Red Hat Enterprise Linux 8

네트워크 보안

보안 네트워크 및 네트워크 통신 구성

Last Updated: 2022-09-04

Red Hat Enterprise Linux 8 네트워크 보안

보안 네트워크 및 네트워크 통신 구성

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

법적 공지

초록

이 제목은 관리자가 네트워크, 연결된 시스템 및 다양한 공격에 대한 네트워크 통신을 보호하는 데 도움이 됩니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

특정 문구에 대한 의견 제출

1. **Multi-page HTML** 형식으로 설명서를 보고 페이지가 완전히 로드된 후 오른쪽 상단 모서리에 **피드백** 버튼이 표시되는지 확인합니다.
2. 커서를 사용하여 주석 처리할 텍스트 부분을 강조 표시합니다.
3. 강조 표시된 텍스트 옆에 표시되는 **피드백 추가** 버튼을 클릭합니다.
4. 의견을 추가하고 **제출** 을 클릭합니다.

Bugzilla를 통해 피드백 제출(등록 필요)

1. Bugzilla 웹 사이트에 로그인합니다.
2. **버전** 메뉴에서 올바른 버전을 선택합니다.
3. **Summary** (요약) 필드에 설명 제목을 입력합니다.
4. **Description** (설명) 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. **Submit Bug**를 클릭하십시오.

1장. OPENSSH로 두 시스템 간의 보안 통신 사용

SSH(Secure Shell)는 클라이언트-서버 아키텍처를 사용하여 두 시스템 간에 보안 통신을 제공하고 사용자가 서버 호스트 시스템에 원격으로 로그인할 수 있는 프로토콜입니다. FTP 또는 Telnet과 같은 다른 원격 통신 프로토콜과 달리 SSH는 로그인 세션을 암호화하여 침입자가 연결에서 암호화되지 않은 암호를 수집하지 못하게 합니다.

Red Hat Enterprise Linux에는 일반 **openssh** 패키지, **openssh-server** 패키지, **openssh-clients** 패키지 등 기본 **OpenSSH** 패키지가 포함되어 있습니다. **OpenSSH** 패키지에는 **OpenSSH**가 암호화된 통신을 제공할 수 있는 몇 가지 중요한 암호화 라이브러리를 설치하는 **OpenSSL** 패키지 **openssl-libs**가 있어야 합니다.

1.1. SSH 및 OPENSSH

SSH(Secure Shell)는 원격 시스템에 로그인하고 해당 시스템에서 명령을 실행하는 프로그램입니다. SSH 프로토콜은 비보안 네트워크를 통해 신뢰할 수 없는 두 호스트 간에 안전한 암호화된 통신을 제공합니다. 보안 채널을 통해 X11 연결 및 임의의 TCP/IP 포트를 전달할 수도 있습니다.

SSH 프로토콜은 원격 쉘 로그인 또는 파일 복사에 사용할 때 두 시스템 간 통신 가로채기 및 특정 호스트의 가장과 같은 보안 위협을 완화합니다. 이는 SSH 클라이언트와 서버가 디지털 서명을 사용하여 신원을 확인하기 때문입니다. 또한 클라이언트와 서버 시스템 간의 모든 통신이 암호화됩니다.

호스트 키는 SSH 프로토콜에서 호스트를 인증합니다. 호스트 키는 OpenSSH가 처음 설치되거나 호스트가 처음 부팅될 때 자동으로 생성되는 암호화 키입니다.

OpenSSH는 Linux, UNIX 및 유사한 운영 체제에서 지원하는 SSH 프로토콜의 구현입니다. OpenSSH 클라이언트와 서버 모두에 필요한 코어 파일을 포함합니다. OpenSSH 제품군은 다음 사용자 공간 툴로 구성됩니다.

- **SSH** 는 원격 로그인 프로그램(SSH 클라이언트)입니다.
- **sshd** 는 OpenSSH SSH 데몬입니다.
- **SCP** 는 안전한 원격 파일 복사 프로그램입니다.
- **SFTP** 는 보안 파일 전송 프로그램입니다.
- **SSH-agent** 는 개인 키 캐싱을 위한 인증 에이전트입니다.
- **ssh-add** 는 개인 키 ID를 **ssh-agent** 에 추가합니다.
- **SSH-keygen은 ssh** 에 대한 인증 키를 생성, 관리 및 변환합니다.
- **SSH-copy-id** 는 원격 SSH 서버의 **authorized_keys** 파일에 로컬 공개 키를 추가하는 스크립트입니다.
- **SSH-keyscan** 은 SSH 공개 호스트 키를 수집합니다.

SSH에는 현재 버전 1과 최신 버전 2의 두 가지 버전이 있습니다. RHEL의 OpenSSH 제품군은 SSH 버전 2만 지원합니다. 버전 1에서 알려진 취약점에 취약하지 않는 향상된 key-exchange 알고리즘을 가지고 있습니다.

RHEL의 핵심 암호화 하위 시스템 중 하나인 OpenSSH는 시스템 전체 암호화 정책을 사용합니다. 이렇게 하면 기본 구성에서 약한 암호 제품군 및 암호화 알고리즘이 비활성화됩니다. 정책을 수정하려면 관리자는 **update-crypto-policies** 명령을 사용하여 설정을 조정하거나 시스템 전체 암호화 정책을 수동으로 비활성화해야 합니다.

OpenSSH 제품군은 두 가지 구성 파일 세트(즉, **ssh**,**scp**, **sftp**)를 사용하며 다른 하나는 서버(**sshd** 데몬)에 사용됩니다.

시스템 전체 SSH 구성 정보는 **/etc/ssh/** 디렉토리에 저장됩니다. 사용자별 SSH 구성 정보는 사용자 홈 디렉터리의 **~/.ssh/**에 저장됩니다. OpenSSH 구성 파일의 자세한 목록은 **sshd(8)** 도움말 페이지의 **FILES** 섹션을 참조하십시오.

추가 리소스

- **man -k ssh** 명령을 사용하여 나열되는 도움말 페이지
- 시스템 전체 암호화 정책 사용

1.2. OPENSSH 서버 구성 및 시작

사용자 환경에 필요할 수 있고 OpenSSH 서버를 시작하는 데 필요할 수 있는 기본 구성에는 다음 절차를 사용합니다. 기본 RHEL 설치 후에는 **sshd** 데몬이 이미 시작되고 서버 호스트 키가 자동으로 생성됩니다.

사전 요구 사항

- **openssh-server** 패키지가 설치되어 있어야 합니다.

절차

1. 현재 세션에서 **sshd** 데몬을 시작하고 부팅 시 자동으로 시작되도록 설정합니다.

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

2. **/etc/ssh/sshd_config** 구성 파일의 **ListenAddress** 지시문의 경우 기본값 **0.0.0.0** (IPv4) 또는 **::** (IPv6) 이외의 다른 주소를 지정하고 느린 동적 네트워크 구성을 사용하려면 **network-online.target** 대상 장치에 대한 종속성을 **sshd.service** 장치 파일에 추가합니다. 이를 위해 다음 내용으로 사용하여 **/etc/systemd/system/sshd.service.d/local.conf** 파일을 만듭니다.

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

3. **/etc/ssh/sshd_config** 구성 파일의 OpenSSH 서버 설정이 시나리오의 요구 사항을 충족하는지 검토합니다.

4. 선택적으로 **/etc/issue** 파일을 편집하여 클라이언트가 인증하기 전에 OpenSSH 서버에서 표시하는 시작 메시지를 변경합니다. 예를 들면 다음과 같습니다.

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

Banner 옵션이 **/etc/ssh/sshd_config**에서 주석 처리되지 않고 해당 값에 **/etc/issue**가 포함되어 있는지 확인하십시오.

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

로그인에 성공한 후 표시되는 메시지를 변경하려면 서버에서 **/etc/motd** 파일을 편집해야 합니다. 자세한 내용은 **pam_motd** 도움말 페이지를 참조하십시오.

5. **systemd** 구성을 다시 로드하고 **sshd** 를 다시 시작하여 변경 사항을 적용합니다.

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

검증

1. **sshd** 데몬이 실행 중인지 확인합니다.

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

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

2. SSH 클라이언트를 사용하여 SSH 서버에 연결합니다.

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

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

추가 리소스

- **sshd(8)** 및 **sshd_config(5)** 도움말 페이지

1.3. 키 기반 인증을 위한 OPENSSH 서버 설정

시스템 보안을 강화하려면 OpenSSH 서버에서 암호 인증을 비활성화하여 키 기반 인증을 시행합니다.

사전 요구 사항

- **openssh-server** 패키지가 설치되어 있어야 합니다.
- **sshd** 데몬이 서버에서 실행되고 있어야 합니다.

절차

1. 텍스트 편집기에서 **/etc/ssh/sshd_config** 구성을 엽니다. 예를 들면 다음과 같습니다.

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

2. **PasswordAuthentication** 옵션을 **no**로 변경합니다.

```
PasswordAuthentication no
```

새 기본 설치 이외의 시스템에서 **PubkeyAuthentication no**가 설정되지 않았으며 **ChallengeResponseAuthentication** 지시문이 **no**로 설정되어 있는지 확인합니다. 콘솔 또는 대역 외 액세스를 사용하지 않고 원격으로 연결하는 경우 암호 인증을 비활성화하기 전에 키 기반 로그인 프로세스를 테스트합니다.

3. NFS로 마운트된 홈 디렉토리에서 키 기반 인증을 사용하려면 **use_nfs_home_dirs** SELinux 부울을 활성화합니다.

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

4. **sshd** 데몬을 다시 로드하여 변경 사항을 적용합니다.

```
# systemctl reload sshd
```

추가 리소스

- **sshd(8)**, **sshd_config(5)** 및 **setsebool(8)** 도움말 페이지

1.4. SSH 키 쌍 생성

다음 절차에 따라 로컬 시스템에서 SSH 키 쌍을 생성하고 생성된 공개 키를 OpenSSH 서버에 복사합니다. 서버가 적절하게 구성된 경우 암호를 제공하지 않고 OpenSSH 서버에 로그인할 수 있습니다.

중요

다음 단계를 **root**로 완료하면 **root**만 키를 사용할 수 있습니다.

절차

1. SSH 프로토콜의 버전 2에 대한 ECDSA 키 쌍을 생성하려면 다음을 수행합니다.

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

```
|.oo..o=++        |
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+.  .  . .    |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

ssh-keygen -t ed25519 명령을 입력하여 **ssh-keygen** 명령 또는 Ed25519 키 쌍과 함께 **-t rsa** 옵션을 사용하여 RSA 키 쌍을 생성할 수도 있습니다.

2. 공개 키를 원격 머신에 복사하려면 다음을 수행합니다.

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

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

세션에서 **ssh-agent** 프로그램을 사용하지 않는 경우 이전 명령은 가장 최근에 수정된 **~/.ssh/id*.pub** 공개 키를 아직 설치하지 않은 경우 복사합니다. 다른 공개 키 파일을 지정하거나 **ssh-agent**로 메모리에 캐시된 키보다 파일의 키 우선 순위를 지정하려면 **ssh-copy-id** 명령을 **-i** 옵션과 함께 사용합니다.

참고

시스템을 다시 설치하고 이전에 생성된 키 쌍을 유지하려면 **~/.ssh/** 디렉터리를 백업합니다. 다시 설치한 후 홈 디렉터리로 복사합니다. **root**를 포함하여 시스템의 모든 사용자에 대해 이 작업을 수행할 수 있습니다.

검증

1. 암호를 제공하지 않고 OpenSSH 서버에 로그인합니다.

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

추가 리소스

- **ssh-keygen(1)** 및 **ssh-copy-id(1)** 도움말 페이지

1.5. 스마트 카드에 저장된 SSH 키 사용

Red Hat Enterprise Linux를 사용하면 OpenSSH 클라이언트의 스마트 카드에 저장된 RSA 및 ECDSA 키를 사용할 수 있습니다. 다음 절차에 따라 암호 대신 스마트 카드로 인증을 활성화합니다.

사전 요구 사항

- 클라이언트 측에서 **opensc** 패키지가 설치되고 **pcscd** 서비스가 실행 중입니다.

절차

1. PKCS #11 URI를 포함하여 OpenSC PKCS #11 모듈에서 제공하는 모든 키를 나열하고 출력을 *keys.pub* 파일에 저장합니다.

```
$ ssh-keygen -D pkcs11: > keys.pub
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3NzaC1yc2E...KKZMzcQZzx
pkcs11:id=%02;object=SIGN%20pubkey;token=SSH%20key;manufacturer=piv_II?module-
path=/usr/lib64/pkcs11/opensc-pkcs11.so
ecdsa-sha2-nistp256 AAA...J0hkYnnsM=
pkcs11:id=%01;object=PIV%20AUTH%20pubkey;token=SSH%20key;manufacturer=piv_II?
module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

2. 원격 서버(*example.com*)에서 스마트 카드를 사용하여 인증을 활성화하려면 공개 키를 원격 서버로 전송합니다. 이전 단계에서 만든 *keys.pub*와 함께 **ssh-copy-id** 명령을 사용하십시오.

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

3. 1단계에서 **ssh-keygen -D** 명령의 출력에서 ECDSA 키를 사용하여 *example.com* 에 연결하려면 다음과 같이 키를 고유하게 참조하는 URI의 하위 집합만 사용할 수 있습니다.

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

4. **~/.ssh/config** 파일에서 동일한 URI 문자열을 사용하여 구성을 영구적으로 만들 수 있습니다.

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

OpenSSH는 **p11-kit-proxy** 래퍼를 사용하고 OpenSC PKCS #11 모듈은 PKCS#11 Kit에 등록되므로 이전 명령을 간소화할 수 있습니다.

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

PKCS #11 URI의 **id=** 부분을 건너뛰면 OpenSSH는 proxy 모듈에서 사용할 수 있는 모든 키를 로드합니다. 이렇게 하면 필요한 입력 횟수가 줄어듭니다.

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

추가 리소스

- Fedora 28: OpenSSH에서 스마트 카드 지원 개선
- **p11-kit(8)**, **opensc.conf(5)**, **pcscd(8)**, **ssh(1)** 및 **ssh-keygen(1)** 도움말 페이지

1.6. OPENSSH의 보안 강화

다음 팁은 OpenSSH를 사용할 때 보안을 강화하는 데 도움이 됩니다. **/etc/ssh/sshd_config** OpenSSH 구성 파일의 변경 사항을 적용하려면 **sshd** 데몬을 다시 로드해야 합니다.

```
# systemctl reload sshd
```

중요

대부분의 보안 강화 구성 변경으로 최신 알고리즘 또는 암호 제품군을 지원하지 않는 클라이언트와의 호환성이 줄어듭니다.

비보안 연결 프로토콜 비활성화

- SSH를 실제로 효과적으로 수행하려면 OpenSSH 제품군으로 대체되는 안전하지 않은 연결 프로토콜을 사용하지 않도록 합니다. 그렇지 않으면 Telnet을 사용하여 로그인할 때 나중에 하나의 세션이 캡처되도록 SSH를 사용하여 사용자 암호를 보호할 수 있습니다. 이러한 이유로 telnet, rsh, rlogin 및 ftp와 같은 비보안 프로토콜을 비활성화하는 것이 좋습니다.

키 기반 인증 활성화 및 암호 기반 인증 비활성화

- 인증에 대한 암호 비활성화 및 키 쌍만 허용하면 공격 면적이 줄어들고 사용자의 시간도 절약할 수 있습니다. 클라이언트에서 **ssh-keygen** 툴을 사용하여 키 쌍을 생성하고 **ssh-copy-id** 유틸리티를 사용하여 OpenSSH 서버의 클라이언트에서 공개 키를 복사합니다. OpenSSH 서버에서 암호 기반 인증을 비활성화하려면 **/etc/ssh/sshd_config**를 편집하고 **PasswordAuthentication** 옵션을 **no**로 변경합니다.

  ```
  PasswordAuthentication no
  ```

키 유형

- **ssh-keygen** 명령은 기본적으로 RSA 키 쌍을 생성하지만 **-t** 옵션을 사용하여 ECDSA 또는 Ed25519 키를 생성하도록 지시할 수 있습니다. ECDSA(Elliptic Curve Digital Signature Algorithm)는 동등한 대칭 키 강점에서 RSA보다 더 나은 성능을 제공합니다. 짧은 키도 생성합니다. Ed25519 공개 키 알고리즘은 RSA, DSA 및 ECDSA보다 더 빠르고 안전하며 더 빠릅니다. OpenSSH는 RSA, ECDSA 및 Ed25519 서버 호스트 키가 누락된 경우 자동으로 생성합니다. RHEL에서 호스트 키 생성을 구성하려면 **sshd-keygen@.service** 인스턴스화 서비스를 사용합니다. 예를 들어, RSA 키 유형의 자동 생성을 비활성화하려면 다음을 실행합니다.

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

- SSH 연결에 대한 특정 키 유형을 제외하려면 **/etc/ssh/sshd_config**에서 관련 행을 주석 처리하고 **sshd** 서비스를 다시 로드합니다. 예를 들어 Ed25519 호스트 키만 허용하려면 다음을 수행합니다.

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

기본값 이외의 포트

- 기본적으로 **sshd** 데몬은 TCP 포트 22에서 수신 대기합니다. 포트를 변경하면 자동화된 네트워크 스캔을 기반으로 하는 공격에 대한 시스템 노출이 줄어들고 따라서 모호성을 통해 보안이 강화됩니다. **/etc/ssh/sshd_config** 구성 파일에서 **Port** 지시문을 사용하여 포트를 지정할 수 있습니다. 또한 기본이 아닌 포트를 사용할 수 있도록 기본 SELinux 정책을 업데이트해야 합니다. 이렇게 하려면 **policycoreutils-python-utils** 패키지에서 **semanage** 툴을 사용합니다.

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

  또한 **firewalld** 구성을 업데이트합니다.

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

  이전 명령에서 *port_number*를 **Port** 지시문을 사용하여 지정된 새 포트 번호로 바꿉니다.

Root 로그인 없음

- 특정 사용 사례에서 root 사용자로 로그인할 가능성이 필요하지 않은 경우 **PermitRootLogin** 설정 지시문을 **/etc/ssh/sshd_config** 파일에서 **no**로 설정하는 것이 좋습니다. root 사용자로 로그인할 가능성을 비활성화하여 관리자는 일반 사용자로 로그인한 후 권한이 있는 명령을 실행하는 사용자를 감사한 다음 root 권한을 얻을 수 있습니다.
  또는 **prohibit-password**로 **PermitRootLogin**을 설정합니다.

```
PermitRootLogin prohibit-password
```

  이렇게 하면 root로 로그인하는 데 암호를 사용하는 대신 키 기반 인증을 사용하고 무차별 강제 공격을 방지하여 위험을 줄입니다.

X 보안 확장 사용

- Red Hat Enterprise Linux 클라이언트의 X 서버는 X 보안 확장을 제공하지 않습니다. 따라서 클라이언트는 X11 전달을 사용하여 신뢰할 수 없는 SSH 서버에 연결할 때 다른 보안 계층을 요청할 수 없습니다. 대부분의 애플리케이션은 이 확장 기능을 사용하여 실행할 수 없습니다.
  기본적으로 **/etc/ssh/ssh_config.d/05-redhat.conf** 파일의 **ForwardX11Trusted** 옵션은 **yes**로 설정되어 있으며 **ssh -X remote_machine** (신뢰할 수 없는 호스트)과 **ssh -Y remote_machine** (신뢰할 수 있는 호스트) 명령 사이에 차이가 없습니다.

  시나리오에 X11 전달 기능이 전혀 필요하지 않은 경우 **/etc/ssh/sshd_config** 구성 파일의 **X11Forwarding** 지시문을 **no**로 설정합니다.

특정 사용자, 그룹 또는 도메인에 대한 액세스 제한

- **/etc/ssh/sshd_config** 구성 파일 서버의 **AllowUsers** 및 **AllowGroups** 지시문을 사용하면 특정 사용자, 도메인 또는 그룹만 OpenSSH 서버에 연결할 수 있습니다. **AllowUsers** 및 **AllowGroups**를 결합하여 보다 정확하게 액세스를 제한할 수 있습니다. 예를 들면 다음과 같습니다.

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

이전 구성 행은 192.168.1.2 주소가 있는 시스템을 제외하고 192.168.1.* 및 10.0.0.* 서브넷의 모든 사용자로부터의 연결을 허용합니다. 모든 사용자는 **example-group** 그룹에 있어야 합니다. OpenSSH 서버는 다른 모든 연결을 거부합니다.

허용 목록(허용으로 시작하는 디렉터리)을 사용하는 것은 차단 목록(거부로 시작하는 옵션)을 사용하는 것보다 더 안전합니다. allowlists는 새로운 권한 없는 사용자 또는 그룹도 차단하기 때문입니다.

시스템 전체 암호화 정책 변경

- OpenSSH는 RHEL 시스템 전체 암호화 정책을 사용하며, 기본 시스템 전체 암호화 정책 수준은 현재 위협 모델에 대한 보안 설정을 제공합니다. 암호화 설정을 보다 엄격하게 수행하려면 현재 정책 수준을 변경합니다.

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

- OpenSSH 서버의 시스템 전체 암호화 정책을 옵트아웃하려면 **/etc/sysconfig/sshd** 파일에서 **CRYPTO_POLICY=** 변수로 줄의 주석 처리를 해제합니다. 이 변경 후 **/etc/ssh/sshd_config** 파일의 **Ciphers**, **MACs**, **KexAlgoritms**, **GSSAPIKexAlgorithms** 섹션에 지정하는 값은 재정의되지 않습니다. 이 작업에는 암호화 옵션 구성에 대한 전문 지식이 필요합니다.

- 자세한 내용은 보안 강화에서 시스템 전체 암호화 정책 사용 을 참조하십시오.

추가 리소스

- **sshd_config(5)**, **ssh-keygen(1)**, **crypto-policies(7)** 및 **update-crypto-policies(8)** 도움말 페이지

1.7. SSH 건너뛰기 호스트를 사용하여 원격 서버에 연결

건너뛰기 호스트라고도 하는 중간 서버를 통해 로컬 시스템을 원격 서버에 연결하려면 다음 절차를 사용하십시오.

사전 요구 사항

- 건너뛰기 호스트는 로컬 시스템의 SSH 연결을 허용합니다.

- 원격 서버는 건너뛰기 호스트에서만 SSH 연결을 허용합니다.

절차

1. 로컬 시스템에서 **~/.ssh/config** 파일을 편집하여 건너뛰기를 정의합니다. 예를 들면 다음과 같습니다.

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

   - **Host** 매개 변수는 **ssh** 명령에서 사용할 수 있는 호스트의 이름 또는 별칭을 정의합니다. 이 값은 실제 호스트 이름과 일치할 수 있지만 임의의 문자열일 수도 있습니다.

   - **HostName** 매개 변수는 건너뛰기 호스트의 실제 호스트 이름 또는 IP 주소를 설정합니다.

2. **ProxyJump** 지시문을 사용하여 원격 서버 건너뛰기를 로컬 시스템의 **~/.ssh/config** 파일에 추가합니다. 예를 들면 다음과 같습니다.

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

3. 로컬 시스템을 사용하여 이동 서버를 통해 원격 서버에 연결합니다.

```
$ ssh remote-server
```

이전 명령은 구성 단계 1과 2를 생략하면 **ssh -J jump-server1 remote-server** 명령과 동일합니다.

참고

더 많은 건너뛰기 서버를 지정할 수 있으며 전체 호스트 이름을 제공할 때 구성 파일에 호스트 정의 추가를 건너뛸 수도 있습니다. 예를 들면 다음과 같습니다.

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

건너뛰기 서버의 사용자 이름 또는 SSH 포트가 원격 서버의 이름과 포트와 다른 경우 이전 명령에서 호스트 이름 전용 표기법을 변경합니다. 예를 들면 다음과 같습니다.

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

추가 리소스

- **ssh_config(5)** 및 **ssh(1)** 도움말 페이지

1.8. SSH-AGENT를 사용하여 SSH 키가 있는 원격 시스템에 연결

SSH 연결을 시작할 때마다 암호를 입력하지 않으려면 **ssh-agent** 유틸리티를 사용하여 개인 SSH 키를 캐시할 수 있습니다. 개인 키와 암호는 안전하게 유지됩니다.

사전 요구 사항

- SSH 데몬이 실행되고 네트워크를 통해 연결할 수 있는 원격 호스트가 있습니다.
- IP 주소 또는 호스트 이름 및 인증 정보를 통해 원격 호스트에 로그인합니다.
- 암호를 사용하여 SSH 키 쌍을 생성하고 공개 키를 원격 시스템으로 전송했습니다.

절차

1. 선택 사항: 키를 사용하여 원격 호스트에 인증할 수 있는지 확인합니다.

   a. SSH를 사용하여 원격 호스트에 연결합니다.

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

   b. 개인 키에 대한 액세스 권한을 부여할 키를 만드는 동안 설정한 암호를 입력합니다.

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

2. **ssh-agent**를 시작합니다.

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

3. **ssh-agent**에 키를 추가합니다.

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

검증

- 선택 사항: SSH를 사용하여 호스트 시스템에 로그인합니다.

```
$ ssh example.user1@198.51.100.1

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

암호를 입력할 필요가 없습니다.

1.9. 추가 리소스

- **sshd(8)**, **ssh(1)**, **scp(1)**, **sftp(1)**, **ssh-keygen(1)**, **ssh-copy-id(1)**, **ssh_config(5)**, **sshd_config(5)**, **update-crypto-policies(8)** 및 **crypto-policies(7)** 도움말 페이지
- OpenSSH 홈 페이지
- 비표준 구성을 사용하여 애플리케이션 및 서비스에 대한 SELinux 구성
- firewalld를 사용하여 네트워크 트래픽 제어

2장. SSH 시스템 역할을 사용하여 보안 통신 구성

관리자는 SSHD 시스템 역할을 사용하여 SSH 서버를 구성하고 SSH 시스템 역할을 사용하여 Red Hat Ansible Automation Platform을 사용하여 동시에 여러 RHEL 시스템에서 일관되게 SSH 클라이언트를 구성할 수 있습니다.

2.1. SSH SERVER 시스템 역할 변수

SSH 서버 시스템 역할 플레이북에서는 기본 설정 및 제한 사항에 따라 SSH 구성 파일의 매개변수를 정의할 수 있습니다.

이러한 변수를 구성하지 않으면 시스템 역할은 RHEL 기본값과 일치하는 **sshd_config** 파일을 생성합니다.

모든 경우에 부울이 **sshd** 구성에서 **yes** 및 **no**로 올바르게 렌더링됩니다. 목록을 사용하여 여러 줄 구성 항목을 정의할 수 있습니다. 예를 들면 다음과 같습니다.

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

다음과 같이 렌더링됩니다.

```
ListenAddress 0.0.0.0
ListenAddress ::
```

SSH 서버 시스템 역할의 변수

sshd_enable

False로 설정하면 역할이 완전히 비활성화됩니다. 기본값은 **True**입니다.

sshd_skip_defaults

True 로 설정하면 시스템 역할이 기본값이 적용되지 않습니다. 대신 **sshd** dict 또는 **sshd_Key** 변수를 사용하여 전체 구성 기본값 세트를 지정합니다. 기본값은 **False**입니다.

sshd_manage_service

False로 설정하면 서비스가 관리되지 않으므로 부팅 시 활성화되지 않으며 시작 또는 다시 로드되지 않습니다. Ansible service 모듈이 현재 AIX에 대해 **enabled**되지 않으므로 컨테이너 또는 AIX 내부에서 실행되는 경우를 제외하고 기본값은 **True**입니다.

sshd_allow_reload

False로 설정하면 구성이 변경된 후 **sshd**가 다시 로드되지 않습니다. 이는 문제 해결에 도움이 될 수 있습니다. 변경된 구성을 적용하려면 **sshd**를 수동으로 다시 로드합니다. 기본값은 AIX를 제외하고 **sshd_manage_service** 와 동일한 값입니다. 여기서 **sshd_manage_service** 기본값은 **False** 이지만 **sshd_allow_reload** 의 기본값은 **True** 입니다.

sshd_install_service

True로 설정하면 역할은 **sshd** 서비스에 대한 서비스 파일을 설치합니다. 이렇게 하면 운영 체제에 제공된 파일이 재정의됩니다. 두 번째 인스턴스를 구성하고 **sshd_service** 변수도 변경하지 않는 한 **True** 로 설정하지 마십시오. 기본값은 **False**입니다.
역할은 다음 변수에서 가리키는 파일을 템플릿으로 사용합니다.

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

sshd_service

이 변수는 두 번째 **sshd** 서비스 인스턴스를 구성하는 데 유용한 **sshd** 서비스 이름을 변경합니다.

sshd

구성이 포함된 dict입니다. 예를 들면 다음과 같습니다.

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

sshd_*OptionName*

dict 대신 **sshd_** 접두사와 옵션 이름으로 구성된 간단한 변수를 사용하여 옵션을 정의할 수 있습니다. simple 변수는 **sshd** dict의 값을 재정의합니다. 예를 들면 다음과 같습니다.

```
sshd_Compression: no
```

sshd_match 및 sshd_match_1을 sshd_match_9로 재정의

dicts 목록 또는 일치 섹션의 경우 dict에 불과합니다. 이러한 변수는 **sshd** dict에 정의된 대로 일치하는 블록을 재정의하지 않습니다. 결과 구성 파일에 모든 소스가 반영됩니다.

SSH Server 시스템 역할의 보조 변수

이러한 변수를 사용하여 지원되는 각 플랫폼에 해당하는 기본값을 재정의할 수 있습니다.

sshd_packages

이 변수를 사용하여 설치된 패키지의 기본 목록을 재정의할 수 있습니다.

sshd_config_owner, sshd_config_group, and **sshd_config_mode**

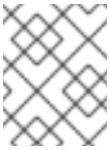
이 역할에서 이러한 변수를 사용하여 생성하는 **openssh** 구성 파일의 소유권 및 권한을 설정할 수 있습니다.

sshd_config_file

이 역할이 **openssh** 서버 구성이 생성된 경로입니다.

sshd_config_namespace

이 변수의 기본값은 null입니다. 즉, 역할이 시스템 기본값을 포함하여 구성 파일의 전체 콘텐츠를 정의함을 의미합니다. 또는 이 변수를 사용하여 드롭인 디렉터리를 지원하지 않는 시스템의 단일 플레이북에 있는 다른 역할 또는 여러 위치에서 이 역할을 호출할 수 있습니다. **sshd_skip_defaults** 변수는 무시되며 이 경우 시스템 기본값이 사용되지 않습니다.
이 변수를 설정하면 역할이 지정된 구성을 지정된 네임스페이스 아래에 기존 구성 파일의 구성 스니펫에 배치합니다. 시나리오에 역할을 여러 번 적용해야 하는 경우 각 애플리케이션에 대해 다른 네임스페이스를 선택해야 합니다.



참고

openssh 구성 파일의 제한 사항은 계속 적용됩니다. 예를 들어 구성 파일에 지정된 첫 번째 옵션만 대부분의 구성 옵션에 적용됩니다.

기술적으로, 역할은 기존 구성 파일의 이전 일치 블록과 관계없이 적용되도록 다른 일치 블록을 포함하지 않는 한 "Match all" 블록에 코드 조각을 배치합니다. 이를 통해 다양한 역할 호출에서 충돌하지 않는 옵션을 구성할 수 있습니다.

sshd_binary

openssh 의 **sshd** 실행 파일의 경로입니다.

sshd_service

sshd 서비스의 이름입니다. 기본적으로 이 변수에는 대상 플랫폼에서 사용하는 **sshd** 서비스의 이름이 포함됩니다. 또한 역할에서 **sshd_install_service** 변수를 사용하는 경우 사용자 지정 **sshd** 서비스의 이름을 설정할 수도 있습니다.

sshd_verify_hostkeys

기본값은 **auto** 입니다. **auto**로 설정하면 생성된 구성 파일에 있는 모든 호스트 키가 나열되고 존재하지 않는 경로가 생성됩니다. 또한 권한 및 파일 소유자는 기본값으로 설정됩니다. 이 기능은 배포 단계에서 역할을 사용하여 첫 번째 시도에서 서비스를 시작할 수 있는지 확인하는 데 유용합니다. 이 확인을 비활성화하려면 이 변수를 빈 목록 **[]**으로 설정합니다.

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

이러한 변수를 사용하여 **sshd_verify_hostkeys**에서 호스트 키에 대한 소유권 및 권한을 설정합니다.

sshd_sysconfig

RHEL 기반 시스템에서 이 변수는 **sshd** 서비스에 대한 추가 세부 정보를 구성합니다. **true** 로 설정하면 이 역할은 다음 구성에 따라 **/etc/sysconfig/sshd** 구성 파일도 관리합니다. 기본값은 **false**입니다.

sshd_sysconfig_override_crypto_policy

RHEL에서 **true** 로 설정하면 이 변수가 시스템 전체 암호화 정책을 재정의합니다. 기본값은 **false**입니다.

sshd_sysconfig_use_strong_rng

RHEL 기반 시스템에서 이 변수는 **sshd** 에서 인수로 지정된 바이트 수를 사용하여 **openssl** 임의 번호 생성기를 다시 작성할 수 있습니다. 기본값은 **0**으로 이 기능을 비활성화합니다. 시스템에 하드웨어 임의 번호 생성기가 없는 경우 이 값을 설정하지 마십시오.

2.2. SSH 서버 시스템 역할을 사용하여 OPENSSH 서버 구성

SSH Server 시스템 역할을 사용하여 Ansible 플레이북을 실행하여 여러 SSH 서버를 구성할 수 있습니다.

참고

SSH 및 SSHD 구성을 변경하는 다른 시스템 역할(예: ID 관리 RHEL 시스템 역할)과 함께 SSH 서버 시스템 역할을 사용할 수 있습니다. 구성을 덮어쓰지 않도록 하려면 SSH Server 역할에서 네임스페이스(RHEL 8 및 이전 버전) 또는 드롭인 디렉터리(RHEL 9)를 사용하는지 확인합니다.

사전 요구 사항

- SSHD 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.
  제어 노드에서 다음이 있어야 합니다.
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible을 기반으로 하는 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 ansible , **ansible -playbook**, **docker** 및 **podman** 와 같은 커넥터, 많은 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 가져오고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법을 참조하십시오.

RHEL 8.6 및 9.0에는 Ansible 명령줄 유틸리티, 명령 및 일부 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)가 도입되었습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며, 제한된 지원 범위가 있습니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식베이스 문서에 포함된 Ansible Core 패키지의 지원 범위를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일이 있어야 합니다.

절차

1. SSH 서버 시스템 역할에 대한 예제 플레이북을 복사합니다.

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

2. 텍스트 편집기를 사용하여 복사된 플레이북을 엽니다. 예를 들면 다음과 같습니다.

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

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

플레이북은 다음을 수행하도록 구성된 SSH 서버로 관리 노드를 구성합니다.

- 암호 및 **root** 사용자 로그인이 비활성화되어 있습니다
- 암호 및 **root** 사용자 로그인은 서브넷 **192.0.2.0/24**에서만 활성화됩니다.

환경 설정에 따라 변수를 수정할 수 있습니다. 자세한 내용은 SSH 서버 시스템 역할 변수를 참조하십시오.

3. 선택 사항: 플레이북 구문을 확인합니다.

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

4. 인벤토리 파일에서 플레이북을 실행합니다.

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

...

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

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

검증

1. SSH 서버에 로그인합니다.

```
$ ssh user1@10.1.1.1
```

다음과 같습니다.

- ***user1*** 은 SSH 서버의 사용자입니다.
- ***10.1.1.1***은 SSH 서버의 IP 주소입니다.

2. SSH 서버에서 **sshd_config** 파일의 내용을 확인합니다.

```
$ vim /etc/ssh/sshd_config

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

3. **192.0.2.0/24** 서브넷에서 root로 서버에 연결할 수 있는지 확인합니다.

   a. IP 주소를 확인합니다.

   ```
   $ hostname -I
   192.0.2.1
   ```

   IP 주소가 **192.0.2.1** - **192.0.2.254** 범위 내에 있는 경우 서버에 연결할 수 있습니다.

   b. **root**로 서버에 연결합니다 :

   ```
   $ ssh root@10.1.1.1
   ```

추가 리소스

- **/usr/share/doc/rhel-system-roles/sshd/README.md** 파일.
- **ansible-playbook(1)** 도움말 페이지.

2.3. SSH 클라이언트 시스템 역할 변수

SSH 클라이언트 시스템 역할 플레이북에서는 기본 설정 및 제한 사항에 따라 클라이언트 SSH 구성 파일의 매개변수를 정의할 수 있습니다.

이러한 변수를 구성하지 않으면 시스템 역할은 RHEL 기본값과 일치하는 글로벌 **ssh_config** 파일을 생성합니다.

모든 경우에 부울이 **ssh** 구성에서 **yes** 또는 **no** 로 올바르게 렌더링됩니다. 목록을 사용하여 여러 줄 구성 항목을 정의할 수 있습니다. 예를 들면 다음과 같습니다.

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

다음과 같이 렌더링됩니다.

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

참고

구성 옵션은 대소문자를 구분합니다.

SSH 클라이언트 시스템 역할의 변수

ssh_user

시스템 역할이 사용자별 구성을 수정하는 기존 사용자 이름을 정의할 수 있습니다. 사용자별 구성은 지정된 사용자의 **~/.ssh/config** 에 저장됩니다. 기본값은 모든 사용자에 대한 글로벌 구성을 수정하는 null입니다.

ssh_skip_defaults

기본값은 **auto** 입니다. **자동으로 설정하면 시스템** 역할은 시스템 전체 구성 파일 **/etc/ssh/ssh_config** 를 작성하고 여기에 RHEL 기본값을 유지합니다. **ssh_drop_in_name** 변수를 정의하여 드롭인 구성 파일을 생성하면 **ssh_skip_defaults** 변수가 자동으로 비활성화됩니다.

ssh_drop_in_name

시스템 전체 드롭인 디렉터리에 배치되는 드롭인 구성 파일의 이름을 정의합니다. 이름은 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 템플릿에서 수정할 구성 파일을 참조하는 데 사용됩니다. 시스템이 드롭인 디렉터리를 지원하지 않는 경우 기본값은 null입니다. 시스템이 드롭인 디렉터리를 지원하는 경우 기본값은 **00-ansible** 입니다.

주의

시스템이 드롭인 디렉터리를 지원하지 않는 경우 이 옵션을 설정하면 플레이가 실패합니다.

제안된 형식은 **NN-name** 입니다. 여기서 **NN** 은 구성 파일의 순서를 지정하는 데 사용되는 두 자리 숫자이고, **name**은 파일의 콘텐츠 또는 소유자에 대한 설명이 포함된 이름입니다.

ssh

구성 옵션과 해당 값이 포함된 dict입니다.

ssh_*OptionName*

dict 대신 **ssh_** 접두사 및 옵션 이름으로 구성된 간단한 변수를 사용하여 옵션을 정의할 수 있습니다. 단순 변수는 **ssh** dict의 값을 재정의합니다.

ssh_additional_packages

이 역할은 가장 일반적인 사용 사례에 필요한 **openssh** 및 **openssh-clients** 패키지를 자동으로 설치합니다. 추가 패키지를 설치해야 하는 경우(예: 호스트 기반 인증에 **openssh-keysign**) 이 변수에 지정할 수 있습니다.

ssh_config_file

역할이 생성된 구성 파일을 저장하는 경로입니다. 기본값:

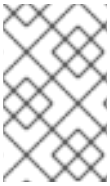
- 시스템에 드롭인 디렉터리가 있는 경우 기본값은 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 템플릿으로 정의됩니다.
- 시스템에 드롭인 디렉터리가 없으면 기본값은 **/etc/ssh/ssh_config** 입니다.
- **ssh_user** 변수가 정의된 경우 기본값은 **~/.ssh/config** 입니다.

ssh_config_owner, ssh_config_group, ssh_config_mode

생성된 구성 파일의 소유자, 그룹 및 모드입니다. 기본적으로 파일 소유자는 **root:root**이며 모드는 **0644** 입니다. **ssh_user** 가 정의되면 모드는 **0600** 이고 소유자와 그룹은 **ssh_user** 변수에 지정된 사용자 이름에서 파생됩니다.

2.4. SSH 클라이언트 시스템 역할을 사용하여 OPENSSH 클라이언트 구성

SSH 클라이언트 시스템 역할을 사용하여 Ansible 플레이북을 실행하여 여러 SSH 클라이언트를 구성할 수 있습니다.



참고

SSH 및 SSHD 구성을 변경하는 다른 시스템 역할(예: ID 관리 RHEL 시스템 역할)과 함께 SSH 클라이언트 시스템 역할을 사용할 수 있습니다. 구성을 덮어쓰지 않도록 하려면 SSH Client 역할이 드롭인 디렉토리(RHEL 8의 기본)를 사용하는지 확인합니다.

사전 요구 사항

- SSH 클라이언트 시스템 역할을 사용하여 구성하려는 하나 이상의 *관리형 노드에* 대한 액세스 및 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.
  제어 노드에서 다음이 있어야 합니다.
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible을 기반으로 하는 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 ansible , **ansible -playbook**, **docker** 및 **podman** 와 같은 커넥터, 많은 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 가져오고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법을 참조하십시오.

RHEL 8.6 및 9.0에는 Ansible 명령줄 유틸리티, 명령 및 일부 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)가 도입되었습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며, 제한된 지원 범위가 있습니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식베이스 문서에 포함된 Ansible Core 패키지의 지원 범위를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일이 있어야 합니다.

절차

1. 다음 내용으로 새 ***playbook.yml*** 파일을 생성합니다.

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

이 플레이북은 다음 구성을 사용하여 관리 노드에서 **root** 사용자의 SSH 클라이언트 기본 설정을 구성합니다.

- 압축이 활성화됩니다.
- ControlMaster 멀티플렉싱이 **auto** 로 설정되어 있습니다.
- ***example .com 호스트에 연결하는 예제*** 별칭은 ***user1*** 입니다.
- ***예제 호스트 별칭이 생성되어 example .com*** 호스트와 ***user1 사용자*** 이름이 인 에 대한 연결을 나타냅니다.
- X11 전달이 비활성화되어 있습니다.

선택적으로 환경 설정에 따라 이러한 변수를 수정할 수 있습니다. 자세한 내용은 SSH 시스템 역할 변수를 참조하십시오.

2. 선택 사항: 플레이북 구문을 확인합니다.

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

검증

- 텍스트 편집기에서 SSH 구성 파일을 열어 관리 노드에 올바른 구성이 있는지 확인합니다. 예를 들면 다음과 같습니다.

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

위에 표시된 예제 플레이북의 애플리케이션을 적용한 후에는 구성 파일에 다음 내용이 포함되어야 합니다.

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

2.5. 비독점 구성에 대해 SSH 서버 시스템 역할 사용

일반적으로 SSH Server 시스템 역할을 적용하면 전체 구성을 덮어씁니다. 이 문제는 다른 시스템 역할 또는 플레이북과 같이 설정을 이전에 조정한 경우 문제가 될 수 있습니다. 다른 옵션을 유지하면서 선택한 구성 옵션에 대해서만 SSH 서버 시스템 역할을 적용하려면 제외 구성을 사용할 수 있습니다.

RHEL 8 이전 버전에서는 구성 스니펫을 사용하여 비독점 구성을 적용할 수 있습니다.

사전 요구 사항

- SSH 서버 시스템 역할을 사용하여 구성하려는 하나 이상의 *관리형 노드에* 대한 액세스 및 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.
  제어 노드에서 다음이 있어야 합니다.
  - **ansible-core** 패키지가 설치되어 있습니다.
  - 관리 노드를 나열하는 인벤토리 파일이 있어야 합니다.
  - 다른 RHEL 시스템 역할에 대한 플레이북입니다. 자세한 내용은 역할 적용을 참조하십시오.

절차

1. **sshd_config_namespace** 변수가 있는 구성 스니펫을 플레이북에 추가합니다.

```
---
- hosts: all
  tasks:
  - name: <Configure SSHD to accept some useful environment variables>
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd_config_namespace: <my-application>
      sshd:
        # Environment variables to accept
        AcceptEnv:
          LANG
          LS_COLORS
          EDITOR
```

인벤토리에 플레이북을 적용하면 역할이 없는 경우 **/etc/ssh/sshd_config** 파일에 다음 스니펫을 추가합니다.

```
# BEGIN sshd system role managed block: namespace <my-application>
Match all
  AcceptEnv LANG LS_COLORS EDITOR
# END sshd system role managed block: namespace <my-application>
```

검증

- 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

추가 리소스

- **/usr/share/doc/rhel-system-roles/sshd/README.md** 파일.
- **ansible-playbook(1)** 도움말 페이지.

3장. TLS 계획 및 구현

TLS(Transport Layer Security)는 네트워크 통신을 보호하는 데 사용되는 암호화 프로토콜입니다. 기본 키 교환 프로토콜, 인증 방법 및 암호화 알고리즘을 구성하여 시스템 보안 설정을 강화할 때 지원되는 클라이언트 범위를 넓혀 보안을 강화해야 합니다. 반대로 엄격한 보안 설정은 클라이언트와의 호환성이 제한되어 일부 사용자가 시스템에서 잠길 수 있습니다. 가장 엄격한 사용 가능한 구성을 대상으로 하고 호환성을 위해 필요한 경우에만 완화해야 합니다.

3.1. SSL 및 TLS 프로토콜

SSL(Secure Sockets Layer) 프로토콜은 원래 Netscape Corporation에서 인터넷을 통한 보안 통신을 위한 메커니즘을 제공하기 위해 개발되었습니다. 그 후 이 프로토콜은 IETF(Internet Engineering Task Force)에서 채택했으며 TLS(Transport Layer Security)로 이름이 변경되었습니다.

TLS 프로토콜은 애플리케이션 프로토콜 계층과 TCP/IP와 같은 신뢰할 수 있는 전송 계층 사이에 있습니다. 애플리케이션 프로토콜과 독립적이므로 다음과 같이 다양한 프로토콜 아래에 계층화할 수 있습니다. HTTP, FTP, SMTP 등.

프로토콜 버전	사용 권장 사항
SSL v2	사용하지 마십시오. 심각한 보안 취약점이 있습니다. RHEL 7 이후 코어 암호화 라이브러리에서 제거되었습니다.
SSL v3	사용하지 마십시오. 심각한 보안 취약점이 있습니다. RHEL 8 이후 코어 암호화 라이브러리에서 제거되었습니다.
TLS 1.0	사용하지 않는 것이 좋습니다. 상호 운용성을 보장하고 최신 암호화 제품군을 지원하지 않는 방식으로 완화할 수 없는 문제가 있습니다. RHEL 8에서는 **LEGACY** 시스템 전체 암호화 정책 프로필에서만 사용할 수 있습니다.
TLS 1.1	필요한 경우 상호 운용성을 목적으로 사용합니다. 최신 암호화 제품군을 지원하지 않습니다. RHEL 8에서는 **LEGACY** 정책에서만 사용할 수 있습니다.
TLS 1.2	최신 AEAD 암호화 제품군을 지원합니다. 이 버전은 모든 시스템 전체 암호화 정책에서 활성화되지만, 이 프로토콜의 선택적 부분에는 취약점이 포함되어 있으며 TLS 1.2에서는 오래된 알고리즘도 허용합니다.
TLS 1.3	권장 버전입니다. TLS 1.3은 문제가 있는 알려진 옵션을 제거하고, 협상 핸드셰이크를 더 많이 암호화하여 추가적인 프라이버시를 제공하며 보다 효율적인 최신 암호화 알고리즘을 사용하여 더 빠르게 사용될 수 있습니다. TLS 1.3은 모든 시스템 차원의 암호화 정책에서도 활성화됩니다.

추가 리소스

- IETF: TLS(Transport Layer Security) 프로토콜 버전 1.3 .

3.2. RHEL 8에서 TLS의 보안 고려 사항

RHEL 8에서는 시스템 전체 암호화 정책 때문에 암호화 관련 고려 사항이 크게 단순화됩니다. **DEFAULT** 암호화 정책은 TLS 1.2 및 1.3만 허용합니다. 시스템이 이전 버전의 TLS를 사용하여 연결을 협상할 수 있도록 하려면 애플리케이션에서 다음과 같은 암호화 정책을 선택하거나 **update-crypto-policies** 명령을 사

용하여 **LEGACY** 정책으로 전환해야 합니다. 자세한 내용은 시스템 전체 암호화 정책 사용 을 참조하십시오.

RHEL 8에 포함된 라이브러리에서 제공하는 기본 설정은 대부분의 배포에 충분히 안전합니다. TLS 구현에서는 가능한 경우 또는 기존 클라이언트 또는 서버의 연결을 방지할 수 없는 보안 알고리즘을 사용합니다. 보안 알고리즘 또는 프로토콜을 지원하지 않는 레거시 클라이언트나 서버가 연결되지 않거나 연결할 수 없는 엄격한 보안 요구 사항을 충족하는 환경에서 강화된 설정을 적용합니다.

TLS 구성을 강화하는 가장 간단한 방법은 **update-crypto-policies --set FUTURE** 명령을 사용하여 시스템 전체 암호화 정책 수준을 **FUTURE**로 전환하는 것입니다.

주의

LEGACY 암호화 정책에 대해 비활성화된 알고리즘은 RHEL 8 보안에 대한 Red Hat의 비전을 따르지 않으며 보안 속성은 신뢰할 수 없습니다. 다시 활성화하는 대신 이러한 알고리즘 사용에서 벗어나는 것이 좋습니다. 예를 들어 이전 하드웨어와의 상호 운용성을 위해 다시 활성화하기로 결정한 경우, 이를 안전하지 않은 것으로 처리하고 네트워크 상호 작용을 별도의 네트워크 세그먼트에 격리하는 등의 추가 보호 조치를 적용합니다. 공용 네트워크에서 사용하지 마십시오.

RHEL 시스템 전체 암호화 정책을 따르거나 설정에 맞는 사용자 지정 암호화 정책을 생성하기로 결정한 경우 사용자 정의 구성에서 선호하는 프로토콜, 암호화 제품군 및 키 길이에 다음 권장 사항을 사용하십시오.

3.2.1. 프로토콜

최신 버전의 TLS는 최상의 보안 메커니즘을 제공합니다. 이전 버전의 TLS에 대한 지원을 포함하는 강력한 이유가 없으면 시스템에서 최소 TLS 버전 1.2를 사용하여 연결을 협상할 수 있습니다.

RHEL 8은 TLS 버전 1.3을 지원하더라도 이 프로토콜의 일부 기능은 RHEL 8 구성 요소에서 완전히 지원되지는 않습니다. 예를 들어 연결 대기 시간을 줄이는 0-RTT(round Trip Time) 기능은 Apache 웹 서버에서 아직 완전히 지원하지 않습니다.

3.2.2. 암호화 제품군

최신의 더욱 안전한 암호화 제품군은 보안이 떨어지는 오래된 제품군보다 선호되어야 합니다. 항상 암호화 또는 인증을 전혀 제공하지 않는 eNULL 및 aNULL 암호화 제품군 사용을 비활성화합니다. 가능한 경우 심각한 단점이 있는 RC4 또는 HMAC-MD5를 기반으로 하는 암호 제품군도 비활성화해야 합니다. 즉, 의도적으로 약해졌던 수출 암호 모음에도 동일하게 적용됩니다. 따라서 쉽게 중단할 수 있습니다.

즉시 안전하지 않지만 128비트 미만의 보안을 제공하는 암호화 제품군은 짧은 유효 기간 동안 고려해서는 안 됩니다. 128비트의 보안을 사용하는 알고리즘은 적어도 몇 년 동안 중단되지 않을 수 있으므로 강력하게 권장합니다. 3DES 암호화는 168비트 사용을 알리지만 실제로 112비트의 보안을 제공합니다.

항상 서버 키가 손상된 경우에도 암호화된 데이터의 기밀성을 보장하는 PFS(forward secrecy)를 지원하는 암호화 제품군을 선호합니다. 이 규칙은 빠른 RSA 키 교환을 제한하지만 ECDHE 및 DHE를 사용할 수 있습니다. 이 둘 중 ECDHE는 더 빠르고 선호되는 선택입니다.

또한 Oracle 공격에 취약하지 않으므로 CBC-GCM 이상의 AEAD 암호를 선호해야 합니다. 또한 대부분의 경우 AES-GCM은 특히 하드웨어에 AES용 암호화 가속기가 있는 경우 CBC 모드에서 AES보다 빠릅니다.

또한 ECDSA 인증서와 함께 ECDSA 키 교환을 사용할 때는 트랜잭션이 순수 RSA 키 교환보다 훨씬 빠릅니다. 레거시 클라이언트를 지원하기 위해 서버에 두 쌍의 인증서와 키(새 클라이언트용)와 RSA 키가 있는 키(기존 클라이언트의 경우)를 설치할 수 있습니다.

3.2.3. 공개 키 길이

RSA 키를 사용하는 경우 항상 SHA-256에서 서명한 3072비트 이상의 키 길이를 선호하므로 true 128비트의 보안에 충분합니다.

주의

시스템의 보안은 체인에서 가장 약한 링크만큼 강력합니다. 예를 들어 강력한 암호만으로는 좋은 보안이 보장되지 않습니다. 키와 인증서는 키에 서명하는 데 CA(인증 기관)에서 사용하는 해시 기능 및 키뿐만 아니라 중요합니다.

추가 리소스

- RHEL 8의 시스템 전체 암호화 정책 .
- **update-crypto-policies(8)** 도움말 페이지.

3.3. 애플리케이션에서 TLS 구성 강화

RHEL에서 시스템 전체 암호화 정책 은 암호화 라이브러리를 사용하는 애플리케이션에서 알려진 안전하지 않은 프로토콜, 암호 또는 알고리즘을 허용하지 않도록 하는 편리한 방법을 제공합니다.

사용자 지정 암호화 설정으로 TLS 관련 구성을 강화하려면 이 섹션에 설명된 암호화 구성 옵션을 사용하고 필요한 최소 용량의 시스템 전체 암호화 정책을 재정의할 수 있습니다.

사용하도록 선택한 구성에 관계없이 항상 서버 애플리케이션에서 *서버 측 암호 순서*를 적용해야 합니다. 사용할 암호화 제품군은 구성하는 순서에 따라 결정됩니다.

3.3.1. Apache HTTP server구성

Apache HTTP Server는 TLS 요구 사항에 따라 **OpenSSL** 및 **NSS** 라이브러리를 모두 사용할 수 있습니다. RHEL 8에서는 eponymous 패키지를 통해 **mod_ssl** 기능을 제공합니다.

```
# yum install mod_ssl
```

mod_ssl 패키지는 **Apache HTTP Server**의 TLS 관련 설정을 수정하는 데 사용할 수 있는 **/etc/httpd/conf.d/ssl.conf** 구성 파일을 설치합니다.

httpd-manual 패키지를 설치하여 TLS 구성을 포함하여 **Apache HTTP Server** 에 대한 전체 문서를 가져옵니다. **/etc/httpd/conf.d/ssl.conf** 구성 파일에서 사용 가능한 지시문은 **/usr/share/httpd/manual/mod/mod_ssl.html** 파일에 자세히 설명되어 있습니다. 다양한 설정의 예는 **/usr/share/httpd/manual/ssl/ssl_howto.html** 파일에 설명되어 있습니다.

/etc/httpd/conf.d/ssl.conf 구성 파일의 설정을 수정할 때 최소한 다음 세 가지 지시문을 고려해야 합니다.

SSLProtocol

이 지시문을 사용하여 허용하려는 TLS 또는 SSL 버전을 지정합니다.

SSLCipherSuite

이 지시문을 사용하여 선호하는 암호화 제품군을 지정하거나 허용하지 않을 암호화 제품군을 비활성화합니다.

SSLHonorCipherOrder

연결 클라이언트가 지정한 암호 순서를 준수하는지 확인하기 위해 이 지시문의 주석을 **on** 으로 설정합니다.

예를 들어 TLS 1.2 및 1.3 프로토콜만 사용하려면 다음을 수행합니다.

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

자세한 내용은 다양한 유형의 서버 배포 문서의 Apache HTTP Server에서 TLS 암호화 구성 장을 참조하십시오.

3.3.2. Nginx HTTP 및 프록시 서버 구성

Nginx 에서 TLS 1.3 지원을 활성화하려면 **/etc/nginx/nginx.conf** 구성 파일의 **server** 섹션에 있는 **ssl_protocols** 옵션에 **TLSv1.3** 값을 추가합니다.

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

자세한 내용은 다양한 유형의 서버 배포 문서의 Nginx 웹 서버에 TLS 암호화 추가 장을 참조하십시오.

3.3.3. Dovecot 메일 서버 구성

TLS를 사용하도록 **Dovecot** 메일 서버의 설치를 구성하려면 **/etc/dovecot/conf.d/10-ssl.conf** 구성 파일을 수정합니다. **/usr/share/doc/dovecot/SSL/DovecotConfiguration.txt** 파일에서 사용 가능한 일부 기본 구성 지시문에 대한 설명은 **Dovecot** 표준 설치와 함께 설치됩니다.

/etc/dovecot/conf.d/10-ssl.conf 구성 파일의 설정을 수정할 때 다음 세 지시문을 최소한으로 고려해야 합니다.

ssl_protocols

이 지시문을 사용하여 허용 또는 비활성화하려는 TLS 또는 SSL 버전을 지정합니다.

ssl_cipher_list

이 지시문을 사용하여 선호하는 암호화 제품군을 지정하거나 허용하지 않을 암호화 제품군을 비활성화합니다.

ssl_prefer_server_ciphers

연결 클라이언트가 지정한 암호 순서를 준수하는지 확인하기 위해 이 지시문의 주석을 제거하고 **yes** 로 설정합니다.

예를 들어 **/etc/dovecot/conf.d/10-ssl.conf** 의 다음 행에서는 TLS 1.1 이상만 허용합니다.

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

추가 리소스

- RHEL 8에서 다양한 유형의 서버 배포
- **config(5)** 및 **ciphers(1)** 도움말 페이지.
- TLS(Trans gram Transport Layer Security) 및 DTLS(데이터그램 전송 계층 보안)의 안전한 사용을 위한 권장 사항.
- Mozilla SSL 구성 생성기.
- SSL 서버 테스트.

4장. IPSEC을 사용하여 VPN 구성

RHEL 8에서는 **Libreswan** 애플리케이션에서 지원하는 **IPsec** 프로토콜을 사용하여 VF(가상 사설 네트워크)를 구성할 수 있습니다.

4.1. IPSEC VPN 구현인 LIBRESWAN

RHEL에서는 Libreswan 애플리케이션에서 지원하는 IPsec 프로토콜을 사용하여 VPN(Virtual Private Network)을 구성할 수 있습니다. Libreswan은 Openswan 애플리케이션의 연속이며, Openswan 설명서의 많은 예제는 Libreswan과 상호 운용할 수 있습니다.

VPN용 IPsec 프로토콜은 인터넷 키 교환(IKE) 프로토콜을 사용하여 구성됩니다. IPsec과 IKE라는 용어는 서로 바꿔 사용할 수 있습니다. IPsec VPN은 IKE VPN, IKEv2 VPN, XAUTH VPN, Cisco VPN 또는 IKE/IPsec VPN이라고도 합니다. L2TP(Layer 2 tunneling Protocol)도 사용하는 IPsec VPN의 변형은 일반적으로 **선택적** 리포지토리에서 제공하는 **xl2tpd** 패키지가 필요한 L2TP/IPsec VPN이라고 합니다.

Libreswan은 오픈 소스 사용자 공간 IKE 구현입니다. IKE v1 및 v2는 사용자 수준 데몬으로 구현됩니다. IKE 프로토콜도 암호화됩니다. IPsec 프로토콜은 Linux 커널에 의해 구현되며 Libreswan은 VPN 터널 구성을 추가하고 제거하도록 커널을 구성합니다.

IKE 프로토콜은 UDP 포트 500 및 4500을 사용합니다. IPsec 프로토콜은 두 가지 프로토콜로 구성됩니다.

- 프로토콜 번호 50이 있는 ESP(Security Payload)가 캡슐화되었습니다.
- 프로토콜 번호 51이 있는 인증된 헤더 (AH)

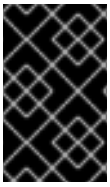
AH 프로토콜은 사용하지 않는 것이 좋습니다. AH 사용자는 null 암호화를 사용하여 ESP로 마이그레이션하는 것이 좋습니다.

IPsec 프로토콜은 다음과 같은 두 가지 작동 모드를 제공합니다.

- 터널 모드(기본값)
- 전송 모드.

IKE 없이 IPsec을 사용하여 커널을 구성할 수 있습니다. 이를 *Manual Keying*이라고 합니다. **ip xfrm** 명령을 사용하여 수동 인증도 구성할 수 있지만 보안상의 이유로 이 방법은 권장되지 않습니다. netlink를 사용하여 Linux 커널과 Libreswan 인터페이스. Linux 커널에서 패킷 암호화 및 암호 해독이 수행됩니다.

Libreswan은 NSS(Network Security Services) 암호화 라이브러리를 사용합니다. Libreswan과 NSS는 모두*FIPS*(*Federal Information Processing Standard*) Publicationdatabind와 함께 사용하도록 인증되었습니다.



중요

Libreswan 및 Linux 커널에서 구현하는 IKE/IPsec VPN은 RHEL에서 사용하는 데 권장되는 유일한 VPN 기술입니다. 이로 인한 위험을 이해하지 못한 채 다른 VPN 기술을 사용하지 마십시오.

RHEL에서 Libreswan은 기본적으로 **시스템 전체 암호화 정책을** 따릅니다. 이렇게 하면 Libreswan이 IKEv2를 기본 프로토콜로 포함한 현재 위협 모델에 대해 보안 설정을 사용하도록 합니다. 자세한 내용은 시스템 전체 암호화 정책 사용을 참조하십시오.

Libreswan은 IKE/IPsec이 피어 프로토콜의 피어이기 때문에 "source" 및 "server" 및 "client"라는 용어를 사용하지 않습니다. 대신 "왼쪽"과 "오른쪽"이라는 용어를 사용하여 엔드 포인트(호스트)를 나타냅니다. 또

한 대부분의 경우 두 엔드포인트 모두에서 동일한 구성을 사용할 수 있습니다. 그러나 일반적으로 관리자는 항상 로컬 호스트에 "왼쪽"을 사용하고 원격 호스트에 "오른쪽"을 사용하도록 선택합니다.

leftid 및 **rightid** 옵션은 인증 프로세스에서 해당 호스트를 식별하는 역할을 합니다. 자세한 내용은 **ipsec.conf(5)** 도움말 페이지를 참조하십시오.

4.2. LIBRESWAN의 인증 방법

Libreswan은 각각 다른 시나리오에 맞는 여러 인증 방법을 지원합니다.

Pre-Shared 키(PSK)

PSK(*Pre-Shared Key*)는 가장 간단한 인증 방법입니다. 보안상의 이유로, 64개의 임의 문자보다 짧은 PSK를 사용하지 마십시오. FIPS 모드에서 PSK는 사용된 무결성 알고리즘에 따라 최소 요구 사항을 준수해야 합니다. **authby=secret** 연결을 사용하여 PSK를 설정할 수 있습니다.

원시 RSA 키

*원시 RSA 키*는 일반적으로 정적 host-host 또는 subnet-to-subnet IPsec 구성에 사용됩니다. 각 호스트는 다른 모든 호스트의 공개 RSA 키를 사용하여 수동으로 구성하고 Libreswan은 각 호스트 쌍 간에 IPsec 터널을 설정합니다. 이 방법은 많은 호스트에 대해 잘 확장되지 않습니다.

ipsec newhostkey 명령을 사용하여 호스트에서 원시 RSA 키를 생성할 수 있습니다. **ipsec showhostkey** 명령을 사용하여 생성된 키를 나열할 수 있습니다. CKA ID 키를 사용하는 연결 구성에는 **leftrsasigkey=** 행이 필요합니다. 원시 RSA 키에 **authby=rsasig** 연결 옵션을 사용합니다.

X.509 인증서

*X.509 인증서*는 일반적으로 공통 IPsec 게이트웨이에 연결된 호스트로 대규모 배포에 사용됩니다. 중앙 *인증 기관* (CA)은 호스트 또는 사용자의 RSA 인증서에 서명합니다. 이 중앙 CA는 개별 호스트 또는 사용자의 취소를 포함하여 신뢰 릴레이를 담당합니다.

예를 들어 **openssl** 명령 및 NSS **certutil** 명령을 사용하여 X.509 인증서를 생성할 수 있습니다. Libreswan은 **leftcert=** 구성 옵션에서 인증서의 닉네임을 사용하여 NSS 데이터베이스에서 사용자 인증서를 읽기 때문에 인증서를 생성할 때 nickname을 제공합니다.

사용자 정의 CA 인증서를 사용하는 경우 NSS(Network Security Services) 데이터베이스로 가져와야 합니다. **ipsec import** 명령을 사용하여 PKCS #12 형식의 인증서를 Libreswan NSS 데이터베이스로 가져올 수 있습니다.

주의

Libreswan에는 RFC 4945의 3.1 절에 설명된 모든 피어 인증서에 대한 주제 대체 이름(SAN)으로 인터넷 키 교환(IKE) 피어 ID가 필요합니다. **require-id-on-certificated=** 옵션을 변경하여 이 검사를 비활성화하면 시스템이 메시지 가로채기(man-in-the-middle) 공격에 취약해질 수 있습니다.

SHA-1 및 SHA-2와 함께 RSA를 사용하는 X.509 인증서를 기반으로 하는 인증에 **authby=rsasig** 연결 옵션을 사용합니다. **authby=** 를 **ecdsa** 및 RSA Probabilistic Signature Scheme(RSASSA-PSS)로 설정하여 SHA-2를 사용하여 ECDSA 디지털 서명에 대해 추가로 제한할 수 있습니다. **authby=rsa-sha2** 를 통해 SHA-2를 사용한 디지털 서명 기반 인증입니다. 기본값은 **authby=rsasig,ecdsa** 입니다.

인증서 및 **authby=** 서명 메서드가 일치해야 합니다. 이는 상호 운용성을 높이고 하나의 디지털 서명 시스템에서 인증을 보존합니다.

NULL 인증

*NULL 인증*은 인증없이 메시 암호화를 얻는 데 사용됩니다. 액티브 공격으로부터 보호하지만 활성 공격으로부터 보호하지 않습니다. 그러나 IKEv2는 비대칭 인증 방법을 허용하므로 Internetscale opportunistic IPsec에도 NULL 인증을 사용할 수 있습니다. 이 모델에서 클라이언트는 서버를 인증하지만 서버는 클라이언트를 인증하지 않습니다. 이 모델은 TLS를 사용하는 보안 웹 사이트와 유사합니다. NULL 인증을 위해 **authby=null** 을 사용합니다.

섀도우 컴퓨터 보호

앞에서 언급한 인증 방법 외에도 PPK(*Post-quantum Pre-shared Key*) 방법을 사용하여 텀 컴퓨터가 통해 가능한 공격으로부터 보호할 수 있습니다. 개별 클라이언트 또는 클라이언트 그룹은 대역 외 구성된 사전 공유 키에 해당하는 PPK ID를 지정하여 자체 PPK를 사용할 수 있습니다.

IKEv1을 사전 공유 키와 함께 사용하면 취약성 공격자로부터 보호할 수 있습니다. IKEv2의 재설계는 기본적으로 이러한 보호를 제공하지 않습니다. Libreswan은 Quantum *Pre-shared Key* (PPK)를 사용하여 IKEv2 연결을 멱등 공격으로부터 보호합니다.

선택적 PPK 지원을 활성화하려면 연결 정의에 **ppk=yes** 를 추가합니다. PPK를 요구하려면 **ppk=insist** 를 추가합니다. 그런 다음 각 클라이언트에 대역 외가 전달되는 비밀 값으로 PPK ID를 부여할 수 있습니다(및 가급적 확장 안전). PPK는 사전 단어를 기반으로 하지 않고 무작위로 매우 강력해야 합니다. PPK ID 및 PPK 데이터는 **ipsec.secrets**에 저장됩니다. 예를 들면 다음과 같습니다.

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

PPKS 옵션은 정적 PPK를 나타냅니다. 이 실험적 기능은 일회용 동적 PPK를 사용합니다. 각 연결에서 1회 패드의 새 부분이 PPK로 사용됩니다. 사용하면 파일 내의 동적 PPK의 해당 부분을 다시 사용하지 않도록 0으로 덮어씁니다. 더 이상 일회성-패드 자료가 남아 있지 않으면 연결이 실패합니다. 자세한 내용은 **ipsec.secrets(5)** 도움말 페이지를 참조하십시오.

주의

동적 PPK의 구현은 지원되지 않는 기술 프리뷰로 제공됩니다. 주의해서 사용하십시오.

4.3. LIBRESWAN 설치

이 절차에서는 Libreswan IPsec/IKE VPN 구현을 설치 및 시작하는 단계를 설명합니다.

사전 요구 사항

- **AppStream** 리포지토리가 활성화되어 있어야 합니다.

절차

1. **libreswan** 패키지를 설치합니다.

```
# yum install libreswan
```

2. Libreswan을 다시 설치하는 경우 이전 데이터베이스 파일을 제거하고 새 데이터베이스를 생성합니다.

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

3. **ipsec** 서비스를 시작하고 부팅 시 서비스를 자동으로 시작합니다.

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

4. **ipsec** 서비스를 추가하여 IKE, ESP 및 AH 프로토콜에 500 및 4500/UDP 포트를 허용하도록 방화벽을 구성합니다.

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

4.4. 호스트 대 호스트 VPN 생성

원시 RSA 키의 인증을 사용하여 두 호스트 간에 호스트 간 IPsec VPN을 생성하려면 두 호스트 *간* 호스트 간 IPsec VPN을 생성하려면 두 호스트 모두에서 다음 명령을 입력합니다.

사전 요구 사항

- *Libreswan이 설치되고* ***ipsec*** *서비스가 각 노드에서 시작됩니다.*

절차

1. *각 호스트에 원시 RSA 키 쌍을 생성합니다.*

```
# ipsec newhostkey
```

2. *이전 단계에서 생성된 키의* ***ckaid****가 반환되었습니다. 예를 들면 왼쪽에서 다음 명령과* ***함께*** *cutord 를 사용합니다.*

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

이전 명령의 출력에서 구성에 필요한 ***leftrsasigkey=*** *행을 생성했습니다. 두 번째 호스트에서 동일한 작업을 수행합니다(오른쪽):*

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

3. ***/etc/ipsec.d/*** *디렉터리에 새* ***my_host-to-host.conf*** *파일을 만듭니다. 이전 단계에서* ***ipsec showhostkey*** *명령의 출력에서 RSA 호스트 키를 새 파일로 작성합니다. 예를 들면 다음과 같습니다.*

```
conn mytunnel
    leftid=@west
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east
```

```
right=192.1.2.45
rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
authby=rsasig
```

4. 키를 가져온 후 **ipsec** 서비스를 다시 시작하십시오.

```
# systemctl restart ipsec
```

5. 연결을 로드합니다.

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

6. 터널을 설정합니다.

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

7. **ipsec** 서비스가 시작될 때 터널을 자동으로 시작하려면 연결 정의에 다음 행을 추가합니다.

```
auto=start
```

4.5. 사이트 간 VPN 구성

사이트 간 IPsec VPN을 생성하려면 두 네트워크를 결합함으로써 두 호스트 간의 IPsec 터널이 생성됩니다. 따라서 호스트는 하나 이상의 서브넷의 트래픽이 통과할 수 있도록 구성된 엔드포인트 역할을 합니다. 따라서 호스트를 네트워크의 원격 부분에 대한 게이트웨이로 간주할 수 있습니다.

사이트-사이트 VPN의 구성은 하나 이상의 네트워크 또는 서브넷이 구성 파일에 지정해야 한다는 점에서 호스트 대 호스트 VPN과만 다릅니다.

사전 요구 사항

- 호스트 대 호스트 VPN이 이미 구성되어 있습니다.

절차

1. host-to-host VPN의 구성으로 파일을 새 파일로 복사합니다. 예를 들면 다음과 같습니다.

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

2. 이전 단계에서 만든 파일에 서브넷 구성을 추가합니다. 예를 들면 다음과 같습니다.

```
conn mysubnet
    also=mytunnel
    leftsubnet=192.0.1.0/24
    rightsubnet=192.0.2.0/24
    auto=start

conn mysubnet6
    also=mytunnel
    leftsubnet=2001:db8:0:1::/64
    rightsubnet=2001:db8:0:2::/64
    auto=start
```

```
# the following part of the configuration file is the same for both host-to-host and site-to-site
connections:

conn mytunnel
    leftid=@west
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east
    right=192.1.2.45
    rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    authby=rsasig
```

4.6. 원격 액세스 VPN 구성

로드 전사는 사용자를 모바일 클라이언트 및 동적으로 할당된 IP 주소로 이동하고 있습니다. 모바일 클라이언트는 X.509 인증서를 사용하여 인증합니다.

다음 예제에서는 **IKEv2** 에 대한 구성을 보여주며 **IKEv1** XAUTH 프로토콜 사용을 방지합니다.

서버에서 다음을 수행합니다.

```
conn roadwarriors
    ikev2=insist
    # support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    fragmentation=yes
    left=1.2.3.4
    # if access to the LAN is given, enable this, otherwise use 0.0.0.0/0
    # leftsubnet=10.10.0.0/16
    leftsubnet=0.0.0.0/0
    leftcert=gw.example.com
    leftid=%fromcert
    leftxauthserver=yes
    leftmodecfgserver=yes
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # pick an IP address pool to assign to remote users
    # 100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
    rightaddresspool=100.64.13.100-100.64.13.254
    # if you want remote clients to use some local DNS zones and servers
    modecfgdns="1.2.3.4, 5.6.7.8"
    modecfgdomains="internal.company.com, corp"
    rightxauthclient=yes
    rightmodecfgclient=yes
    authby=rsasig
    # optionally, run the client X.509 ID through pam to allow or deny client
    # pam-authorize=yes
    # load connection, do not initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

모바일 클라이언트에서 로드 해커의 장치는 이전 구성의 약간의 변형을 사용합니다.

```
conn to-vpn-server
    ikev2=insist
    # pick up our dynamic IP
    left=%defaultroute
    leftsubnet=0.0.0.0/0
    leftcert=myname.example.com
    leftid=%fromcert
    leftmodecfgclient=yes
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
    # rightsubnet=10.10.0.0/16
    rightsubnet=0.0.0.0/0
    fragmentation=yes
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasig
    # allow narrowing to the server's suggested assigned IP and remote subnet
    narrowing=yes
    # support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    # initiate connection
    auto=start
```

4.7. 메시 VPN 구성

임의의 VPN이라고도 하는 메시 VPN 네트워크는 모든 노드가 IPsec을 사용하여 통신하는 네트워크입니다. 구성에서는 IPsec을 사용할 수 없는 노드에 예외를 허용합니다. 메시 VPN 네트워크는 두 가지 방법으로 구성할 수 있습니다.

- *IPsec이 필요합니다.*
- *IPsec을 선호하지만 대체를 사용하여 일반 텍스트 통신을 허용합니다.*

노드 간 인증은 X.509 인증서 또는 DNSSEC(DNS Security Extensions)를 기반으로 할 수 있습니다.

다음 절차에서는 X.509 인증서를 사용합니다. 이러한 인증서는 Dogtag Certificate System과 같은 모든 종류의 CA(인증 기관) 관리 시스템을 사용하여 생성할 수 있습니다. Dogtag는 각 노드의 인증서를 개인 키, 노드 인증서 및 기타 노드의 X.509 인증서의 유효성을 검사하는 데 사용되는 루트 CA 인증서가 포함된 PKCS #12 형식(.p12 파일)에서 사용할 수 있다고 가정합니다.

각 노드에는 X.509 인증서를 제외하고 동일한 구성이 있습니다. 이를 통해 네트워크의 기존 노드를 재구성하지 않고 새 노드를 추가할 수 있습니다. PKCS #12 파일에는 "간단한 이름"이 필요합니다. 이 경우 친숙한 이름을 참조하는 구성 파일이 모든 노드에서 동일하게 유지되도록 이름 "노드"를 사용합니다.

사전 요구 사항

- *Libreswan이 설치되고 **ipsec** 서비스가 각 노드에서 시작됩니다.*

절차

1. *각 노드에서 PKCS #12 파일을 가져옵니다. 이 단계에서는 PKCS #12 파일을 생성하는 데 사용되는 암호가 필요합니다.*

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

2. ***IPsec 필수(사설), IPsec(공유 옵션) 및 No IPsec*** *(공유) 프로필에 대해 다음 세 가지 연결 정의를 만듭니다.*

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

conn private
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=drop
 negotiationshunt=drop
 # left
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert
       leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup

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

3. *적절한 범주에 네트워크의 IP 주소를 추가합니다. 예를 들어 모든 노드가 10.15.0.0/16 네트워크에 있고 모든 노드에 IPsec 암호화가 필요한 경우 다음을 수행하십시오.*

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

4. *특정 노드(예: 10.15.34.0/24)가 IPsec과 함께 작동하도록 허용하려면 다음을 사용하여 private-or-clear 그룹에 해당 노드를 추가합니다.*

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

5. *호스트를 정의하려면 (예: 10.15.1.2) 를 명확한 그룹으로 IPsec 을 사용할 수 없는 호스트를 정의하려면 다음을 사용합니다.*

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

***/etc/ipsec.d/policies** 디렉터리의 파일은 각 새 노드의 템플릿에서 생성하거나 Puppet 또는 Ansible 을 사용하여 프로비저닝할 수 있습니다.*

모든 노드에는 예외 또는 트래픽 흐름 예상과 동일한 목록이 있습니다. 따라서 IPsec 이 필요하고 다른 노드에서 IPsec 을 사용할 수 없기 때문에 두 개의 노드가 통신할 수 없습니다.

6. *노드를 재시작하여 구성된 메시에 추가합니다.*

```
# systemctl restart ipsec
```

7. *노드 추가를 완료하면 **ping** 명령으로 IPsec 터널을 열 수 있습니다. 노드가 연 터널을 보려면 다음을 수행합니다.*

```
# ipsec trafficstatus
```

4.8. FIPS 호환 IPSEC VPN 배포

Libreswan 기반의 FIPS 호환 IPsec VPN 솔루션을 배포하려면 다음 절차를 사용하십시오. 다음 단계를 사용하면 사용할 수 있는 암호화 알고리즘과 FIPS 모드에서 Libreswan 에 대해 비활성화된 암호화 알고리즘도 식별할 수 있습니다.

사전 요구 사항

- ***AppStream** 리포지토리가 활성화되어 있어야 합니다.*

절차

1. ***libreswan** 패키지를 설치합니다.*

```
# yum install libreswan
```

2. *Libreswan 을 다시 설치하는 경우 이전 NSS 데이터베이스를 제거하십시오.*

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

3. ***ipsec** 서비스를 시작하고 부팅 시 서비스를 자동으로 시작합니다.*

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

4. ***ipsec** 서비스를 추가하여 IKE, ESP 및 AH 프로토콜에 500 및 4500/UDP 포트를 허용하도록 방화벽을 구성합니다.*

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

5. *시스템을 FIPS 모드로 전환합니다.*

-

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

6. *커널이 FIPS 모드로 전환되도록 시스템을 다시 시작하십시오.*

```
# reboot
```

검증

1. *Libreswan이 FIPS 모드에서 실행 중인지 확인하려면 다음을 수행합니다.*

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

2. *또는* ***systemd*** *저널의* ***ipsec*** *유닛 항목을 확인합니다.*

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

3. *FIPS 모드에서 사용 가능한 알고리즘을 보려면 다음을 수행합니다.*

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

4. *FIPS 모드에서 비활성화된 알고리즘을 쿼리하려면 다음을 수행합니다.*

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

```
DH algorithm MODP1024 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant
```

5. FIPS 모드에서 허용되는 모든 알고리즘 및 암호를 나열하려면 다음을 수행합니다.

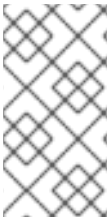
```
# ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
{256,192,*128}  aes_ccm, aes_ccm_c
{256,192,*128}  aes_ccm_b
{256,192,*128}  aes_ccm_a
[*192]  3des
{256,192,*128}  aes_gcm, aes_gcm_c
{256,192,*128}  aes_gcm_b
{256,192,*128}  aes_gcm_a
{256,192,*128}  aesctr
{256,192,*128}  aes
{256,192,*128}  aes_gmac
sha, sha1, sha1_96, hmac_sha1
sha512, sha2_512, sha2_512_256, hmac_sha2_512
sha384, sha2_384, sha2_384_192, hmac_sha2_384
sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
null, dh0
dh14
dh15
dh16
dh17
dh18
ecp_256, ecp256
ecp_384, ecp384
ecp_521, ecp521
```

추가 리소스

- *시스템 전체 암호화 정책 사용*.

4.9. 암호로 IPSEC NSS 데이터베이스 보호

기본적으로 IPsec 서비스는 처음 시작하는 동안 비어 있는 암호를 사용하여 NSS(Network Security Services) 데이터베이스를 생성합니다. 다음 단계를 사용하여 암호 보호를 추가합니다.



참고

이전 RHEL 버전 6.6 버전에서는 NSS 암호화 라이브러리가 FIPS 6443 레벨 2 표준에 대해 인증되었기 때문에 FIPShiera 요구 사항을 충족하는 암호로 IPsec NSS 데이터베이스를 보호해야 했습니다. RHEL 8에서는 NIST 인증 NSS를 이 표준의 수준1로 인증했으며 이 상태에는 데이터베이스에 대한 암호 보호가 필요하지 않습니다.

사전 요구 사항

- **/etc/ipsec.d/** 디렉토리에는 NSS 데이터베이스 파일이 포함되어 있습니다.

절차

1. Libreswan의 **NSS** 데이터베이스에 대해 암호 보호를 활성화합니다.

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

Enter new password:
```

2. 이전 단계에서 설정한 암호가 포함된 **/etc/ipsec.d/nsspassword** 파일을 만듭니다. 예를 들면 다음과 같습니다.

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

nsspassword 파일은 다음 구문을 사용합니다.

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

기본 NSS 소프트웨어 토큰은 **NSS Certificate DB** 입니다. 시스템이 FIPS 모드에서 실행 중인 경우 토큰 이름은 **NSS FIPS 140-2 Certificate DB** 입니다.

3. 시나리오에 따라 **nsspassword** 파일을 완료한 후 **ipsec** 서비스를 시작하거나 다시 시작합니다.

```
# systemctl restart ipsec
```

검증

1. NSS 데이터베이스에 비어 있지 않은 암호를 추가한 후 **ipsec** 서비스가 실행 중인지 확인합니다.

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

2. 선택적으로 **Journal** 로그에 초기화가 성공했는지 확인하는 항목이 포함되어 있는지 확인합니다.

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

추가 리소스

- **certutil(1)** 도움말 페이지.

- 정부 표준 지식 베이스 문서.

4.10. TCP를 사용하도록 IPSEC VPN 구성

Libreswan은 RFC 8229에 설명된 대로 IKE 및 IPsec 패킷을 TCP 캡슐화를 지원합니다. 이 기능을 사용하면 UDP를 통해 전송되는 트래픽을 방지하고 ESB(Security Payload)를 캡슐화하는 네트워크에서 IPsec VPN을 설정할 수 있습니다. TCP를 대체 또는 기본 VPN 전송 프로토콜로 사용하도록 VPN 서버와 클라이언트를 구성할 수 있습니다. TCP 캡슐화는 성능 비용이 늘어날 수 있으므로 시나리오에서 UDP가 영구적으로 차단된 경우에만 TCP를 기본 VPN 프로토콜로 사용하십시오.

사전 요구 사항

- 원격 액세스 VPN 이 이미 구성되어 있습니다.

절차

1. **config setup** 섹션의 **/etc/ipsec.conf** 파일에 다음 옵션을 추가합니다.

```
listen-tcp=yes
```

2. UDP를 처음 시도하지 못하면 TCP 캡슐화를 대체 옵션으로 사용하려면 클라이언트의 연결 정의에 다음 두 옵션을 추가합니다.

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

또는 UDP가 영구적으로 차단되었음을 알고 있는 경우 클라이언트 연결 구성에서 다음 옵션을 사용합니다.

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

추가 리소스

- IETF RFC 8229: IKE 및 IPsec Packets의 TCP 캡슐화 .

4.11. IPSEC 연결 속도를 높이기 위해 ESP 하드웨어 오프로드 자동 감지 및 사용 구성

하드웨어로 캡슐화된 보안 페이로드(ESP)를 오프로드하면 이더넷을 통해 IPsec 연결을 가속화할 수 있습니다. 기본적으로 Libreswan은 하드웨어가 이 기능을 지원하는지 감지하여 ESP 하드웨어 오프로드를 활성화합니다. 이 절차에서는 기능을 비활성화하거나 명시적으로 사용하도록 설정한 경우 자동 탐지를 활성화하는 방법을 설명합니다.

사전 요구 사항

- 네트워크 카드는 ESP 하드웨어 오프로드를 지원합니다.
- 네트워크 드라이버는 ESP 하드웨어 오프로드를 지원합니다.
- IPsec 연결이 구성되고 작동합니다.

절차

1. *ESP 하드웨어 오프로드 지원의 자동 탐지를 사용해야 하는 연결의 **/etc/ipsec.d/** 디렉토리에서 Libreswan 구성 파일을 편집합니다.*
2. *연결 설정에 **nic-offload** 매개변수가 설정되어 있지 않은지 확인합니다.*
3. ***nic-offload** 를 제거한 경우 **ipsec** 서비스를 다시 시작합니다.*

```
# systemctl restart ipsec
```

검증

네트워크 카드가 ESP 하드웨어 오프로드 지원을 지원하는 경우 다음 단계에 따라 결과를 확인하십시오.

1. *IPsec 연결에 사용하는 이더넷 장치의 **tx_ipsec** 및 **rx_ipsec** 카운터를 표시합니다.*

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

2. *IPsec 터널을 통해 트래픽을 전송합니다. 예를 들어 원격 IP 주소를 ping 합니다.*

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

3. *이더넷 장치의 **tx_ipsec** 및 **rx_ipsec** 카운터를 다시 표시합니다.*

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

카운터 값이 증가하면 ESP 하드웨어 오프로드가 작동합니다.

추가 리소스

- *IPsec을 사용하여 VPN 구성*

4.12. IPSEC 연결을 가속화하도록 본딩에 ESP 하드웨어 오프로드 구성

하드웨어로 ESB(Security Payload)를 오프로드하면 IPsec 연결 속도가 빨라집니다. 네트워크 본딩을 장애 조치의 이유로 사용하는 경우 ESP 하드웨어 오프로드를 구성하는 절차와 일반 이더넷 장치를 사용하는 절차가 다릅니다. 예를 들어 이 시나리오에서는 본딩에 대한 오프로드 지원을 활성화하고 커널은 설정을 본딩 포트에 적용합니다.

사전 요구 사항

- *본딩의 모든 네트워크 카드는 ESP 하드웨어 오프로드를 지원합니다.*
- *네트워크 드라이버는 본딩 장치에서 ESP 하드웨어 오프로드를 지원합니다. RHEL 에서는 **ixgbe** 드라이버만 이 기능을 지원합니다.*
- *본딩이 구성되고 작동합니다.*
- *본딩에서는 **active-backup** 모드를 사용합니다. 본딩 드라이버는 이 기능에 대해 다른 모드를 지원하지 않습니다.*

- *IPsec 연결이 구성되고 작동합니다.*

절차

1. *네트워크 본딩에서 ESP 하드웨어 오프로드 지원을 활성화합니다.*

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

   *이 명령을 사용하면 **bond0** 연결에서 ESP 하드웨어 오프로드를 지원할 수 있습니다.*

2. ***bond0** 연결을 다시 활성화합니다.*

```
# nmcli connection up bond0
```

3. *ESP 하드웨어 오프로드를 사용해야 하는 연결의 **/etc/ipsec.d/** 디렉토리에서 Libreswan 구성 파일을 편집하고 **nic-offload=yes** 문을 연결 항목에 추가합니다.*

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

4. ***ipsec** 서비스를 다시 시작하십시오.*

```
# systemctl restart ipsec
```

검증

1. *본딩의 활성 포트를 표시합니다.*

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

2. *활성 포트의 **tx_ipsec** 및 **rx_ipsec** 카운터를 표시합니다.*

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

3. *IPsec 터널을 통해 트래픽을 전송합니다. 예를 들어 원격 IP 주소를 ping 합니다.*

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

4. *활성 포트의 **tx_ipsec** 및 **rx_ipsec** 카운터를 다시 표시합니다.*

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

   카운터 값이 증가하면 ESP 하드웨어 오프로드가 작동합니다.

추가 리소스

- 네트워크 본딩 구성
- IPsec을 사용하여 VPN 구성
- 네트워크 보안 문서의 IPsec 장을 사용하여 VPN 구성.

4.13. 시스템 전체 암호화 정책에서 비활성화된 IPSEC 연결 구성

연결에 대한 시스템 전체 암호화 정책 덮어쓰기

RHEL 시스템 전체 암호화 정책은 **%default** 라는 특수 연결을 생성합니다. 이 연결에는 the **ikev2,esp** 및 **ike** 옵션의 기본값이 포함되어 있습니다. 그러나 연결 구성 파일에 언급된 옵션을 지정하여 기본값을 재정의할 수 있습니다.

예를 들어 다음 구성에서는 AES 및 SHA-1 또는 SHA-2와 함께 IKEv1을 사용하고 IPsec(ESP)을 AES-GCM 또는 AES-CBC와 함께 사용하는 연결을 허용합니다.

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

AES-GCM은 IPsec (ESP) 및 IKEv2에서는 사용할 수 있지만 IKEv1에는 사용할 수 없습니다.

모든 연결에 대한 시스템 전체 암호화 정책 비활성화

모든 IPsec 연결에 대한 시스템 전체 암호화 정책을 비활성화하려면 **/etc/ipsec.conf** 파일에서 다음 행을 주석 처리하십시오.

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

그런 다음 연결 구성 파일에 **ikev2=never** 옵션을 추가합니다.

추가 리소스

- 시스템 전체 암호화 정책 사용.

4.14. IPSEC VPN 구성 문제 해결

IPsec VPN 구성과 관련된 문제는 여러 가지 주요 이유로 발생합니다. 이러한 문제가 발생하면 문제의 원인이 다음 시나리오에 해당하는지 확인하고 해당 솔루션을 적용할 수 있습니다.

기본 연결 문제 해결

VPN 연결에 대한 대부분의 문제는 관리자가 구성 옵션과 일치하지 않는 엔드포인트를 구성한 새로운 배포에서 발생합니다. 또한 작동 중인 구성은 새로 호환되지 않는 값 때문에 갑자기 작동을 중지할 수 있습니다. 이는 관리자가 구성을 변경한 결과일 수 있습니다. 또는 관리자가 암호화 알고리즘과 같은 특정 옵션에 대해 다양한 기본값을 사용하여 펌웨어 업데이트 또는 패키지 업데이트를 설치할 수 있습니다.

IPsec VPN 연결이 설정되었는지 확인하려면 다음을 수행하십시오.

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

출력이 비어 있거나 연결 이름이 인 항목이 표시되지 않으면 터널이 손상됩니다.

문제가 연결에 있는지 확인하려면 다음을 수행하십시오.

1. vpn.example.com 연결을 다시 로드합니다.

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

2. 다음으로 VPN 연결을 시작합니다.

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

방화벽 관련 문제

가장 일반적인 문제는 IPsec 엔드포인트 중 하나 또는 엔드포인트 사이의 라우터에 있는 방화벽이 모든 IKE(Internet Key Exchange) 패킷을 삭제하는 것입니다.

- IKEv2의 경우 다음 예제와 유사한 출력은 방화벽에 문제가 있음을 나타냅니다.

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

- IKEv1의 경우 시작 명령의 출력은 다음과 같습니다.

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

IPsec을 설정하는 데 사용되는 IKE 프로토콜은 암호화되므로 **tcpdump** 도구를 사용하여 제한된 문제 하위 집합만 해결할 수 있습니다. 방화벽이 IKE 또는 IPsec 패킷을 삭제하는 경우 **tcpdump** 유틸리티를 사용하여 원인을 찾을 수 있습니다. 그러나 **tcpdump** 는 IPsec VPN 연결의 다른 문제를 진단할 수 없습니다.

- **eth0** 인터페이스에서 VPN과 암호화된 모든 데이터의 협상을 캡처하려면 다음을 수행합니다.

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

일치하지 않는 알고리즘, 프로토콜 및 정책

VPN 연결에서는 엔드포인트에 IKE 알고리즘, IPsec 알고리즘 및 IP 주소 범위가 일치해야 합니다. 불일치가 발생하면 연결에 실패합니다. 다음 방법 중 하나를 사용하여 일치하지 않는 경우 알고리즘, 프로토콜 또는 정책을 조정하여 수정합니다.

- 원격 엔드포인트가 IKE/IPsec을 실행 중이 아닌 경우 이를 나타내는 ICMP 패킷이 표시됩니다. 예를 들면 다음과 같습니다.

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

- 일치하지 않는 알고리즘의 예:

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

- 일치하지 않는 IPsec 알고리즘의 예:

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

  일치하는 버전과 일치하지 않으면 응답 없이 원격 엔드포인트가 요청을 삭제할 수도 있었습니다. 이는 모든 IKE 패킷을 삭제하는 방화벽과 동일합니다.

- IKEv2 (Traffic Selectors - TS)의 일치하지 않는 IP 주소 범위의 예:

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

- IKEv1의 일치하지 않는 IP 주소 범위의 예:

```
# ipsec auto --up vpn.example.com
...
031 "vpn.example.com" #2: STATE_QUICK_I1: 60 second timeout exceeded after 0
```

```
retransmits.  No acceptable response to our first Quick Mode message: perhaps peer likes
no proposal
```

- IKEv1에서 PreSharedKeys ()를 사용할 때 양쪽이 동일한 에 배치되지 않으면 전체 IKE 메시지가 읽을 수 없게 됩니다.

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

- IKEv2에서 불일치- 오류로 인해 AUTHENTICATION_FAILED 메시지가 표시됩니다.

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

최대 전송 단위

IKE 또는 IPsec 패킷을 차단하는 방화벽 이외의 네트워킹 문제의 가장 일반적인 원인은 암호화된 패킷의 증가된 패킷 크기와 관련이 있습니다. 최대 전송 단위(MTU)보다 큰 네트워크 하드웨어 조각 패킷(예: 1500바이트). 종종 조각이 손실되고 패킷이 다시 집계되지 않습니다. 이로 인해 작은 규모의 패킷을 사용하는 ping 테스트가 작동하지만 다른 트래픽이 실패하면 간헐적인 오류가 발생합니다. 이 경우 SSH 세션을 설정할 수 있지만 원격 호스트에 'ls -al /usr' 명령을 입력하여 바로 터미널이 중지됩니다.

이 문제를 해결하려면 터널 구성 파일에 **mtu=1400** 옵션을 추가하여 MTU 크기를 줄입니다.

또는 TCP 연결의 경우 MSS 값을 변경하는 iptables 규칙을 활성화합니다.

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

이전 명령에서 시나리오의 문제를 해결하지 않으면 **set-mss** 매개변수에 더 작은 크기를 직접 지정합니다.

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

NAT(네트워크 주소 변환)

IPsec 호스트가 NAT 라우터 역할을 할 때 실수로 패킷을 다시 매핑할 수 있습니다. 다음 예제 구성은 문제를 보여줍니다.

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

주소가 172.16.0.1인 시스템에는 NAT 규칙이 있습니다.

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

주소 10.0.2.33의 시스템이 패킷을 192.168.0.1로 보내는 경우 라우터는 IPsec 암호화를 적용하기 전에 소스 10.0.2.33을 172.16.0.1로 변환합니다.

그런 다음 소스 주소가 10.0.2.33인 패킷이 더 이상 **conn myvpn** 설정과 일치하지 않으며 IPsec은 이 패킷을 암호화하지 않습니다.

이 문제를 해결하려면 라우터의 대상 IPsec 서브넷 범위에 대해 NAT를 제외하는 규칙을 삽입합니다. 예를 들면 다음과 같습니다.

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

커널 IPsec 하위 시스템 버그

예를 들어 버그로 인해 IKE 사용자 공간과 IPsec 커널이 동기화 해제되는 경우 커널 IPsec 하위 시스템이 실패할 수 있습니다. 이러한 문제를 확인하려면 다음을 수행합니다.

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

이전 명령의 출력에 0이 아닌 값은 문제가 있음을 나타냅니다. 이 문제가 발생하면 새 지원 케이스를 열고 해당 IKE 로그와 함께 이전 명령의 출력을 연결합니다.

Libreswan 로그

Libreswan 로그는 기본적으로 **syslog** 프로토콜을 사용합니다. **journalctl** 명령을 사용하여 IPsec과 관련된 로그 항목을 찾을 수 있습니다. 로그에 대한 해당 항목은 **pluto** IKE 데몬에 의해 전송되므로 "pluto" 키워드를 검색합니다. 예를 들면 다음과 같습니다.

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

ipsec 서비스에 대한 실시간 로그를 표시하려면 다음을 수행합니다.

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

기본 로깅 수준에 설정 문제가 표시되지 않으면 **/etc/ipsec.conf** 파일의 **config setup** 섹션에 **plutodebug=all** 옵션을 추가하여 디버그 로그를 활성화합니다.

디버그 로깅은 많은 항목을 생성하며, **journald** 또는 **syslogd** 서비스 속도에 따라 **syslog** 메시지가 제한될 수 있습니다. 전체 로그가 있는지 확인하려면 로깅을 파일로 리디렉션합니다. **/etc/ipsec.conf**를 편집하고 **config setup** 섹션에 **logfile=/var/log/pluto.log**를 추가합니다.

추가 리소스

- 로그 파일을 사용하여 문제 해결.
- **tcpdump(8)** 및 **ipsec.conf(5)** 도움말 페이지.
- firewalld 사용 및 구성

4.15. 추가 리소스

- **ipsec(8)**, **ipsec.conf(5)**, **ipsec.secrets(5)**, **ipsec_auto(8)** 및 **ipsec_rsasigkey(8)** 도움말 페이지.

- ***/usr/share/doc/libreswan-version/*** *directory*.
- *업스트림 프로젝트의 웹사이트*.
- *Libreswan 프로젝트 Wiki*.
- *모든 Libreswan 도움말 페이지*.
- *NIST 특수 발행 800-77: IPsec VPN 가이드*.

5장. VPN RHEL 시스템 역할을 사용하여 IPSEC으로 VPN 연결 구성

VPN 시스템 역할을 사용하면 Red Hat Ansible Automation Platform을 사용하여 RHEL 시스템에서 VPN 연결을 구성할 수 있습니다. 호스트 간, 네트워크 간, VPN 원격 액세스 서버 및 메시 구성을 설정하는 데 사용할 수 있습니다.

호스트 간 연결의 경우 역할은 필요에 따라 기본 매개 변수를 사용하여 **vpn_connections** 목록에 있는 각 호스트 쌍 간에 VPN 터널을 설정합니다. 또는 나열된 모든 호스트 간에 기회 메시 구성을 생성하도록 구성할 수 있습니다. 이 역할은 **호스트에** 있는 호스트의 이름이 Ansible 인벤토리에 사용되는 호스트의 이름과 동일하며, 이러한 이름을 사용하여 터널을 구성할 수 있다고 가정합니다.

참고

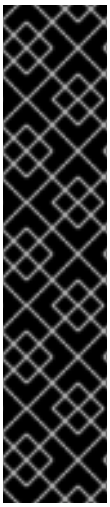
VPN RHEL 시스템 역할은 현재 VPN 공급자로서 IPsec 구현인 Libreswan만 지원합니다.

5.1. VPN 시스템 역할을 사용하여 IPSEC을 사용하여 호스트 간 VPN 생성

VPN 시스템 역할을 사용하여 인벤토리 파일에 나열된 모든 관리 노드를 구성할 제어 노드에서 Ansible 플레이북을 실행하여 호스트 간 연결을 구성할 수 있습니다.

사전 요구 사항

- VPN 시스템 역할을 사용하여 구성할 시스템인 하나 이상의 관리 노드에 대한 액세스 및 사용 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.
  제어 노드에서 다음이 있어야 합니다.
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.



중요

RHEL 8.0-8.5는 Ansible을 기반으로 하는 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 ansible , **ansible -playbook**, **docker** 및 **podman** 와 같은 커넥터, 많은 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 가져오고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법을 참조하십시오.

RHEL 8.6 및 9.0에는 Ansible 명령줄 유틸리티, 명령 및 일부 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)가 도입되었습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며, 제한된 지원 범위가 있습니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식베이스 문서에 포함된 Ansible Core 패키지의 지원 범위를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일이 있어야 합니다.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
- name: Host to host VPN
  hosts: managed_node1, managed_node2
  roles:
```

```
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
```

이 플레이북은 시스템 역할로 자동 생성된 키와 함께 사전 공유 키 인증을 사용하여 ***managed_node1-to- managed_node2*** *연결을 구성합니다.*

2. *선택 사항: 다음 섹션을 호스트 목록에 추가하여 인벤토리 파일에 나열되지 않은 관리 호스트에서 외부 호스트로* ***의*** *연결을 구성합니다.*

```
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
          external_node:
            hostname: 192.0.2.2
```

그러면 두 개의 추가 연결, 즉 ***managed_node1-to-external_node*** *및* ***managed_node2-to-external_node****가 구성됩니다*.

참고

연결은 외부 노드가 아닌 관리형 노드에서만 구성됩니다.

1. *선택 사항:* ***vpn_connections*** *내의 추가 섹션을 사용하여 관리되는 노드에 여러 VPN 연결을 지정할 수 있습니다(예: 컨트롤 플레인 및 데이터 플레인).*

```
- name: Multiple VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed_node1:
            hostname: 192.0.2.0 # IP for the control plane
          managed_node2:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed_node1:
            hostname: 10.0.0.1 # IP for the data plane
          managed_node2:
            hostname: 10.0.0.2
```

2. *선택 사항: 환경 설정에 따라 변수를 수정할 수 있습니다. 자세한 내용은* ***/usr/share/doc/rhel-system-roles/vpn/README.md*** *파일을 참조하십시오.*

3. *선택 사항: 플레이북 구문을 확인합니다.*

```
# ansible-playbook --syntax-check /path/to/file/playbook.yml -i /path/to/file/inventory_file
```

4. *인벤토리 파일에서 플레이북을 실행합니다.*

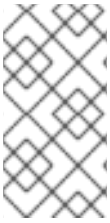
```
# ansible-playbook -i /path/to/file/inventory_file /path/to/file/playbook.yml
```

검증

1. *관리형 노드에서 연결이 성공적으로 로드되었는지 확인합니다.*

```
# ipsec status | grep connection.name
```

connection.name을 이 노드의 연결 이름으로 교체합니다(예: ***managed_node1-to-managed_node2****).*



참고

기본적으로 역할은 각 시스템의 관점에서 생성하는 각 연결에 대해 설명이 포함된 이름을 생성합니다. 예를 들어 ***managed_node1과 managed_node 2 간에 연결을 생성하는 경우 managed_node 1에서 이 연결의 설명이 managed_node 1-to-managed_node2이지만 managed_node2*** *의 경우 연결의 이름은* ***managed_node2-to-managed_node1*** *입니다.*

1. *관리형 노드에서 연결이 성공적으로 시작되었는지 확인합니다.*

```
# ipsec trafficstatus | grep connection.name
```

2. *선택 사항: 연결이 성공적으로 로드되지 않은 경우 다음 명령을 입력하여 연결을 수동으로 추가합니다. 이렇게 하면 연결 설정 실패 이유를 나타내는 더 구체적인 정보를 제공합니다.*

```
# ipsec auto --add connection.name
```

참고

연결을 로드하고 시작하는 동안 발생한 모든 오류는 로그에 보고되며, ***/var/log/pluto.log****에서 확인할 수 있습니다. 이러한 로그는 구문 분석하기 어렵기 때문에 대신 표준 출력에서 로그 메시지를 가져오기 위해 수동으로 연결을 추가합니다.*

5.2. VPN 시스템 역할을 사용하여 IPSEC을 통한 OPPORTUNISTIC 메시 VPN 연결 생성

VPN 시스템 역할을 사용하여 인벤토리 파일에 나열된 모든 관리 노드를 구성할 제어 노드에서 Ansible 플레이북을 실행하여 인증을 위해 인증서를 사용하는 opportunistic 메시 VPN 연결을 구성할 수 있습니다.

인증서로 인증은 플레이북에서 ***auth_method: cert*** *매개 변수를 정의하여 구성합니다. VPN 시스템 역할은* ***/etc/ipsec.d*** *디렉터리에 정의된 IPsec Network Security Services(NSS) 암호화 라이브러리에 필요한 인증서가 포함되어 있다고 가정합니다. 기본적으로 노드 이름은 인증서 닉네임으로 사용됩니다. 이 예에서는* ***managed_node1*** *입니다. 인벤토리의* ***cert_name*** *속성을 사용하여 다른 인증서 이름을 정의할 수 있습니다.*

다음 예제 절차에서는 Ansible 플레이북을 실행하는 시스템인 제어 노드가 관리 노드(192.0.2.0/24)와 동일한 클래스 없는 상호 도메인 라우팅(CIDR) 번호를 공유하며 IP 주소는 192.0.2.7입니다. 따라서 제어 노드는 CIDR 192.0.2.0/24에 대해 자동으로 생성되는 프라이빗 정책에 따릅니다.

플레이 중에 SSH 연결 손실을 방지하기 위해 제어 노드에 대한 명확한 정책이 정책 목록에 포함됩니다. CIDR이 기본값과 같은 정책 목록에도 항목이 있습니다. 이는 이 플레이북이 기본 정책의 규칙을 재정의하여 private-or-clear 대신 비공개로 만들기 때문입니다.

사전 요구 사항

- *VPN 시스템 역할을 사용하여 구성할 시스템인 하나 이상의 관리 노드에 대한 액세스 및 사용 권한.*
  - *모든 관리형 노드에서 **/etc/ipsec.d** 디렉터리의 NSS 데이터베이스에는 피어 인증에 필요한 모든 인증서가 포함되어 있습니다. 기본적으로 노드 이름은 인증서 닉네임으로 사용됩니다.*
- *Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.*
  제어 노드에서 다음이 있어야 합니다.
  - ***ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.*

중요

*RHEL 8.0-8.5는 Ansible을 기반으로 하는 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 ansible , **ansible -playbook**, **docker** 및 **podman** 와 같은 커넥터, 많은 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 가져오고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법을 참조하십시오.*

*RHEL 8.6 및 9.0에는 Ansible 명령줄 유틸리티, 명령 및 일부 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)가 도입되었습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며, 제한된 지원 범위가 있습니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식베이스 문서에 포함된 Ansible Core 패키지의 지원 범위를 참조하십시오.*

- *관리 노드를 나열하는 인벤토리 파일이 있어야 합니다.*

절차

1. *다음 내용으로 새 **playbook.yml** 파일을 생성합니다.*

```
- name: Mesh VPN
  hosts: managed_node1, managed_node2, managed_node3
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
            cidr: 192.0.2.7/32
```

2. *선택 사항: 환경 설정에 따라 변수를 수정할 수 있습니다. 자세한 내용은* ***/usr/share/doc/rhel-system-roles/vpn/README.md*** *파일을 참조하십시오.*

3. *선택 사항: 플레이북 구문을 확인합니다.*

```
# ansible-playbook --syntax-check playbook.yml
```

4. *인벤토리 파일에서 플레이북을 실행합니다.*

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

5.3. 추가 리소스

- *VPN 시스템 역할에 사용되는 매개변수 및 역할에 대한 자세한 내용은* ***/usr/share/doc/rhel-system-roles/vpn/README.md*** *파일을 참조하십시오.*

- ***ansible-playbook*** *명령에 대한 자세한 내용은* ***ansible-playbook(1)*** *도움말 페이지를 참조하십시오.*

6장. MACSEC을 사용하여 동일한 물리적 네트워크에서 계층 2 트래픽 암호화

MACsec을 사용하여 두 장치(point-to-point) 간의 통신을 보호할 수 있습니다. 예를 들어, 지사 사무실은 중앙 사무실과 Metro-Ethernet 연결을 통해 연결되어 있으며, 사무실을 연결하는 두 호스트에서 MACsec을 구성하여 보안을 강화할 수 있습니다.

Media Access Control Security(MACsec)는 다음을 포함하여 이더넷 링크를 통해 다양한 트래픽 유형을 보호하는 계층 2 프로토콜입니다.

- DHCP(동적 호스트 구성 프로토콜)
- 주소 확인 프로토콜(ARP)
- 인터넷 프로토콜 버전 4 / 6 (**IPv4** / **IPv**6) 및
- TCP 또는 UDP와 같은 IP를 통한 트래픽

MACsec은 기본적으로 GCM-AES-128 알고리즘을 사용하여 LAN의 모든 트래픽을 암호화하고 인증하며, 사전 공유 키를 사용하여 참가자 호스트 간 연결을 설정합니다. 사전 공유 키를 변경하려면 MACsec을 사용하는 네트워크의 모든 호스트에서 NM 구성을 업데이트해야 합니다.

MACsec 연결은 이더넷 네트워크 카드, VLAN 또는 터널 장치와 같은 이더넷 장치를 상위로 사용합니다. MACsec 장치에서만 암호화된 연결을 사용하는 다른 호스트와 통신하도록 IP 구성을 설정하거나 상위 장치에서 IP 구성을 설정할 수도 있습니다. 후자의 경우, 암호화되지 않은 연결을 사용하고 MACsec 장치를 사용하여 암호화된 연결에 MACsec 장치를 사용하여 다른 호스트와 통신하는 데 상위 장치를 사용할 수 있습니다.

MACsec에는 특별한 하드웨어가 필요하지 않습니다. 예를 들어 호스트와 스위치 간의 트래픽만 암호화하려는 경우를 제외하고 모든 스위치를 사용할 수 있습니다. 이 시나리오에서는 스위치에서 MACsec도 지원해야 합니다.

즉, MACsec을 구성하는 일반적인 두 가지 방법이 있습니다.

- 호스트 대 호스트 및
- 다른 호스트로 전환한 후 다른 호스트로 전환하는 호스트

중요

MACsec은 동일한(물리적 또는 가상) LAN에 있는 호스트 간에만 사용할 수 있습니다.

6.1. NMCLI를 사용하여 MACSEC 연결 구성

nmcli 유틸리티를 사용하여 MACsec을 사용하도록 이더넷 인터페이스를 구성할 수 있습니다. 이 절차에서는 이더넷을 통해 연결된 두 호스트 간에 MACsec 연결을 생성하는 방법을 설명합니다.

절차

1. MACsec을 구성하는 첫 번째 호스트에서 다음을 수행합니다.
   - 사전 공유 키에 대해 연결 연결 키(CAK) 및 연결 연결 키 이름(CKN)을 만듭니다.
     a. 16바이트 16진수 CAK를 생성합니다.

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

b. *32 바이트 16 진수 CKN 을 생성합니다.*

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

2. *두 호스트 모두에서 MACsec 연결을 통해 연결하려고 합니다.*

3. *MACsec 연결을 생성합니다.*

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

macsec.mka-cak *및* ***macsec.mka-ckn*** *매개변수에서 이전 단계에서 생성된 CAK 및 CKN 을 사용합니다. MACsec 보호 네트워크의 모든 호스트에서 값이 동일해야 합니다.*

4. *MACsec 연결에서 IP 설정을 구성합니다.*

a. ***IPv4*** *설정을 구성합니다. 예를 들어 정적* ***IPv4*** *주소, 네트워크 마스크, 기본 게이트웨이 및 DNS 서버를* ***macsec0*** *연결로 설정하려면 다음을 입력합니다.*

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

b. ***IPv6*** *설정을 구성합니다. 예를 들어 정적* ***IPv6*** *주소, 네트워크 마스크, 기본 게이트웨이 및 DNS 서버를* ***macsec0*** *연결로 설정하려면 다음을 입력합니다.*

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

5. *연결을 활성화합니다.*

```
# nmcli connection up macsec0
```

검증 단계

1. *트래픽이 암호화되었는지 확인합니다.*

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

2. *선택 사항: 암호화되지 않은 트래픽을 표시합니다.*

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

3. *MACsec 통계 표시:*

```
# ip macsec show
```

4. *각 보호 유형에 대한 개별 카운터 표시: 무결성 전용(암호화) 및 암호화(encrypt)*

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

6.2. 추가 리소스

- *MACsec: 네트워크 트래픽 블로그를 암호화하는 다른 솔루션입니다*.

7장. FIREWALLD 사용 및 구성

방화벽 은 시스템을 외부로부터 원하지 않는 트래픽으로부터 보호하는 방법입니다. 사용자가 방화벽 규칙 집합을 정의하여 호스트 시스템에서 들어오는 네트워크 트래픽을 제어할 수 있습니다. 이러한 규칙은 들어오는 트래픽을 정렬하고 차단하거나 을 통해 허용하는 데 사용됩니다.

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

firewalld 는 트래픽 관리를 간소화하는 영역과 서비스의 개념을 사용합니다. 영역은 사전 정의된 규칙 세트입니다. 네트워크 인터페이스 및 소스를 영역에 할당할 수 있습니다. 허용되는 트래픽은 컴퓨터가 연결된 네트워크에 따라 달라지며 이 네트워크가 할당된 보안 수준입니다. 방화벽 서비스는 특정 서비스에 대해 들어오는 트래픽을 허용하기 위해 필요한 모든 설정을 포괄하고 영역 내에서 적용되는 사전 정의된 규칙입니다.

서비스는 네트워크 통신에 하나 이상의 포트 또는 주소를 사용합니다. 방화벽은 포트를 기반으로 통신을 필터링합니다. 서비스에 대한 네트워크 트래픽을 허용하려면 포트를 열어야 합니다. **firewalld** 는 명시적으로 open으로 설정되지 않은 포트의 모든 트래픽을 차단합니다. trusted와 같은 일부 영역에서 기본적으로 모든 트래픽을 허용합니다.

nftables 백엔드가 있는 **firewalld** 는 **--direct** 옵션을 사용하여 사용자 지정 **nftables** 규칙을 **firewalld** 에 전달하는 것을 지원하지 않습니다.

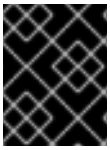
7.1. FIREWALLD 시작하기

이 섹션에서는 **firewalld** 에 대한 정보를 제공합니다.

7.1.1. 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. 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.
- **nftables**: **nftables** 유틸리티를 사용하여 전체 네트워크에 대해 등의 복잡하고 성능 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 Red Hat Enterprise Linux에서 계속 작동합니다. 새로운 방화벽 스크립트의 경우 **nftables** 를 사용하는 것이 좋습니다.



중요

서로 다른 방화벽 서비스가 서로 영향을 미치지 않도록 하려면 RHEL 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

7.1.2. 영역

firewalld 는 사용자가 해당 네트워크 내에서 인터페이스와 트래픽을 배치하기로 결정한 신뢰 수준에 따라 네트워크를 다른 영역으로 분리하는 데 사용할 수 있습니다. 연결은 하나의 영역에만 포함될 수 있지만 영역은 많은 네트워크 연결에 사용할 수 있습니다.

NetworkManager 는 인터페이스 영역의 **firewalld** 를 알립니다. 다음을 사용하여 인터페이스에 영역을 할당할 수 있습니다.

- **NetworkManager**
- **firewall-config** 도구
- **firewall-cmd** 명령줄 도구
- RHEL 웹 콘솔

세 번째 방법은 적절한 **NetworkManager** 구성 파일만 편집할 수 있습니다. 웹 콘솔, **firewall-cmd 또는 firewall- config** 를 사용하여 인터페이스의 영역을 변경하는 경우 요청은 **NetworkManager** 로 전달되며**firewalld** 에서 처리되지 않습니다.

사전 정의된 영역은 **/usr/lib/firewalld/zones/** 디렉토리에 저장되며 사용 가능한 모든 네트워크 인터페이스에 즉시 적용할 수 있습니다. 이러한 파일은 수정된 후에만 **/etc/firewalld/zones/** 디렉토리에 복사됩니다. 사전 정의된 영역의 기본 설정은 다음과 같습니다.

블록

들어오는 네트워크 연결은 IPv4의 경우 **IPv4** 및 icmp6-adm-prohibited의 icmp-host-prohibited 메시지와 함께 거부됩니다 **.** 시스템 내에서 시작된 네트워크 연결만 가능합니다.

dmz

내부 네트워크에 대한 제한된 액세스 권한으로 공개적으로 액세스할 수 있는 비유형 영역에 있는 컴퓨터. 선택한 연결만 허용됩니다.

drop

들어오는 네트워크 패킷은 알림 없이 삭제됩니다. 나가는 네트워크 연결만 가능합니다.

external

특히 라우터의 경우 마스커레이딩이 활성화된 외부 네트워크에서 사용합니다. 네트워크의 다른 컴퓨터는 컴퓨터가 손상되지 않도록 신뢰하지 않습니다. 선택한 연결만 허용됩니다.

홈

주로 네트워크의 다른 컴퓨터를 신뢰하는 가정에서 사용할 수 있습니다. 선택한 연결만 허용됩니다.

internal

네트워크의 다른 컴퓨터를 대부분 신뢰하는 경우 내부 네트워크에서 사용할 수 있습니다. 선택한 연결만 허용됩니다.

public

네트워크의 다른 컴퓨터를 신뢰하지 않는 공공 부문의 경우. 선택한 연결만 허용됩니다.

trusted

모든 네트워크 연결이 허용됩니다.

work

네트워크의 다른 컴퓨터를 대부분 신뢰하는 직장에서 사용할 수 있습니다. 선택한 연결만 허용됩니다.

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

참고

네트워크 영역 이름은 쉽게 설명해야 하며 사용자가 합리적인 결정을 신속하게 내릴 수 있도록 해야 합니다. 보안 문제를 방지하려면 기본 영역 구성을 검토하고 요구 사항 및 위험 평가에 따라 불필요한 서비스를 비활성화합니다.

추가 리소스

- ***firewalld.zone(5)*** *도움말 페이지.*

7.1.3. 사전 정의된 서비스

서비스는 로컬 포트, 프로토콜, 소스 포트 및 대상의 목록일 뿐만 아니라 서비스가 활성화된 경우 자동으로 로드되는 방화벽 도우미 모듈 목록입니다. 서비스를 사용하면 사용자는 포트 열기, 프로토콜 정의, 패킷 전달 활성화 등의 여러 작업을 한 단계로 수행할 수 있으므로 사용자는 시간을 절약할 수 있습니다.

서비스 구성 옵션과 일반 파일 정보는 ***firewalld.service(5)*** *도움말 페이지에 설명되어 있습니다. 서비스는 다음과 같은 형식으로 이름이 지정된 개별 XML 구성 파일인* ***service-name.xml*** *을 통해 지정됩니다. 프로토콜 이름은* ***firewalld*** *의 서비스 또는 애플리케이션 이름보다 우선합니다.*

그래픽 firewall ***-config 도구, firewall-cmd 및 firewall- offline -cmd*** *를 사용하여 서비스를 추가하고 제거할 수 있습니다.*

또는 ***/etc/firewalld/services/*** *디렉터리에서 XML 파일을 편집할 수도 있습니다. 사용자가 서비스를 추가하거나 변경하지 않은 경우* ***/etc/firewalld/services/*** *에 해당 XML 파일이 없습니다. 서비스를 추가하거나 변경하려면* ***/usr/lib/firewalld/services/*** *디렉터리의 파일을 템플릿으로 사용할 수 있습니다.*

추가 리소스

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

7.1.4. firewalld 시작

절차

1. ***firewalld*** *를 시작하려면* ***root*** *로 다음 명령을 입력합니다.*

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

2. *시스템을 시작할 때* ***firewalld*** *가 자동으로 시작되도록 하려면* ***root*** *로 다음 명령을 입력합니다.*

```
# systemctl enable firewalld
```

7.1.5. firewalld 중지

절차

1. ***firewalld*** *를 중지하려면* ***root*** *로 다음 명령을 입력합니다.*

```
# systemctl stop firewalld
```

2. *시스템을 시작할 때* ***firewalld*** *가 자동으로 시작되지 않도록 하려면 다음을 수행합니다.*

```
# systemctl disable firewalld
```

3. firewalld **D-Bus** 인터페이스에 액세스하고 다른 서비스에 **firewalld** 가 필요한 경우에도 **firewalld** 를 시작하지 않도록하려면 다음을 수행하십시오.

```
# systemctl mask firewalld
```

7.1.6. 영구 firewalld 구성 확인

예를 들어 **firewalld** 구성 파일을 수동으로 편집한 후 관리자는 변경 사항이 올바른지 확인하려고 합니다. 이 섹션에서는 **firewalld** 서비스의 영구 구성을 확인하는 방법을 설명합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. **firewalld** 서비스의 영구 구성을 확인합니다.

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

영구 구성이 유효한 경우 명령은 **성공을** 반환합니다. 그렇지 않으면 명령에서 다음과 같은 추가 세부 정보가 있는 오류를 반환합니다.

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

7.2. FIREWALLD의 현재 상태 및 설정 보기

이 섹션에서는 **firewalld** 의 현재 상태, 허용된 서비스 및 현재 설정에 대한 정보를 다룹니다.

7.2.1. firewalld의 현재 상태 보기

방화벽 서비스 **firewalld** 는 기본적으로 시스템에 설치됩니다. **firewalld** CLI 인터페이스를 사용하여 서비스가 실행 중인지 확인합니다.

절차

1. 서비스 상태를 보려면 다음을 수행합니다.

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

2. 서비스 상태에 대한 자세한 내용은 **systemctl status** 하위 명령을 사용합니다.

```
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor pr
  Active: active (running) since Mon 2017-12-18 16:05:15 CET; 50min ago
    Docs: man:firewalld(1)
```

```
Main PID: 705 (firewalld)
  Tasks: 2 (limit: 4915)
 CGroup: /system.slice/firewalld.service
         └─705 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
```

7.2.2. GUI를 사용하여 허용된 서비스 보기

그래픽 **firewall-config** 도구를 사용하여 서비스 목록을 보려면 **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **방화벽** 을 입력한 다음 **Enter** 키를 누릅니다. **firewall-config** 도구가 나타납니다. 이제 **Services** (서비스) 탭에서 서비스 목록을 볼 수 있습니다.

명령줄을 사용하여 그래픽 방화벽 구성 도구를 시작할 수 있습니다.

사전 요구 사항

- **firewall-config** 패키지를 설치했습니다.

절차

- 명령줄을 사용하여 그래픽 방화벽 구성 도구를 시작하려면 다음을 수행합니다.

```
$ firewall-config
```

Firewall Configuration(방화벽 구성) 창이 열립니다. 이 명령은 일반 사용자로 실행할 수 있지만, 관리자 암호를 묻는 메시지가 표시되는 경우가 있습니다.

7.2.3. CLI를 사용하여 firewalld 설정 보기

CLI 클라이언트를 사용하면 현재 방화벽 설정을 다양한 볼 수 있습니다. list **-all** 옵션은 **firewalld** 설정에 대한 전체 개요를 표시합니다.

firewalld 는 영역을 사용하여 트래픽을 관리합니다. zone **옵션으로** 영역을 지정하지 않으면 활성 네트워크 인터페이스 및 연결에 할당된 기본 영역에서 명령이 효과적입니다.

절차

- 기본 영역에 대한 모든 관련 정보를 나열하려면 다음을 수행합니다.

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

- *설정을 표시할 영역을 지정하려면* ***--zone=zone-name 인수를 firewall- cmd --list-all*** *명령에 추가합니다. 예를 들면 다음과 같습니다.*

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

- *서비스 또는 포트와 같은 특정 정보에 대한 설정을 보려면 특정 옵션을 사용합니다. 명령 도움말 페이지를 사용하여* ***firewalld*** *도움말 페이지를 참조하거나 옵션 목록을 가져옵니다.*

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

- *현재 영역에서 허용되는 서비스를 보려면 다음을 수행합니다.*

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

참고

CLI 도구를 사용하여 특정 하위 파트의 설정을 나열하는 경우 해석하기 어려울 수 있습니다. 예를 들어 ***SSH*** *서비스를 허용하고* ***firewalld*** *는 서비스에 필요한 포트(22)를 엽니다. 나중에 허용된 서비스를 나열하는 경우 목록에* ***SSH*** *서비스가 표시되지만 열려 있는 포트를 나열하는 경우 아무것도 표시되지 않습니다. 따라서* ***--list-all*** *옵션을 사용하여 전체 정보를 받는 것이 좋습니다.*

7.3. FIREWALLD를 사용하여 네트워크 트래픽 제어

이 섹션에서는 ***firewalld*** *를 사용하여 네트워크 트래픽을 제어하는 방법에 대해 설명합니다.*

7.3.1. CLI를 사용하여 긴급한 경우 모든 트래픽 비활성화

시스템 공격과 같은 긴급 상황에서는 모든 네트워크 트래픽을 비활성화하고 공격자를 차단할 수 있습니다.

절차

1. *네트워킹 트래픽을 즉시 비활성화하려면 에서 패닉 모드를 전환합니다.*

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

중요

패닉 모드를 활성화하면 모든 네트워킹 트래픽이 중지됩니다. 따라서 시스템에 대한 물리적 액세스 권한이 있거나 직렬 콘솔을 사용하여 로그인하는 경우에만 사용해야 합니다.

2. *패닉 모드를 끄면 방화벽을 영구 설정으로 되돌립니다. 패닉 모드를 끄려면 다음을 입력합니다.*

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

검증

- *패닉 모드가 켜져 있는지 또는 꺼져 있는지 여부를 확인하려면 다음을 사용합니다.*

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

7.3.2. CLI를 사용하여 사전 정의된 서비스로 트래픽 제어

*트래픽을 제어하는 가장 간단한 방법은 사전 정의된 서비스를 **firewalld** 에 추가하는 것입니다. 그러면 필요한 모든 포트가 열리고 서비스 정의 파일에 따라 다른 설정을 수정합니다.*

절차

1. *서비스가 아직 허용되지 않은지 확인합니다.*

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

2. *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 ...
[trimmed for clarity]
```

3. *허용된 서비스에 서비스를 추가합니다.*

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

4. *새 설정을 영구적으로 설정합니다.*

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

7.3.3. GUI를 사용하여 사전 정의된 서비스로 트래픽 제어

다음 절차에서는 그래픽 사용자 인터페이스를 사용하여 사전 정의된 서비스로 네트워크 트래픽을 제어하는 방법을 설명합니다.

사전 요구 사항

- ***firewall-config** 패키지를 설치했습니다.*

절차

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

c. *신뢰할 수 있는 각 서비스 유형의 확인란을 선택하거나 선택한 영역에서 서비스를 차단하는 확인란을 선택 취소합니다.*

2. *서비스를 편집하려면 다음을 수행합니다.*

a. ***firewall-config** 도구를 시작합니다.*

b. ***Configuration** (구성) 이라는 레이블이 지정된 메뉴에서 **Permanent** (영구) 를 선택합니다. **Services** (서비스) 창의 아래쪽에 추가 아이콘 및 메뉴 버튼이 표시됩니다.*

c. *구성할 서비스를 선택합니다.*

***Ports** (포트), **Protocols** (프로토콜) 및 **Source Port(소스 포트)** 탭을 사용하면 선택한 서비스에 대한 포트, 프로토콜 및 소스 포트를 추가, 변경 및 제거할 수 있습니다. modules 탭은 **Netfilter** 도우미 모듈을 구성하는 데 사용됩니다. **Destination(대상)** 탭에서는 특정 대상 주소 및 인터넷 프로토콜(IP **4 또는 IPv 6) 으로 트래픽을 제한할 수 있습니다**.*

참고

***런타임 모드에서 서비스 설정을 변경할 수 없습니다**.*

7.3.4. 새 서비스 추가

*그래픽 firewall **-config 도구, firewall-cmd 및 firewall- offline -cmd** 를 사용하여 서비스를 추가하고 제거할 수 있습니다. 또는 **/etc/firewalld/services/** 에서 XML 파일을 편집할 수도 있습니다. 서비스가 추가 또는 변경되지 않은 경우 **/etc/firewalld/services/** 에 해당 XML 파일이 없습니다. 서비스를 추가하거나 변경하려면 **/usr/lib/firewalld/services/** 파일을 템플릿으로 사용할 수 있습니다.*

참고

*서비스 이름은 영숫자여야 하며, 추가로 **_** (마이즈코어) 및 **-** (대시) 만 포함할 수 있습니다.*

절차

*터미널에서 새 서비스를 추가하려면 **firewalld** 가 활성 상태가 아닌 경우 **firewall -cmd 또는 firewall-offline-cmd** 를 사용합니다.*

1. *다음 명령을 입력하여 새롭고 빈 서비스를 추가합니다.*

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

2. *로컬 파일을 사용하여 새 서비스를 추가하려면 다음 명령을 사용합니다.*

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

*추가 **--name=service-name** 옵션을 사용하여 서비스 이름을 변경할 수 있습니다.*

3. *서비스 설정이 변경되는 즉시 서비스의 업데이트된 사본이 **/etc/firewalld/services/** 에 배치됩니다.*
***root** 로 다음 명령을 입력하여 서비스를 수동으로 복사할 수 있습니다.*

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

***firewalld** 는 첫 번째 위치에 있는 **/usr/lib/firewalld/services** 에서 파일을 로드합니다. 파일이*

/etc/firewalld/services 에 배치되고 유효한 경우 파일이 /usr/lib/firewalld/services 에서 일치하는 파일을 재정의합니다. /usr/lib/firewalld/services 의 재정의된 파일은 /etc/firewalld/services 에서 일치하는 파일을 제거하거나 firewalld 가 서비스의 기본값을 로드하라는 요청을 받은 즉시 사용됩니다. 이는 영구 환경에만 적용됩니다. 런타임 환경에서도 이러한 폴백을 가져오려면 다시 로드해야 합니다.

7.3.5. GUI를 사용하여 포트 열기

방화벽을 통한 트래픽을 특정 포트에 대한 트래픽을 허용하려면 GUI에서 포트를 열 수 있습니다.

사전 요구 사항

- *firewall-config 패키지를 설치했습니다.*

절차

1. *firewall-config 도구를 시작하고 변경할 설정의 네트워크 영역을 선택합니다.*
2. *Ports(포트) 탭을 선택하고 오른쪽에 있는 Add(추가) 버튼을 클릭합니다. Port and Protocol(포트 및 프로토콜) 창이 열립니다.*
3. *허용할 포트의 포트 번호 또는 범위를 입력합니다.*
4. *목록에서 tcp 또는 udp 를 선택합니다.*

7.3.6. GUI를 사용하여 프로토콜로 트래픽 제어

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

사전 요구 사항

- *firewall-config 패키지를 설치했습니다.*

절차

1. *firewall-config 도구를 시작하고 변경할 설정의 네트워크 영역을 선택합니다.*
2. *Protocols(프로토콜) 탭을 선택하고 오른쪽에 있는 Add(추가) 버튼을 클릭합니다. Protocol(프로토콜) 창이 열립니다.*
3. *목록에서 프로토콜을 선택하거나 Other Protocol(기타 프로토콜) 확인란을 선택하고 필드에 프로토콜을 입력합니다.*

7.3.7. GUI를 사용하여 소스 포트 열기

특정 포트에서 방화벽을 통한 트래픽을 허용하려면 GUI를 사용할 수 있습니다.

사전 요구 사항

- *firewall-config 패키지를 설치했습니다.*

절차

1. *firewall-config 도구를 시작하고 변경할 설정의 네트워크 영역을 선택합니다.*

2. ***Source Port(소스 포트)*** *탭을 선택하고 오른쪽에 있는* ***Add*** *(추가) 단추를 클릭합니다.* ***Source Port(소스 포트)*** *창이 열립니다.*

3. *허용할 포트의 포트 번호 또는 범위를 입력합니다. 목록에서* ***tcp*** *또는* ***udp*** *를 선택합니다.*

7.4. CLI를 사용하여 포트 제어

포트는 운영 체제에서 네트워크 트래픽을 수신 및 구분하고 이에 따라 시스템 서비스로 전달할 수 있는 논리적 장치입니다. 일반적으로 포트에서 수신 대기하는 데몬으로 표시됩니다. 즉, 이 포트로 들어오는 모든 트래픽을 기다립니다.

일반적으로 시스템 서비스는 예약된 표준 포트에서 수신 대기합니다. 예를 들어 ***httpd*** *데몬은 포트 80에서 수신 대기합니다. 그러나 기본적으로 시스템 관리자는 보안 또는 기타 이유로 다른 포트에서 수신 대기하도록 데몬을 구성합니다.*

7.4.1. 포트 열기

열려 있는 포트를 통해 시스템은 보안 위험을 나타내는 외부에서 액세스할 수 있습니다. 일반적으로 포트를 닫고 특정 서비스에 필요한 경우에만 포트를 엽니다.

절차

현재 영역에서 열려 있는 포트 목록을 가져오려면 다음을 수행합니다.

1. *허용되는 모든 포트를 나열합니다.*

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

2. *포트를 허용된 포트에 추가하여 들어오는 트래픽에 대해 엽니다.*

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

포트 유형은 ***tcp,udp,sctp*** *또는* ***dccp*** *입니다. 유형은 네트워크 통신 유형과 일치해야 합니다.*

3. *새 설정을 영구적으로 설정합니다.*

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

포트 유형은 ***tcp,udp,sctp*** *또는* ***dccp*** *입니다. 유형은 네트워크 통신 유형과 일치해야 합니다.*

7.4.2. 포트 닫기

열려 있는 포트가 더 이상 필요하지 않은 경우 ***firewalld*** *에서 해당 포트를 닫습니다. 포트를 열린 상태로 유지하면 보안 위험이 있으므로 사용하지 않는 즉시 모든 불필요한 포트를 종료하는 것이 좋습니다.*

절차

포트를 종료하려면 허용된 포트 목록에서 제거합니다.

1. *허용되는 모든 포트를 나열합니다.*

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

주의

*이 명령은 포트로 열린 포트 목록만 제공합니다. 서비스로 열린 열려 있는 포트는 볼 수 없습니다. 따라서 --list- **ports 대신 --list- all** 옵션을 사용해야 합니다.*

2. *포트를 허용된 포트에서 제거하여 들어오는 트래픽에 대해 종료합니다.*

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

3. *새 설정을 영구적으로 설정합니다.*

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

7.5. 시스템 역할을 사용하여 포트 구성

*RHEL(Red Hat Enterprise Linux) **firewalld** System Role을 사용하여 들어오는 트래픽에 대해 로컬 방화벽에서 포트를 열거나 닫고 재부팅 시 새 구성을 유지할 수 있습니다. 이 예제에서는 **HTTPS** 서비스에 대한 들어오는 트래픽을 허용하도록 기본 영역을 구성하는 방법을 설명합니다.*

Ansible 제어 노드에서 다음 절차를 실행합니다.

사전 요구 사항

- ***firewalld** 시스템 역할을 사용하여 구성하려는 하나 이상의 관리형 노드에 대한 액세스 및 권한.*
- *Red Hat Ansible Engine이 다른 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.*
- ***ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.*
- *플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리형 노드에 적절한 **sudo** 권한이 있습니다.*
- *호스트는 NetworkManager를 사용하여 네트워크를 구성합니다.*

절차

1. *플레이북에서 지침을 실행하려는 호스트가 아직 의도하지 않은 경우 이 호스트의 IP 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.*

```
node.example.com
```

2. *다음 콘텐츠를 사용하여 **~/adding-and-removing-ports.yml** 플레이북을 생성합니다.*

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

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

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

permanent: true 옵션을 사용하면 재부팅 시 새 설정이 유지됩니다.

3. 플레이북을 실행합니다.

- **root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

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

- 관리 호스트에 사용자로 연결하려면 다음을 입력합니다.

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

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지가 표시됩니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

검증

1. 관리형 노드에 연결합니다.

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

2. **HTTPS** 서비스와 연결된 **443/tcp** 포트가 열려 있는지 확인합니다.

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

추가 리소스

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

7.6. FIREWALLD 영역 작업

영역은 들어오는 트래픽을 보다 투명하게 관리하는 개념을 나타냅니다. 영역은 네트워킹 인터페이스에 연결되거나 다양한 소스 주소가 할당됩니다. 각 영역에 대해 개별적으로 방화벽 규칙을 관리하므로 복잡한 방화벽 설정을 정의하고 트래픽에 적용할 수 있습니다.

7.6.1. 영역 나열

다음 절차에서는 명령줄을 사용하여 영역을 나열하는 방법을 설명합니다.

절차

1. 시스템에서 사용할 수 있는 영역을 보려면 다음을 수행합니다.

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

firewall-cmd --get-zones 명령은 시스템에서 사용할 수 있는 모든 영역을 표시하지만 특정 영역에 대한 세부 정보는 표시하지 않습니다.

2. 모든 영역에 대한 자세한 정보를 보려면 다음을 수행합니다.

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

3. 특정 영역에 대한 자세한 정보를 보려면 다음을 수행합니다.

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

7.6.2. 특정 영역에 대한 firewalld 설정 수정

cli를 사용하여 사전 정의된 서비스로 트래픽 제어 및 *cli를 사용하여 포트 제어* 는 현재 작업 영역의 범위에서 서비스를 추가하거나 포트를 수정하는 방법을 설명합니다. 때로는 다른 영역에 규칙을 설정해야 합니다.

절차

- 다른 영역에서 작업하려면 **--zone=zone-name** 옵션을 사용합니다. 예를 들어 영역 public 에서 **SSH** 서비스를 허용하려면 다음을 수행합니다.

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

7.6.3. 기본 영역 변경

시스템 관리자는 구성 파일의 네트워킹 인터페이스에 영역을 할당합니다. 인터페이스가 특정 영역에 할당되지 않은 경우 기본 영역에 할당됩니다. **firewalld** 서비스를 다시 시작하면 **firewalld** 가 기본 영역에 대한 설정을 로드하여 활성화합니다.

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 절차에 따라 설정은 **--permanent** 옵션이 없어도 영구적인 설정입니다.*

7.6.4. 영역에 네트워크 인터페이스 할당

사용 중인 인터페이스의 영역을 변경하여 다양한 영역에 대해 다양한 규칙 집합을 정의한 다음 설정을 빠르게 변경할 수 있습니다. 여러 인터페이스를 사용하면 각 인터페이스를 통해 들어오는 트래픽을 구분하도록 특정 영역을 설정할 수 있습니다.

절차

특정 인터페이스에 영역을 할당하려면 다음을 수행합니다.

1. *활성 영역과 여기에 할당된 인터페이스를 나열합니다.*

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

2. *인터페이스를 다른 영역에 할당합니다.*

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

7.6.5. nmcli를 사용하여 연결에 영역 할당

*다음 절차에서는 **nmcli** 유틸리티를 사용하여 **NetworkManager** 연결에 **firewalld** 영역을 추가하는 방법을 설명합니다.*

절차

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

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

2. *연결을 활성화합니다.*

```
# nmcli connection up profile
```

7.6.6. ifcfg 파일에서 네트워크 연결에 수동으로 영역 할당

NetworkManager 에서 연결을 관리하는 경우 사용하는 영역을 알고 있어야 합니다. 모든 네트워크 연결에 대해 이동식 장치가 있는 컴퓨터의 위치에 따라 다양한 방화벽 설정의 유연성을 제공하는 영역을 지정할 수 있습니다. 따라서 회사 또는 홈과 같은 다양한 위치에 대해 영역 및 설정을 지정할 수 있습니다.

절차

- *연결 영역을 설정하려면 **/etc/sysconfig/network-scripts/ifcfg-connection_name** 파일을 편집하고 이 연결에 영역을 할당하는 행을 추가합니다.*

```
ZONE=zone_name
```

7.6.7. 새 영역 생성

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

절차

1. *새 영역을 생성합니다.*

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

2. *새 영역이 영구 설정에 추가되었는지 확인합니다.*

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

3. *새 설정을 영구적으로 설정합니다.*

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

7.6.8. 영역 구성 파일

영역은 영역 구성 파일을 사용하여 만들 수도 있습니다. 이 방법은 새 영역을 만들어야 할 때 유용할 수 있지만 다른 영역의 설정을 재사용하고 약간만 변경하려고 합니다.

firewalld *영역 구성 파일에는 영역에 대한 정보가 포함되어 있습니다. XML 파일 형식의 영역 설명, 서비스, 포트, 프로토콜, icmp-blocks, masquerade, forward-ports 및 리치 언어 규칙입니다. 파일 이름은 현재* ***zone-name 의 길이를 17자로 제한하는 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****) 및 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>
```

해당 영역의 설정을 변경하려면 섹션을 추가하거나 제거하여 포트, 서비스 등을 추가합니다.

추가 리소스

- ***firewalld.zone*** *매뉴얼 페이지*

7.6.9. 영역 대상을 사용하여 들어오는 트래픽에 대한 기본 동작 설정

모든 영역에 대해 추가로 지정되지 않은 들어오는 트래픽을 처리하는 기본 동작을 설정할 수 있습니다. 이러한 동작은 영역의 대상을 설정하여 정의됩니다. 네 가지 옵션이 있습니다.

- ***수락****: 특정 규칙에 의해 허용되지 않는 패킷을 제외하고 들어오는 모든 패킷을 허용합니다.*
- ***거부****: 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.*
- ***default***: ***REJECT*** *와 유사한 동작이지만 특정 시나리오에서 특별한 의미가 있습니다. 자세한 내용은* ***firewall-cmd(1)*** *도움말 페이지의* ***Adapt 및 Query Zones 및 Policies*** *섹션을 참조하십시오.*

절차

영역의 대상을 설정하려면 다음을 수행합니다.

1. *특정 영역에 대한 정보를 나열하여 기본 대상을 확인합니다.*

```
# 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

- ***firewall-cmd(1)*** *도움말 페이지*

7.7. 소스에 따라 영역을 사용하여 들어오는 트래픽 관리

영역을 사용하여 소스에 따라 들어오는 트래픽을 관리할 수 있습니다. 이를 통해 들어오는 트래픽을 정렬하고 다른 영역을 통해 라우팅하여 해당 트래픽으로 연결할 수 있는 서비스를 허용하거나 허용하지 않도록 할 수 있습니다.

소스를 영역에 추가하면 영역이 활성화되고 해당 소스에서 들어오는 트래픽이 이를 통해 전송됩니다. 지정된 소스의 트래픽에 적절하게 적용되는 각 영역에 대해 다른 설정을 지정할 수 있습니다. 하나의 네트워크 인터페이스만 있는 경우에도 더 많은 영역을 사용할 수 있습니다.

7.7.1. 소스 추가

들어오는 트래픽을 특정 영역으로 라우팅하려면 소스를 해당 영역에 추가합니다. 소스는 IP 주소이거나 클래스 없는 도메인 간 라우팅(CIDR) 표기법의 IP 마스크일 수 있습니다.

참고

중복되는 네트워크 범위를 사용하여 여러 영역을 추가하는 경우 영역 이름으로 영숫자로 정렬되며 첫 번째 영역만 고려됩니다.

- *현재 영역에 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 ***신뢰할 수 있는*** *영역의 192.168.2.15 에서 들어오는 모든 트래픽을 허용합니다.*

절차

1. *사용 가능한 모든 영역을 나열합니다.*

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

2. *영구 모드에서 소스 IP를 신뢰할 수 있는 영역에 추가합니다.*

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

3. *새 설정을 영구적으로 설정합니다.*

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

7.7.2. 소스 제거

영역에서 소스를 제거하면 소스에서 들어오는 트래픽이 줄어듭니다.

절차

1. *필수 영역에 허용된 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 설정합니다.*

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

7.7.3. 소스 포트 추가

원본 포트에 따라 트래픽을 정렬할 수 있도록 하려면 ***--add-source-port*** *옵션을 사용하여 소스 포트를 지정합니다. 이 옵션을* ***--add-source*** *옵션과 결합하여 트래픽을 특정 IP 주소 또는 IP 범위로 제한할 수도 있습니다.*

절차

- *소스 포트를 추가하려면 다음을 수행합니다.*

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

7.7.4. 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>
```

7.7.5. 특정 도메인에만 서비스를 허용하려면 영역 및 소스를 사용합니다.

특정 네트워크의 트래픽이 시스템에서 서비스를 사용하도록 허용하려면 영역 및 소스를 사용합니다. 다음 절차에서는 다른 트래픽이 차단되는 동안 **192.0.2.0/24** 네트워크에서 HTTP 트래픽만 허용합니다.

주의

*이 시나리오를 구성할 때 **기본** 타겟이 있는 영역을 사용합니다. 타겟이 **ACCEPT** 로 설정된 영역을 사용하면 **192.0.2.0/24** 의 트래픽에 대해 모든 네트워크 연결이 수락되기 때문에 보안 위험이 있습니다.*

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 시작된 트래픽을 영역을 통해 라우팅합니다.

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

3. **http** 서비스를 **내부** 영역에 추가합니다.

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

4. 새 설정을 영구적으로 설정합니다.

```
# 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

- ***firewalld.zones(5)*** *도움말 페이지*

7.8. 영역 간에 전달된 트래픽 필터링

정책 오브젝트를 사용하면 사용자가 정책에 유사한 권한이 필요한 다양한 ID를 그룹화할 수 있습니다. 트래픽 방향에 따라 정책을 적용할 수 있습니다.

정책 오브젝트 기능은 firewalld에서 정방향 및 출력 필터링을 제공합니다. 다음은 로컬 호스트 VM에 대한 액세스를 통해 호스트를 연결할 수 있도록 다양한 영역 간 트래픽을 필터링하는 데 firewalld를 사용하는 방법을 설명합니다.

7.8.1. 정책 오브젝트와 영역 간의 관계

Policy 개체를 사용하면 서비스, 포트 및 리치 규칙과 같은 firewalld의 기본 규칙을 정책에 연결할 수 있습니다. 상태 저장 및 단방향 방식으로 영역 간에 통과하는 트래픽에 정책 오브젝트를 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* ***ANY*** *는 수신 및 송신 영역 목록에 사용되는 심볼릭 영역입니다.*

- ***HOST*** *심볼릭 영역을 사용하면 에서 시작되는 트래픽에 대한 정책이 허용되거나 firewalld를 실행하는 호스트의 대상이 있습니다.*
- ***ANY*** *심볼릭 영역은 현재 및 향후 모든 영역에 정책을 적용합니다.* ***ANY*** *심볼릭 영역은 모든 영역의 와일드카드 역할을 합니다.*

7.8.2. 우선순위를 사용하여 정책 정렬

여러 정책을 동일한 트래픽 집합에 적용할 수 있으므로 적용할 수 있는 정책에 대한 우선 순위 순서를 생성하는 데 우선순위를 사용해야 합니다.

정책을 정렬할 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 이전에 실행됩니다. 우선순위가 높은 값은 더 낮은 값보다 우선합니다.

다음 규칙은 정책 우선순위에 적용됩니다.

- *부정 우선 순위가 있는 정책은 영역의 규칙 앞에 적용됩니다.*
- *긍정적인 우선 순위가 있는 정책은 영역의 규칙 뒤에 적용됩니다.*
- *우선 순위 0은 예약되어 있으므로 사용할 수 없습니다.*

7.8.3. 정책 오브젝트를 사용하여 로컬 호스트 컨테이너와 호스트에 물리적으로 연결된 네트워크 간의 트래픽 필터링

정책 오브젝트 기능을 사용하면 컨테이너 및 가상 시스템 트래픽을 필터링할 수 있습니다.

절차

1. *새 정책 만들기.*

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

2. *모든 트래픽을 차단합니다.*

```
# firewall-cmd --permanent --policy podmanToHost --set-target REJECT

# firewall-cmd --permanent --policy podmanToHost --add-service dhcp

# firewall-cmd --permanent --policy podmanToHost --add-service dns
```

참고

기본적으로 호스트에 대한 모든 트래픽을 차단한 다음 호스트에 필요한 서비스를 선택적으로 여는 것이 좋습니다.

3. *정책과 함께 사용할 수신 영역을 정의합니다.*

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

4. *정책과 함께 사용할 송신 영역을 정의합니다.*

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

검증

- *정책에 대한 정보를 확인합니다.*

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

7.8.4. 정책 오브젝트의 기본 대상 설정

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

- ***ACCEPT*** *- 패킷을 수락*
- ***DROP*** *- 원하지 않는 패킷 삭제*
- ***REJECT*** *- ICMP 응답으로 원하지 않는 패킷 거부*
- ***CONTINUE (기본값)*** *- 패킷은 다음 정책 및 영역의 규칙에 따라 적용됩니다.*

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

검증

- *정책에 대한 정보 확인*

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

7.9. FIREWALLD를 사용하여 NAT 구성

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

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

7.9.1. 다양한 NAT 유형: 마스커레이드, 소스 NAT, 대상 NAT, 리디렉션

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 공급자는 **10.0.0.0/8** 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하며 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 해당 범위에서 패킷의 소스 IP 주소를 공용 IP 주소로 매핑합니다.
마스커레이딩과 SNAT 모두 매우 유사합니다. 차이점은 다음과 같습니다.

- 마스커레이딩은 나가는 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 나가는 인터페이스에서 동적 IP 주소를 사용하는 경우 마스커레이딩을 사용합니다.
- SNAT는 패킷의 소스 IP 주소를 지정된 IP로 설정하고 나가는 인터페이스의 IP를 동적으로 조회하지 않습니다. 따라서 SNAT는 마스커레이딩보다 빠릅니다. 나가는 인터페이스에서 고정 IP 주소를 사용하는 경우 SNAT를 사용합니다.

대상 NAT(DNAT)

이 NAT 유형을 사용하여 들어오는 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 들어오는 트래픽을 이 서버로 리디렉션할 수 있습니다.

리디렉션

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

7.9.2. IP 주소 마스커레이딩 구성

다음 절차에서는 시스템에서 IP 마스커레이딩을 활성화하는 방법을 설명합니다. IP 마스커레이딩은 인터넷에 액세스할 때 게이트웨이 뒤의 개별 시스템을 숨깁니다.

절차

1. IP 마스커레이드가 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. 그렇지 않으면 종료 상태 **1** 로 **no** 를 출력합니다. **zone(영역)** 을 생략하면 기본 영역이 사용됩니다.

2. IP 마스커레이딩을 사용하려면 **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.

7.10. 포트 전달

이 메서드를 사용하여 포트를 리디렉션하는 것은 IPv4 기반 트래픽에서만 작동합니다. IPv6 리디렉션 설정의 경우 리치 규칙을 사용해야 합니다.

외부 시스템으로 리디렉션하려면 마스커레이딩을 활성화해야 합니다. 자세한 내용은 IP 주소 마스커레이딩 구성을 참조하십시오.

참고

로컬 전달을 구성한 호스트에서 리디렉션된 포트를 통해 서비스에 액세스할 수 없습니다.

7.10.1. 리디렉션할 포트 추가

firewalld 를 사용하면 시스템의 특정 포트에 도달하는 들어오는 트래픽이 선택한 다른 내부 포트 또는 다른 시스템의 외부 포트에 전달되도록 포트 리디렉션을 설정할 수 있습니다.

사전 요구 사항

- 한 포트에서 다른 포트 또는 다른 주소로 트래픽을 리디렉션하기 전에 패킷이 도착하는 포트, 사용되는 프로토콜, 리디렉션하려는 세 가지 사항을 알아야 합니다.

절차

1. 포트를 다른 포트로 리디렉션하려면 다음을 수행합니다.

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

2. 다른 IP 주소에서 다른 포트로 포트를 리디렉션하려면 다음을 수행합니다.

   a. 전달할 포트를 추가합니다.

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

   b. 마스커레이드를 활성화합니다.

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

7.10.2. 동일한 머신에서 TCP 포트 80을 포트 88으로 리디렉션

단계에 따라 TCP 포트 80을 포트 88으로 리디렉션합니다.

절차

1. 포트 80을 TCP 트래픽의 포트 88로 리디렉션합니다.

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

2. 새 설정을 영구적으로 설정합니다.

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

3. 포트가 리디렉션되는지 확인합니다.

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

7.10.3. 리디렉션된 포트 제거

다음 절차에서는 리디렉션된 포트를 제거하는 방법을 설명합니다.

절차

1. 리디렉션된 포트를 제거하려면 다음을 수행합니다.

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

2. 다른 주소로 리디렉션되는 전달된 포트를 제거하려면 다음을 수행합니다.

   a. 전달된 포트를 제거합니다.

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

   b. 마스커레이드를 비활성화합니다.

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

7.10.4. 동일한 머신의 포트 88으로 전달된 TCP 포트 80 제거

다음 절차에서는 포트 리디렉션을 제거하는 방법을 설명합니다.

절차

1. 리디렉션된 포트를 나열합니다.

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

2. 방화벽에서 리디렉션된 포트를 제거합니다.

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

3. 새 설정을 영구적으로 설정합니다.

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

7.11. ICMP 요청 관리

ICMP(Internet Control Message Protocol)는 다양한 네트워크 장치에서 오류 메시지 및 연결 문제를 나타내는 운영 정보를 전송하는 데 사용하는 지원 프로토콜입니다(예: 요청된 서비스를 사용할 수 없음). **ICMP**는 시스템 간에 데이터를 교환하는 데 사용되지 않으므로 TCP 및 UDP와 같은 전송 프로토콜과 다릅니다.

안타깝게도 **ICMP** 메시지, 특히 **에코 요청 및 에코 를 사용하여 네트워크에 대한 정보를 공개하고** 다양한 유형의 사기 활동에 이러한 정보를 오용할 수 있습니다. 따라서 **firewalld** 를 사용하면 **ICMP** 요청을 차단하여 네트워크 정보를 보호할 수 있습니다.

7.11.1. ICMP 요청 나열 및 차단

ICMP 요청 나열

ICMP 요청은 **/usr/lib/firewalld/icmptypes/** 디렉터리에 있는 개별 XML 파일에 설명됩니다. 이러한 파일을 읽고 요청에 대한 설명을 확인할 수 있습니다. **firewall-cmd** 명령은 **ICMP** 요청 조작을 제어합니다.

- 사용 가능한 모든 **ICMP** 유형을 나열하려면 다음을 수행합니다.

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

- **ICMP** 요청은 IPv4, IPv6 또는 두 프로토콜 모두에서 사용할 수 있습니다. **ICMP** 요청이 사용된 프로토콜을 보려면 다음을 수행합니다.

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

- **ICMP** 요청 상태는 요청이 현재 차단되었거나 그렇지 않은 경우 **no** 인 경우 **yes** 를 표시합니다. **ICMP** 요청이 현재 차단되었는지 확인하려면 다음을 실행합니다.

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

ICMP 요청 차단 또는 차단 해제

서버에서 **ICMP** 요청을 차단하면 일반적으로 수행하는 정보를 제공하지 않습니다. 그러나 이는 정보가 전혀 제공되지 않음을 의미하지는 않습니다. 클라이언트는 특정 **ICMP** 요청이 차단되고 있음을 수신합니다(거부됨). **ICMP** 요청을 차단하면 특히 IPv6 트래픽에서 통신 문제가 발생할 수 있으므로 주의해서 고려해야 합니다.

- **ICMP** 요청이 현재 차단되었는지 확인하려면 다음을 실행합니다.

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

- **ICMP** 요청을 차단하려면 다음을 수행합니다.

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

- **ICMP** 요청에 대한 블록을 제거하려면 다음을 수행합니다.

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

정보를 제공하지 않고 ICMP 요청 차단

일반적으로 **ICMP** 요청을 차단하면 클라이언트가 차단하고 있음을 알 수 있습니다. 따라서 실시간 IP 주소를 스니핑하는 잠재적인 공격자는 IP 주소가 온라인 상태임을 계속 확인할 수 있습니다. 이 정보를 완전히 숨기려면 **ICMP** 요청을 모두 삭제해야 합니다.

- 모든 **ICMP** 요청을 차단하고 삭제하려면 다음을 수행합니다.
- 영역의 대상을 **DROP** 으로 설정합니다.

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

이제 명시적으로 허용된 트래픽을 제외하고 **ICMP** 요청을 포함한 모든 트래픽이 삭제됩니다.

특정 **ICMP 요청을 차단 및 삭제하고 다른 ICMP** 요청을 허용하려면 다음을 수행합니다.

1. 영역의 대상을 **DROP** 으로 설정합니다.

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

2. ICMP 블록 인버전을 추가하여 모든 **ICMP** 요청을 한 번에 차단합니다.

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

3. 허용하려는 **ICMP 요청에 대한 ICMP** 블록을 추가합니다.

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

4. 새 설정을 영구적으로 설정합니다.

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

블록 inversion 은 **ICMP** 요청의 설정을 변환하므로 이전에 차단되지 않은 모든 요청이 **DROP** 으로 변경되므로 영역의 대상이 차단되므로 차단됩니다. 차단된 요청은 차단되지 않습니다. 즉, 요청 차단을 해제하려면 차단 명령을 사용해야 합니다.

블록 인버전을 완전히 허용 설정으로 되돌리려면 다음을 수행합니다.

1. 영역의 대상을 **기본값** 또는 **ACCEPT** 로 설정합니다.

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

2. **ICMP** 요청에 대해 추가된 모든 블록을 제거합니다.

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

3. **ICMP** 블록 인버전을 제거합니다.

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

4. 새 설정을 영구적으로 설정합니다.

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

7.11.2. GUI를 사용하여 ICMP 필터 구성

- **ICMP** 필터를 활성화하거나 비활성화하려면 **firewall-config** 도구를 시작하고 메시지가 필터링될 네트워크 영역을 선택합니다. **ICMP Filter(ICMP 필터)** 탭을 선택하고 필터링할 각 **ICMP** 메시지 유형의 확인란을 선택합니다. 확인란을 지워 필터를 비활성화합니다. 이 설정은 방향에 따라 설정되며 기본값은 모든 것을 허용합니다.

- **ICMP 필터를 전환하려면 오른쪽에 있는 Invert Filter** (검색 필터) 확인란을 클릭합니다. 표시된 **ICMP** 유형만 허용되고 다른 모든 유형이 거부됩니다. DROP 대상을 사용하는 영역에서는 해당 대상이 삭제됩니다.

7.12. FIREWALLD를 사용하여 IP 세트 설정 및 제어

firewalld 에서 지원하는 IP 세트 유형 목록을 보려면 root로 다음 명령을 입력합니다.

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

7.12.1. CLI를 사용하여 IP 세트 옵션 구성

IP 세트는 **firewalld** 영역에서 소스로, 리치 규칙의 소스로도 사용할 수 있습니다. Red Hat Enterprise Linux 에서 기본 방법은 직접 규칙에서 **firewalld** 로 생성된 IP 세트를 사용하는 것입니다.

- 영구 환경에서 **firewalld** 로 알려진 IP 세트를 나열하려면 다음 명령을 **root** 로 사용하십시오.

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

- 새 IP 세트를 추가하려면 영구 환경을 **root** 로 사용하여 다음 명령을 사용하십시오.

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

이전 명령은 이름 test 및 **IPv4** 의 **hash:net 유형으로** 새 IP 집합을 생성합니다. **IPv6** 에 사용할 IP 세트를 만들려면 **--option=family=inet6** 옵션을 추가합니다. 런타임 환경에서 새 설정을 적용하려면 **firewalld** 를 다시 로드합니다.

- 다음 명령을 사용하여 **root** 로 새 IP 세트를 나열합니다.

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

- IP 세트에 대한 자세한 정보를 얻으려면 **root** 로 다음 명령을 사용하십시오.

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

```
options:
entries:
```

현재 IP 세트에는 항목이 없습니다.

- 테스트 IP 세트에 항목을 추가하려면 **root** 로 다음 명령을 사용하십시오.

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

이전 명령은 IP 주소 192.168.0.1 을 IP 집합에 추가합니다.

- IP 세트의 현재 항목 목록을 가져오려면 다음 명령을 **root** 로 사용하십시오.

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

- IP 주소 목록이 포함된 파일을 생성합니다. 예를 들면 다음과 같습니다.

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

IP 집합의 IP 주소 목록이 있는 파일에는 행당 항목이 포함되어야 합니다. 해시, 세미콜론 또는 빈 행으로 시작하는 행은 무시됩니다.

- iplist.txt 파일의 주소를 추가하려면 **root** 로 다음 명령을 사용하십시오.

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

- IP 세트의 확장 항목 목록을 보려면 **root** 로 다음 명령을 사용하십시오.

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

- IP 세트에서 주소를 제거하고 업데이트된 항목 목록을 확인하려면 다음 명령을 **루트로** 사용합니다.

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

- IP 세트를 영역에 소스로 추가하여 영역으로 설정된 IP 세트에 나열된 주소에서 들어오는 모든 트래픽을 처리할 수 있습니다. 예를 들어 테스트 IP 세트를 드롭 영역에 소스로 추가하여 테스트 IP 세트에 나열된 모든 항목에서 들어오는 모든 패킷을 삭제하려면 **root** 로 다음 명령을 사용합니다.

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

소스의 **ipset:** 접두사는 소스 가 IP 주소 또는 주소 범위가 아닌 IP 세트임을 표시합니다.

IP 집합의 생성 및 제거만 영구 환경으로 제한되며, **--permanent** 옵션을 사용하지 않고 런타임 환경에서 다른 모든 IP 집합 옵션을 사용할 수도 있습니다.

주의

Red Hat은 **firewalld** 를 통해 관리되지 않는 IP 세트를 사용하지 않는 것이 좋습니다. 이러한 IP 세트를 사용하려면 세트를 참조하려면 영구적인 직접 규칙이 필요하며 이러한 IP 세트를 생성하려면 사용자 지정 서비스를 추가해야 합니다. 이 서비스는 **firewalld** 를 시작하기 전에 시작해야 합니다. 그렇지 않으면 **firewalld** 가 이러한 세트를 사용하여 직접 규칙을 추가할 수 없습니다. **/etc/firewalld/direct.xml** 파일을 사용하여 영구적인 직접 규칙을 추가할 수 있습니다.

7.13. 리치 규칙 우선순위 지정

기본적으로 리치 규칙은 규칙 동작을 기반으로 구성됩니다. 예를 들어 **거부** 규칙은 **허용** 규칙보다 우선합니다. 리치 규칙의 **priority** 매개 변수는 관리자가 리치 규칙과 실행 순서를 세부적으로 제어할 수 있습니다.

7.13.1. 우선순위 매개변수가 규칙을 다른 체인으로 구성하는 방법

리치 규칙에서 **우선순위** 매개변수를 **-32768** 및 **32767** 사이의 숫자로 설정할 수 있으며 더 낮은 값이 우선합니다.

firewalld 서비스는 우선 순위 값을 기반으로 다른 체인으로 규칙을 구성합니다.

- 0보다 낮은 우선 순위: 규칙이 **_pre** 접미사가 있는 체인으로 리디렉션됩니다.
- 우선순위 0: 규칙이 **_post** 접미사가 있는 체인으로 리디렉션됩니다.
- 우선 순위는 0: 작업을 기반으로 하는 규칙은 **_log, _ deny 또는 _ allow** 를 사용하여 체인으로 리디렉션됩니다.

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

7.13.2. 리치 규칙의 우선 순위 설정

절차에서는 **priority** 매개 변수를 사용하여 다른 규칙에서 허용되거나 거부되지 않는 모든 트래픽을 기록하는 리치 규칙을 생성하는 방법의 예를 설명합니다. 이 규칙을 사용하여 예기치 않은 트래픽에 플래그를 지정할 수 있습니다.

절차

1. 우선 순위가 매우 낮은 리치 규칙을 추가하여 다른 규칙과 일치하지 않는 모든 트래픽을 기록합니다.

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

명령은 로그 항목 수를 분당 **5** 개로 추가로 제한합니다.

2. 선택적으로 이전 단계에서 명령을 생성한 **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
  }
}
```

7.14. 방화벽 잠금 구성

로컬 애플리케이션 또는 서비스는 방화벽 구성이 **root** (예: **libvirt**)로 실행되는 경우 변경할 수 있습니다. 이 기능을 사용하면 관리자가 방화벽 구성을 잠글 수 있으므로 애플리케이션이 없거나 잠금 허용 목록에 추가된 애플리케이션만 방화벽 변경 사항을 요청할 수 있습니다. 기본적으로 잠금 설정이 비활성화됩니다. 활성화된 경우 로컬 애플리케이션 또는 서비스에서 방화벽에 대한 원하지 않는 구성 변경 사항이 없는지 확인할 수 있습니다.

7.14.1. CLI를 사용하여 잠금 구성

다음 절차에서는 명령줄을 사용하여 잠금을 활성화하거나 비활성화하는 방법을 설명합니다.

- 잠금이 활성화되었는지 여부를 쿼리하려면 **root** 로 다음 명령을 사용하십시오.

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

잠금이 활성화된 경우 명령이 종료 상태 **0** 으로 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 로 **no** 를 출력합니다.

- 잠금을 활성화하려면 **root** 로 다음 명령을 입력합니다.

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

- 잠금을 비활성화하려면 **root** 로 다음 명령을 사용하십시오.

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

7.14.2. CLI를 사용하여 잠금 허용 목록 옵션 구성

잠금 허용 목록에는 명령, 보안 컨텍스트, 사용자 및 사용자 ID가 포함될 수 있습니다. 허용 목록의 명령 항목이 별표 "*"로 끝나면 해당 명령으로 시작하는 모든 명령줄이 일치합니다. "*"가 없으면 인수를 포함한 절대 명령이 일치해야 합니다.

- 컨텍스트는 실행 중인 애플리케이션 또는 서비스의 보안(SELinux) 컨텍스트입니다. 실행 중인 애플리케이션의 컨텍스트를 가져오려면 다음 명령을 사용합니다.

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

이 명령은 실행 중인 모든 애플리케이션을 반환합니다. 출력을 **grep** 도구를 통해 파이프하여 관심 있는 애플리케이션을 가져옵니다. 예를 들면 다음과 같습니다.

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

- 허용 목록에 있는 모든 명령행을 나열하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에 명령을 추가하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에서 명령을 제거하려면 **root** 로 다음 명령을 입력합니다.

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

- 명령이 허용 목록에 있는지 쿼리하려면 **root** 로 다음 명령을 입력합니다.

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

이 명령은 true인 경우 종료 상태 **0** 으로 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 로 **no** 를 출력합니다.

- 허용 목록에 있는 모든 보안 컨텍스트를 나열하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에 컨텍스트 컨텍스트를 추가하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에서 컨텍스트 컨텍스트를 제거하려면 **root** 로 다음 명령을 입력합니다.

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

- 컨텍스트 컨텍스트 가 허용 목록에 있는지 쿼리하려면 **root** 로 다음 명령을 입력합니다.

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

종료 상태 **0** (true인 경우)을 사용하여 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 에서 **no** 를 출력합니다.

- 허용 목록에 있는 모든 사용자 ID를 나열하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에 사용자 ID uid 를 추가하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에서 사용자 ID uid 를 제거하려면 **root** 로 다음 명령을 입력합니다.

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

- 사용자 ID uid 가 허용 목록에 있는지 쿼리하려면 다음 명령을 입력합니다.

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

종료 상태 **0** (true 인 경우)을 사용하여 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 에서 **no** 를 출력합니다.

- 허용 목록에 있는 모든 사용자 이름을 나열하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에 사용자 이름 사용자를 추가하려면 **root** 로 다음 명령을 입력합니다.

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

- 허용 목록에서 사용자 이름 사용자를 제거하려면 **root** 로 다음 명령을 입력합니다.

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

- 사용자 이름 사용자가 허용 목록에 있는지 쿼리하려면 다음 명령을 입력합니다.

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

종료 상태 **0** (true 인 경우)을 사용하여 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 에서 **no** 를 출력합니다.

7.14.3. 설정 파일을 사용하여 잠금 허용 목록 옵션 구성

기본 허용 목록 구성 파일에는 **NetworkManager** 컨텍스트와 **libvirt** 의 기본 컨텍스트가 포함되어 있습니다. 사용자 ID 0도 목록에 있습니다.

+ allowlist 구성 파일은 **/etc/firewalld/** 디렉토리에 저장됩니다.

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
   <user id="0"/>
 </whitelist>
```

다음은 사용자 ID가 **815** 인 사용자에 대해 **firewall-cmd** 유틸리티에 대한 모든 명령을 활성화하는 허용 목록 구성 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 ID** 와 **사용자 이름을** 모두 표시하지만 하나의 옵션만 필요합니다. Python은 인터프리터이며 명령줄 앞에 추가됩니다. 다음과 같은 특정 명령을 사용할 수도 있습니다.*

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

*이 예제에서는 **--lockdown-on** 명령만 허용됩니다.*

*Red Hat Enterprise Linux에서는 모든 유틸리티가 **/usr/bin/** 디렉토리에 배치되고 **/bin/** 디렉토리는 **/usr/bin/** 디렉터리에 연결됩니다. 즉, **root** 로 입력될 때 **firewall-cmd** 의 경로를 **/bin/firewall-cmd** 로 해석할 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 기존에 루트가 아닌 **사용자에게** 만 사용되는 **/usr/bin/firewall-cmd** 경로 외에 허용 목록에 명령 경로를 추가해야 합니다.*

*명령의 name 특성 끝에 있는 * 는 이 문자열로 시작하는 모든 명령이 일치함을 의미합니다. * 가 없는 경우 인수를 포함한 절대 명령이 일치해야 합니다.*

7.15. FIREWALLD 영역에서 다양한 인터페이스 또는 소스 간 트래픽 전달 활성화

*영역 내 전달은 **firewalld** 영역 내의 인터페이스 또는 소스 간 트래픽 전달을 활성화하는 **firewalld** 기능입니다.*

7.15.1. 기본 타겟이 ACCEPT로 설정된 영역 내 전달과 영역의 차이점

*영역 내 전달이 활성화되면 단일 **firewalld** 영역 내의 트래픽이 하나의 인터페이스 또는 소스에서 다른 인터페이스 또는 소스로 이동할 수 있습니다. zone은 인터페이스 및 소스의 신뢰 수준을 지정합니다. 신뢰 수준이 같으면 인터페이스 또는 소스 간 통신이 가능합니다.*

***firewalld** 의 기본 영역에서 영역 내 전달을 활성화하면 현재 기본 영역에 추가된 인터페이스 및 소스에만 적용됩니다.*

***firewalld** 의 **신뢰할 수 있는** 영역은 **ACCEPT** 로 설정된 기본 대상을 사용합니다. 이 영역은 전달된 모든 트래픽을 수락하며 영역 내 전달은 해당 트래픽에 적용되지 않습니다.*

다른 기본 대상 값의 경우 전달된 트래픽은 기본적으로 삭제되며, 신뢰할 수 있는 영역을 제외한 모든 표준 영역에 적용됩니다.

7.15.2. 이더넷과 Wi-Fi 네트워크 간에 트래픽 전달을 위해 영역 내 전달

*영역 내 전달을 사용하여 동일한 **firewalld** 영역에 있는 인터페이스와 소스 간에 트래픽을 전달할 수 있습니다. 예를 들어 이 기능을 사용하여 **enp1s0에 연결된 이더넷 네트워크와 wlp0s 20** 에 연결된 Wi-Fi 네트워크 간에 트래픽을 전달합니다.*

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 활성화하려는 인터페이스가* ***내부*** *영역과 다른 영역에 할당되지 않았는지 확인합니다.*

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

3. *인터페이스가 현재* ***내부*** *이외의 영역에 할당되어 있는 경우 인터페이스를 다시 할당합니다.*

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

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

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

5. *영역 내 전달을 활성화합니다.*

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

검증

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

1. *영역 전달을 활성화한 호스트의* ***enp1s0*** *인터페이스와 동일한 네트워크에 있는 호스트에 로그인합니다.*

2. ***ncat*** *을 사용하여* 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 실행되는 에코 서버에 연결합니다.*

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

5. *임의의 내용을 입력하고* ***Enter*** *키를 누르고 텍스트가 다시 전송되었는지 확인합니다.*

추가 리소스

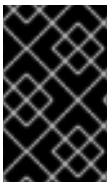
- ***firewalld.zones(5)*** *도움말 페이지*

7.16. ANSIBLE 에서 RHEL SYSTEM ROLES 를 사용하여 FIREWALLD 설정 구성

Ansible 방화벽 시스템 역할을 사용하여 한 번에 여러 클라이언트에서 ***firewalld*** *서비스 설정을 구성할 수 있습니다. 이 해결책:*

- *효율적인 입력 설정을 포함하는 인터페이스를 제공합니다.*
- *의도한 모든* ***firewalld*** *매개 변수를 한 위치에 유지합니다.*

제어 노드에서 **firewall** 역할을 실행한 후 System Role은 **firewalld** 매개변수를 관리형 노드에 즉시 적용하고 재부팅 시에도 영구합니다.



중요

RHEL 채널을 통해 제공되는 RHEL 시스템 역할은 RHEL 고객은 기본 **AppStream** 리포지토리에서 RPM 패키지로 사용할 수 있습니다. RHEL System Roles는 Ansible Automation Hub를 통해 Ansible 서브스크립션을 보유한 고객에게 컬렉션으로도 사용할 수 있습니다.

7.16.1. 방화벽 RHEL 시스템 역할 소개

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

firewalld 서비스의 자동화된 구성을 위해 RHEL 시스템 역할의 **rhel-system-roles.firewall** 역할이 도입되었습니다. **rhel-system-roles** 패키지에는 이 시스템 역할과 참조 문서가 포함되어 있습니다.

자동 방식으로 하나 이상의 시스템에 **firewalld** 매개 변수를 적용하려면 플레이북에서 **방화벽** 시스템 역할 변수를 사용합니다. 플레이북은 텍스트 기반 YAML 형식으로 작성된 하나 이상의 플레이 목록입니다.

인벤토리 파일을 사용하여 Ansible에서 구성할 시스템 세트를 정의할 수 있습니다.

firewall 역할을 사용하면 다양한 **firewalld** 매개변수를 구성할 수 있습니다. 예를 들면 다음과 같습니다.

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

추가 리소스

- **/usr/share/doc/rhel-system-roles/firewall/** 디렉토리에 있는 **README.md** 및 **README.html** 파일
- 플레이북 작업
- 인벤토리를 구축하는 방법

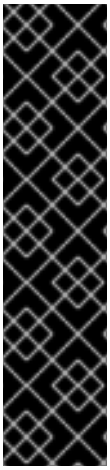
7.16.2. 하나의 로컬 포트에서 다른 로컬 포트로 들어오는 트래픽 전달

rhel-system-roles.firewall 역할을 사용하면 여러 관리 대상 호스트에 지속적으로 적용된 **firewalld** 매개변수를 원격으로 구성할 수 있습니다.

사전 요구 사항

- RHEL 서브스크립션의 인타이틀먼트는 제어 노드에 **ansible-core** 및 **rhel-system-roles** 패키지를 설치했습니다.
- 관리 호스트의 인벤토리는 제어 시스템에 있으며 Ansible은 이러한 호스트에 연결할 수 있습니다.
- 관리 호스트에서 Ansible 플레이북을 실행할 수 있는 권한이 있습니다.

- *플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리 호스트에 대한 적절한 **sudo** 권한이 있습니다.*
- *인벤토리 파일은 플레이북에서 작업을 수행해야 하는 호스트를 나열합니다. 이 절차의 플레이북은 **testinservers** 그룹의 호스트에서 실행됩니다.*



중요

*RHEL 8.0 - 8.5는 Ansible을 기반으로 하는 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 ansible , **ansible -playbook**, **docker** 및 **podman** 와 같은 커넥터, 플러그인 및 모듈 전체와 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine을 다운로드하고 설치하는 방법을 참조하십시오.*

*RHEL 8.6 이상에서는 Ansible 명령줄 유틸리티, 명령 및 일부 기본 제공 Ansible 플러그인 세트를 포함하는 Ansible Core(**ansible-core** RPM으로 제공)를 도입했습니다. AppStream 리포지토리는 제한된 지원 범위가 있는 **ansible-core** 를 제공합니다. RHEL 9 AppStream 에 포함된 ansible-core 패키지에 대한 지원 범위를 검토하여 자세한 내용을 확인할 수 있습니다.*

절차

1. *~/**port_forwarding.yml** 파일을 생성하고 다음 내용을 추가합니다.*

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

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

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

*이 파일은 플레이북을 나타내며 일반적으로 **인벤토리** 파일에서 선택한 특정 관리 호스트에 대해 실행되는 플레이 라고 하는 정렬된 작업 목록을 포함합니다. 이 경우 플레이북은 관리 호스트의 **testingservers** 그룹에 대해 실행됩니다.*

*플레이의 **hosts** 키는 플레이가 실행되는 호스트를 지정합니다. 이 키의 값 또는 값을 관리 호스트의 개별 이름 또는 **인벤토리** 파일에 정의된 호스트 그룹으로 제공할 수 있습니다.*

***tasks** 섹션에는 **vars** 섹션에 언급된 매개 변수와 값을 구성할 시스템 역할을 지정하는 **include_role** 키가 있습니다.*

***vars** 섹션에는 **firewall** 이라는 역할 변수가 포함되어 있습니다. 이 변수는 사전 값 목록입니다. 관리 호스트에서 **firewalld** 에 적용할 매개변수를 지정합니다. 예제 역할은 포트 8080으로 들어오는 트래픽을 포트 443으로 전달합니다. 설정은 즉시 적용되며 재부팅해도 유지됩니다.*

2. *필요한 경우 플레이북의 구문이 올바른지 확인합니다.*

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

playbook: port_forwarding.yml
```

이 예에서는 플레이북의 성공적인 확인을 보여줍니다.

3. *플레이북을 실행합니다.*

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

검증

- *관리 호스트에서 다음을 수행합니다.*
  - *호스트를 다시 시작하여 재부팅 후 **firewalld** 설정이 여전히 제 위치에 있는지 확인합니다.*

```
# reboot
```

  - ***firewalld** 설정을 표시합니다.*

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

추가 리소스

- *RHEL 시스템 역할 시작하기*
- ***/usr/share/doc/rhel-system-roles/firewall/** 디렉토리에 있는 **README.html** 및 **README.md** 파일*
- *인벤토리 빌드*
- *Ansible 구성*
- *플레이북 작업*
- *변수 사용*
- *역할*

7.16.3. 시스템 역할을 사용하여 포트 구성

*RHEL(Red Hat Enterprise Linux) **firewalld** System Role을 사용하여 들어오는 트래픽에 대해 로컬 방화벽에서 포트를 열거나 닫고 재부팅 시 새 구성을 유지할 수 있습니다. 이 예제에서는 **HTTPS** 서비스에 대한 들어오는 트래픽을 허용하도록 기본 영역을 구성하는 방법을 설명합니다.*

Ansible 제어 노드에서 다음 절차를 실행합니다.

사전 요구 사항

- ***firewalld** 시스템 역할을 사용하여 구성하려는 하나 이상의 관리형 노드에 대한 액세스 및 권한.*
- *Red Hat Ansible Engine이 다른 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.*
- ***ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.*
- *플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리형 노드에 적절한 **sudo** 권한이 있습니다.*
- *호스트는 NetworkManager를 사용하여 네트워크를 구성합니다.*

절차

1. *플레이북에서 지침을 실행하려는 호스트가 아직 의도하지 않은 경우 이 호스트의 IP 또는 이름을* ***/etc/ansible/hosts*** *Ansible 인벤토리 파일에 추가합니다.*

```
node.example.com
```

2. *다음 콘텐츠를 사용하여* ***~/adding-and-removing-ports.yml*** *플레이북을 생성합니다.*

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

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

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

permanent: true *옵션을 사용하면 재부팅 시 새 설정이 유지됩니다.*

3. *플레이북을 실행합니다.*

- ***root*** *사용자로 관리 호스트에 연결하려면 다음을 입력합니다.*

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

- *관리 호스트에 사용자로 연결하려면 다음을 입력합니다.*

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

--ask-become-pass *옵션을 사용하면* ***ansible-playbook*** *명령에서* ***-u user_name*** *옵션에 정의된 사용자의* ***sudo*** *암호를 입력하라는 메시지가 표시됩니다.*

-u user_name *옵션을 지정하지 않으면* ***ansible-playbook*** *은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.*

검증

1. *관리형 노드에 연결합니다.*

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

2. ***HTTPS*** *서비스와 연결된* ***443/tcp*** *포트가 열려 있는지 확인합니다.*

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

추가 리소스

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

7.16.4. firewalld RHEL 시스템 역할을 사용하여 NetNamespace firewalld 영역 구성

시스템 관리자는 RHEL ***firewalld*** *시스템 역할을 사용하여* ***enp1s0*** *인터페이스에* ***dmz*** *영역을 구성하여 영역에 대한* ***HTTPS*** *트래픽을 허용할 수 있습니다. 이렇게 하면 외부 사용자가 웹 서버에 액세스할 수 있습니다.*

사전 요구 사항

- *VPN 시스템 역할을 사용하여 구성할 시스템인 하나 이상의 관리 노드에 대한 액세스 및 사용 권한.*
- *Red Hat Ansible Engine이 다른 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.*
- *관리 노드를 나열하는 인벤토리 파일입니다.*
- ***ansible*** *및* ***rhel-system-roles*** *패키지는 제어 노드에 설치됩니다.*
- *플레이북을 실행할 때* ***root*** *와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리형 노드에 적절한* ***sudo*** *권한이 있습니다.*
- *관리형 노드는* ***NetworkManager*** *를 사용하여 네트워크를 구성합니다.*

절차

1. *다음 콘텐츠를 사용하여* ***~/configuring-a-dmz-using-the-firewall-system-role.yml*** *플레이북을 생성합니다.*

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

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

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

2. *플레이북을 실행합니다.*

   - ***root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.*

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

   - *관리 호스트에 사용자로 연결하려면 다음을 입력합니다.*

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

     ***--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지가 표시됩니다.*

   ***-u user_name** 옵션을 지정하지 않으면 **ansible-playbook** 은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.*

검증

- *관리형 노드에서 **dmz** 영역에 대한 자세한 정보를 확인합니다.*

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

7.17. 추가 리소스

- ***firewalld(1)** 도움말 페이지*
- ***firewalld.conf(5)** 도움말 페이지*
- ***firewall-cmd(1)** 도움말 페이지*
- ***firewall-config(1)** 도움말 페이지*
- ***firewall-offline-cmd(1)** 도움말 페이지*
- ***firewalld.icmptype(5)** 도움말 페이지*
- ***firewalld.ipset(5)** 도움말 페이지*
- ***firewalld.service(5)** 도움말 페이지*
- ***firewalld.zone(5)** 도움말 페이지*

- ***firewalld.direct(5)** 도움말 페이지*
- ***firewalld.lockdown-whitelist(5)***
- ***firewalld.richlanguage(5)***
- ***firewalld.zones(5)** 도움말 페이지*
- ***firewalld.dbus(5)** 도움말 페이지*

8장. NFTABLES 시작하기

nftables 프레임워크는 패킷 분류 기능을 제공합니다. 가장 주목할 만한 기능은 다음과 같습니다.

- 선형 처리 대신 기본 제공 조회 테이블
- **IPv4** 및 **IPv6** 프로토콜 모두를 위한 단일 프레임워크
- 전체 규칙 세트를 가져오기, 업데이트 및 저장하는 대신 모두 원자적으로 적용되는 규칙
- 규칙 세트(**nftrace**) 및 모니터링 추적 이벤트(n **ft** 툴의)에서 디버깅 및 추적 지원
- 프로토콜별 확장 없이 보다 일관되고 컴팩트한 구문
- 타사 애플리케이션용 Netlink API

nftables 프레임워크는 테이블을 사용하여 체인을 저장합니다. 체인에는 작업을 수행하기 위한 개별 규칙이 포함되어 있습니다. **libnftnl** 라이브러리는 **libmnl** 라이브러리를 통해 **nftables** Netlink API와의 낮은 수준의 상호 작용에 사용할 수 있습니다.

규칙 세트 변경의 효과를 표시하려면 **nft list ruleset** 명령을 사용합니다. 이러한 툴은 테이블, 체인, 규칙, 세트 및 기타 오브젝트를 **nftables** 규칙 세트에 추가하므로 **nft flush ruleset** 명령과 같은 **nftables** 규칙 세트 작업이 이전에 분리된 레거시 명령을 사용하여 설치된 규칙 세트에 영향을 줄 수 있습니다.

8.1. IPTABLES에서 NFTABLES로 마이그레이션

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

8.1.1. 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. 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.**
- **nftables: nftables 유틸리티를 사용하여 전체 네트워크에 대해 등의 복잡하고 성능 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 Red Hat Enterprise Linux에서 계속 작동합니다. 새로운 방화벽 스크립트의 경우 nftables 를 사용하는 것이 좋습니다.**

중요

서로 다른 방화벽 서비스가 서로 영향을 미치지 않도록 하려면 **RHEL** 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

8.1.2. 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

- *시스템이 부팅될 때 nftables 규칙 자동 로드*

8.1.3. 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**

8.1.4. 일반적인 iptables 및 nftables 명령 비교

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

- 모든 규칙 나열:

iptables	nftables
iptables-save	**nft** 목록 규칙 세트

- 특정 테이블 및 체인 나열:

iptables	nftables
iptables -L	**nft** 목록 테이블 **ip filter**
iptables -L INPUT	**nft** 목록 체인 **IP** 필터 **INPUT**
iptables -t nat -L PREROUTING	**nft** 목록 체인 **ip nat PREROUTING**

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

예제: **firewalld**에서 생성한 규칙 나열

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

8.1.5. 추가 리소스

- *iptables: 두 가지 변형과 nftables의 관계*

8.2. NFTABLES 스크립트 작성 및 실행

nftables 프레임워크는 방화벽 규칙을 유지 관리하기 위해 쉘 스크립트를 사용하는 주요 이점을 가져오는 기본 스크립팅 환경을 제공합니다. 스크립트 실행은 **atomic**입니다. 즉, 시스템이 전체 스크립트를 적용하거나 오류가 발생할 경우 실행을 방지합니다. 이렇게 하면 방화벽이 항상 일관된 상태에 있습니다.

또한 **nftables** 스크립트 환경에서는 관리자가 다음을 수행할 수 있습니다.

- 코멘트 추가
- 변수 정의
- 다른 규칙 세트 파일 포함

이 섹션에서는 이러한 기능을 사용하는 방법과 **nftables** 스크립트를 만들고 실행하는 방법을 설명합니다.

nftables 패키지를 설치하면 **Red Hat Enterprise Linux**가 **/etc/nftables/** 디렉토리에 ***.nft** 스크립트를 자동으로 생성합니다. 이러한 스크립트에는 다양한 용도로 테이블과 빈 체인을 생성하는 명령이 포함되어 있습니다.

8.2.1. 지원되는 nftables 스크립트 형식

nftables 스크립팅 환경은 다음 형식으로 스크립트를 지원합니다.

- *nft list ruleset 명령과 동일한 형식으로 스크립트를 작성할 수 있습니다. 규칙 세트는 다음과 같습니다.*

```
#!/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 명령과 동일한 구문을 명령에 사용할 수 있습니다.*

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

# Flush the rule set
flush ruleset

# Create a table
add table inet example_table

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

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

8.2.2. nftables 스크립트 실행 중

nft 유틸리티로 전달하거나 스크립트를 직접 실행하여 nftables 스크립트를 실행할 수 있습니다.

사전 요구 사항

- *이 섹션의 절차에서는 /etc/nftables/example_firewall.nft 파일에 nftables 스크립트를 저장했다고 가정합니다.*

절차

- *nft 유틸리티에 전달하여 nftables 스크립트를 실행하려면 다음을 입력합니다.*

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

- *직접 nftables 스크립트를 실행하려면 다음을 수행합니다.*

a. *한 번만 필요한 단계 :*

i. *스크립트가 다음 shebang 시퀀스로 시작되는지 확인합니다.*

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

중요

f 매개변수를 생략하면 n ft 유틸리티가 스크립트를 읽지 않고 표시됩니다. 오류: 구문 오류, 예기치 않은 줄 바꿈, 예상 문자열.

ii. *선택 사항: 스크립트 소유자를 root 로 설정합니다.*

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

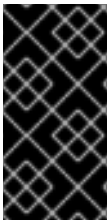
iii. *소유자에 대해 스크립트를 실행 파일로 만듭니다.*

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

b. *스크립트를 실행합니다.*

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

출력이 표시되지 않으면 시스템에서 스크립트를 성공적으로 실행했습니다.



중요

nft 가 스크립트를 성공적으로 실행하는 경우에도 잘못 배치된 규칙, 누락된 매개 변수 또는 스크립트의 기타 문제로 인해 방화벽이 예상대로 작동하지 않을 수 있습니다.

추가 리소스

- **chown(1)** 도움말 페이지
- **chmod(1)** 도움말 페이지
- 시스템이 부팅될 때 **nftables** 규칙 자동 로드

8.2.3. nftables 스크립트에서 주석 사용

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

예 8.1. nftables 스크립트의 주석

주석은 명령 옆에뿐만 아니라 행의 시작 부분에서 시작할 수 있습니다.

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

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

8.2.4. nftables 스크립트에서 변수 사용

nftables 스크립트에서 변수를 정의하려면 **define** 키워드를 사용합니다. 단일 값과 익명 세트를 변수에 저장할 수 있습니다. 더 복잡한 시나리오의 경우 세트 또는 확인 맵을 사용합니다.

단일 값이 있는 변수

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

```
define INET_DEV = enp1s0
```

- $ 기호 다음에 변수 이름을 작성하여 스크립트에서 변수를 사용할 수 있습니다.

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

익명 세트를 포함하는 변수

다음 예제에서는 익명 세트를 포함하는 변수를 정의합니다.

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

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

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

참고

중괄호는 변수가 집합을 나타내고 있음을 나타내기 때문에 규칙에서 사용할 때 특수한 의미 체계가 있습니다.

추가 리소스

- nftables 명령의 세트 사용
- nftables 명령에서 verdict 맵 사용

8.2.5. nftables 스크립트에 파일 포함

nftables 스크립팅 환경을 사용하면 관리자가 include 문을 사용하여 다른 스크립트를 포함할 수 있습니다.

절대 경로 또는 상대 경로 없이 파일 이름만 지정하면 nftables 에 기본 검색 경로의 파일이 포함되며 이는 Red Hat Enterprise Linux에서 /etc 로 설정됩니다.

예 8.2. 기본 검색 디렉토리의 파일 포함

기본 검색 디렉터리에서 파일을 포함하려면 다음을 수행합니다.

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

예 8.3. 디렉토리의 *.nft 파일 모두 포함

/etc/nftables/rulesets/ 디렉토리에 저장된 ***.nft** 로 끝나는 모든 파일을 포함시키려면 다음을 수행하십시오.

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

include 문은 점으로 시작하는 파일과 일치하지 않습니다.

추가 리소스

- **nft(8)** 도움말 페이지에 **Include files** 섹션

8.2.6. 시스템이 부팅될 때 nftables 규칙을 자동으로 로드

nftables systemd 서비스는 **/etc/sysconfig/nftables.conf** 파일에 포함된 방화벽 스크립트를 로드합니다. 이 섹션에서는 시스템이 부팅될 때 방화벽 규칙을 로드하는 방법을 설명합니다.

사전 요구 사항

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

절차

1. **/etc/sysconfig/nftables.conf** 파일을 편집합니다.
   - **nftables** 패키지를 설치할 때 **/etc/nftables/** 에 생성된 ***.nft** 스크립트를 개선하는 경우 이러한 스크립트에 대한 **include** 문의 주석을 제거합니다.

- 스크립트를 처음부터 작성하는 경우 **include** 문을 추가하여 이러한 스크립트를 포함합니다. 예를 들어 **nftables** 서비스가 시작될 때 **/etc/nftables/example.nft** 스크립트를 로드하려면 다음을 추가합니다.

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

2. 선택적으로 시스템을 재부팅하지 않고 방화벽 규칙을 로드하도록 **nftables** 서비스를 시작합니다.

```
# systemctl start nftables
```

3. **nftables** 서비스를 활성화합니다.

```
# systemctl enable nftables
```

추가 리소스

- **nftables** 스크립트 형식 지원

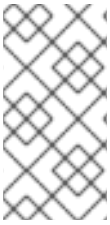
8.3. NFTABLES 테이블, 체인 및 규칙 생성 및 관리

이 섹션에서는 **nftables** 규칙 세트를 표시하는 방법과 이를 관리하는 방법을 설명합니다.

8.3.1. 표준 체인 우선 순위 값 및 텍스트 이름

체인을 만들 때 우선 순위 는 정수 값 또는 동일한 후크 값을 통과하는 체인이 있는 순서를 지정하는 표준 이름을 설정할 수 있습니다.

이름 및 값은 기본 체인을 등록할 때 **xtables** 에서 사용하는 우선 순위에 따라 정의됩니다.



참고

nft list chains 명령은 기본적으로 텍스트 우선 순위 값을 표시합니다. 명령에 **-y** 옵션을 전달하여 숫자 값을 볼 수 있습니다.

예 8.4. 텍스트 값을 사용하여 우선순위 설정

*다음 명령은 표준 우선 순위 값 **50** 을 사용하여 **example_table**에 **example_ chain** 이라는 체인을 생성합니다.*

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

우선순위는 표준 값이므로 텍스트 값을 사용할 수도 있습니다.

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

*표 **8.1.** 표준 우선 순위 이름, 제품군 및 후크 호환성 매트릭스*

이름	현재의	제품군	후크
raw	-300	**ip**, **ip6**, **inet**	모두
mangle	-150	**ip**, **ip6**, **inet**	모두
dstnat	-100	**ip**, **ip6**, **inet**	PREROUTING
filter	0	**ip**, **ip6**, **inet**, **arp**, **netdev**	모두
보안	50	**ip**, **ip6**, **inet**	모두
srcnat	100	**ip**, **ip6**, **inet**	POSTROUTING

모든 제품군은 동일한 값을 사용하지만 브리지 제품군은 다음과 같은 값을 사용합니다.

*표 **8.2.** 브리지 제품군의 표준 우선 순위 이름 및 후크 호환성*

이름	현재의	후크
dstnat	-300	PREROUTING
filter	-200	모두
아웃	100	출력 결과
srcnat	300	POSTROUTING

추가 리소스

- nft(8) 도움말 페이지의 체인 섹션

8.3.2. nftables 규칙 세트 표시

nftables 의 규칙 세트에는 테이블, 체인 및 규칙이 포함됩니다. 이 섹션에서는 규칙 집합을 표시하는 방법을 설명합니다.

절차

- 규칙 세트를 표시하려면 다음을 입력합니다.

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

참고

기본적으로 nftables 는 테이블을 사전 생성하지 않습니다. 그 결과 테이블이 없는 호스트에 규칙 집합을 표시하면 nft list ruleset 명령은 출력이 표시되지 않습니다.

8.3.3. nftables 테이블 생성

nftables 의 테이블은 체인, 규칙, 세트 및 기타 오브젝트 컬렉션이 포함된 네임 스페이스입니다. 이 섹션에서는 테이블을 만드는 방법을 설명합니다.

각 테이블에는 주소 제품군이 정의되어 있어야 합니다. 테이블의 주소 제품군은 테이블 프로세스 유형을 정의합니다. 테이블을 만들 때 다음 주소 제품군 중 하나를 설정할 수 있습니다.

- ip: 일치하는 IPv4 패킷 만 일치합니다. 주소 제품군을 지정하지 않으면 기본값입니다.

- *ip6: IPv6 패킷 만 일치합니다.*
- *inet: IPv4 및 IPv6 패킷과 일치합니다.*
- *arp: IPv4 ARP(Address Resolution Protocol) 패킷과 일치합니다.*
- *bridge: 는 브리지 장치를 통과하는 패킷과 일치합니다.*
- *netdev: 수신에서 패킷 일치.*

절차

1. *nft add table 명령을 사용하여 새 테이블을 만듭니다. 예를 들어 IPv4 및 IPv6 패킷을 처리하는 example_table 이라는 테이블을 생성하려면 다음을 수행합니다.*

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

2. *선택적으로 규칙 세트의 모든 테이블을 나열합니다.*

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

추가 리소스

- *nft(8) 도움말 페이지의 Address family 섹션*
- *nft(8) 도움말 페이지의 테이블 섹션*

8.3.4. nftables 체인 생성

체인은 규칙을 위한 컨테이너입니다. 다음 두 가지 규칙 유형이 있습니다.

- *기본 체인: 기본 체인을 네트워킹 스택의 패킷 진입점으로 사용할 수 있습니다.*

- *일반 체인: 일반 체인을 건너뛰기 대상으로 사용하고 규칙을 더 효과적으로 구성할 수 있습니다.*

이 절차에서는 기존 테이블에 기본 체인을 추가하는 방법을 설명합니다.

사전 요구 사항

- *새 체인을 추가할 테이블입니다.*

절차

1. ***nft add chain** 명령을 사용하여 새 체인을 만듭니다. 예를 들어 **example_ table** 에 **example_chain** 이라는 체인을 생성하려면 다음을 수행합니다.*

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

중요

쉘이 세미콜론을 명령의 끝으로 해석하지 않도록 하려면 세미콜론 앞에 \ 이스케이프 문자 앞에 추가합니다.

*이 체인은 들어오는 패킷을 필터링합니다. **priority** 매개변수는 **nftables** 가 동일한 후크 값이 있는 체인을 처리하는 순서를 지정합니다. 우선순위 값이 더 높은 값보다 우선합니다. **policy** 매개 변수는 이 체인의 규칙에 대한 기본 작업을 설정합니다. 원격으로 서버에 로그인하고 기본 정책을 종료하도록 설정한 경우 다른 규칙에서 원격 액세스를 허용하지 않는 경우 즉시 연결이 끊어집니다.*

2. *선택적으로 모든 체인을 표시합니다.*

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

추가 리소스

- *nft(8) 도움말 페이지의 Address family 섹션*

- *nft(8) 도움말 페이지의 체인 섹션*

8.3.5. nftables 체인 끝에 규칙 추가

이 섹션에서는 기존 nftables 체인의 끝에 규칙을 추가하는 방법을 설명합니다.

사전 요구 사항

- *규칙을 추가할 체인이 있습니다.*

절차

1. *새 규칙을 추가하려면 nft add rule 명령을 사용합니다. 예를 들어 port 22에서 TCP 트래픽을 허용하는 example_ table의 example_ chain 에 규칙을 추가하려면 다음을 수행합니다.*

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

포트 번호 대신 서비스 이름을 지정할 수도 있습니다. 이 예제에서는 포트 번호 22 대신 ssh 를 사용할 수 있습니다. 서비스 이름은 /etc/services 파일의 항목에 따라 포트 번호로 확인됩니다.

2. *필요한 경우 모든 체인과 해당 규칙을 example_table 에 표시합니다.*

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

추가 리소스

- *nft(8) 도움말 페이지의 Address family 섹션*

- *nft(8) 도움말 페이지의 Rules 섹션*

8.3.6. nftables 체인의 시작 부분에 규칙 삽입

이 섹션에서는 기존 nftables 체인의 시작 부분에 규칙을 삽입하는 방법을 설명합니다.

사전 요구 사항

- *규칙을 추가할 체인이 있습니다.*

절차

1. *새 규칙을 삽입하려면 nft 삽입 규칙 명령을 사용합니다. 예를 들어 포트 22에서 TCP 트래픽을 허용하는 example_ table의 example_ chain 에 규칙을 삽입하려면 다음을 수행합니다.*

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

또는 포트 번호 대신 서비스 이름을 지정할 수 있습니다. 이 예제에서는 포트 번호 22 대신 ssh 를 사용할 수 있습니다. 서비스 이름은 /etc/services 파일의 항목에 따라 포트 번호로 확인됩니다.

2. *필요한 경우 모든 체인과 해당 규칙을 example_table 에 표시합니다.*

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

추가 리소스

- *nft(8) 도움말 페이지의 Address family 섹션*
- *nft(8) 도움말 페이지의 Rules 섹션*

8.3.7. nftables 체인의 특정 위치에 규칙 삽입

이 섹션에서는 **nftables** 체인의 기존 규칙 전후에 규칙을 삽입하는 방법을 설명합니다. 이렇게 하면 올바른 위치에 새 규칙을 배치할 수 있습니다.

사전 요구 사항

- 규칙을 추가할 체인이 있습니다.

절차

1. **nft -a list ruleset** 명령을 사용하여 핸들을 포함하여 모든 체인과 해당 규칙을 **example_table** 에 표시합니다.

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

a를 사용하면 프로세스가 표시됩니다. 새 규칙을 다음 단계에서 배치하려면 이 정보가 필요합니다.

2. **example_ table** 의 **example_chain** 체인에 새 규칙을 삽입합니다.

- 핸들 **3** 이전의 포트 **636** 에 **TCP** 트래픽을 허용하는 규칙을 삽입하려면 다음을 입력합니다.

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

- 핸들 **3** 이후 포트 **80** 에서 **TCP** 트래픽을 허용하는 규칙을 추가하려면 다음을 입력합니다.

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

3. 필요한 경우 모든 체인과 해당 규칙을 **example_table** 에 표시합니다.

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

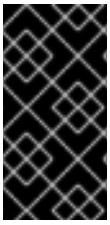
추가 리소스

- **nft(8)** 도움말 페이지의 **Address family** 섹션
- **nft(8)** 도움말 페이지의 **Rules** 섹션

8.4. NFTABLES를 사용하여 NAT 구성

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

- 마스커레이딩
- **SNAT(소스 NAT)**
- 대상 **NAT(DNAT)**
- 리디렉션



중요

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

8.4.1. 다양한 NAT 유형: 마스커레이드, 소스 NAT, 대상 NAT, 리디렉션

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 공급자는 10.0.0.0/8 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하며 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 해당 범위에서 패킷의 소스 IP 주소를 공용 IP 주소로 매핑합니다.

마스커레이딩과 SNAT 모두 매우 유사합니다. 차이점은 다음과 같습니다.

- 마스커레이딩은 나가는 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 나가는 인터페이스에서 동적 IP 주소를 사용하는 경우 마스커레이딩을 사용합니다.
- SNAT는 패킷의 소스 IP 주소를 지정된 IP로 설정하고 나가는 인터페이스의 IP를 동적으로 조회하지 않습니다. 따라서 SNAT는 마스커레이딩보다 빠릅니다. 나가는 인터페이스에서 고정 IP 주소를 사용하는 경우 SNAT를 사용합니다.

대상 NAT(DNAT)

이 NAT 유형을 사용하여 들어오는 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 들어오는 트래픽을 이 서버로 리디렉션할 수 있습니다.

리디렉션

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

8.4.2. nftables를 사용하여 마스커레이딩 구성

마스커레이딩을 사용하면 라우터가 인터페이스를 통해 전송된 패킷의 소스 IP를 인터페이스의 IP 주소로 동적으로 변경할 수 있습니다. 즉, 인터페이스에 새 IP가 할당되면 nftables 에서 소스 IP를 교체할 때 새 IP를 자동으로 사용합니다.

다음 절차에서는 호스트가 ens3 인터페이스를 통해 나가는 패킷의 소스 IP를 ens3 에 설정된 IP로 바꾸는 방법을 설명합니다 .

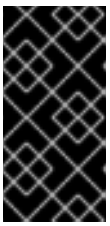
절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. 테이블에 prerouting 및 postrouting 체인을 추가합니다.

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



중요

이전 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크에는 이 체인이 들어오는 패킷 응답과 일치해야 합니다.

쉘이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3** 인터페이스에서 나가는 패킷과 일치하는 사후 라우팅 체인에 규칙을 추가합니다.

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

8.4.3. nftables를 사용하여 소스 NAT 구성

라우터에서 **SNAT(Source NAT)**를 사용하면 인터페이스를 통해 전송된 패킷의 **IP**를 특정 **IP** 주소로 변경할 수 있습니다.

다음 절차에서는 **ens3** 인터페이스를 통해 나가는 패킷의 소스 **IP**를 **192.0.2.1** 로 바꾸는 방법을 설명합니다.

절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. 테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

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

중요

사후 라우팅 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크에서 이 체인이 나가는 패킷 응답과 일치해야 합니다.

쉘이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3** 을 통해 나가는 패킷의 소스 **IP**를 **192.0.2.1**로 대체하는 사후 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

8.4.4. nftables를 사용하여 대상 NAT 구성

대상 **NAT**를 사용하면 라우터의 트래픽을 인터넷에서 직접 액세스할 수 없는 호스트로 리디렉션할 수 있습니다.

다음 절차에서는 라우터의 포트 **80** 및 **443** 으로 전송된 수신 트래픽을 **192.0.2.1 IP** 주소가 있는 호스트로 리디렉션하는 방법을 설명합니다.

절차

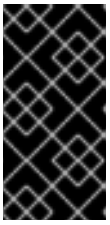
1. 테이블을 생성합니다.

```
# nft add table nat
```

2.

2. 테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

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



중요

사후 라우팅 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크에서 이 체인이 나가는 패킷 응답과 일치해야 합니다.

셸이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **192.0.2.1 IP**가 있는 호스트로 포트 **80** 및 **443** 으로 전송된 **ens3** 인터페이스에서 들어오는 트래픽을 리디렉션하는 **prerouting** 체인에 규칙을 추가합니다.

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

4. 환경에 따라 소스 주소를 변경하려면 **SNAT** 또는 **masquerading** 규칙을 추가합니다.

   a. **ens3** 인터페이스가 동적 **IP** 주소를 사용한 경우 마스커레이딩 규칙을 추가합니다.

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

   b. **ens3** 인터페이스가 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
   ```

추가 리소스

- 다른 **NAT** 유형: 마스커레이딩, 소스 **NAT**, 대상 **NAT**, 리디렉션

8.4.5. nftables를 사용하여 리디렉션 구성

리디렉션 기능은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 대상 네트워크 주소 변환

(DNAT)의 특별한 경우입니다.

다음 절차에서는 수신 트래픽을 로컬 호스트의 포트 22 로 리디렉션하고 포트 2222 로 전달하는 방법을 설명합니다.

절차

1. *테이블을 생성합니다.*

```
# nft add table nat
```

2. *prerouting 체인을 테이블에 추가합니다.*

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

쉘이 음수 우선 순위 값을 nft 명령의 옵션으로 해석하지 않으려면 -- 옵션을 nft 명령에 전달해야 합니다.

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

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

추가 리소스

- *다른 NAT 유형: 마스커레이딩, 소스 NAT, 대상 NAT, 리디렉션*

8.5. NFTABLES 명령의 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.

8.5.1. nftables에서 익명 세트 사용

익명의 세트에는 규칙에서 직접 사용하는 { 22, 80, 443 } 와 같이 curly 대괄호로 묶은 쉼표로 구분된 값이 포함되어 있습니다. IP 주소 또는 기타 일치 기준에도 익명 세트를 사용할 수 있습니다.

익명 세트의 단점은 세트를 변경하려면 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 nftables에서 명명된 세트 사용에 설명된 대로 named sets 를 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 에서 example_chain 에 규칙을 추가하려면 다음을 수행합니다.

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

2. 필요한 경우 모든 체인과 해당 규칙을 example_table 에 표시합니다.

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

8.5.2. nftables에서 이름이 지정된 세트 사용

nftables 프레임워크는 변경 가능한 이름 집합을 지원합니다. 명명된 세트는 테이블 내의 여러 규칙에 사용할 수 있는 요소 목록 또는 범위입니다. 익명 세트보다 다른 이점은 세트를 사용하는 규칙을 교체하지 않고도 명명된 집합을 업데이트할 수 있다는 것입니다.

명명된 집합을 만들 때 집합에 포함된 요소의 유형을 지정해야 합니다. 다음 유형을 설정할 수 있습니다.

- 192.0.2.1 또는 192.0.2.0/24 와 같은 IPv4 주소 또는 범위가 포함된 세트의 ipv4_addr 입니다.
- 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 입니다.*

- *tcp 와 같은 인터넷 프로토콜 유형 목록이 포함된 세트의 inet_proto 입니다.*

- *ssh 와 같은 인터넷 서비스 목록이 포함된 세트의 inet_service.*

- *패킷 표시 목록이 포함된 집합의 표시입니다. 패킷 표시는 임의의 양의 32비트 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 생성하려면 다음을 실행합니다.*

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

   - *IPv4 주소 범위를 저장할 수 있는 세트를 생성하려면 다음을 수행합니다.*

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

   중요

   쉘이 command의 종료로 Semis를 해석하는 것을 방지하려면, 백슬래시를 사용하여 별칭을 이스케이프해야 합니다.

2. *선택적으로 집합을 사용하는 규칙을 만듭니다. 예를 들어 다음 명령은 example_set 의 IPv4 주소에서 모든 패킷을 삭제하는 example_table 의 example_chain 에 규칙을 추가합니다.*

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

example_set 은 여전히 비어 있으므로 현재 규칙의 효과가 없습니다.

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 지정하는 경우 위 예제에서 192.0.2.0/24 와 같이 CIDR(Classless Inter-Domain Routing) 표기법을 사용할 수도 있습니다.

8.5.3. 추가 리소스

- nft(8) 도움말 페이지의 세트 섹션

8.6. NFTABLES 명령에서 VERDICT 맵 사용

사전이라고도 하는verdict 맵을 사용하면 nft 가 일치 기준을 작업에 매핑하여 패킷 정보를 기반으로 작업을 수행할 수 있습니다.

8.6.1. nftables에서 익명 맵 사용

익명 맵은 규칙에서 직접 사용하는 { match_criteria : action } 문입니다. 이 문에는 쉼표로 구분된 여러 매핑이 포함될 수 있습니다.

익명 맵의 단점은 맵을 변경하려면 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 nftables에서 명명된 맵을 사용하여 에 설명된 대로 명명된 맵 을 사용합니다.

이 예제에서는 익명 맵을 사용하여 IPv4 및 IPv6 프로토콜의 TCP 및 UDP 패킷을 다른 체인으로 라우

팅하는 방법을 설명합니다.

절차

1. ***example_table** 을 생성합니다.*

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

2. ***example_table** 에서 **tcp_packets** 체인을 생성합니다.*

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

3. *이 체인의 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 체인의 트래픽을 계산하는 **udp_packets** 에 규칙을 추가합니다.*

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

6. *들어오는 트래픽의 체인을 만듭니다. 예를 들어 들어오는 트래픽을 필터링하는 **example_table** 에 **incoming_traffic** 이라는 체인을 생성하려면 다음을 실행합니다.*

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

7. *수신_**traffic** 에 익명 맵을 사용하여 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 다른 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 패킷 및 바이트 수를 모두 표시합니다.

8.6.2. nftables에서 이름이 지정된 맵 사용

nftables 프레임워크는 이름이 지정된 맵을 지원합니다. 이러한 맵을 테이블 내에서 여러 규칙에 사용할 수 있습니다. 익명 맵보다 다른 이점은 이름을 사용하는 규칙을 교체하지 않고도 명명된 맵을 업데이트할 수 있다는 것입니다.

이름이 지정된 맵을 생성할 때 요소의 유형을 지정해야 합니다.

- 일치하는 부분에 있는 맵의 **ipv4_addr** 에는 **192.0.2.1** 과 같은 **IPv4** 주소가 포함되어 있습니다.
- 일치하는 부분에 **IPv6** 주소가 포함된 맵의 **ipv6_addr (예: 2001:db8:1::1)**
- 일치하는 부분에 대한 **ether_addr** 에는 **52:54:00:6b:66:42** 와 같은 **MAC(Media Access Control)** 주소가 포함되어 있습니다.
- 일치하는 부분에 있는 맵의 **inet_proto** 에는 **tcp** 와 같은 인터넷 프로토콜 유형이 포함되어 있습니다.

- *일치하는 부분에 있는 맵의 **inet_service** 에는 **ssh** 또는 **22** 와 같은 인터넷 서비스 이름 포트 번호가 포함됩니다.*

- *일치하는 부분에 패킷 표시가 포함된 맵의 표시입니다. 패킷 표시는 임의의 양의 **32**비트 정수 값**(0 ~2147483647)**일 수 있습니다.*

- *일치하는 파트가 카운터 값이 포함된 맵의 카운터입니다. 카운터 값은 양의 **64**비트 정수 값일 수 있습니다.*

- *일치하는 부분에 할당량 값이 포함된 맵의 할당량입니다. 할당량 값은 모든 양의 **64**비트 정수 값일 수 있습니다.*

*이 예제에서는 소스 **IP** 주소를 기반으로 들어오는 패킷을 허용하거나 삭제하는 방법을 설명합니다. 명명된 맵을 사용하면 이 시나리오를 구성하는 단일 규칙만 필요하며, **IP** 주소와 작업은 맵에 동적으로 저장됩니다. 이 절차에서는 맵에서 항목을 추가하고 제거하는 방법도 설명합니다.*

절차

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

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

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

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

중요

*쉘이 **command**의 종료로 **Semis**를 해석하는 것을 방지하려면, 백슬래시를 사용하여 별칭을 이스케이프해야 합니다.*

3. *빈 맵을 생성합니다. 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_table에서 모두 example_map 에 정의된 IPv4 주소에 작업을 적용하는 example_table 의 example_chain 에 규칙을 추가합니다.

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

5. example_map 에 IPv4 주소 및 해당 작업을 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 의 패킷을 수락하고 192.0.2.2 에서 패킷을 삭제합니다.

6. 선택적으로 다른 IP 주소 및 action 문을 추가하여 맵을 향상시킵니다.

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

7. 선택적으로 맵에서 항목을 제거합니다.

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

8. 선택적으로 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
  }
}
```

8.6.3. 추가 리소스

- nft(8) 도움말 페이지의 Map 섹션

8.7. NFTABLES를 사용하여 포트 전달 구성

관리자는 포트 전달을 통해 특정 대상 포트로 전송된 패킷을 다른 로컬 또는 원격 포트로 전달할 수 있습니다.

예를 들어 웹 서버에 공용 IP 주소가 없는 경우 방화벽에서 방화벽의 수신 패킷을 웹 서버로 전달하는 포트 전달 규칙을 설정할 수 있습니다. 이 방화벽 규칙을 사용하면 인터넷의 사용자가 방화벽의 IP 또는 호스트 이름을 사용하여 웹 서버에 액세스할 수 있습니다.

8.7.1. 들어오는 패킷을 다른 로컬 포트로 전달

이 섹션에서는 포트 8022 의 수신 IPv4 패킷을 로컬 시스템의 포트 22 로 전달하는 방법에 대한 예를 설명합니다.

절차

1. ip address family로 nat 이라는 테이블을 생성합니다.

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

2. 테이블에 prerouting 및 postrouting 체인을 추가합니다.

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

참고

nft 명령의 옵션으로 쉘이 음수 우선 순위 값을 해석하지 않도록 -- 옵션을 nft 명령에 전달합니다.

3. 포트 8022 에서 수신되는 패킷을 로컬 포트 22 로 리디렉션하는 prerouting 체인에 규칙을 추가합니다.

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

8.7.2. 특정 로컬 포트에서 들어오는 패킷을 다른 호스트로 전달

대상 네트워크 주소 변환(DNAT) 규칙을 사용하여 로컬 포트의 수신 패킷을 원격 호스트에 전달할 수

있습니다. 이를 통해 인터넷 사용자는 개인 IP 주소가 있는 호스트에서 실행되는 서비스에 액세스할 수 있습니다.

이 절차에서는 로컬 포트 443 에서 수신되는 IPv4 패킷을 192.0.2.1 IP 주소로 원격 시스템의 동일한 포트 번호로 전달하는 방법을 설명합니다.

사전 요구 사항

- 패킷을 전달해야 하는 시스템에서 root 사용자로 로그인했습니다.

절차

1. ip address family로 nat 이라는 테이블을 생성합니다.

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

2. 테이블에 prerouting 및 postrouting 체인을 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 명령의 옵션으로 쉘이 음수 우선 순위 값을 해석하지 않도록 -- 옵션을 nft 명령에 전달합니다.

3. 포트 443 에서 수신되는 패킷을 192.0.2.1 의 동일한 포트로 리디렉션하는 prerouting 체인에 규칙을 추가합니다.

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

4. postrouting 체인에 규칙을 추가하여 masquerade 발신 트래픽에 규칙을 추가합니다.

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

5. 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
```

8.8. NFTABLES를 사용하여 연결 양 제한

nftables 를 사용하여 연결 수를 제한하거나 지정된 양의 연결을 설정하려는 IP 주소를 차단하여 너무 많은 시스템 리소스를 사용하지 않도록 할 수 있습니다.

8.8.1. nftables를 사용하여 연결 수 제한

nft 유틸리티의 ct count 매개 변수를 사용하면 관리자가 연결 수를 제한할 수 있습니다. 이 절차에서는 들어오는 연결을 제한하는 방법에 대한 기본 예제를 설명합니다.

사전 요구 사항

- example_table 의 기본 example_chain 이 있습니다.

절차

1. IPv4 주소에 대한 동적 세트를 생성합니다.

```
# nft add set inet example_table example_meter { type ipv4_addr\; flags dynamic \;}
```

2. IPv4 주소에서 SSH 포트(22)에 대한 동시 연결을 허용하는 규칙을 추가하고 동일한 IP에서 추가로 모든 연결을 거부합니다.

```
# nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
```

3. 선택적으로 이전 단계에서 생성한 세트를 표시합니다.

```
# nft list set inet example_table example_meter
table inet example_table {
  meter example_meter {
    type ipv4_addr
    size 65535
    elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2  }
  }
}
```

elements 항목은 현재 규칙과 일치하는 주소를 표시합니다. 이 예제에서 요소는 SSH 포트에 대한 활성 연결이 있는 IP 주소를 나열합니다. 출력에 활성 연결 수 또는 연결이 거부된 경우 출력에 표시되지 않습니다.

8.8.2. 1분 이내에 10개 이상의 새로운 수신 TCP 연결을 시도하는 IP 주소 차단

이 섹션에서는 1분 이내에 10개의 IPv4 TCP 연결을 설정하는 호스트를 일시적으로 차단하는 방법을 설명합니다.

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. *1분 이내에 10개 이상의 TCP 연결을 설정하려는 소스 주소에서 모든 패킷을 삭제하는 규칙을 추가합니다.*

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked meter ratemeter { ip saddr timeout 5m limit rate over 10/minute } drop
```

timeout 5m 매개변수는 nftables 가 5분 후에 자동으로 항목을 제거하여 계측이 오래된 항목으로 채워지지 않도록 정의합니다.

검증

- *미터 콘텐츠를 표시하려면 다음을 입력합니다.*

```
# nft list meter ip filter ratemeter
table ip filter {
  meter ratemeter {
    type ipv4_addr
    size 65535
    flags dynamic,timeout
    elements = { 192.0.2.1 limit rate over 10/minute timeout 5m expires 4m58s224ms }
  }
}
```

8.9. NFTABLES 규칙 디버깅

nftables 프레임워크는 관리자가 규칙을 디버깅하고 패킷과 일치하는 경우 다양한 옵션을 제공합니다. 이 섹션에서는 이러한 옵션에 대해 설명합니다.

8.9.1. 카운터를 사용하여 규칙 생성

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다. 이 섹션에서는 카운터로 새 규칙을 만드는 방법에 대해 설명합니다.

- 기존 규칙에 카운터를 추가하는 프로시저에 대한 자세한 내용은 네트워킹 구성 및 관리에서 기존 규칙에 카운터 추가를 참조하십시오.**For more information on a procedure that adds a counter to an existing rule, see Adding a counter to an existing rule in Configuring and managing networking.**

사전 요구 사항

- 규칙을 추가할 체인이 있습니다.

절차

1. **counter** 매개 변수가 있는 새 규칙을 체인에 추가합니다. 다음 예제에서는 포트 **22**에서 **TCP** 트래픽을 허용하고 이 규칙과 일치하는 패킷 및 트래픽을 계산하는 카운터를 포함하는 규칙을 추가합니다.

```
# 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
  }
}
```

8.9.2. 기존 규칙에 카운터 추가

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다. 이 섹션에서는 기존 규칙에 카운터를 추가하는 방법을 설명합니다.

- 카운터를 사용하여 새 규칙을 추가하는 절차에 대한 자세한 내용은 네트워킹 구성 및 관리에서 *카운터를 사용하여 규칙 생성* 을 참조하십시오.

사전 요구 사항

- 카운터를 추가하려는 규칙이 있습니다.

절차

1. 처리를 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 **counter** 매개 변수로 교체하여 카운터 를 추가합니다. 다음 예제에서는 이전 단계에 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
  }
}
```

8.9.3. 기존 규칙과 일치하는 패킷 모니터링

nft monitor 명령과 함께 **nftables** 의 추적 기능을 사용하면 관리자가 규칙과 일치하는 패킷을 표시할 수 있습니다. 이 절차에서는 규칙의 추적을 활성화하고 이 규칙과 일치하는 패킷을 모니터링하는 방법을 설명합니다.

사전 요구 사항

- *카운터를 추가하려는 규칙이 있습니다.*

절차

1. *처리를 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 메타 **nftrace** 세트 **1** 매개변수로 교체하여 추적 기능을 추가합니다. 다음 예제에서는 이전 단계에 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** 명령을 사용하여 추적을 표시합니다. 다음 예제에서는 **inet example_table example_chain** 을 포함하는 항목만 표시하도록 명령의 출력을 필터링합니다.*

```
# 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 수와 일치하는 트래픽 양에 따라 **nft monitor** 명령은 많은 출력을 표시할 수 있습니다. **grep** 또는 기타 유틸리티를 사용하여 출력을 필터링합니다.*

8.10. NFTABLES 규칙 세트 백업 및 복원

이 섹션에서는 파일에서 **nftables** 규칙을 백업하는 방법과 파일에서 규칙을 복원하는 방법에 대해 설명합니다.

관리자는 규칙과 함께 파일을 사용하여 규칙을 사용하여 규칙을 다른 서버로 전송할 수 있습니다.

8.10.1. 파일에 nftables 규칙 세트 백업

이 섹션에서는 파일에 **nftables** 규칙 세트를 백업하는 방법을 설명합니다.

절차

- **nftables** 규칙을 백업하려면 다음을 수행합니다.
  - **nft** 목록 **ruleset** 형식으로 생성된 형식으로 다음을 수행하십시오.

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

  - **JSON** 형식의 경우:

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

8.10.2. 파일에서 nftables 규칙 세트 복원

이 섹션에서는 **nftables** 규칙 세트를 복원하는 방법을 설명합니다.

절차

- **nftables** 규칙을 복원하려면 다음을 수행합니다.
  - 복원할 파일이 **nft** 목록 규칙 세트에 의해 생성되거나 **n ft** 명령을 직접 포함하는 경우:

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

- 복원할 파일이 JSON 형식인 경우:

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

8.11. 추가 리소스

- Red Hat Enterprise Linux 8에서 nftables 사용
- iptables 뒤에는 어떻게 됩니까? 그 후터, 물론: nftables
- firewalld: future는 nftables입니다.

9장. 네트워크 서비스 보안

Red Hat Enterprise Linux 8은 다양한 유형의 네트워크 서버를 지원합니다. 해당 네트워크 서비스는 시스템 보안을 노출하여 서비스 거부 공격**(DoS)**, 분산 서비스 거부 공격**(DDoS)**, 스크립트 취약성 공격 및 버퍼 오버플로 공격과 같은 다양한 유형의 공격의 위험에 노출될 수 있습니다.

공격에 대한 시스템 보안을 늘리려면 사용하는 활성 네트워크 서비스를 모니터링하는 것이 중요합니다. 예를 들어 네트워크 서비스가 시스템에서 실행 중인 경우 데몬은 네트워크 포트에서 연결을 수신 대기하므로 보안이 저하될 수 있습니다. 네트워크를 통한 공격에 대한 노출을 제한하려면 사용되지 않는 모든 서비스를 꺼야 합니다.

9.1. NETNAMESPACEBIND 서비스 보안

Net Namespacebind 서비스는 **NIS(Network Information Service)** 및 **NFS(Network File System)**와 같은 원격 프로시저 호출**(RPC)** 서비스에 대한 동적 포트 할당 데몬입니다. 인증 메커니즘이 약하고 제어하는 서비스에 다양한 포트를 할당할 수 있기 때문에 **NetNamespace bind** 의 보안을 유지하는 것이 중요합니다.

모든 네트워크에 대한 액세스를 제한하고 서버의 방화벽 규칙을 사용하여 특정 예외를 정의하여 **NetNamespace bind** 를 보호할 수 있습니다.

참고

- **NFSv2** 및 **NFSv3** 서버에는 **NetNamespace bind** 서비스가 필요합니다.
- **NFSv4**에서는 **rpcbind** 서비스가 네트워크에서 수신 대기할 필요가 없습니다.

사전 요구 사항

- **Net Namespacebind** 패키지가 설치되어 있어야 합니다.
- **firewalld** 패키지가 설치되어 서비스가 실행 중입니다.

절차

1. 방화벽 규칙을 추가합니다. 예를 들면 다음과 같습니다.

- **TCP** 연결을 제한하고 **111** 포트를 통해 **192.168.0.0/24** 호스트에서 패키지를 수락합니다.

```
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source
address="192.168.0.0/24" invert="True" drop'
```

- **TCP** 연결을 제한하고 **111** 포트를 통해 로컬 호스트에서 패키지를 수락합니다.

```
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source
address="127.0.0.1" accept'
```

- **UDP** 연결을 제한하고 **111** 포트를 통해 **192.168.0.0/24** 호스트에서 패키지를 수락합니다.

```
# firewall-cmd --permanent --add-rich-rule='rule family="ipv4" port port="111"
protocol="udp" source address="192.168.0.0/24" invert="True" drop'
```

방화벽 설정을 영구적으로 설정하려면 방화벽 규칙을 추가할 때 **--permanent** 옵션을 사용합니다.

2. 방화벽을 다시 로드하여 새 규칙을 적용합니다.

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

검증 단계

- 방화벽 규칙을 나열합니다.

```
# firewall-cmd --list-rich-rule
rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24"
invert="True" drop
rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept
rule family="ipv4" port port="111" protocol="udp" source address="192.168.0.0/24"
invert="True" drop
```

추가 리소스

- **NFSv4** 전용 서버에 대한 자세한 내용은 **NFSv4** 전용 서버 구성 섹션을 참조하십시오.

- *firewalld 사용 및 구성*

9.2. NETNAMESPACE.MOUNTD 서비스 보안

rpc.mountd 데몬은 **NFS** 마운트 프로토콜의 서버 측을 구현합니다. **NFS** 마운트 프로토콜은 **NFS** 버전 2(RFC 9.14) 및 **NFS** 버전 3(RFC 1813)에서 사용됩니다.

서버에 방화벽 규칙을 추가하여 **rpc.mountd** 서비스를 보호할 수 있습니다. 모든 네트워크에 대한 액세스를 제한하고 방화벽 규칙을 사용하여 특정 예외를 정의할 수 있습니다.

사전 요구 사항

- **Net Namespace.mountd** 패키지가 설치되어 있어야 합니다.
- **firewalld** 패키지가 설치되어 서비스가 실행 중입니다.

절차

1. 서버에 방화벽 규칙을 추가합니다. 예를 들면 다음과 같습니다.

   - **192.168.0.0/24** 호스트에서 **mountd** 연결을 수락합니다.

     ```
     # firewall-cmd --add-rich-rule 'rule family="ipv4" service name="mountd" source
     address="192.168.0.0/24" invert="True" drop'
     ```

   - 로컬 호스트에서 **mountd** 연결을 수락합니다.

     ```
     # firewall-cmd --permanent --add-rich-rule 'rule family="ipv4" source address="127.0.0.1"
     service name="mountd" accept'
     ```

   방화벽 설정을 영구적으로 설정하려면 방화벽 규칙을 추가할 때 **--permanent** 옵션을 사용합니다.

2. 방화벽을 다시 로드하여 새 규칙을 적용합니다.

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

검증 단계

- 방화벽 규칙을 나열합니다.

```
# firewall-cmd --list-rich-rule
rule family="ipv4" service name="mountd" source address="192.168.0.0/24" invert="True"
drop
rule family="ipv4" source address="127.0.0.1" service name="mountd" accept
```

추가 리소스

- *firewalld 사용 및 구성*

9.3. NFS 서비스 보안

Kerberos를 사용하여 모든 파일 시스템 작업을 인증하고 암호화하여 네트워크 파일 시스템 버전 4(NFSv4)를 보호할 수 있습니다. NAT(Network Address Translation) 또는 방화벽으로 NFSv4를 사용하는 경우 /etc/default/nfs 파일을 수정하여 위임을 해제할 수 있습니다. 위임은 서버에서 파일 관리를 클라이언트에 위임하는 기술입니다. 반면 NFSv2 및 NFSv3에서는 파일을 잠금 및 마운트하는 데 Kerberos를 사용하지 않습니다.

NFS 서비스는 모든 버전의 NFS에서 TCP를 사용하여 트래픽을 전송합니다. 이 서비스는 RPCSEC_GSS 커널 모듈의 일부로 Kerberos 사용자 및 그룹 인증을 지원합니다.

NFS를 사용하면 원격 호스트가 네트워크를 통해 파일 시스템을 마운트하고 로컬에 마운트된 것처럼 해당 파일 시스템과 상호 작용할 수 있습니다. 중앙 집중식 서버의 리소스를 병합하고 파일 시스템을 공유할 때 /etc/nfsmount.conf 파일의 NFS 마운트 옵션을 추가로 사용자 지정할 수 있습니다.

9.3.1. NFS 서버 보안을 위한 내보내기 옵션

NFS 서버는 /etc/exports 파일에 있는 호스트로 내보낼 디렉터리 및 호스트의 목록 구조를 결정합니다.

주의

내보내기 파일의 구문에 있는 추가 공간은 구성이 크게 변경될 수 있습니다.

다음 예에서 **/tmp/nfs/** 디렉터리는 **bob.example.com** 호스트와 공유되며 읽기 및 쓰기 권한이 있습니다.

```
/tmp/nfs/     bob.example.com(rw)
```

다음 예제는 이전 예제와 동일하지만 읽기 전용 권한으로 **bob.example.com** 호스트에 동일한 디렉토리를 공유하며 호스트 이름 이후의 단일 공백 문자로 읽기 및 쓰기 권한으로 전 세계에 공유합니다.

```
/tmp/nfs/     bob.example.com (rw)
```

showmount -e <hostname> 명령을 입력하여 시스템의 공유 디렉터리를 확인할 수 있습니다.

/etc/exports 파일에서 다음 내보내기 옵션을 사용합니다.

주의

파일 시스템의 하위 디렉터리 내보내기는 안전하지 않으므로 전체 파일 시스템을 내보냅니다. 공격자는 부분적으로 내보낸 파일 시스템의 일부를 내보낼 수 있습니다.

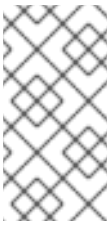
ro

ro 옵션을 사용하여 **NFS** 볼륨을 읽기 전용으로 내보냅니다.

rw

NFS 볼륨에서 읽기 및 쓰기 요청을 허용하려면 **rw** 옵션을 사용합니다. 쓰기 액세스를 허용하면

공격 위험이 증가하기 때문에 이 옵션을 주의해서 사용하십시오.



참고

rw 옵션을 사용하여 디렉터리를 마운트해야 하는 경우 가능한 위험을 줄이기 위해 모든 사용자가 쓸 수 없도록 쓰기 권한이 없는지 확인하십시오.

root_squash

root_squash 옵션을 사용하여 uid /gid 0의 요청을 anonymous uid/gid 에 매핑합니다. 이는 bin 사용자 또는 직원 그룹과 같이 동일하게 민감할 수 있는 다른 uid 또는 gids 에는 적용되지 않습니다.

no_root_squash

no_root_squash 옵션을 사용하여 루트 스쿼시를 끕니다. 기본적으로 NFS 공유는 root 사용자를 권한이 없는 사용자 계정인 nobody 사용자로 변경합니다. 이렇게 하면 생성된 모든 루트 파일의 소유자가 nobody 로 변경되어 setuid 비트 세트를 사용하여 프로그램을 업로드할 수 없습니다. no_root_squash 옵션을 사용하는 경우 원격 루트 사용자는 공유 파일 시스템에서 파일을 변경하고 트로이 목마에서 다른 사용자를 위해 감염된 애플리케이션을 유지할 수 있습니다.

보안

secure 옵션을 사용하여 예약된 포트로 내보내기를 제한합니다. 기본적으로 서버는 예약된 포트를 통해서만 클라이언트 통신을 허용합니다. 그러나 모든 사용자가 여러 네트워크의 클라이언트에서 root 사용자가 쉽게 될 수 있으므로 서버가 예약된 포트를 통한 통신의 권한이 있다고 가정하는 것은 거의 안전하지 않습니다. 따라서 예약된 포트에 대한 제한은 제한된 값입니다. Kerberos, 방화벽 및 특정 클라이언트에 대한 내보내기 제한에 의존하는 것이 좋습니다.

또한 NFS 서버를 내보낼 때 다음과 같은 모범 사례를 고려하십시오.

- *홈 디렉토리를 내보내는 것은 일부 애플리케이션이 일반 텍스트 또는 약하게 암호화된 형식으로 암호를 저장하므로 위험합니다. 애플리케이션 코드를 검토하고 개선하여 위험을 줄일 수 있습니다.*
- *일부 사용자는 SSH 키에 암호를 설정하지 않고 다시 홈 디렉터리에 대한 위험을 초래합니다. 암호를 사용하거나 Kerberos를 사용하여 이러한 위험을 줄일 수 있습니다.*
- *NFS 내보내기만 필수 클라이언트로 제한합니다. NFS 서버에서 showmount -e 명령을 사용하여 서버가 내보내는 내용을 검토합니다. 특히 필요하지 않은 항목을 내보내지 마십시오.*

- 불필요한 사용자가 서버에 로그인하여 공격 위험을 줄이는 것을 허용하지 마십시오. 주기적으로 서버에 액세스할 수 있는 사용자 및 항목을 확인할 수 있습니다.

추가 리소스

- Red Hat Identity Management 사용 시 Kerberos로 NFS 보안
- NFS 서버 구성.
- exports(5) 및 nfs(5) 도움말 페이지

9.3.2. NFS 클라이언트 보안을 위한 마운트 옵션

다음 옵션을 mount 명령에 전달하여 NFS 기반 클라이언트의 보안을 강화할 수 있습니다.

nosuid

nosuid 옵션을 사용하여 set-user-tekton 또는 set -group-tekton 비트를 비활성화합니다. 이렇게 하면 setuid 프로그램을 실행하여 원격 사용자가 더 높은 권한을 얻지 못하며, setuid 옵션과 반대로 이 옵션을 사용할 수 있습니다.

noexec

클라이언트의 모든 실행 파일을 비활성화하려면 noexec 옵션을 사용합니다. 이를 사용하여 사용자가 실수로 공유 파일 시스템에 저장된 파일을 실행하지 못하도록 합니다.

nodev

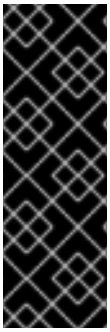
nodev 옵션을 사용하여 클라이언트의 장치 파일을 하드웨어 장치로 처리하지 않도록 합니다.

resvport

resvport 옵션을 사용하여 예약된 포트에 대한 통신을 제한하고 권한 있는 소스 포트를 사용하여 서버와 통신할 수 있습니다. 예약된 포트는 권한 있는 사용자 및 root 사용자와 같은 프로세스를 위해 예약되어 있습니다.

evera 일 수 있습니다.

NFS 서버의 second 옵션을 사용하여 마운트 지점의 파일에 액세스하기 위해 RPCGSS 보안 플레이버를 선택합니다. 유효한 보안 플레이버는 ,sys,tekton5, jenkinsfile5i, jenkinsfile 5p.입니다.



중요

jenkinsfile 5-libs 패키지에서 제공하는 **MIT Kerberos** 라이브러리는 새 배포에서 데이터 암호화 표준(DES) 알고리즘을 지원하지 않습니다. 보안 및 호환성으로 인해 **DES**는 **Kerberos** 라이브러리에서 기본적으로 더 이상 사용되지 않으며 비활성화되어 있습니다. 호환성을 이유로 환경에 **DES**가 필요한 경우가 아니면 **DES** 대신 최신의 보안 알고리즘을 사용하십시오.

추가 리소스

- 공통 **NFS** 마운트 옵션.

9.3.3. 방화벽을 사용하여 NFS 보안

NFS 서버에서 방화벽을 보호하려면 필요한 포트만 열어 둡니다. 다른 서비스에 **NFS** 연결 포트 번호를 사용하지 마십시오.

사전 요구 사항

- **nfs-utils** 패키지가 설치되어 있어야 합니다.
- **firewalld** 패키지가 설치되어 실행 중입니다.

절차

- **NFSv4**에서 방화벽이 **TCP** 포트 **2049** 를 열어야 합니다.
- **NFSv3**에서 **2049** 개의 추가 포트를 엽니다.

1. **rpcbind** 서비스는 **NFS** 포트를 동적으로 할당하므로 방화벽 규칙을 생성할 때 문제가 발생할 수 있습니다. 이 프로세스를 단순화하려면 **/etc/nfs.conf** 파일을 사용하여 사용할 포트를 지정합니다.

   a. **[mountd]** 섹션에 **port = <value>** 형식으로**mountd(mountd)**에 대해 **TCP** 및 **UDP** 포트를 설정합니다.

b. [statd] 섹션에 port =<value> 형식의 TCP 및 UDP 포트를 설정합니다.

2. /etc/nfs.conf 파일에서 NFS 잠금 관리자(nlockmgr)의 TCP 및 UDP 포트를 설정합니다.

a. [lockd] 섹션에서 nlockmgr (#189.statd)의 TCP 포트를 port=value 형식으로 설정합니다. 또는 /etc/modprobe.d/lockd.conf 파일에서 nlm_tcpport 옵션을 사용할 수 있습니다.

b. [lockd] 섹션에서 udp-port=value 형식의 nlockmgr (tekton.statd)에 대해 UDP 포트를 설정합니다. 또는 /etc/modprobe.d/lockd.conf 파일에서 nlm_udpport 옵션을 사용할 수 있습니다.

검증 단계

- NFS 서버의 활성 포트 및 RPC 프로그램을 나열합니다.

```
$ rpcinfo -p
```

추가 리소스

- Red Hat Identity Management 사용 시 Kerberos로 NFS 보안
- exports(5) 및 nfs(5) 도움말 페이지

9.4. FTP 서비스 보안

FTP(File Transfer Protocol)를 사용하여 네트워크를 통해 파일을 전송할 수 있습니다. 사용자 인증을 포함한 서버를 사용하는 모든 FTP 트랜잭션은 암호화되지 않으므로 안전하게 구성되었는지 확인해야 합니다.

RHEL 8에서는 두 개의 FTP 서버를 제공합니다.

- Red Hat Content Accelerator(tux) - FTP 기능이 포함된 커널 공간 웹 서버.

- *매우 안전한 **FTP** 데몬(**EgressIP**) - **FTP** 서비스의 독립형 보안 지향 구현입니다.*

*다음 보안 지침은 **vsftpd FTP** 서비스를 설정하기 위한 것입니다.*

9.4.1. FTP 시작 배너 보안

*사용자가 **FTP** 서비스에 연결하면 **FTP**는 기본적으로 공격자가 시스템의 약점을 식별하는 데 유용할 수 있는 버전 정보가 포함되어 있습니다. 공격자는 기본 배너를 변경하여 이 정보에 액세스하지 못하도록 할 수 있습니다.*

*단일 줄 메시지를 직접 포함하거나 다중 줄 메시지를 포함할 수 있는 별도의 파일을 참조하도록 **/etc/banners/ftp.msg** 파일을 편집하여 사용자 지정 배너를 정의할 수 있습니다.*

절차

- *단일 줄 메시지를 정의하려면 다음 옵션을 **/etc/tekton/ NetNamespace.conf** 파일에 추가합니다.*

```
ftpd_banner=Hello, all activity on ftp.example.com is logged.
```

- *별도의 파일에 메시지를 정의하려면 다음을 수행하십시오.*
  - *배너 메시지를 포함하는 **.msg** 파일을 만듭니다(예: **/etc/banners/ftp.msg**):*

```
########## Hello, all activity on ftp.example.com is logged. ##########
```

*여러 배너의 관리를 단순화하려면 모든 배너를 **/etc/banners/** 디렉토리에 배치합니다.*

  - *배너 파일의 경로를 **/etc/ NetNamespace/tekton.conf** 파일의 **banner_file** 옵션에 추가합니다.*

```
banner_file=/etc/banners/ftp.msg
```

검증

- 수정된 배너를 표시합니다.

```
$ ftp localhost
Trying ::1…
Connected to localhost (::1).
Hello, all activity on ftp.example.com is logged.
```

9.4.2. FTP에서 익명 액세스 및 업로드 방지

기본적으로 **vsftpd** 패키지를 설치하면 디렉터리에 대한 읽기 전용 권한이 있는 익명 사용자를 위한 **/var/ftp/** 디렉토리와 디렉터리 트리가 생성됩니다. 익명 사용자는 데이터에 액세스할 수 있으므로 중요한 데이터를 이러한 디렉터리에 저장하지 마십시오.

시스템의 보안을 강화하기 위해 익명 사용자가 특정 디렉터리에 파일을 업로드하고 익명 사용자가 데이터를 읽을 수 없도록 **FTP** 서버를 구성할 수 있습니다. 다음 절차에서 **anonymous** 사용자는 **root** 사용자가 소유한 디렉터리에 파일을 업로드할 수 있어야 하지만 변경하지 않아야 합니다.

절차

- **/var/ftp/pub/** 디렉토리에 **write-only** 디렉토리를 만듭니다.

```
# mkdir /var/ftp/pub/upload
# chmod 730 /var/ftp/pub/upload
# ls -ld /var/ftp/pub/upload
drwx-wx---. 2 root ftp 4096 Nov 14 22:57 /var/ftp/pub/upload
```

- 다음 행을 **/etc/tekton/tekton.conf** 파일에 추가합니다.

```
anon_upload_enable=YES
anonymous_enable=YES
```

- 선택 사항: 시스템에 **SELinux**를 활성화하고 강제 적용하는 경우 **SELinux** 부울 속성 **allow_ftpd_anon_write** 및 **allow_ftpd_full_access** 를 활성화합니다.

주의

익명 사용자가 디렉터리에서 읽고 쓸 수 있도록 허용하면 서버가 악의적인 소프트웨어의 리포지토리가 될 수 있습니다.

9.4.3. FTP용 사용자 계정 보안

FTP는 인증을 위해 안전하지 않은 네트워크를 통해 암호화되지 않은 사용자 이름과 암호를 전송합니다. 사용자 계정에서 서버에 대한 시스템 사용자 액세스를 거부하여 **FTP**의 보안을 강화할 수 있습니다.

구성에 해당하는 많은 다음 단계를 수행합니다.

절차

- 다음 행을 **/etc/tekton/tekton.conf** 파일에 추가하여 **EgressIP** 서버의 모든 사용자 계정을 비활성화합니다.

```
local_enable=NO
```

- **/etc/pam.d/ octets PAM** 구성 파일에 사용자 이름을 추가하여 특정 계정 또는 특정 계정 그룹(예: **root** 사용자 및 **sudo** 권한이 있는 사용자)에 대한 **FTP** 액세스를 비활성화합니다.

- **/etc/tekton/ftpusers** 파일에 사용자 계정을 추가하여 사용자 계정을 비활성화합니다.

9.4.4. 추가 리소스

- **ftpd_selinux(8)** 도움말 페이지

9.5. HTTP 서버 보안

9.5.1. httpd.conf의 보안 개선 사항

/etc/httpd/conf/httpd.conf 파일에서 보안 옵션을 구성하여 **Apache HTTP** 서버의 보안을 강화할 수 있습니다.

프로덕션에 저장하기 전에 항상 시스템에서 실행 중인 모든 스크립트가 올바르게 작동하는지 확인합니다.

*스크립트 또는 공통 게이트웨이 인터페이스(**CGI**)가 포함된 디렉터리에 대한 쓰기 권한만 있으면 됩니다. 쓰기 권한을 사용하여 디렉터리 소유권을 **root** 사용자로 변경하려면 다음 명령을 입력합니다.*

```
# chown root directory-name
# chmod 755 directory-name
```

***/etc/httpd/conf/httpd.conf** 파일에서 다음 옵션을 구성할 수 있습니다.*

FollowSymLinks

이 지시문은 기본적으로 활성화되어 있으며 디렉터리의 심볼릭 링크를 따릅니다.

Indexes

이 지시문은 기본적으로 활성화되어 있습니다. 사용자가 서버에서 파일을 검색하지 못하도록 이 지시문을 비활성화합니다.

UserDir

*이 지시문은 시스템에 사용자 계정이 있는지 확인할 수 있으므로 기본적으로 비활성화되어 있습니다. **/root/** 이외의 모든 사용자 디렉터리 검색을 활성화하려면 **UserDir enabled** 및 **UserDir disabled root** 지시문을 사용합니다. 비활성화된 계정 목록에 사용자를 추가하려면 **UserDir disabled** 줄에 공백으로 구분된 사용자 목록을 추가합니다.*

ServerTokens

이 지시문은 클라이언트에 다시 전송되는 서버 응답 헤더 필드를 제어합니다. 다음 매개변수를 사용하여 정보를 사용자 지정할 수 있습니다.

ServerTokens Full

*웹 서버 버전 번호, 서버 운영 체제 세부 정보, 설치된 **Apache** 모듈 등 사용 가능한 모든 정보를 제공합니다. 예를 들면 다음과 같습니다.*

```
Apache/2.4.37 (Red Hat Enterprise Linux) MyMod/1.2
```

ServerTokens Full-Release

릴리스 버전과 함께 사용 가능한 모든 정보를 제공합니다. 예를 들면 다음과 같습니다.

> *Apache/2.4.37 (Red Hat Enterprise Linux) (Release 41.module+el8.5.0+11772+c8e0c271)*

ServerTokens Prod / ServerTokens ProductOnly

웹 서버 이름을 제공합니다. 예를 들면 다음과 같습니다.

> *Apache*

ServerTokens Major

웹 서버 주요 릴리스 버전을 제공합니다. 예를 들면 다음과 같습니다.

> *Apache/2*

ServerTokens Minor

웹 서버 마이너 릴리스 버전을 제공합니다. 예를 들면 다음과 같습니다.

> *Apache/2.4*

ServerTokens Min / ServerTokens Minimal

웹 서버 최소 릴리스 버전을 제공합니다. 예를 들면 다음과 같습니다.

> *Apache/2.4.37*

ServerTokens OS

웹 서버 릴리스 버전과 운영 체제를 제공합니다. 예를 들면 다음과 같습니다.

> *Apache/2.4.37 (Red Hat Enterprise Linux)*

***ServerTokens Prod** 옵션을 사용하여 공격자가 시스템에 대한 중요한 정보를 얻는 위험을 줄일 수 있습니다.*

중요

IncludesNoExec 지시문을 제거하지 마십시오. 기본적으로 **Server Side Includes (SSI)** 모듈은 명령을 실행할 수 없습니다. 이를 변경하면 공격자가 시스템에서 명령을 입력할 수 있습니다.

httpd 모듈 제거

httpd 모듈을 제거하여 **HTTP** 서버의 기능을 제한할 수 있습니다. 이 작업을 수행하려면 **/etc/httpd/conf.modules.d/** 또는 **/etc/httpd/conf.d/** 디렉터리에서 구성 파일을 편집합니다. 예를 들어 프록시 모듈을 제거하려면 다음을 수행합니다.

```
echo '# All proxy modules disabled' > /etc/httpd/conf.modules.d/00-proxy.conf
```

추가 리소스

- **Apache HTTP** 서버
- **Apache HTTP** 서버에 대한 **SELinux** 정책 사용자 정의

9.5.2. Nginx 서버 구성 보안

Nginx는 고성능 **HTTP** 및 프록시 서버입니다. 다음 구성 옵션을 사용하여 **Nginx** 구성을 강화할 수 있습니다.

절차

- 버전 문자열을 비활성화하려면 **server_tokens** 구성 옵션을 수정합니다.

```
server_tokens off;
```

이 옵션은 서버 버전 번호와 같은 추가 세부 정보를 표시하지 않습니다. 이 구성은 **Nginx**에서 제공하는 모든 요청에 서버 이름만 표시합니다. 예를 들면 다음과 같습니다.

```
$ curl -sI http://localhost | grep Server
Server: nginx
```

- 특정 **/etc/nginx/ conf** 파일에서 알려진 특정 웹 애플리케이션 취약점을 완화하는 보안 헤더를 추가합니다.

- 예를 들어 **X- frame-Options** 헤더 옵션은 도메인 외부의 모든 페이지를 거부하여 **Nginx**에서 제공하는 콘텐츠를 프레임하고, 클릭재킹 공격을 완화합니다.

```
add_header X-Frame-Options "SAMEORIGIN";
```

- 예를 들어 **x-content-type** 헤더는 일부 이전 브라우저에서 **MIME-type** 스니핑을 방지합니다.

```
add_header X-Content-Type-Options nosniff;
```

- 예를 들어 **X-XSS-Protection** 헤더를 사용하면 **XSS(Cross-Site Scripting)** 필터링을 활성화하여 브라우저가 **Nginx**의 응답에 포함된 악성 콘텐츠를 렌더링하지 않도록 합니다.

```
add_header X-XSS-Protection "1; mode=block";
```

- 당신은 대중에 노출된 서비스를 제한하고 그들이하는 것을 제한하고, 예를 들어, 방문자로부터 수락 할 수 있습니다.

```
limit_except GET {
   allow 192.168.1.0/32;
   deny  all;
}
```

스니펫은 **GET** 및 **HEAD** 를 제외한 모든 메서드에 대한 액세스를 제한합니다.

- 예를 들어 **HTTP** 메서드를 비활성화할 수 있습니다.

```
# Allow GET, PUT, POST; return "405 Method Not Allowed" for all others.
if ( $request_method !~ ^(GET|PUT|POST)$ ) {
   return 405;
}
```

- **Nginx** 웹 서버에서 제공하는 데이터를 보호하도록 **SSL**을 구성할 수 있습니다. **HTTPS**를 통해서만 서비스를 제공하는 것이 좋습니다. 또한 **Mozilla SSL** 구성 생성기를 사용하여 **Nginx** 서버에서 **SSL**을 활성화하기 위한 보안 구성 프로필을 생성할 수 있습니다. 생성된 구성을 사용하면 알려진 취약한 프로토콜(예: **SSLv2** 및 **SSLv3**), 암호 및 해시 알고리즘(예: **3DES** 및 **MD5**)이 비활성화됩니다. **SSL** 서버 테스트를 사용하여 구성이 최신 보안 요구 사항을 충족하는지 확인할 수도 있습니다.

추가 리소스

- Mozilla SSL 구성 생성기
- SSL 서버 테스트

9.6. 인증된 로컬 사용자에 대한 액세스 제한으로 POSTGRESQL 보안

PostgreSQL은 오브젝트 관계 데이터베이스 관리 시스템(DBMS)입니다. Red Hat Enterprise Linux에서 PostgreSQL은 postgresql-server 패키지에서 제공합니다.

클라이언트 인증을 구성하여 공격 위험을 줄일 수 있습니다. 데이터베이스 클러스터의 데이터 디렉터리에 저장된 pg_hba.conf 구성 파일은 클라이언트 인증을 제어합니다. 호스트 기반 인증을 위한 PostgreSQL을 구성하려면 절차를 따르십시오.

절차

1. PostgreSQL 설치:

```
# yum install postgresql-server
```

2. 다음 옵션 중 하나를 사용하여 데이터베이스 스토리지 영역을 초기화합니다.

   a. initdb 유틸리티 사용:

```
$ initdb -D /home/postgresql/db1/
```

   d 옵션을 사용하는 initdb 명령은 지정한 디렉터리가 아직 존재하지 않는 경우(예: /home/postgresql/db1/)를 생성합니다. 그러면 이 디렉토리에는 데이터베이스에 저장된 모든 데이터와 클라이언트 인증 구성 파일도 포함됩니다.

   b. postgresql-setup 스크립트 사용:

```
$ postgresql-setup --initdb
```

   기본적으로 스크립트는 /var/lib/pgsql/data/ 디렉터리를 사용합니다. 이 스크립트는 시

스템 관리자가 기본 데이터베이스 클러스터 관리를 지원합니다.

3. 인증된 로컬 사용자가 사용자 이름을 사용하여 모든 데이터베이스에 액세스할 수 있도록 하려면 **pg_hba.conf** 파일에서 다음 행을 수정합니다.

```
local   all             all                                     trust
```

이 문제는 데이터베이스 사용자를 생성하고 로컬 사용자가 없는 계층화된 애플리케이션을 사용하는 경우 문제가 될 수 있습니다. 시스템의 모든 사용자 이름을 명시적으로 제어하지 않으려면 **pg_hba.conf** 파일에서 로컬 행 항목을 제거합니다.

4. 데이터베이스를 다시 시작하여 변경 사항을 적용합니다.

```
# systemctl restart postgresql
```

이전 명령은 데이터베이스를 업데이트하고 구성 파일의 구문도 확인합니다.

9.7. MEMCACHED 서비스 보안

Memcached는 고성능 분산 메모리 개체 캐싱 시스템입니다. 데이터베이스 로드를 줄여 동적 웹 애플리케이션의 성능을 향상시킬 수 있습니다.

Memcached는 데이터베이스 호출, **API** 호출 또는 페이지 렌더링 결과에 따른 문자열 및 오브젝트와 같은 작은 임의 데이터의 작은 청크를 위한 메모리 내 키-값 저장소입니다. **Memcached**를 사용하면 활용도가 낮은 영역에서 메모리를 더 많은 메모리가 필요한 애플리케이션에 할당할 수 있습니다.

2018년, 다수의 인터넷에 노출된 **Memcached** 서버를 악용하여 **DDoS** 수정 공격의 취약점이 발견되었다. 이러한 공격은 전송에 **UDP** 프로토콜을 사용하는 **Memcached** 통신을 활용합니다. 이 공격은 몇 백 바이트 크기의 요청이 몇 메가바이트 또는 수백 메가바이트의 크기를 생성 할 수있는 높은 변동 비율 때문에 효과적이었습니다.

대부분의 경우 **memcached** 서비스를 공용 인터넷에 노출할 필요가 없습니다. 이러한 노출은 자체 보안 문제가 발생할 수 있으므로 원격 공격자가 **Memcached**에 저장된 정보를 유출하거나 수정할 수 있습니다.

이 섹션을 따라 가능한 **DDoS** 공격에 대해 **Memcached** 서비스를 사용하여 시스템을 강화할 수 있습니다.

9.7.1. jenkinsfile에 대해 Memcached 강화

보안 위험을 완화하려면 구성에 적용할 수 있는 다음 여러 단계를 수행하십시오.

절차

- LAN에서 방화벽을 구성합니다. Memcached 서버에 로컬 네트워크에서만 액세스할 수 있는 경우 memcached 서비스에서 사용하는 포트로 외부 트래픽을 라우팅하지 마십시오. 예를 들어, 허용되는 포트 목록에서 기본 포트 11211 을 제거합니다.

```
# firewall-cmd --remove-port=11211/udp
# firewall-cmd --runtime-to-permanent
```

- 애플리케이션과 동일한 시스템에서 단일 Memcached 서버를 사용하는 경우 memcached 를 설정하여 localhost 트래픽만 청취합니다. /etc/sysconfig/memcached 파일에서 OPTIONS 값을 수정합니다.

```
OPTIONS="-l 127.0.0.1,::1"
```

- SASL(Simple Authentication and Security Layer) 인증을 활성화합니다.

1. /etc/sasl2/memcached.conf 파일을 수정하거나 추가합니다.

```
sasldb_path: /path.to/memcached.sasldb
```

2. SASL 데이터베이스에 계정을 추가합니다.

```
# saslpasswd2 -a memcached -c cacheuser -f /path.to/memcached.sasldb
```

3. memcached 사용자 및 그룹에 대해 데이터베이스에 액세스할 수 있는지 확인합니다.

```
# chown memcached:memcached /path.to/memcached.sasldb
```

4. /etc/sysconfig/memcached 파일의 OPTIONS 매개변수에 -S 값을 추가하여 Memcached에서 SASL 지원을 활성화합니다.

-

```
OPTIONS="-S"
```

5. ***Memcached** 서버를 다시 시작하여 변경 사항을 적용합니다.*

```
# systemctl restart memcached
```

6. ***SASL** 데이터베이스에서 생성된 사용자 이름과 암호를 애플리케이션의 **Memcached** 클라이언트 구성에 추가합니다.*

- ***TLS**를 사용하여 **Memcached** 클라이언트 및 서버 간 통신을 암호화합니다.*

1. */etc/sysconfig/memcached* *파일의 **OPTIONS** 매개변수에 **-Z** 값을 추가하여 **TLS**를 사용하여 **Memcached** 클라이언트와 서버 간의 암호화된 통신을 활성화합니다.*

```
OPTIONS="-Z"
```

2. ***-o ssl_chain_cert** 옵션을 사용하여 **PEM** 형식으로 인증서 체인 파일 경로를 추가합니다.*

3. ***-o ssl_key** 옵션을 사용하여 개인 키 파일 경로를 추가합니다.*