6_addr**
- 일치하는 부분에 미디어 액세스 제어(MAC) 주소가 포함된 맵의 **ether_addr** (예 : **52:54:00:6b:66:42**).
- **inet_proto** 일치하는 부분이 있는 맵의 경우 **tcp** 와 같은 인터넷 프로토콜 유형이 포함되어 있습니다.
- **inet_service** 일치하는 맵의 경우 **ssh** 또는 **22** 와 같은 인터넷 서비스 이름 포트 번호가 포함되어 있습니다.
- 일치하는 부분에 패킷 표시 가 포함된 맵에 표시됩니다. 패킷 마크는 모든 양의 32 비트 정수 값 (**0** ~ **2147483647**) 일 수 있습니다.
- 일치하는 부분에 카운터 값이 포함된 맵의 카운터입니다. counter 값은 모든 양의 64 비트 정수 값이 될 수 있습니다.
- 일치하는 부분에 할당량 값이 포함된 맵의 할당량입니다. 할당량 값은 모든 양의 64 비트 정수 값일 수 있습니다.

예를 들어 소스 IP 주소에 따라 들어오는 패킷을 허용하거나 삭제할 수 있습니다. 명명된 맵을 사용하면 IP 주소와 작업이 맵에 동적으로 저장되는 동안 이 시나리오를 구성하는 단일 규칙만 필요합니다.

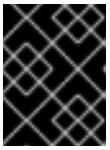
절차

1. *테이블을 만듭니다. 예를 들어, IPv4 패킷을 처리하는 **example_table** 이라는 테이블을 생성하려면 다음을 실행합니다.*

```
# nft add table ip example_table
```

2. *체인을 만듭니다. 예를 들어 **example_table** 에서 **example_chain** 이라는 체인을 생성하려면 다음을 수행합니다.*

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



중요

쉘이 명령 끝부분을 해석하지 못하도록 하려면 백슬래시를 사용하여 host를 이스케이프해야 합니다.

3. *빈 맵을 만듭니다. 예를 들어 IPv4 주소에 대한 맵을 생성하려면 다음을 수행합니다.*

```
# nft add map ip example_table example_map { type ipv4_addr : verdict \; }
```

4. *맵을 사용하는 규칙을 생성합니다. 예를 들어 다음 명령은 **example_table** 의 **example_chain** 에 규칙을 추가하여 **example_map** 에 모두 정의된 IPv4 주소에 작업을 적용합니다.*

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. *IPv4 주소와 해당 작업을 **example_map** 에 추가합니다.*

```
# nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

*이 예제에서는 작업에 대한 IPv4 주소 매핑을 정의합니다. 위에서 만든 규칙과 함께 방화벽은 **192.0.2.1** 에서 패킷을 수락하고 **192.0.2.2** 에서 패킷을 삭제합니다.*

6. *선택 사항: 다른 IP 주소 및 action 문을 추가하여 맵을 개선합니다.*

```
# nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7. *선택 사항: 맵에서 항목을 제거합니다.*

```
# nft delete element ip example_table example_map { 192.0.2.1 }
```

8. *선택 사항: 규칙 세트를 표시합니다.*

```
# 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. 추가 리소스

- *시스템의 **nft(8)** 도움말 페이지의 **Maps** 섹션*

2.7. 예제: NFTABLES 스크립트를 사용하여 LAN 및 DMZ 보호

*RHEL 라우터의 **nftables** 프레임워크를 사용하여 내부 LAN의 네트워크 클라이언트와 DMZ의 웹 서버를 인터넷 및 기타 네트워크에서 무단 액세스로부터 보호하는 방화벽 스크립트를 작성하고 설치합니다.*

중요

이 예는 예시 목적으로만 사용되며 특정 요구 사항이 있는 시나리오를 설명합니다.

*방화벽 스크립트는 네트워크 인프라 및 보안 요구 사항에 따라 크게 달라집니다. 사용자 환경에 대한 스크립트를 작성할 때 **nftables** 방화벽의 개념을 알아보려면 이 예제를 사용합니다.*

2.7.1. 네트워크 조건

이 예제의 네트워크에는 다음 조건이 있습니다.

- *라우터는 다음 네트워크에 연결되어 있습니다.*
  - *인터페이스 **enp1s0**을 통한 인터넷*
  - *내부 LAN through 인터페이스 **enp7s0***
  - ***enp8s0**을 통한 DMZ*
- *라우터의 인터넷 인터페이스에는 정적 IPv4 주소(**203.0.113.1**)와 IPv6 주소(**2001:db8:a::1**)가 할당되어 있습니다.*
- *내부 LAN의 클라이언트는 **10.0.0.0/24** 범위의 개인 IPv4 주소만 사용합니다. 결과적으로 LAN에서 인터넷으로 전송되는 경우 소스 네트워크 주소 변환(SNAT)이 필요합니다.*
- *내부 LAN의 관리자는 IP 주소 **10.0.0.100** 및 **10.0.0.200** 을 사용합니다.*
- *DMZ는 **198.51.100.0/24** 및 **2001:db8:b::/56** 범위의 공용 IP 주소를 사용합니다.*
- *DMZ의 웹 서버는 **198.51.100.5** 및 **2001:db8:b::5** IP 주소를 사용합니다.*
- *라우터는 LAN 및 DMZ에 있는 호스트에 대한 캐싱 DNS 서버 역할을 합니다.*

2.7.2. 방화벽 스크립트에 대한 보안 요구 사항

*다음은 예제 네트워크의 **nftables** 방화벽에 대한 요구 사항입니다.*

- *라우터는 다음을 수행할 수 있어야 합니다.*

- DNS 쿼리를 반복적으로 확인합니다.
- 루프백 인터페이스에서 모든 연결을 수행합니다.

- 내부 LAN의 클라이언트는 다음을 수행할 수 있어야 합니다.
  - 라우터에서 실행 중인 캐싱 DNS 서버를 쿼리합니다.
  - DMZ의 HTTPS 서버에 액세스합니다.
  - 인터넷의 모든 HTTPS 서버에 액세스합니다.
- 관리자는 SSH를 사용하여 라우터 및 DMZ의 모든 서버에 액세스할 수 있어야 합니다.
- DMZ의 웹 서버는 다음을 수행할 수 있어야 합니다.
  - 라우터에서 실행 중인 캐싱 DNS 서버를 쿼리합니다.
  - 인터넷의 HTTPS 서버에 액세스하여 업데이트를 다운로드합니다.
- 인터넷의 호스트는 다음을 수행할 수 있어야 합니다.
  - DMZ의 HTTPS 서버에 액세스합니다.
- 또한 다음과 같은 보안 요구 사항이 있습니다.
  - 명시적으로 허용되지 않은 연결 시도는 삭제해야 합니다.
  - 삭제된 패킷이 기록되어야 합니다.

2.7.3. 삭제된 패킷의 로깅 구성

기본적으로 **systemd** 는 삭제된 패킷과 같은 커널 메시지를 저널에 기록합니다. 또한 이러한 항목을 별도의 파일에 기록하도록 **rsyslog** 서비스를 구성할 수 있습니다. 로그 파일이 무한대로 확장되지 않도록 하려면 순환 정책을 구성합니다.

사전 요구 사항

- **rsyslog** 패키지가 설치되어 있어야 합니다.
- **rsyslog** 서비스가 실행 중입니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/ECDHE.d/nftables.conf** 파일을 만듭니다.

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

이 구성을 사용하여 **rsyslog** 서비스는 /var/log/ECDHE 대신 **/var/log/nftables.log** 파일에 패킷을 로그했습니다.

2. **rsyslog** 서비스를 다시 시작하십시오.

```
# systemctl restart rsyslog
```

3. 크기가 10MB를 초과하는 경우 /etc/logrotate.d/nftables.log를 교체하여 /var/log/nftables.log 를 순환하도록 /etc/logrotate.d/nftables 파일을 만듭니다.

```
/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 설정은 다음 순환 작업 중에 30일이 지난 순환 로그를 제거하도록 정의합니다.

추가 리소스

- rsyslog.conf(5) 및 logrotate(8) 도움말 페이지

2.7.4. nftables 스크립트 작성 및 활성화

이 예는 RHEL 라우터에서 실행되며 내부 LAN 및 DMZ의 웹 서버에서 클라이언트를 보호하는 nftables 방화벽 스크립트입니다. 예제에 사용된 방화벽의 네트워크 및 요구 사항에 대한 자세한 내용은 방화벽 스크립트에 대한 네트워크 조건 및 보안 요구 사항을 참조하십시오.

주의

이 nftables 방화벽 스크립트는 데모 목적으로만 사용됩니다. 환경 및 보안 요구 사항에 맞게 조정하지 않고 사용하지 마십시오.

사전 요구 사항

- 네트워크는 네트워크 조건에 설명된 대로 구성됩니다.

절차

1. 다음 콘텐츠를 사용하여 /etc/nftables/firewall.nft 스크립트를 만듭니다.

```
# 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. **/etc/nftables/firewall.nft** 스크립트를 **/etc/sysconfig/nftables.conf** 파일에 포함합니다.

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

3. **IPv4** 전달을 활성화합니다.

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

4. **nftables** 서비스를 활성화하고 시작합니다.

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

검증

1. 선택 사항: nftables 규칙 세트를 확인합니다.

```
# nft list ruleset
...
```

2. 방화벽에서 방지하는 액세스를 시도합니다. 예를 들어 DMZ에서 SSH를 사용하여 라우터에 액세스하십시오.

```
# ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. 로깅 설정에 따라 검색합니다.

- 차단된 패킷의 systemd 저널:

```
# 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 ...
```

- 차단된 패킷의 /var/log/nftables.log 파일:

```
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. NFTABLES를 사용하여 포트 전달 구성

포트 전달을 사용하면 관리자가 특정 대상 포트로 전송된 패킷을 다른 로컬 또는 원격 포트로 전달할 수 있습니다.

예를 들어 웹 서버에 공용 IP 주소가 없는 경우 방화벽의 포트 80 및 443 의 수신 패킷을 웹 서버로 전달하는 방화벽에서 포트 전달 규칙을 설정할 수 있습니다. 이 방화벽 규칙을 사용하면 인터넷 사용자는 방화벽의 IP 또는 호스트 이름을 사용하여 웹 서버에 액세스할 수 있습니다.

2.8.1. 들어오는 패킷을 다른 로컬 포트로 전달

nftables 를 사용하여 패킷을 전달할 수 있습니다. 예를 들어 8022 포트의 수신 IPv4 패킷을 로컬 시스템의 포트 22 로 전달할 수 있습니다.

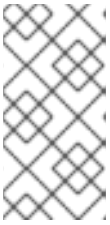
절차

1. *ip address family*를 사용하여 *nat* 라는 테이블을 만듭니다.

```
# nft add table ip nat
```

2. 사전 설정 및 사 후 체인 을 테이블에 추가합니다.

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



참고

nft 명령에 *--* 옵션을 전달하여 쉘에서 음수 우선 순위 값을 *nft* 명령의 옵션으로 해석하지 못하도록 합니다.

3. 포트 *802*에서 들어오는 패킷을 로컬 포트 *22* 로 리디렉션하는 사전 할당 체인에 규칙을 추가합니다.

```
# nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

2.8.2. 특정 로컬 포트에서 들어오는 패킷을 다른 호스트로 전달

대상 네트워크 주소 변환*(DNAT)* 규칙을 사용하여 로컬 포트에서 들어오는 패킷을 원격 호스트로 전달할 수 있습니다. 이를 통해 인터넷의 사용자는 개인 *IP* 주소가 있는 호스트에서 실행되는 서비스에 액세스할 수 있습니다.

예를 들어 로컬 포트 *443* 에서 들어오는 *IPv4* 패킷을 *192.0.2.1 IP* 주소를 사용하여 원격 시스템의 동일한 포트 번호로 전달할 수 있습니다.

사전 요구 사항

- 패킷을 전달해야 하는 시스템에 *root* 사용자로 로그인되어 있습니다.

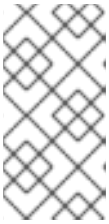
절차

1. ip address family를 사용하여 nat 라는 테이블을 만듭니다.

```
# nft add table ip nat
```

2. 사전 설정 및 사 후 체인 을 테이블에 추가합니다.

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



참고

nft 명령에 -- 옵션을 전달하여 쉘에서 음수 우선 순위 값을 nft 명령의 옵션으로 해석하지 못하도록 합니다.

3. 포트 443 에서 들어오는 패킷을 192.0.2.1 의 동일한 포트로 리디렉션하는 사전 할당 체인에 규칙을 추가합니다.

```
# nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4. 사 후 체인에 규칙을 추가하여 나가는 트래픽을 마스커레이드합니다.

```
# nft add rule ip nat postrouting daddr 192.0.2.1 masquerade
```

5. 패킷 전달 활성화:

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

2.9. NFTABLES를 사용하여 연결 양 제한

nftables 를 사용하여 연결 수를 제한하거나 지정된 연결 양을 설정하려는 IP 주소를 차단하여 너무 많은 시스템 리소스를 사용할 수 있습니다.

2.9.1. nftables를 사용하여 연결 수 제한

nft 유틸리티의 ct count 매개변수를 사용하면 IP 주소당 동시 연결 수를 제한할 수 있습니다. 예를 들어 이 기능을 사용하여 각 소스 IP 주소가 호스트에 대한 두 개의 병렬 SSH 연결만 설정할 수 있도록 구성

할 수 있습니다.

절차

1. ***inet** 주소 제품군을 사용하여 **filter** 테이블을 생성합니다.*

```
# nft add table inet filter
```

2. ***inet** 필터 테이블에 입력 체인을 추가합니다.*

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

3. ***IPv4** 주소에 대한 동적 세트를 생성합니다.*

```
# nft add set inet filter limit-ssh { type ipv4_addr\; flags dynamic \;}
```

4. ***IPv4** 주소에서 **SSH** 포트(22)에 동시에 들어오는 연결만 허용하는 입력 체인에 규칙을 추가하고 동일한 **IP**에서 추가 연결을 모두 거부합니다.*

```
# nft add rule inet filter input tcp dport ssh ct state new add @limit-ssh { ip saddr ct count over 2 } counter reject
```

검증

1. *동일한 **IP** 주소에서 호스트로의 두 개 이상의 새 동시 **SSH** 연결을 설정합니다. 두 연결이 이미 설정된 경우 **nftables**는 **SSH** 포트에 대한 연결을 거부합니다.*

2. ***limit-ssh** 측정을 표시합니다.*

```
# nft list set inet filter limit-ssh
table inet filter {
  set limit-ssh {
    type ipv4_addr
    size 65535
    flags dynamic
    elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2 }
  }
}
```

***elements** 항목은 현재 규칙과 일치하는 주소를 표시합니다. 이 예에서 요소는 **SSH** 포트에*

대한 활성 연결이 있는 IP 주소를 나열합니다. 출력에 활성 연결 수 또는 연결이 거부된 경우 출력은 표시되지 않습니다.

2.9.2. 1분 이내에 10개 이상의 들어오는 TCP 연결을 시도하는 IP 주소 차단

1분 이내에 10개 이상의 IPv4 TCP 연결을 설정하는 호스트를 일시적으로 차단할 수 있습니다.

절차

1. ip address family로 filter 테이블을 생성합니다.

```
# nft add table ip filter
```

2. 필터 테이블에 입력 체인을 추가합니다.

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

3. filter 테이블에 denylist라 는 세트를 추가합니다.

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

이 명령은 IPv4 주소에 대한 동적 세트를 생성합니다. timeout 5m 매개변수는 설정된 값이 오래된 항목으로 채워지지 않도록 5분 후에 nftables 가 자동으로 항목을 제거함을 정의합니다.

4. 1분 내에 새 TCP 연결을 허용하려는 호스트의 소스 IP 주소를 거부 목록에 자동으로 추가하는 규칙을 추가합니다.

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

추가 리소스

- nftables에서 명명된 세트 사용

2.10. NFTABLES 규칙 디버깅

nftables 프레임워크는 관리자가 규칙을 디버그하고 패킷이 일치하는 경우 다양한 옵션을 제공합니다.

2.10.1. 카운터를 사용하여 규칙 생성

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다.

- 기존 규칙에 카운터를 추가하는 프로시저에 대한 자세한 내용은 기존 규칙에 카운터 추가를 참조하십시오.

사전 요구 사항

- 규칙을 추가하려는 체인이 있습니다.

절차

1. **counter** 매개 변수를 사용하여 새 규칙을 체인에 추가합니다. 다음 예제에서는 포트 **22**에서 **TCP** 트래픽을 허용하는 카운터가 있는 규칙을 추가하고 이 규칙과 일치하는 패킷 및 트래픽을 계산합니다.

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

2. 카운터 값을 표시하려면 다음을 수행합니다.

```
# 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. 기존 규칙에 카운터 추가

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다.

- 카운터를 사용하여 새 규칙을 추가하는 프로시저에 대한 자세한 내용은 카운터를 사용하여 규칙 생성을 참조하십시오.

사전 요구 사항

- *카운터를 추가하려는 규칙이 있습니다.*

절차

1. *핸들을 포함하여 체인의 규칙을 표시합니다.*

```
# 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. *규칙을 대체하지만 **counter** 매개 변수로 교체하여 카운터 를 추가합니다. 다음 예제에서는 이전 단계에서 표시된 규칙을 교체하고 카운터를 추가합니다.*

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter accept
```

3. *카운터 값을 표시하려면 다음을 수행합니다.*

```
# 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. 기존 규칙과 일치하는 패킷 모니터링

__nft monitor__ 명령과 함께 __nftables__ 의 추적 기능을 사용하면 관리자가 규칙과 일치하는 패킷을 표시할 수 있습니다. 이 규칙과 일치하는 패킷을 모니터링하는 데 사용하는 규칙에 대한 추적을 활성화할 수 있습니다.

사전 요구 사항

- *카운터를 추가하려는 규칙이 있습니다.*

절차

1. *핸들을 포함하여 체인의 규칙을 표시합니다.*

```
# 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. *규칙을 교체하지만* **meta nftrace set 1** *매개변수로 추적 기능을 추가합니다. 다음 예제에서는 이전 단계에서 표시된 규칙을 교체하고 추적을 활성화합니다.*

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace set 1 accept
```

3. **nft monitor** *명령 을 사용하여 추적을 표시합니다. 다음 예제에서는 명령의* *출력을 필터링하여* **inet example_table example_chain** *이 포함된 항목만 표시합니다.*

```
# 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)
...
```

주의

추적이 활성화된 규칙 수 및 일치하는 트래픽 양에 따라 **nft monitor** *명령은 많은 출력을 표시할 수 있습니다.* **grep** *또는 기타 유틸리티를 사용하여 출력을 필터링합니다.*

2.11. NFTABLES 규칙 세트 백업 및 복원

nftables *규칙을 파일에 백업하고 나중에 복원할 수 있습니다. 또한 관리자는 규칙과 함께 파일을 사용하여 다른 서버로 규칙을 전송할 수도 있습니다.*

2.11.1. 파일에 nftables 규칙 세트 백업

nft 유틸리티를 사용하여 nftables 규칙 세트를 파일로 백업할 수 있습니다.

절차

- nftables 규칙을 백업하려면 다음을 수행합니다.
  - nft list ruleset 형식으로 생성된 형식의 경우:

```
# nft list ruleset > file.nft
```

  - JSON 형식의 경우:

```
# nft -j list ruleset > file.json
```

2.11.2. 파일에서 nftables 규칙 세트 복원

파일에서 nftables 규칙 세트를 복원할 수 있습니다.

절차

- nftables 규칙을 복원하려면 다음을 수행합니다.
  - 복원할 파일이 nft list ruleset 에 의해 생성된 형식으로 되어 있거나 nft 명령이 직접 포함 된 경우:

```
# nft -f file.nft
```

  - 복원할 파일이 JSON 형식인 경우:

```
# nft -j -f file.json
```

2.12. 추가 리소스

- *Red Hat Enterprise Linux 8에서 nftables 사용*
- *iptables의 뒤에는 무엇이 있습니까? 후속 조치: nftables*
- *Firewalld: 미래는 nftables입니다.*

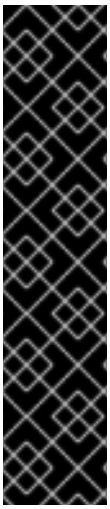
3장. DDOS 공격을 방지하기 위해 고성능 트래픽 필터링에 XDP-FILTER 사용

nftables, Express Data Path(XDP) 프로세스와 같은 패킷 필터와 비교하여 네트워크 패킷을 네트워크 인터페이스에서 바로 삭제합니다. 따라서 **XDP**는 방화벽 또는 기타 애플리케이션에 도달하기 전에 패키지의 다음 단계를 결정합니다. 결과적으로 **XDP** 필터는 더 적은 리소스가 필요하며, 분산 서비스 거부 **(DDoS)** 공격을 방어하기 위해 기존 패킷 필터보다 훨씬 더 높은 속도로 네트워크 패킷을 처리할 수 있습니다. 예를 들어, 테스트 중에 **Red Hat**은 단일 코어에서 초당 **26**만 개의 네트워크 패킷을 삭제했으며 동일한 하드웨어에서 **nftables** 의 드롭 속도보다 상당히 높습니다.

xdp-filter 유틸리티는 **XDP**를 사용하여 들어오는 네트워크 패킷을 허용하거나 삭제합니다. 특정 간에 트래픽을 필터링하는 규칙을 생성할 수 있습니다.

- **IP** 주소
- **MAC** 주소
- 포트

xdp-filter 에 상당히 높은 패킷 처리 속도가 있더라도 **nftables** 와 동일한 기능이 없습니다. **XDP**를 사용하여 패킷 필터링을 시연하려면 **xdp-filter** 를 개념적 유틸리티로 고려해 보십시오. 또한 자체 **XDP** 애플리케이션을 작성하는 방법을 더 잘 이해하기 위해 유틸리티 코드를 사용할 수 있습니다.



중요

AMD 및 **Intel 64**비트 이외의 아키텍처에서 **xdp-filter** 유틸리티는 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약**(SLA)**에서 지원되지 않으며 기능적으로 완전하지 않을 수 있으며 **Red Hat**은 프로덕션에 사용하지 않는 것이 좋습니다. 이러한 프리뷰를 통해 향후 제품 기능에 조기에 액세스할 수 있으므로 고객은 개발 과정에서 기능을 테스트하고 피드백을 제공할 수 있습니다.

기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 **Red Hat** 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

3.1. XDP-FILTER 규칙과 일치하는 네트워크 패킷 삭제

xdp-filter 를 사용하여 네트워크 패킷을 삭제할 수 있습니다.

- *특정 대상 포트*
- *특정 **IP** 주소*
- *특정 **MAC** 주소*

***xdp-filter** 의 **allow** 정책은 모든 트래픽이 허용되고 필터는 특정 규칙과 일치하는 네트워크 패킷만 삭제하도록 정의합니다. 예를 들어 삭제할 패킷의 소스 **IP** 주소를 알고 있는 경우 이 방법을 사용합니다.*

사전 요구 사항

- ***xdp-tools** 패키지가 설치되어 있습니다.*
- ***XDP** 프로그램을 지원하는 네트워크 드라이버입니다.*

절차

1. ***xdp-filter** 를 로드하여 **enp1s0** 과 같은 특정 인터페이스에서 들어오는 패킷을 처리하십시오.*

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

*기본적으로 **xdp-filter** 는 **allow** 정책을 사용하고 유틸리티는 모든 규칙과 일치하는 트래픽만 삭제합니다.*

*선택적으로 **-f** 기능 옵션을 사용하여 **tcp,ipv4** 또는 **ethernet** 과 같은 특정 기능만 활성화합니다. 필요한 기능만 로드하면 패킷 처리 속도가 향상됩니다. 여러 기능을 활성화하려면 쉼표로 구분합니다.*

*오류와 함께 명령이 실패하면 네트워크 드라이버는 **XDP** 프로그램을 지원하지 않습니다.*

2. *일치하는 패킷을 삭제하는 규칙을 추가합니다. 예를 들면 다음과 같습니다.*

- 포트 **22** 로 들어오는 패킷을 삭제하려면 다음을 입력합니다.

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

이 명령은 **TCP** 및 **UDP** 트래픽과 일치하는 규칙을 추가합니다. 특정 프로토콜만 일치시키려면 **-p protocol** 옵션을 사용합니다.

- **192.0.2.1** 에서 들어오는 패킷을 삭제하려면 다음을 입력합니다.

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

xdp-filter 는 **IP** 범위를 지원하지 않습니다.

- **MAC** 주소 **00:53:00:07:BE** 에서 들어오는 패킷을 삭제하려면 다음을 입력합니다.

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

검증

- 다음 명령을 사용하여 삭제 및 허용된 패킷에 대한 통계를 표시합니다.

```
# xdp-filter status
```

추가 리소스

- 시스템의 **XDP-filter(8)** 도움말 페이지

- 개발자이고 **xdp-filter** 의 코드에 관심이 있는 경우 **Red Hat** 고객 포털에서 해당 소스 **RPM(SRPM)**을 다운로드하여 설치합니다.

3.2. XDP-FILTER 규칙과 일치하는 항목을 제외한 모든 네트워크 패킷 삭제

xdp-filter 를 사용하여 네트워크 패킷만 허용할 수 있습니다.

- 특정 대상 포트(From and to a specific destination port)
- 특정 IP 주소의 출처 및 추가Adding from and to a specific IP address
- 특정 MAC 주소 (From and to specific MAC address)

이렇게 하려면 필터가 특정 규칙과 일치하는 항목을 제외하고 모든 네트워크 패킷을 삭제하도록 정의하는 **xdp-filter** 의 거부 정책을 사용합니다. 예를 들어 삭제할 패킷의 소스 IP 주소를 모르는 경우 이 방법을 사용합니다.

주의

인터페이스에서 **xdp-filter** 를 로드할 때 거부 하도록 기본 정책을 설정하면 커널은 특정 트래픽을 허용하는 규칙을 생성할 때까지 이 인터페이스에서 모든 패킷을 즉시 삭제합니다. 시스템에서 잠기지 않도록 하려면 명령을 로컬로 입력하거나 다른 네트워크 인터페이스를 통해 호스트에 연결합니다.

사전 요구 사항

- **xdp-tools** 패키지가 설치되어 있습니다.
- 로컬 또는 트래픽을 필터링할 네트워크 인터페이스를 사용하여 호스트에 로그인되어 있습니다.
- **XDP** 프로그램을 지원하는 네트워크 드라이버입니다.

절차

1. **xdp-filter** 를 로드하여 **enp1s0** 과 같은 특정 인터페이스에서 패킷을 처리하십시오.

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

선택적으로 **-f** 기능 옵션을 사용하여 **tcp,ipv4** 또는 **ethernet** 과 같은 특정 기능만 활성화합니다. 필요한 기능만 로드하면 패킷 처리 속도가 향상됩니다. 여러 기능을 활성화하려면 쉼표로 구분합니다.

오류와 함께 명령이 실패하면 네트워크 드라이버는 **XDP** 프로그램을 지원하지 않습니다.

2. 일치하는 패킷을 허용하는 규칙을 추가합니다. 예를 들면 다음과 같습니다.

- **22** 번의 패킷을 허용하려면 다음을 입력합니다.

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

이 명령은 **TCP** 및 **UDP** 트래픽과 일치하는 규칙을 추가합니다. 특정 프로토콜만 일치시키려면 **-p protocol** 옵션을 명령에 전달합니다.

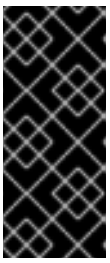
- **192.0.2.1** 패킷을 허용하려면 다음을 입력합니다.

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

xdp-filter 는 **IP** 범위를 지원하지 않습니다.

- **MAC** 주소로 패킷을 허용하려면 **00:53:00:07:BE** 를 입력합니다.

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



중요

xdp-filter 유틸리티는 상태 저장 패킷 검사를 지원하지 않습니다. 이를 위해서는 **-m mode** 옵션을 사용하여 모드를 설정하지 않거나 시스템이 나가는 트래픽에 응답하는 들어오는 트래픽을 허용하는 명시적 규칙을 추가해야 합니다.

검증

- 다음 명령을 사용하여 삭제 및 허용된 패킷에 대한 통계를 표시합니다.

```
# xdp-filter status
```

-

추가 리소스

- *시스템의 **XDP-filter(8)** 도움말 페이지*

- *개발자이고 **xdp-filter** 의 코드에 관심이 있는 경우 **Red Hat** 고객 포털에서 해당 소스 **RPM(SRPM)**을 다운로드하여 설치합니다.*