Red Hat Enterprise Linux 9

네트워크 보안

보안 네트워크 및 네트워크 통신 구성

Last Updated: 2024-08-06

Red Hat Enterprise Linux 9 네트워크 보안

보안 네트워크 및 네트워크 통신 구성

법적 공지

초록

네트워크 보안을 개선하고 데이터 유출 및 침입의 위험을 낮추는 도구와 기술을 알아보십시오.

차례

RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

Jira를 통해 피드백 제출 (등록 필요)

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

1장. 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 설치될 때 또는 호스트가 처음 부팅될 때 자동으로 생성되는 암호화 키입니다.

OpenSSH는 Linux, UNIX 및 유사한 운영 체제에서 지원하는 SSH 프로토콜 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 파일 복사 프로그램입니다.
- **SFTP** 는 안전한 파일 전송 프로그램입니다.
- **SSH-agent** 는 개인 키를 캐싱하기 위한 인증 에이전트입니다.
- **ssh-add** 는 **ssh-agent** 에 개인 키 ID를 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.

참고

RHEL 9에서는 기본적으로 SFTP(Secure File Transfer Protocol)가 SSH 파일 전송 프로토콜(SCP)으로 교체됩니다. 이는 SCP가 이미 보안 문제를 발생했기 때문입니다(예: CVE-2020-15778).

사용자의 시나리오에서 SFTP를 사용할 수 없거나 호환되지 않는 경우 **-O** 옵션을 사용하여 원래 SCP/RCP 프로토콜을 강제로 사용할 수 있습니다.

자세한 내용은 Red Hat Enterprise Linux 9에서 OpenSSH SCP 프로토콜 사용 중단 문서를 참조하십시오.

RHEL의 OpenSSH 제품군은 SSH 버전 2만 지원합니다. 이전 버전 1에서 알려진 악용에 취약하지 않은 향상된 키 교환 알고리즘이 있습니다.

OpenSSH는 RHEL의 핵심 암호화 하위 시스템 중 하나로 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 클라이언트 프로그램(즉, **ssh**,**scp**, **sftp**)과 서버(**sshd** 데몬)에 대한 두 가지 구성 파일 세트를 사용합니다.

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 나열된 도움말 페이지
- 시스템 전체 암호화 정책 사용

1.2. OPENSSH 서버 구성 및 시작

OpenSSH 서버의 시작 배너 및 IP 주소를 변경할 수 있습니다. 느린 동적 네트워크 구성으로 전환할 수도 있습니다.

기본 RHEL 설치 후 **sshd** 데몬이 이미 시작되고 서버 호스트 키가 자동으로 생성됩니다.

사전 요구 사항

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

절차

1. **sshd** 서비스가 아직 실행되지 않은 경우 현재 세션에서 이를 시작하고 부팅 시 자동으로 시작되도록 설정합니다.

```
# systemctl enable --now 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 시나리오 요구 사항을 충족하는지 검토합니다.

4. 선택적으로 **/etc/issue** 파일을 편집하여 클라이언트를 인증하기 전에 OpenSSH 서버가 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
```

새로운 기본 설치 이외의 시스템에서 **PubkeyAuthentication no** 가 설정되지 않았으며 **Kbd interactiveAuthentication** 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

로컬 시스템에서 SSH 키 쌍을 생성하고 생성된 공개 키를 OpenSSH 서버에 복사하여 암호를 제공하지 않고 OpenSSH 서버에 로그인할 수 있습니다. 이 옵션을 허용하도록 서버를 구성해야 합니다.

사전 요구 사항

- OpenSSH 서버에 연결할 Linux 사용자로 로그인되어 있습니다. **root** 로 다음 단계를 완료하면 **root** 만 키를 사용할 수 있습니다.

절차

1. SSH 프로토콜의 버전 2에 대한 ECDSA 키 쌍을 생성합니다.

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

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 복사합니다.

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

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

&lt ***;username> @ <ssh-server-example.com*** >을 사용자 인증 정보로 바꿉니다.

세션에서 **ssh-agent** 프로그램을 사용하지 않는 경우 이전 명령은 가장 최근에 수정된 **~/.ssh/id*.pub** 공개 키가 아직 설치되지 않은 경우 공개 키를 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.

3. 선택 사항: 이전에 생성된 키 쌍을 시스템의 재설치 사이에 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 <username>@<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

OpenSSH 클라이언트의 스마트 카드에 저장된 RSA 및 ECDSA 키를 사용할 수 있습니다. 스마트 카드를 사용한 인증은 기본 암호 인증을 대체합니다.

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

OpenSSH를 사용할 때 보안을 강화하기 위해 시스템을 조정할 수 있습니다.

/etc/ssh/sshd_config OpenSSH 구성 파일의 변경 사항을 적용하려면 **sshd** 데몬을 다시 로드해야 합니다.

```
# systemctl reload sshd
```

주의

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

비보안 연결 프로토콜 비활성화

- SSH를 효과적으로 사용하려면 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** 유틸리

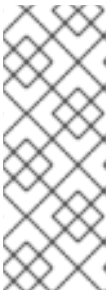
티를 사용하여 OpenSSH 서버의 클라이언트에서 공개 키를 복사합니다. OpenSSH 서버에서 암호 기반 인증을 비활성화하려면 **/etc/ssh/sshd_config**를 편집하고 **PasswordAuthentication** 옵션을 **no**로 변경합니다.

```
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 힘에서 RSA보다 더 나은 성능을 제공합니다. 또한 더 짧은 키를 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
```



참고

cloud-init 가 활성화된 이미지에서 **ssh-keygen** 단위가 자동으로 비활성화됩니다. 이는 **ssh-keygen 템플릿** 서비스가 **cloud-init** 툴을 방해하고 호스트 키 생성에 문제가 발생할 수 있기 때문입니다. 이러한 문제를 방지하려면 **cloud-init** 가 실행 중인 경우 **etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** 드롭인 구성 파일에서 **ssh-keygen** 장치를 비활성화합니다.

- 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
```

중요

Ed25519 알고리즘은 FIPS-140과 호환되지 않으며 OpenSSH는 FIPS 모드에서 Ed25519 키와 작동하지 않습니다.

기본값 이외의 포트

- 기본적으로 **sshd** 데몬은 TCP 포트 22에서 수신 대기합니다. 포트를 변경하면 자동화된 네트워크 스캔을 기반으로 시스템이 공격에 노출되는 것을 줄이며 비만에 의해 보안이 향상됩니다.
  /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 --remove-port=22/tcp
# firewall-cmd --runtime-to-permanent
```

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

Root 로그인

- **PermitRootLogin**은 기본적으로 **prohibit-password** 로 설정됩니다. 이렇게 하면 root로 로그인하는 데 암호를 사용하는 대신 키 기반 인증을 사용하고 무차별 강제 공격을 방지하여 위험을 줄입니다.

주의

관리자가 권한이 있는 명령을 실행하는 사용자를 감사할 수 없기 때문에 root 사용자로 로그인 활성화는 안전한 방법이 아닙니다. 관리 명령을 사용하려면 로그인하고 **sudo**를 대신 사용합니다.

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/50-redhat.conf** 파일의 **ForwardX11Trusted** 옵션은 **yes** 로 설정되고 **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.

  OpenSSH 서버는 **/etc/ssh/sshd_config** 의 모든 Allow 및 Deny 지시문을 전달하는 연결만 허용합니다. 예를 들어 **AllowUsers** 지시문이 **AllowGroups** 지시문에 나열된 그룹에 포함되지 않은 사용자를 나열하는 경우 사용자는 로그인할 수 없습니다.

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
```

주의

시스템이 인터넷에서 통신할 경우 **FUTURE** 정책의 엄격한 설정으로 인해 상호 운용성 문제에 직면할 수 있습니다.

시스템 전체 암호화 정책을 통해 SSH 프로토콜의 특정 암호만 비활성화할 수도 있습니다. 자세한 내용은 보안 강화 문서의 하위 정책을 사용하여 시스템 전체 암호화 정책 사용자 지정 섹션을 참조하십시오.

OpenSSH 서버에 대한 시스템 전체 암호화 정책을 비활성화하려면 /etc/ssh/sshd_config.d/ 디렉터리에 있는 드롭인 구성 파일에 암호화 정책을 지정하려면 **/etc/ssh/sshd_config.d/** 디렉터리에 있는 **암호화 정책을 지정하여 두 자리 숫자 접두사가 50-redhat.conf 파일 앞에 있고 .conf 접미사를 .conf** 접미사로 지정합니다.

자세한 내용은 **sshd_config(5)** 도움말 페이지를 참조하십시오.

OpenSSH 클라이언트에 대한 시스템 전체 암호화 정책을 비활성화하려면 다음 작업 중 하나를 수행합니다.

- 지정된 사용자의 경우 **~/.ssh/ config 파일의 사용자별 구성으로 글로벌 ssh_ config** 를 재정의합니다.
- 전체 시스템의 경우 **/etc/ssh/ssh_config.d/** 디렉터리에 있는 드롭인 구성 파일에 암호화 정책을 지정하고 두 자리 숫자 접두사가 **50-redhat.conf** 파일 앞에 있고 **.conf** 접미사(예: **49-crypto-policy-override.conf**)를 사용합니다.

추가 리소스

- **sshd_config(5)**, **ssh-keygen(1)**, **crypto-policies(7)** 및 **update-crypto-policies(8)** 도움말 페이지
- 보안 강화 문서에서 시스템 전체 암호화 정책 사용 .
- ssh 서비스에 대한 특정 알고리즘 및 암호를 비활성화하는 방법만 문서입니다.

1.7. SSH 건너뛰기 호스트를 사용하여 원격 서버에 연결

건너뛰기 호스트라고도 하는 중간 서버를 통해 로컬 시스템을 원격 서버에 연결할 수 있습니다.

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.

4. 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.example
.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. SSH 시스템 역할과 보안 통신 구성

관리자는 **sshd** 시스템 역할을 사용하여 SSH 서버와 **ssh** 시스템 역할을 구성하여 Ansible Core 패키지를 사용하여 여러 RHEL 시스템에서 동시에 SSH 클라이언트를 일관되게 구성할 수 있습니다.

1.9.1. sshd RHEL 시스템 역할의 변수

sshd 시스템 역할 플레이북에서는 환경 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 ::
```

추가 리소스

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

1.9.2. sshd RHEL 시스템 역할을 사용하여 OpenSSH 서버 구성

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

참고

SSH 및 SSHD 구성을 변경하는 다른 RHEL 시스템 역할(예: Identity Management RHEL 시스템 역할)과 함께 **sshd** RHEL 시스템 역할을 사용할 수 있습니다. 구성을 덮어쓰지 않으려면 **sshd** 역할에서 네임스페이스(RHEL 8 및 이전 버전) 또는 드롭인 디렉터리(RHEL 9)를 사용하는지 확인합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

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

```
---
- name: SSH server configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure sshd to prevent root and password login except from particular subnet
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd:
          PermitRootLogin: no
```

```
      PasswordAuthentication: no
      Match:
        - Condition: "Address 192.0.2.0/24"
          PermitRootLogin: yes
          PasswordAuthentication: yes
```

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.

2. 플레이북 구문을 확인합니다.

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. Playbook을 실행합니다.

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

검증

1. SSH 서버에 로그인합니다.

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

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

```
$ cat /etc/ssh/sshd_config.d/00-ansible_system_role.conf
#
# Ansible managed
#
PasswordAuthentication no
PermitRootLogin no
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@<ssh_server>
   ```

추가 리소스

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

1.9.3. ssh RHEL 시스템 역할의 변수

ssh 시스템 역할 플레이북에서는 환경 설정 및 제한 사항에 따라 클라이언트 SSH 구성 파일의 매개 변수를 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.

추가 리소스

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

1.9.4. ssh RHEL 시스템 역할을 사용하여 OpenSSH 클라이언트 구성

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

참고

SSH 및 SSHD 구성을 변경하는 다른 시스템 역할(예: Identity Management RHEL 시스템 역할)과 함께 **ssh** RHEL 시스템 역할을 사용할 수 있습니다. 구성을 덮어쓰지 않도록 하려면 **ssh** 역할에 드롭인 디렉터리(RHEL 8 이상에서 기본값)를 사용해야 합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

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

```
---
- name: SSH client configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: "Configure ssh clients"
      ansible.builtin.include_role:
        name: rhel-system-roles.ssh
      vars:
        ssh_user: root
        ssh:
          Compression: true
          GSSAPIAuthentication: no
          ControlMaster: auto
          ControlPath: ~/.ssh/.cm%C
          Host:
            - Condition: example
              Hostname: server.example.com
              User: user1
        ssh_ForwardX11: no
```

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

- 압축이 활성화됩니다.
- ControlMaster 멀티플렉싱이 **auto** 로 설정되어 있습니다.
- **server. example.com** 호스트에 연결하는 예제 별칭은 **user1** 입니다.
- **예제** 호스트 별칭이 생성되며, 이 별칭은 **user1** 사용자 이름으로 을 호스팅하는 **server.example.com** 에 대한 연결을 나타냅니다.
- X11 전달이 비활성화되어 있습니다.

2. 플레이북 구문을 확인합니다.

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. Playbook을 실행합니다.

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

검증

- SSH 구성 파일을 표시하여 관리 노드에 올바른 구성이 있는지 확인합니다.

```
# cat ~/root/.ssh/config
# Ansible managed
Compression yes
```

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

추가 리소스

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

1.9.5. 비독점 구성에 sshd RHEL 시스템 역할 사용

일반적으로 **sshd** 시스템 역할을 적용하면 전체 구성을 덮어씁니다. 이전에 구성을 조정한 경우(예: 다른 시스템 역할 또는 플레이북)를 사용하는 경우 문제가 있을 수 있습니다. 다른 옵션을 그대로 유지하면서 선택한 구성 옵션에 대해서만 **sshd** 시스템 역할을 적용하려면 제외된 구성을 사용할 수 있습니다.

포함되지 않은 구성을 적용할 수 있습니다.

- 구성 스니펫을 사용하여 RHEL 8 및 이전 버전에서.
- 드롭인 디렉터리에서 파일을 사용하여 RHEL 9 이상에서 다음을 수행합니다. 기본 구성 파일은 드롭인 디렉터리에 이미 **/etc/ssh/sshd_config.d/00-ansible_system_role.conf** 로 배치되어 있습니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.
   - RHEL 8 이상을 실행하는 관리형 노드의 경우:

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

```
        LANG
        LS_COLORS
        EDITOR
```

- RHEL 9 이상을 실행하는 관리형 노드의 경우:

```
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure sshd to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
        sshd:
          # Environment variables to accept
          AcceptEnv:
            LANG
            LS_COLORS
            EDITOR
```

sshd_config_file 변수에서 **sshd** 시스템 역할이 구성 옵션을 쓰는 **.conf** 파일을 정의합니다. 두 자리 접두사(예: ***42-***)를 사용하여 구성 파일을 적용할 순서를 지정합니다.

2. 플레이북 구문을 확인합니다.

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. Playbook을 실행합니다.

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

검증

- SSH 서버의 구성을 확인합니다.
  - RHEL 8 이상을 실행하는 관리형 노드의 경우:

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

  - RHEL 9 이상을 실행하는 관리형 노드의 경우:

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

추가 리소스

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

1.10. 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
- 비표준 구성을 사용하여 애플리케이션 및 서비스에 대한 SELinux 구성
- firewalld를 사용하여 네트워크 트래픽 제어

2장. TLS 키 및 인증서 생성 및 관리

TLS(Transport Layer Security) 프로토콜을 사용하여 두 시스템 간에 전송된 통신을 암호화할 수 있습니다. 이 표준은 개인 및 공개 키, 디지털 서명 및 인증서와 함께ECDHE 암호화를 사용합니다.

2.1. TLS 인증서

TLS(Transport Layer Security)는 클라이언트-서버 애플리케이션이 정보를 안전하게 전달할 수 있도록 하는 프로토콜입니다. TLS는 공개 및 개인 키 쌍의 시스템을 사용하여 클라이언트와 서버 간에 전송된 통신을 암호화합니다. TLS는 SSL(Secure Sockets Layer)의 후속 프로토콜입니다.

TLS는 X.509 인증서를 사용하여 호스트 이름 또는 조직과 같은 ID를 디지털 서명을 사용하여 공개 키에 바인딩합니다. X.509는 공개 키 인증서의 형식을 정의하는 표준입니다.

보안 애플리케이션의 인증은 애플리케이션 인증서의 공개 키 값의 무결성에 따라 달라집니다. 공격자가 공개 키를 자체 공개 키로 교체하면 실제 애플리케이션을 가장하고 데이터 보안 액세스 권한을 얻을 수 있습니다. 이러한 유형의 공격을 방지하려면 모든 인증서에 CA(인증 기관)의 서명이 있어야 합니다. CA는 인증서의 공개 키 값의 무결성을 확인하는 신뢰할 수 있는 노드입니다.

CA는 디지털 서명을 추가하고 인증서를 발행하여 공개 키에 서명합니다. 디지털 서명은 CA의 개인 키로 인코딩되는 메시지입니다. CA의 공개 키는 CA 인증서를 배포하여 애플리케이션에서 사용할 수 있습니다. 애플리케이션은 CA의 디지털 서명을 CA의 공개 키로 디코딩하여 인증서가 유효하게 서명되었는지 확인합니다.

CA에서 서명한 인증서를 사용하려면 공개 키를 생성하여 서명을 위해 CA로 보내야 합니다. 이를 CSR(인증서 서명 요청)이라고 합니다. CSR에는 인증서에 대한 고유 이름(DN)도 포함됩니다. 두 가지 유형의 인증서에 제공할 수 있는 DN 정보에는 해당 국가의 경우 2자리 국가 코드, 주/도, 시/도, 도시 이름, 조직 이름, 이메일 주소 등이 포함될 수 있습니다. 현재 많은 상용 CA는 주체 대체 이름 확장을 선호하고 CSR의 DN을 무시합니다.

RHEL은 TLS 인증서 작업을 위한 두 가지 주요 툴킷을 제공합니다. gnutls 및 OpenSSL. openssl 패키지에서 **openssl** 유틸리티를 사용하여 인증서를 생성, 읽기, 서명 및 확인할 수 있습니다. **gnutls-utils** 패키지에서 제공하는 **certtool** 유틸리티는 다른 구문과 백엔드의 다양한 라이브러리 세트를 사용하여 동일한 작업을 수행할 수 있습니다.

추가 리소스

- RFC 5280: Internet X.509 공개 키 인프라 인증서 및 인증서 해지 목록 (CRL) 프로필
- **OpenSSL(1)**, **x509(1)**, **ca(1)**, **req(1)** 및 **certtool(1)** 매뉴얼 페이지

2.2. OPENSSL을 사용하여 개인 CA 생성

개인 인증 기관(CA)은 시나리오가 내부 네트워크 내의 엔티티를 확인해야 하는 경우에 유용합니다. 예를 들어, 컨트롤에서 서명된 인증서 또는 상용 CA를 구입하지 않으려면 개인 CA를 사용하여 인증으로 VPN 게이트웨이를 만들 때 사용합니다. 이러한 사용 사례의 인증서에 서명하기 위해 개인 CA는 자체 서명된 인증서를 사용합니다.

사전 요구 사항

- **sudo** 를 사용하여 관리자 명령을 입력할 수 있는 **루트** 권한 또는 권한이 있어야 합니다. 이러한 권한이 필요한 명령은 **#** 으로 표시됩니다.

절차

1. CA의 개인 키를 생성합니다. 예를 들어 다음 명령은 256비트 Elliptic Curve Digital Signature Algorithm(ECDSA) 키를 생성합니다.

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

키 생성 프로세스의 시간은 호스트의 하드웨어 및 엔트로피, 선택한 알고리즘 및 키 길이에 따라 달라집니다.

2. 이전 명령에서 생성된 개인 키를 사용하여 서명된 인증서를 생성합니다.

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

생성된 **ca.crt** 파일은 10년 동안 다른 인증서에 서명하는 데 사용할 수 있는 자체 서명된 CA 인증서입니다. 프라이빗 CA의 경우 <ECDHE CA>를 CN(일반 이름)으로 임의의 문자열로 교체할 수 있습니다.

3. *CA의 개인 키에 대해 보안 권한을 설정합니다. 예를 들면 다음과 같습니다.*

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

다음 단계

- *자체 서명된 CA 인증서를 클라이언트 시스템의 신뢰 앵커로 사용하려면 CA 인증서를 클라이언트에 복사하여 클라이언트의 시스템 전체 신뢰 저장소에 **root** 로 추가합니다.*

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

자세한 내용은 3 장. 공유 시스템 인증서 사용 를 참조하십시오.

검증

1. *CSR(인증서 서명 요청)을 생성하고 CA를 사용하여 요청에 서명합니다. CA는 CSR을 기반으로 인증서를 성공적으로 생성해야 합니다. 예를 들면 다음과 같습니다.*

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

자세한 내용은 2.5 절. "개인 CA를 사용하여 OpenSSL과 함께 CSR의 인증서 발행" 를 참조하십시오.

2. *자체 서명된 CA에 대한 기본 정보를 표시합니다.*

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

```
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
…
```

3. *개인 키의 일관성을 확인합니다.*

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

18wRpXoCVC9vcPki7WIT+gnmCI+hRANCAARb9NxIvkaVjFhOoZbGp/HtIQxbM78E
lwbDP0BI624xBJ8gK68ogSaq2x4SdezFdV1gNeKScDcU+Pj2pELldmdF
-----END PRIVATE KEY-----
```

추가 리소스

- ***openssl(1)***, ***ca(1)***, ***genpkey(1)***, ***x509(1)***, and ***req(1)*** man pages

2.3. OPENSSL 을 사용하여 TLS 서버 인증서의 개인 키와 CSR 생성

TLS 암호화 통신 채널은 CA(인증 기관)의 유효한 TLS 인증서가 있는 경우에만 사용할 수 있습니다. 인증서를 받으려면 먼저 서버의 개인 키와 CSR(인증서 서명 요청)을 만들어야 합니다.

절차

1. *서버 시스템에서 개인 키를 생성합니다. 예를 들면 다음과 같습니다.*

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

2. *선택 사항: 선택한 텍스트 편집기를 사용하여 CSR 생성을 간소화하는 구성 파일을 준비합니다. 예를 들면 다음과 같습니다.*

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

[req]
distinguished_name = dn
prompt = no

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

[alt_name]
DNS.1 = <example.com>
DNS.2 = <server.example.com>
```

```
IP.1 = <192.168.0.1>
IP.2 = <::1>
IP.3 = <127.0.0.1>
```

extendedKeyUsage = serverAuth 옵션은 인증서 사용을 제한합니다.

3. 이전에 생성한 개인 키를 사용하여 CSR을 생성합니다.

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

-config 옵션을 생략하면 **req** 유틸리티에서 추가 정보를 입력하라는 메시지를 표시합니다. 예를 들면 다음과 같습니다.

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

다음 단계

- 서명하기 위해 선택한 CA에 CSR을 제출합니다. 또는 신뢰할 수 있는 네트워크 내의 내부 사용 시나리오의 경우 개인 CA를 사용하여 서명합니다. 자세한 내용은 2.5절. "개인 CA를 사용하여 OpenSSL과 함께 CSR의 인증서 발행"를 참조하십시오.

검증

1. CA에서 요청된 인증서를 가져온 후 인증서의 사람이 읽을 수 있는 부분이 요구 사항과 일치하는지 확인합니다.

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

```
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

추가 리소스

- ***openssl(1)***, ***x509(1)***, ***genpkey(1)***, ***req(1)***, and ***config(5)*** man pages

2.4. OPENSSL 을 사용하여 TLS 클라이언트 인증서의 개인 키와 CSR 생성

TLS 암호화 통신 채널은 CA(인증 기관)의 유효한 TLS 인증서가 있는 경우에만 사용할 수 있습니다. 인증서를 받으려면 먼저 클라이언트의 개인 키와 CSR(인증서 서명 요청)을 생성해야 합니다.

절차

1. *클라이언트 시스템에서 개인 키를 생성합니다. 예를 들면 다음과 같습니다.*

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

2. *선택 사항: 선택한 텍스트 편집기를 사용하여 CSR 생성을 간소화하는 구성 파일을 준비합니다. 예를 들면 다음과 같습니다.*

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

[req]
distinguished_name = dn
prompt = no

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

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

extendedKeyUsage = clientAuth *옵션은 인증서 사용을 제한합니다.*

3. *이전에 생성한 개인 키를 사용하여 CSR을 생성합니다.*

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

-config *옵션을 생략하면* ***req*** *유틸리티에서 추가 정보를 입력하라는 메시지를 표시합니다. 예를 들면 다음과 같습니다.*

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
…
```

```
Common Name (eg, your name or your server's hostname) []: <client.example.com>
Email Address []: <client@example.com>
```

다음 단계

- *서명하기 위해 선택한 CA에 CSR을 제출합니다. 또는 신뢰할 수 있는 네트워크 내의 내부 사용 시나리오의 경우 개인 CA를 사용하여 서명합니다. 자세한 내용은 2.5절. "개인 CA를 사용하여 OpenSSL과 함께 CSR의 인증서 발행"를 참조하십시오.*

검증

1. *사용자가 읽을 수 있는 인증서 부분이 요구 사항과 일치하는지 확인합니다. 예를 들면 다음과 같습니다.*

```
$ openssl x509 -text -noout -in <client-cert.crt>
Certificate:
…
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Subject Alternative Name:
                email:client@example.com
…
```

추가 리소스

- ***openssl(1)***, ***x509(1)***, ***genpkey(1)***, ***req(1)***, and ***config(5)*** man pages

2.5. 개인 CA를 사용하여 OPENSSL과 함께 CSR의 인증서 발행

시스템이 TLS 암호화 통신 채널을 설정할 수 있도록 하려면 CA(인증 기관)에서 유효한 인증서를 제공해야 합니다. 개인 CA가 있는 경우 시스템에서 인증서 서명 요청(CSR)에 서명하여 요청된 인증서를 생성할 수 있습니다.

사전 요구 사항

- *이미 개인 CA가 구성되어 있습니다. 자세한 내용은 2.2절. "OpenSSL을 사용하여 개인 CA 생성"를 참조하십시오.*
- *CSR이 포함된 파일이 있습니다. 2.3절. "OpenSSL을 사용하여 TLS 서버 인증서의 개인 키와 CSR 생성"에서 CSR 생성의 예를 확인할 수 있습니다.*

절차

1. *선택 사항: 선택한 텍스트 편집기를 사용하여 인증서에 확장을 추가하기 위해 OpenSSL 구성 파일을 준비합니다. 예를 들면 다음과 같습니다.*

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

[client-cert]
extendedKeyUsage = clientAuth
```

2. ***x509*** *유틸리티를 사용하여 CSR을 기반으로 인증서를 생성합니다. 예를 들면 다음과 같습니다.*

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

추가 리소스

- ***OpenSSL(1)***, ***ca(1)*** *및* ***x509(1)*** *매뉴얼 페이지*

2.6. GNUTLS를 사용하여 개인 CA 생성

개인 인증 기관(CA)은 시나리오가 내부 네트워크 내의 엔티티를 확인해야 하는 경우에 유용합니다. 예를 들어, 컨트롤에서 서명된 인증서 또는 상용 CA를 구입하지 않으려면 개인 CA를 사용하여 인증으로 VPN 게이트웨이를 만들 때 사용합니다. 이러한 사용 사례의 인증서에 서명하기 위해 개인 CA는 자체 서명된 인증서를 사용합니다.

사전 요구 사항

- ***sudo*** *를 사용하여 관리자 명령을 입력할 수 있는* ***루트*** *권한 또는 권한이 있어야 합니다. 이러한 권한이 필요한 명령은* ***#*** *으로 표시됩니다.*
- *이미 시스템에 GnuTLS를 설치했습니다. 그렇지 않은 경우 다음 명령을 사용할 수 있습니다.*

```
$ dnf install gnutls-utils
```

절차

1. *CA의 개인 키를 생성합니다. 예를 들어 다음 명령은 256 비트 ECDSA(Elliptic Curve Digital Signature Algorithm) 키를 생성합니다.*

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

키 생성 프로세스의 시간은 호스트의 하드웨어 및 엔트로피, 선택한 알고리즘 및 키 길이에 따라 달라집니다.

2. *인증서에 대한 템플릿 파일을 생성합니다.*

   a. *선택한 텍스트 편집기로 파일을 생성합니다. 예를 들면 다음과 같습니다.*

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

   b. *필요한 인증 세부 정보를 포함하도록 파일을 편집합니다.*

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

```
ca
cert_signing_key
crl_signing_key
```

3. 1단계에서 생성된 개인 키를 사용하여 서명된 인증서를 생성합니다.
생성된 < ca.crt > 파일은 1년 동안 다른 인증서에 서명하는 데 사용할 수 있는 자체 서명된 CA 인증서입니다. < ca.crt > 파일은 공개 키(인증서) 입니다. 로드된 파일 < ca.key& gt;는 개인 키입니다. 이 파일을 안전한 위치에 보관해야 합니다.

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

4. CA의 개인 키에 대해 보안 권한을 설정합니다. 예를 들면 다음과 같습니다.

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

다음 단계

- 자체 서명된 CA 인증서를 클라이언트 시스템의 신뢰 앵커로 사용하려면 CA 인증서를 클라이언트에 복사하여 클라이언트의 시스템 전체 신뢰 저장소에 **root** 로 추가합니다.

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

자세한 내용은 3 장. 공유 시스템 인증서 사용 를 참조하십시오.

검증

1. 자체 서명된 CA에 대한 기본 정보를 표시합니다.

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

2. CSR(인증서 서명 요청)을 생성하고 CA를 사용하여 요청에 서명합니다. CA는 CSR을 기반으로 인증서를 성공적으로 생성해야 합니다. 예를 들면 다음과 같습니다.

   a. CA의 개인 키를 생성합니다.

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

   b. 선택한 텍스트 편집기에서 새 구성 파일을 엽니다. 예를 들면 다음과 같습니다.

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

   c. 필요한 인증 세부 정보를 포함하도록 파일을 편집합니다.

```
signing_key
encryption_key
key_agreement

tls_www_server

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

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

d. *이전에 생성된 개인 키를 사용하여 요청을 생성합니다.*

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

e. *인증서를 생성하고 CA의 개인 키로 서명합니다.*

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

추가 리소스

- ***certtool(1)*** 및 ***trust(1)*** 도움말 페이지

2.7. GNUTLS를 사용하여 TLS 서버 인증서에 대한 개인 키 및 CSR 생성

인증서를 받으려면 먼저 서버의 개인 키와 CSR(인증서 서명 요청)을 만들어야 합니다.

절차

1. *서버 시스템에서 개인 키를 생성합니다. 예를 들면 다음과 같습니다.*

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

2. *선택 사항: 선택한 텍스트 편집기를 사용하여 CSR 생성을 간소화하는 구성 파일을 준비합니다. 예를 들면 다음과 같습니다.*

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

tls_www_server

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

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

3. 이전에 생성한 개인 키를 사용하여 CSR을 생성합니다.

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

--template 옵션을 생략하면 **certool** 유틸리티에서 추가 정보를 입력하라는 메시지를 표시합니다. 예를 들면 다음과 같습니다.

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

다음 단계

- 서명하기 위해 선택한 CA에 CSR을 제출합니다. 또는 신뢰할 수 있는 네트워크 내의 내부 사용 시나리오의 경우 개인 CA를 사용하여 서명합니다. 자세한 내용은 2.9 절. "개인 CA를 사용하여 GnuTLS와 CSR의 인증서 발행"를 참조하십시오.

검증

1. CA에서 요청된 인증서를 가져온 후 인증서의 사람이 읽을 수 있는 부분이 요구 사항과 일치하는지 확인합니다.

```
$ certtool --certificate-info --infile <example-server.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
```

```
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
...
```

추가 리소스

- ***certtool(1)*** 도움말 페이지

2.8. GNUTLS를 사용하여 TLS 클라이언트 인증서에 대한 개인 키 및 CSR 생성

인증서를 받으려면 먼저 클라이언트의 개인 키와 CSR(인증서 서명 요청)을 생성해야 합니다.

절차

1. 클라이언트 시스템에서 개인 키를 생성합니다. 예를 들면 다음과 같습니다.

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

2. 선택 사항: 선택한 텍스트 편집기를 사용하여 CSR 생성을 간소화하는 구성 파일을 준비합니다. 예를 들면 다음과 같습니다.

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

tls_www_client

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

3. 이전에 생성한 개인 키를 사용하여 CSR을 생성합니다.

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

--template 옵션을 생략하면 ***certtool*** 유틸리티에서 추가 정보를 입력하라는 메시지를 표시합니다. 예를 들면 다음과 같습니다.

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

다음 단계

- 서명하기 위해 선택한 CA에 CSR을 제출합니다. 또는 신뢰할 수 있는 네트워크 내의 내부 사용 시나리오의 경우 개인 CA를 사용하여 서명합니다. 자세한 내용은 2.9절. "개인 CA를 사용하여 GnuTLS와 CSR의 인증서 발행"를 참조하십시오.

검증

1. 사용자가 읽을 수 있는 인증서 부분이 요구 사항과 일치하는지 확인합니다. 예를 들면 다음과 같습니다.

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

추가 리소스

- **certtool(1)** 도움말 페이지

2.9. 개인 CA를 사용하여 GNUTLS와 CSR의 인증서 발행

시스템이 TLS 암호화 통신 채널을 설정할 수 있도록 하려면 CA(인증 기관)에서 유효한 인증서를 제공해야 합니다. 개인 CA가 있는 경우 시스템에서 인증서 서명 요청(CSR)에 서명하여 요청된 인증서를 생성할 수 있습니다.

사전 요구 사항

- 이미 개인 CA가 구성되어 있습니다. 자세한 내용은 2.6절. "GnuTLS를 사용하여 개인 CA 생성"를 참조하십시오.
- CSR이 포함된 파일이 있습니다. 2.7절. "GnuTLS를 사용하여 TLS 서버 인증서에 대한 개인 키 및 CSR 생성"에서 CSR을 생성하는 예를 찾을 수 있습니다.

절차

1. 선택 사항: 선택한 텍스트 편집기를 사용하여 인증서에 확장 기능을 추가하기 위해 GnuTLS 구성 파일을 준비합니다. 예를 들면 다음과 같습니다.

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

2. **certtool** 유틸리티를 사용하여 CSR을 기반으로 인증서를 생성합니다. 예를 들면 다음과 같습니다.

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

추가 리소스

- ***certtool(1)*** *도움말 페이지*

3장. 공유 시스템 인증서 사용

공유 시스템 인증서 스토리지를 사용하면 NSS, GnuTLS, OpenSSL 및 Java에서 시스템 인증서 앵커 및 블록 목록 정보를 검색할 기본 소스를 공유할 수 있습니다. 기본적으로 신뢰 저장소에는 긍정 및 부정적인 신뢰를 포함한 Mozilla CA 목록이 포함되어 있습니다. 시스템을 사용하면 코어 Mozilla CA 목록을 업데이트하거나 다른 인증서 목록을 선택할 수 있습니다.

3.1. 시스템 전체 신뢰 저장소

RHEL의 통합 시스템 전체 신뢰 저장소는 **/etc/pki/ca-trust/** 및 **/usr/share/pki/ca-trust-source/** 디렉터리에 있습니다. **/usr/share/pki/ca-trust-source/**의 신뢰 설정은 **/etc/pki/ca-trust/**의 설정보다 우선 순위가 낮습니다.

인증서 파일은 설치된 하위 디렉터리에 따라 처리됩니다.

- 신뢰 앵커는 다음과 같습니다.
  - **/usr/share/pki/ca-trust-source/anchors/** 또는
  - **/etc/pki/ca-trust/source/anchors/**.
- 신뢰할 수 없는 인증서는 에 저장됩니다.
  - **/usr/share/pki/ca-trust-source/blocklist/** 또는
  - **/etc/pki/ca-trust/source/blocklist/**.
- 확장 BEGINTED 파일 형식의 인증서는 에 있습니다.
  - **/usr/share/pki/ca-trust-source/** 또는
  - **/etc/pki/ca-trust/source/**.

참고

계층적 암호화 시스템에서 신뢰 앵커는 다른 당사자가 신뢰할 수 있는 것으로 간주하는 신뢰할 수 있는 엔티티입니다. X.509 아키텍처에서 루트 인증서는 신뢰 체인이 파생되는 신뢰 앵커입니다. 체인 검증을 사용하려면 신뢰할 수 있는 당사자가 신뢰 앵커에 먼저 액세스할 수 있어야 합니다.

추가 리소스

- **update-ca-trust(8)** 및 **trust(1)** 도움말 페이지

3.2. 새 인증서 추가

새로운 신뢰 소스로 시스템에서 애플리케이션을 승인하려면 해당 인증서를 시스템 전체 저장소에 추가하고 **update-ca-trust** 명령을 사용합니다.

사전 요구 사항

- **ca-certificates** 패키지가 시스템에 있습니다.

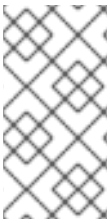
절차

1. 간단한 PEM 또는 DER 파일 형식의 인증서를 시스템에서 신뢰하는 CA 목록에 추가하려면 인증서 파일을 **/usr/share/pki/ca-trust-source/anchors/** 또는 **/etc/pki/ca-trust/source/anchors/** 디렉터리에 복사합니다.

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

2. 시스템 전체 신뢰 저장소 구성을 업데이트하려면 **update-ca-trust** 명령을 사용합니다.

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



참고

Firefox 브라우저에서 **update-ca-trust** 를 이전에 실행하지 않고 추가된 인증서를 사용할 수 있지만 모든 CA를 변경한 후 **update-ca-trust** 명령을 입력합니다. 또한 Firefox, Chromium 및 GNOME 웹 캐시 파일과 같은 브라우저의 캐시를 지우거나 브라우저를 다시 시작하여 현재 시스템 인증서 구성을 로드해야 할 수도 있습니다.

추가 리소스

- **update-ca-trust(8)** 및 **trust(1)** 도움말 페이지

3.3. 신뢰할 수 있는 시스템 인증서 관리

trust 명령은 공유 시스템 전체 신뢰 저장소에서 인증서를 관리하는 편리한 방법을 제공합니다.

- 신뢰 앵커를 나열, 추출, 추가, 제거 또는 변경하려면 **trust** 명령을 사용합니다. 이 명령에 대한 기본 도움말을 보려면 인수 없이 또는 **--help** 지시문을 사용하여 입력합니다.

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

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

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

- 모든 시스템 신뢰 앵커 및 인증서를 나열하려면 **trust list** 명령을 사용합니다.

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

pkcs11:id=%a6%b3%e1%2b%2b%49%b6%d7%73%a1%aa%94%f5%01%e7%73%65%4c%ac%50;type=cert
    type: certificate
```

```
    label: ACEDICOM Root
    trust: anchor
    category: authority
...
```

- 신뢰 앵커를 시스템 전체 신뢰 저장소에 저장하려면 **trust anchor** 하위 명령을 사용하고 인증서 경로를 지정합니다. & lt;path.to/certificate.crt >를 인증서 및 파일 이름의 경로로 바꿉니다.

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

- 인증서를 제거하려면 인증서의 경로 또는 인증서의 ID를 사용합니다.

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

추가 리소스

- **trust** 명령의 모든 하위 명령은 상세한 기본 도움말을 제공합니다. 예를 들면 다음과 같습니다.

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

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

추가 리소스

- **update-ca-trust(8)** 및 **trust(1)** 도움말 페이지

4장. 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.

4.1. 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 핵심 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 9에서는 모든 암호화 정책에서 비활성화됨.
TLS 1.1	필요한 경우 상호 운용성을 목적으로 사용합니다. 최신 암호화 제품군을 지원하지 않습니다. RHEL 9에서는 모든 암호화 정책에서 비활성화됨.
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 전체 암호화 정책에서도 활성화됩니다.

추가 리소스

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

4.2. RHEL 9의 TLS에 대한 보안 고려 사항

*RHEL 9에서 TLS 구성은 시스템 전체 암호화 정책 메커니즘을 사용하여 수행됩니다. 1.2 미만의 TLS 버전은 더 이상 지원되지 않습니다. **DEFAULT**, **FUTURE**, **LEGACY** 암호화 정책은 TLS 1.2 및 1.3만 허용합니다. 자세한 내용은 시스템 전체 암호화 정책 사용 을 참조하십시오.*

RHEL 9에 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 Red Hat의 RHEL 9 보안 비전을 준수하지 않으며 보안 속성을 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.

4.2.1. 프로토콜

최신 버전의 TLS는 최상의 보안 메커니즘을 제공합니다. TLS 1.2는 **LEGACY** 암호화 정책을 사용하는 경우에도 최소 버전입니다. 암호화 정책을 비활성화하거나 사용자 지정 정책을 제공하면 이전 프로토콜 버전을 다시 활성화할 수 있지만 결과적으로 생성되는 구성은 지원되지 않습니다.

RHEL 9는 TLS 버전 1.3을 지원하지만 RHEL 9 구성 요소에서 이 프로토콜의 일부 기능을 완전히 지원하는 것은 아닙니다. 예를 들어 연결 대기 시간을 줄이는 0-RTT(round Trip Time) 기능은 Apache 웹 서버에서 아직 완전히 지원하지 않습니다.

주의

FIPS 모드에서 실행되는 RHEL 9.2 이상 시스템은 모든 TLS 1.2 연결이 FIPS 140-3 표준의 필요에 따라 확장 마스터 보안(ECDSA) 확장(RFC 7627)을 사용해야 함을 강제합니다. 따라서 ECDSA 또는 TLS 1.3을 지원하지 않는 레거시 클라이언트는 FIPS 모드에서 실행되는 RHEL 9 서버에 연결할 수 없으며 FIPS 모드의 RHEL 9 클라이언트는 ECDSA 없이 TLS 1.2만 지원하는 서버에 연결할 수 없습니다. Red Hat Enterprise Linux 9.2에서 적용된 TLS 확장 "확장 마스터 시크릿"을 참조하십시오.

4.2.2. 암호화 제품군

보안이 강화된 최신 암호화 제품군을 안전하지 않은 이전 암호 제품군에 우선적으로 사용해야 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 128 비트 미만의 보안을 제공하는 암호화 제품군은 짧은 유효 기간 동안 고려해서는 안 됩니다. 128 비트의 보안을 사용하는 알고리즘은 적어도 몇 년 동안 중단되지 않을 수 있으므로 강력하게 권장합니다. 3DES 암호화는 168 비트 사용을 알리지만 실제로 112 비트의 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.

4.2.3. 공개 키 길이

RSA 키를 사용하는 경우 항상 적어도 SHA-256에 의해 서명된 3072 비트의 키 길이를 선호하며, 이는 실제 128 비트의 보안에 충분히 큰 영향을 미칩니다.

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.

4.3. 애플리케이션에서 TLS 구성 강화

RHEL에서 *시스템 전체 암호화 정책은* 암호화 라이브러리를 사용하는 애플리케이션에서 알려진 비보안 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.

사용할 구성이 무엇이든 관계없이 서버 애플리케이션이 서버 측 암호 순서 를 강제 적용하므로 사용할 암호 모음은 구성하는 순서에 따라 결정됩니다.

4.3.1. TLS를 사용하도록 Apache HTTP 서버 구성

Apache HTTP Server는 TLS 요구 사항에 따라 **OpenSSL** 및 **NSS** 라이브러리를 모두 사용할 수 있습니다. RHEL 9는 eponymous 패키지를 통해 **mod_ssl** 기능을 제공합니다.

```
# dnf 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 가져

옵니다. **/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_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 *웹 서버 배포 및 역방향 프록시* 문서의 *Apache HTTP Server에서 TLS 암호화 구성* 장을 참조하십시오.

4.3.2. TLS를 사용하도록 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 *웹 서버 배포 및 역방향 프록시* 문서의 *Nginx 웹 서버에 TLS 암호화 추가* 장을 참조하십시오.

4.3.3. TLS를 사용하도록 Dovecot 메일 서버 구성

TLS를 사용하도록 **Dovecot** 메일 서버의 설치를 구성하려면 **/etc/dovecot/conf.d/10-ssl.conf** 구성 파일을 수정합니다. 해당 파일에서 사용 가능한 기본 구성 지시문 중 일부에 대한 설명은 **/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt** 파일의 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

- *웹 서버 및 역방향 프록시 배포*
- ***config(5)** 및 **ciphers(1)** 도움말 페이지.*
- ***TLS(Transport Layer Security)** 및 **DTLS(Datagram Transport Layer Security)**에 대한 보안 권장 사항입니다.*
- ***Mozilla SSL** 구성 생성기.*
- ***SSL** 서버 테스트.*

5장. IPSEC을 사용하여 VPN 구성

RHEL 9에서는 Libreswan 애플리케이션에서 지원하는 IPsec 프로토콜을 사용하여 VPN(가상 사설 네트워크)을 구성할 수 있습니다.

5.1. LIBRESWAN AS AN IPSEC VPN 구현

RHEL에서는 Libreswan 애플리케이션에서 지원하는 IPsec 프로토콜을 사용하여 VPN(Virtual Private Network)을 구성할 수 있습니다. Libreswan은 Openswan 애플리케이션에 대한 연속이며, Openswan 문서의 많은 예제는 Libreswan과 교환 가능합니다.

VPN용 IPsec 프로토콜은 IKEv(Internet Key Exchange) 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 추가 및 제거하도록 커널을 설정합니다.

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

- 프로토콜 번호 50이 있는ESP(Security Payload)를 캡슐화합니다.
- 인증된 헤더(AH)는 프로토콜 번호 51이 있습니다.

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

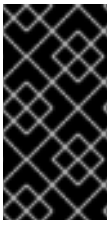
IPsec 프로토콜은 다음 두 가지 작업 모드를 제공합니다.

- 터널 모드(기본값)

- *전송 모드.*

IKE 없이 IPsec을 사용하여 커널을 구성할 수 있습니다. 이를 수동 키링 이라고 합니다. ip xfrm 명령을 사용하여 수동 인증도 구성할 수 있지만 보안상의 이유로 이 방법은 권장되지 않습니다. Libreswan은 Netlink 인터페이스를 사용하여 Linux 커널과 통신합니다. 커널은 패킷 암호화 및 암호 해독을 수행합니다.

Libreswan은 NSS(Network Security Services) 암호화 라이브러리를 사용합니다. NSS는FIPS(Federal Information Processing Standard) 발행 140-2와 함께 사용하도록 인증되었습니다.



중요

Libreswan 및 Linux 커널에서 구현되는 IKE/IPsec VPN은 RHEL에서 사용할 수 있는 유일한 VPN 기술입니다. 위험을 이해 하지 않고 다른 VPN 기술을 사용 하지 마십시오.

RHEL에서 Libreswan은 기본적으로 시스템 전체 암호화 정책을 따릅니다. 이를 통해 Libreswan은 IKEv2를 기본 프로토콜로 포함한 현재 위협 모델에 대한 보안 설정을 사용할 수 있습니다. 자세한 내용은 시스템 전체 암호화 정책 사용을 참조하십시오.

Libreswan은 IKE/IPsec이 피어 프로토콜로 피어링되기 때문에 "source" 및 "destination" 또는 "server" 및 "client"라는 용어를 사용하지 않습니다. 대신 "left" 및 "right"라는 용어를 사용하여 엔드 포인트(호스트)를 나타냅니다. 또한 대부분의 경우 두 끝점 모두에서 동일한 구성을 사용할 수 있습니다. 그러나 일반적으로 관리자는 로컬 호스트에 "left"를 사용하고 원격 호스트에 대해 "right"를 사용하도록 선택합니다.

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

5.2. 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 키는 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 해지를 포함하여 신뢰를 중계합니다.

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-2의 RSA를 사용하는 X.509 인증서에 따라 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. 이로 인해 상호 운용성이 증가하고 하나의 디지털

서명 시스템에서 인증을 유지합니다.

NULL 인증

NULL 인증은 인증없이 메시 암호화를 얻는 데 사용됩니다. 액티브 공격으로부터 보호하지만 활성 공격으로부터 보호하지 않습니다. 그러나 **IKEv2**는 비대칭 인증 방법을 허용하므로 **Internetscale opportunistic IPsec**에도 **NULL** 인증을 사용할 수 있습니다. 이 모델에서 클라이언트는 서버를 인증하지만 서버는 클라이언트를 인증하지 않습니다. 이 모델은 **TLS**를 사용하는 보안 웹 사이트와 유사합니다. **NULL** 인증을 위해 **authby=null** 을 사용합니다.

Quantum Computer에 대한 보호

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

IKEv1을 사전 공유 키와 함께 사용하면 정크 공격자로부터 보호됩니다. **IKEv2**의 재 설계는 이러한 보호 기능을 기본적으로 제공하지 않습니다. **Libreswan**은 압 공격으로부터 **IKEv2** 연결을 보호하기 위해 **Post-quantum Pre-shared Key (PPK)**를 사용하여 **IKEv2** 연결을 보호합니다.

선택적 **PPK** 지원을 활성화하려면 연결 정의에 **ppk=yes** 를 추가합니다. **PPK**를 요구하려면 **ppk=insist** 를 추가합니다. 그런 다음, 각 클라이언트에 범위를 벗어난 비밀 값이 있는 **PPK ID**를 제공할 수 있습니다(및 더 바람직하게 압축하는 경우). **PPK**는 사전 단어를 기반으로 하지 않고 무작위로 매우 강력해야 합니다. **PPK ID** 및 **PPK** 데이터는 **ipsec.secrets** 파일에 저장됩니다. 예를 들면 다음과 같습니다.

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

PPKS 옵션은 정적 **PPK**를 나타냅니다. 이 실험적 기능은 일회용 동적 **PPK**를 사용합니다. 각 연결에서 **1**회패드의 새 부분이 **PPK**로 사용됩니다. 사용하면 파일 내의 동적 **PPK**의 해당 부분을 다시 사용하지 않도록 **0**으로 덮어씁니다. 더 이상 일회성 편집기가 남아 있지 않으면 연결이 실패합니다. 자세한 내용은 **ipsec.secrets(5)** 도움말 페이지를 참조하십시오.

주의

동적 **PPK**의 구현은 지원되지 않는 기술 프리뷰로 제공됩니다. 주의해서 사용하십시오.

5.3. LIBRESWAN 설치

Libreswan IPsec/IKE 구현을 통해 VPN을 설정하려면 해당 패키지를 설치하고 ipsec 서비스를 시작하고 방화벽에서 서비스를 허용해야 합니다.

사전 요구 사항

- **AppStream 리포지토리가 활성화되어 있어야 합니다.**

절차

1. **libreswan 패키지를 설치합니다.**

```
# dnf install libreswan
```

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

```
# systemctl stop ipsec
# rm /var/lib/ipsec/nss/*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
```

5.4. 호스트 대 호스트 VPN 생성

원시 RSA 키의 인증을 사용하여 왼쪽 및 오른쪽 이라는 두 호스트 간에 host-to-host IPsec VPN을 생성하도록 Libreswan을 구성할 수 있습니다.

사전 요구 사항

- ***Libreswan**이 설치되어 있고 **ipsec** 서비스가 각 노드에 시작됩니다.*

절차

1. *각 호스트에 원시 **RSA** 키 쌍을 생성합니다.*

```
# ipsec newhostkey
```

2. *이전 단계에서 생성된 키의 **ckaid**가 반환되었습니다. 왼쪽에서 다음 명령과 함께 **ckaid** 를 사용하십시오. 예를 들면 다음과 같습니다.*

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

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

```
# 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
```

5.5. 사이트 간 VPN 구성

두 개의 네트워크에 참여하여 사이트 간 **IPsec VPN**을 생성하려면 두 호스트 간의 **IPsec** 터널을 생성합니다. 따라서 호스트는 하나 이상의 서브넷의 트래픽이 통과할 수 있도록 구성된 엔드포인트 역할을 합니다. 따라서 호스트를 네트워크의 원격 부분에 대한 게이트웨이로 간주할 수 있습니다.

사이트 간 **VPN**의 구성은 구성 파일에 하나 이상의 네트워크 또는 서브넷을 지정해야 한다는 점에서 호스트 투 호스트 **VPN**과만 다릅니다.

사전 요구 사항

- 호스트 간 **VPN**이 이미 구성되어 있습니다.

절차

1. 호스트 투 호스트 **VPN**의 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
```

5.6. 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, 길 전사의 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
```

5.7. 메시 VPN 구성

*임의 의 **VPN**이라고도 하는 메시 **VPN** 네트워크는 모든 노드가 **IPsec**을 사용하여 통신하는 네트워크입니다. 이 구성은 **IPsec**을 사용할 수 없는 노드에 대한 예외를 허용합니다. 메시 **VPN** 네트워크는 두 가지 방법으로 구성할 수 있습니다.*

- ***IPsec**이 필요합니다.*
- ***IPsec**을 선호하지만, 일반 텍스트 통신을 대체합니다.*

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

*이러한 연결은 **right=%opportunisticgroup** 항목에 정의된 **opportunistic IPsec** 을 제외하고 일반 **Libreswan** 구성이므로 **opportunistic IPsec**에 일반 **IKEv2** 인증 방법을 사용할 수 있습니다. 일반적인 인증 방법은 일반적으로 공유 **CA**(인증 기관)를 사용하여 **X.509** 인증서를 기반으로 호스트가 서로 인증하는 것입니다. 클라우드 배포에서는 일반적으로 표준 절차의 일부로 클라우드에 있는 각 노드의 인증서를 발급합니다.*

중요

*손상된 호스트 하나로 인해 그룹 **PSK**도 손상될 수 있으므로 **PreSharedKey(PSK)** 인증을 사용하지 마십시오.*

***NULL** 인증을 사용하여 수동 공격자로부터만 보호하는 인증 없이 노드 간에 암호화를 배포할 수 있습니다.*

*다음 절차에서는 **X.509** 인증서를 사용합니다. **Dogtag Certificate System**과 같은 모든 종류의 **CA** 관리 시스템을 사용하여 이러한 인증서를 생성할 수 있습니다. **Dogtag**는 각 노드의 인증서를 개인 키, 노드 인증서 및 다른 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 이름 "노드"를 사용하는 "이름"이 필요하며, 이를 통해 친숙한 이름을 참조하는 구성 파일이 모든 노드에 대해 동일할 수 있습니다.*

사전 요구 사항

- *Libreswan**이 설치되어 있고 **ipsec** 서비스가 각 노드에서 시작됩니다.*
- *새 **NSS** 데이터베이스가 초기화됩니다.*

1. *이전 **NSS** 데이터베이스가 이미 있는 경우 이전 데이터베이스 파일을 제거하십시오.*

```
# systemctl stop ipsec
# rm /var/lib/ipsec/nss/*db
```

2. *다음 명령을 사용하여 새 데이터베이스를 초기화할 수 있습니다.*

```
# ipsec initnss
```

절차

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

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

2. *필요한 **IPsec (private)**에 대해 다음 세 가지 연결 정의 **(private-or-clear)** 및 **No IPsec (clear)** 프로파일에 대해 다음 세 가지 연결 정의를 만듭니다.*

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

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

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

(1) ***auto** 변수에는 몇 가지 옵션이 있습니다.*

opportunistic IPsec과 함께 온 디맨드 연결 옵션을 사용하여 IPsec 연결을 시작하거나 항상 활성화할 필요가 없는 명시적으로 구성된 연결에 사용할 수 있습니다. 이 옵션은 커널에 트랩 XFRM 정책을 설정하여 해당 정책과 일치하는 첫 번째 패킷을 수신할 때 IPsec 연결을 시작할 수 있습니다.

다음 옵션을 사용하여 Opportunistic IPsec 또는 명시적으로 구성된 연결을 사용하는지 여부에 관계없이 IPsec 연결을 효과적으로 구성하고 관리할 수 있습니다.

추가 옵션

연결 구성을 로드하고 원격 시작에 응답하기 위해 준비합니다. 그러나 연결은 로컬 측에서 자동으로 시작되지 않습니다. ipsec auto --up 명령을 사용하여 IPsec 연결을 수동으로 시작할 수 있습니다.

시작 옵션

연결 구성을 로드하고 원격 시작에 응답하기 위해 준비합니다. 또한 원격 피어에 대한 연결을 즉시 시작합니다. 영구 및 항상 활성 연결에 이 옵션을 사용할 수 있습니다.

2

leftid 및 rightid 변수는 IPsec 터널 연결의 오른쪽과 왼쪽 채널을 식별합니다. 이러한 변수를 사용하여 구성된 경우 로컬 IP 주소의 값 또는 로컬 인증서의 제목 DN을 가져올 수 있습니다.

3

leftcert 변수는 사용하려는 NSS 데이터베이스의 닉네임을 정의합니다.

3. *네트워크의 IP 주소를 해당 카테고리에 추가합니다. 예를 들어 모든 노드가 10.15.0.0/16 네트워크에 있고 모든 노드가 IPsec 암호화를 사용해야 하는 경우 다음을 수행합니다.*

```
# 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 해당 노드를 private-or-clear 그룹에 추가합니다.*

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

5. *IPsec을 clear 그룹으로 할 수 없는 호스트(예: 10.15.1.2)를 정의하려면 다음을 사용합니다.*

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

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

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

6. 노드를 재시작하여 구성된 메시에 추가합니다.

```
# systemctl restart ipsec
```

검증

1. **ping** 명령을 사용하여 **IPsec** 터널을 엽니다.

```
# ping <nodeYYY>
```

2. 가져온 인증서를 사용하여 **NSS** 데이터베이스를 표시합니다.

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

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

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

3. 노드에서 열려 있는 터널을 확인합니다.

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

추가 리소스

- **IPsec.conf(5)** 도움말 페이지.

- **authby** 변수에 대한 자세한 내용은 6.2를 참조하십시오. Libreswan의 인증 방법.

5.8. FIPS 호환 IPSEC VPN 배포

Libreswan을 사용하여 **FIPS** 호환 **IPsec VPN** 솔루션을 배포할 수 있습니다. 이를 위해 사용 가능한 암호화 알고리즘과 **FIPS** 모드에서 **Libreswan**에 대해 비활성화된 암호화 알고리즘을 식별할 수 있습니다.

사전 요구 사항

- **AppStream** 리포지토리가 활성화되어 있어야 합니다.

절차

1. **libreswan** 패키지를 설치합니다.

```
# dnf install libreswan
```

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

```
# systemctl stop ipsec

# rm /var/lib/ipsec/nss/*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. ***Libreswan**이 **FIPS** 모드에서 실행 중인지 확인합니다.*

```
# 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 Mode: YES
```

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

```
# ipsec pluto --selftest 2>&1 | head -6
Initializing NSS using read-write database "sql:/var/lib/ipsec/nss"
FIPS Mode: YES
NSS crypto library initialized
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity support [disabled]
```

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 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 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//"
aes_ccm, aes_ccm_c
aes_ccm_b
aes_ccm_a
NSS(CBC)  3des
NSS(GCM)  aes_gcm, aes_gcm_c
NSS(GCM)  aes_gcm_b
NSS(GCM)  aes_gcm_a
NSS(CTR)  aesctr
NSS(CBC)  aes
aes_gmac
NSS       sha, sha1, sha1_96, hmac_sha1
NSS       sha512, sha2_512, sha2_512_256, hmac_sha2_512
NSS       sha384, sha2_384, sha2_384_192, hmac_sha2_384
NSS       sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
NSS(MODP) null, dh0
NSS(MODP) dh14
NSS(MODP) dh15
NSS(MODP) dh16
NSS(MODP) dh17
NSS(MODP) dh18
NSS(ECP)  ecp_256, ecp256
NSS(ECP)  ecp_384, ecp384
NSS(ECP)  ecp_521, ecp521
```

추가 리소스

- 시스템 전체 암호화 정책 사용.

5.9. 암호로 IPSEC NSS 데이터베이스 보호

기본적으로 **IPsec** 서비스는 처음 시작하는 동안 비어 있는 암호를 사용하여 **NSS(Network Security Services)** 데이터베이스를 생성합니다. 보안을 강화하기 위해 암호 보호를 추가할 수 있습니다.

사전 요구 사항

- **/var/lib/ipsec/nss/** 디렉터리에는 **NSS** 데이터베이스 파일이 포함되어 있습니다.

절차

1. **Libreswan에 대해 NSS 데이터베이스에 대한 암호 보호를 활성화합니다.**

```
# certutil -N -d sql:/var/lib/ipsec/nss
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 암호를 포함하는 **/etc/ipsec.d/nsspassword** 파일을 생성합니다. 예를 들면 다음과 같습니다.

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

nsspassword 파일은 다음 구문을 사용합니다.

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

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

추가 리소스

- **certutil(1)** 도움말 페이지.
- 규정 준수 활동 및 정부 표준 지식 베이스 문서의 **FIPS 140-2** 및 **FIPS 140-3**.

5.10. TCP를 사용하도록 IPSEC VPN 구성

Libreswan은 **RFC 8229**에 설명된 대로 **IKE** 및 **IPsec** 패킷을 **TCP** 캡슐화를 지원합니다. 이 기능을 사용하면 **UDP**를 통해 전송되는 트래픽을 방지하고 **ESB(Security Payload)**를 캡슐화하는 네트워크에서 **IPsec VPN**을 설정할 수 있습니다. **TCP**를 대체 또는 기본 **VPN** 전송 프로토콜로 사용하도록 **VPN** 서버와 클라이언트를 구성할 수 있습니다. **TCP** 캡슐화는 성능 비용이 늘어날 수 있으므로 시나리오에서 **UDP**가 영구적으로 차단된 경우에만 **TCP**를 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. **UDP**를 처음 시도하지 못하면 **TCP** 캡슐화를 대체 옵션으로 사용하려면 클라이언트의 연결 정의에 다음 두 옵션을 추가합니다.

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

또는 **UDP**가 영구적으로 차단되었음을 알고 있는 경우 클라이언트 연결 구성에서 다음 옵션

을 사용합니다.

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

추가 리소스

- IETF RFC 8229: IKE 및 IPsec 패킷의 TCP 캡처.

5.11. 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. 기능을 비활성화하거나 명시적으로 활성화한 경우 자동 탐지로 다시 전환할 수 있습니다.

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 검색을 사용해야 하는 연결의 /etc/ipsec.d/ 디렉터리에서 Libreswan 구성 파일을 편집합니다.
2. nic-offload 매개변수가 연결 설정에 설정되지 않았는지 확인합니다.
3. nic-offload 를 제거한 경우 ipsec 서비스를 다시 시작합니다.

```
# systemctl restart ipsec
```

검증

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 ping합니다.

```
# 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

5.12. IPSEC 연결 속도를 높이기 위해 본딩에서 ESP 하드웨어 오프로드 구성

하드웨어로 ESB(Security Payload)를 오프로드하면 IPsec 연결 속도가 빨라집니다. 장애 조치(failover) 이유로 네트워크 본딩을 사용하는 경우 ESP 하드웨어 오프로드 구성의 요구 사항과 절차는 일반 이더넷 장치를 사용하는 것과 다릅니다. 예를 들어, 이 시나리오에서는 본딩에서 오프로드 지원을 활성화하며 커널은 본딩 포트에 설정을 적용합니다.

사전 요구 사항

- 본딩의 모든 네트워크 카드는 ESP 하드웨어 오프로드를 지원합니다.
- 네트워크 드라이버는 본딩 장치에서 ESP 하드웨어 오프로드를 지원합니다. RHEL에서는 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/** 디렉터리에서 **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
- **IPsec**을 사용하여 **VPN** 구성

5.13. RHEL 시스템 역할을 사용하여 IPSEC으로 VPN 연결 구성

vpn 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL** 시스템에서 **VPN** 연결을 구성할 수 있습니다. 이를 사용하여 **host-to-host, network-to-network, VPN Remote Access Server** 및 메시 구성을 설정할 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 **opportunistic** 메시 구성을 생성하도록 구성할 수 있습니다. 이 역할은 호스트에 있는 호스트의 이름이 **Ansible** 인벤토리에 사용된 호스트의 이름과 동일하고 해당 이름을 사용하여 터널을 구성할 수 있다고 가정합니다.

참고

vpn RHEL 시스템 역할은 현재 IPsec 구현인 Libreswan만 VPN 공급자로 지원합니다.

5.13.1. vpn RHEL 시스템 역할을 사용하여 IPsec으로 호스트 간 VPN 생성

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

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.

절차

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

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

이 플레이북은 시스템 역할에 의해 자동으로 생성되는 키와 함께 사전 공유 키 인증을 사용하여 managed-node-01.example.com-to-managed-node-02.example.com 연결을 구성합니다. vpn_manage_firewall 및 vpn_manage_selinux 둘 다 true 로 설정되므로 vpn 역할은 firewall 및 selinux 역할을 사용하여 vpn 역할에서 사용하는 포트를 관리합니다.

인벤토리 파일에 나열되지 않은 외부 호스트로의 관리 호스트에서 연결을 구성하려면 다음 섹션을 호스트 vpn_connections 목록에 추가합니다.

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

그러면 하나의 추가 연결을 구성합니다 . **managed-node-01.example.com-to-<external_node>**

참고

연결은 관리형 노드에서만 구성되며 외부 노드에는 구성되지 않습니다.

2.

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

```
- name: Multiple VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 192.0.2.0 # IP for the control plane
          managed-node-02.example.com:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 10.0.0.1 # IP for the data plane
          managed-node-02.example.com:
            hostname: 10.0.0.2
```

3.

플레이북 구문을 확인합니다.

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

4.

Playbook을 실행합니다.

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

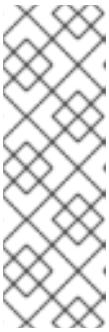
검증

1.

관리형 노드에서 연결이 성공적으로 로드되었는지 확인합니다.

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

& lt;connection_name >을 이 노드의 연결 이름으로 바꿉니다(예: managed_node1-to-managed_node 2) .



참고

기본적으로 역할은 각 시스템의 관점에서 생성하는 각 연결에 대해 설명이 포함된 이름을 생성합니다. 예를 들어 managed_node1 과 managed_node2 간의 연결을 생성할 때 managed_node1 에서 이 연결의 설명적인 이름은 managed_node1-to-managed_node2 이지만, managed_node2 에서 연결 이름은 managed_node2-to-managed_node1 입니다.

2.

관리형 노드에서 연결이 성공적으로 시작되었는지 확인합니다.

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

3.

선택 사항: 연결이 성공적으로 로드되지 않으면 다음 명령을 입력하여 연결을 수동으로 추가합니다. 이렇게 하면 연결 설정 실패 이유를 나타내는 더 구체적인 정보가 제공됩니다.

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

참고

연결을 로드하고 시작하는 동안 발생할 수 있는 오류는 /var/log/pluto.log 파일에 보고됩니다. 이러한 로그는 구문 분석하기 어렵기 때문에 대신 표준 출력에서 로그 메시지를 가져오는 데 수동으로 연결을 추가합니다.

추가 리소스

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

5.13.2. vpn RHEL 시스템 역할을 사용하여 IPsec을 통한 opportunistic 메시 VPN 연결 생성

vpn 시스템 역할을 사용하여 제어 노드에서 Ansible 플레이북을 실행하여 인증에 인증서를 사용하는 opportunistic 메시 VPN 연결을 구성할 수 있습니다. 이 연결은 인벤토리 파일에 나열된 모든 관리 노드를 구성합니다.

사전 요구 사항

- *컨트롤 노드 및 관리형 노드를 준비했습니다.*
- *관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.*
- *관리 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.*
- */etc/ipsec.d/ 디렉터리의 NSS(Network Security Services) 암호화 라이브러리에는 필요한 인증서가 포함되어 있습니다.*

절차

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

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

```
      cidr: 192.0.2.0/24
    - policy: clear
      cidr: 192.0.2.7/32
  vpn_manage_firewall: true
  vpn_manage_selinux: true
```

*인증서를 사용한 인증은 플레이북에 **auth_method: cert** 매개변수를 정의하여 구성됩니다. 기본적으로 노드 이름은 인증서 닉네임으로 사용됩니다. 이 예에서는 **managed-node-01.example.com** 입니다. 인벤토리에서 **cert_name** 속성을 사용하여 다른 인증서 이름을 정의할 수 있습니다.*

*이 예제 절차에서는 **Ansible** 플레이북을 실행하는 시스템인 제어 노드에서 관리 노드 **(192.0.2.0/24)**와 동일한 **classless inter-domain routing (CIDR)** 번호를 공유하며 **IP** 주소 **192.0.2.7**을 갖습니다. 따라서 제어 노드는 **CIDR 192.0.2.0/24**에 대해 자동으로 생성되는 개인 정책에 속합니다.*

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

***vpn_manage_firewall** 및 **vpn_manage_selinux** 둘 다 **true** 로 설정되므로 **vpn** 역할은 **firewall** 및 **selinux** 역할을 사용하여 **vpn** 역할에서 사용하는 포트를 관리합니다.*

2. *플레이북 구문을 확인합니다.*

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. ***Playbook**을 실행합니다.*

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

추가 리소스

- */usr/share/ansible/roles/rhel-system-roles.vpn/README.md* 파일

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

5.14. 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.*

5.15. 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** 끝점 중 하나 또는 끝점 간의 라우터에 있는 방화벽이 모든 인터넷 키 교환 **(IKE)** 패킷을 삭제하는 것입니다.*

- ***IKEv2**의 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
...
```

- ***IKEv1**의 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
...
```

***IPsec**을 설정하는 데 사용되는 **IKE** 프로토콜이 암호화되어 있으므로 **tcpdump** 도구를 사용하여 문제의 제한된 하위 집합만 해결할 수 있습니다. 방화벽이 **IKE** 또는 **IPsec** 패킷을 삭제하는 경우 **tcpdump** 유틸리티를 사용하여 해당 원인을 찾을 수 있습니다. 그러나 **tcpdump** 는 **IPsec VPN** 연결의 다른 문제를 진단할 수 없습니다.*

- ***eth0** 인터페이스에서 **VPN**과 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/IPsec**을 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
```

*일치하지 않는 **IKE** 버전으로 인해 응답 없이 원격 끝점에서 요청을 삭제할 수도 있습니다. 이는 모든 **IKE** 패킷을 삭제하는 방화벽과 동일합니다.*

- ***IKEv2**의 일치하지 않는 **IP** 주소 범위 예(트래픽 선택기 **- TS**)*

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

- ***IKEv1**의 일치하지 않는 **IP** 주소 범위 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
```

- ***IKEv1**에서 **PreSharedKeys (PSK)**를 사용하는 경우 양쪽이 동일한 **PSK**에 배치되지 않으면 전체 **IKE** 메시지가 읽을 수 없게됩니다.*

```
# 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
```

- ***IKEv2*에서 *mismatched-PSK* 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. 네트워크 하드웨어 조각은 최대 전송 단위(*MTU*)보다 큰 패킷(예: *1500*바이트)입니다. 종종 조각이 손실되고 패킷이 다시 조립되지 않습니다. 이로 인해 크기가 작은 패킷을 사용하는 *ping* 테스트가 작동할 때 간헐적으로 오류가 발생하지만 다른 트래픽은 실패합니다. 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 **172.16.0.1**인 시스템에는 **NAT** 규칙이 있습니다.

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

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

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

이 문제를 해결하려면 라우터의 대상 **IPsec** 서브넷 범위에 대해 **NAT**를 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 **0**이 아닌 값은 문제가 있음을 나타냅니다. 이 문제가 발생하면 새 *지원 케이스*를 열고 해당 **IKE** 로그와 함께 이전 명령의 출력을 연결합니다.

Libreswan 로그

기본적으로 **syslog** 프로토콜을 사용하는 **Libreswan** 로그입니다. **journalctl** 명령을 사용하여 **IPsec**과 관련된 로그 항목을 찾을 수 있습니다. 로그에 대한 해당 항목이 **pluto IKE** 데몬에서 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 수준에서 구성 문제가 표시되지 않는 경우 **/etc/ipsec.conf** 파일의 구성 설정 섹션에 **plutodebug=all** 옵션을 추가하여 디버그 로그를 활성화합니다.

디버그 로깅은 많은 항목을 생성하며 **journald** 또는 **syslogd** 서비스는 **syslog** 메시지를 제한할 수 있습니다. 전체 로그를 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

5.16. 추가 리소스

- **IPsec(8), ipsec.conf(5), ipsec.secrets(5), ipsec_auto(8), ipsec_rsasigkey(8)** 도움말 페이지.
- **/usr/share/doc/libreswan-version/ directory.**
- **Libreswan Project Wiki.**
- 모든 **Libreswan** 도움말 페이지.
- **NIST** 특별 발행 **800-77: IPsec VPN**에 대한 가이드.

6장. 네트워크 서비스 보안

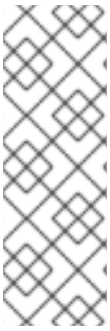
Red Hat Enterprise Linux 9는 다양한 유형의 네트워크 서버를 지원합니다. 네트워크 서비스는 DoS(서비스 거부 공격), DDoS(Distributed Denial of Service Attack), 스크립트 취약점 공격, 버퍼 오버플로 공격과 같은 다양한 유형의 공격에 시스템 보안을 노출할 수 있습니다.

공격에 대한 시스템 보안을 강화하려면 사용하는 활성 네트워크 서비스를 모니터링하는 것이 중요합니다. 예를 들어 네트워크 서비스가 시스템에서 실행되는 경우 해당 데몬은 네트워크 포트의 연결을 수신 대기하므로 보안이 저하될 수 있습니다. 네트워크를 통한 공격에 대한 노출을 제한하려면 사용되지 않은 모든 서비스를 해제해야 합니다.

6.1. DIFFIEBIND 서비스 보안

ResourceOverride bind 서비스는 원격 프로시저 호출(RPC) 서비스(Network Information Service) 및 NFS(Network File System)와 같은 서비스를 위한 동적 포트 할당 데몬입니다. 이 메커니즘은 약한 인증 메커니즘을 가지고 있으며 제어하는 서비스에 대해 광범위한 포트를 할당할 수 있으므로 Diffie bind를 보호하는 것이 중요합니다.

모든 네트워크에 대한 액세스를 제한하고 서버의 방화벽 규칙을 사용하여 특정 예외를 정의하여 Diffie bind 를 보호할 수 있습니다.



참고

- NFSv3 서버에는 Diffie bind 서비스가 필요합니다.
- NFSv4 에서는 rpcbind 서비스가 필요하지 않습니다.

사전 요구 사항

- alice bind 패키지가 설치되어 있어야 합니다.
- firewalld 패키지가 설치되었으며 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

- 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 구성을참조하십시오.

- *firewalld 사용 및 구성*

6.2. DIFFIE.MOUNTD 서비스 보안

rpc.mountd 데몬은 **NFS** 마운트 프로토콜의 서버 측을 구현합니다. **NFS** 마운트 프로토콜은 **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

- **alice .mountd** 패키지가 설치되어 있습니다.
- **firewalld** 패키지가 설치되었으며 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

- 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

6.3. NFS 서비스 보안

Kerberos를 사용하여 모든 파일 시스템 작업을 인증하고 암호화하여 NFSv4(Network File System 버전)를 보호할 수 있습니다. NAT(Network Address Translation) 또는 방화벽으로 NFSv4를 사용하는 경우 /etc/default/nfs 파일을 수정하여 위임을 해제할 수 있습니다. 위임은 서버가 파일 관리를 클라이언트에 위임하는 기술입니다.

반면 NFSv3에서는 파일 잠금 및 마운트에 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 파일에 NFS 마운트 옵션을 추가로 사용자 지정할 수 있습니다.

6.3.1. NFS 서버 보안을 위한 내보내기 옵션

NFS 서버에서는 /etc/exports 파일의 호스트로 내보낼 파일 시스템과 디렉터리의 목록 구조를 결정합니다.

주의

/etc/exports 파일의 구문에 있는 추가 공백으로 인해 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 항목과 동일하지만, bob.example.com 호스트와 동일한 디렉터리를 읽기 전용 권한으로 공유하고 호스트 이름 뒤의 단일 공백 문자로 인해 읽기 및 쓰기 권한으로 세계와 공유합니다.

```
/tmp/nfs/     bob.example.com (rw)
```

showmount -e < hostname > 명령을 입력하여 시스템의 공유 디렉토리를 확인할 수 있습니다.

/etc/exports 파일에서 다음 내보내기 옵션을 사용할 수 있습니다.

주의

파일 시스템의 하위 디렉터리 내보내기는 안전하지 않으므로 전체 파일 시스템을 내보냅니다. 공격자는 부분적으로 내보낸 파일 시스템의 내보내기되지 않은 부분에 액세스할 수 있습니다.

ro

NFS 볼륨을 읽기 전용으로 내보냅니다.

rw

NFS 볼륨에서 읽기 및 쓰기 요청을 허용합니다. 쓰기 액세스를 허용하므로 이 옵션을 신중하게 사용하면 공격 위험이 높아집니다. 시나리오에 rw 옵션을 사용하여 디렉터리를 마운트해야 하는 경우 가능한 위험을 줄이기 위해 모든 사용자가 쓸 수 없는지 확인합니다.

root_squash

uid/gid 0의 요청을 anonymous uid/gid 에 매핑합니다. 이는 bin 사용자 또는 직원 그룹과 같이 동일하게 민감한 다른 uid 또는 gid에는 적용되지 않습니다.

no_root_squash

루트 스쿼시를 끕니다. 기본적으로 NFS 공유에서는 root 사용자를 권한이 없는 사용자 계정인 nobody 사용자로 변경합니다. 이렇게 하면 생성된 모든 루트 파일의 소유자가 nobody 로 변경되어 setuid 비트가 설정된 프로그램 업로드가 방지됩니다. no_root_squash 옵션을 사용하는 경우 원격 루트 사용자는 공유 파일 시스템의 파일을 변경하고 다른 사용자를 위해 trojans의 애플리케이션을 유지할 수 있습니다.

보안

내보내기를 예약된 포트로 제한합니다. 기본적으로 서버는 예약된 포트를 통해서만 클라이언트 통신을 허용합니다. 그러나 모든 사용자가 여러 네트워크의 클라이언트에서 루트 사용자가 되기 때문에 예약된 포트를 통한 통신이 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.

- *일부 애플리케이션은 일반 텍스트 또는 약한 암호화 형식으로 암호를 저장하기 때문에 홈 디렉터리를 내보내는 것은 위험합니다. 애플리케이션 코드를 검토 및 개선하여 위험을 줄일 수 있습니다.*

- *일부 사용자는 SSH 키에 암호를 설정하지 않아 홈 디렉터리의 위험이 다시 발생합니다. 암호를 강제 사용하거나 Kerberos를 사용하여 이러한 위험을 줄일 수 있습니다.*

- *NFS 내보내기를 필수 클라이언트로만 제한합니다. NFS 서버에서 showmount -e 명령을 사용하여 서버가 내보내는 내용을 검토합니다. 특히 필요하지 않은 항목을 내보내지 마십시오.*

- *불필요한 사용자가 서버에 로그인하여 공격 위험을 줄이도록 허용하지 마십시오. 정기적으로 서버에 액세스할 수 있는 사용자 및 항목을 확인할 수 있습니다.*

추가 리소스

- Red Hat Identity Management 사용 시 IdM에서 자동 사용

- exports(5) 및 nfs(5) 도움말 페이지

6.3.2. NFS 클라이언트 보안을 위한 마운트 옵션

다음 옵션을 mount 명령에 전달하여 NFS 기반 클라이언트의 보안을 강화할 수 있습니다.

nosuid

nosuid 옵션을 사용하여 set-user-identifier 또는 set-group-identifier 비트를 비활성화합니다. 이렇게 하면 원격 사용자가 setuid 프로그램을 실행하여 더 높은 권한을 얻지 못하며 setuid 옵션 대신 이 옵션을 사용할 수 있습니다.

noexec

noexec 옵션을 사용하여 클라이언트의 실행 파일을 모두 비활성화합니다. 이를 사용하여 사용자가 공유 파일 시스템에 저장된 파일을 실수로 실행하지 못하도록 합니다.

nodev

nodev 옵션을 사용하여 클라이언트의 장치 파일을 하드웨어 장치로 처리하지 않도록 합니다.

resvport

resvport 옵션을 사용하여 예약된 포트로의 통신을 제한하고 권한 있는 소스 포트를 사용하여 서버와 통신할 수 있습니다. 예약된 포트는 권한 있는 사용자 및 root 사용자와 같은 프로세스를 위해 예약됩니다.

sec

NFS 서버에서 sec 옵션을 사용하여 마운트 지점의 파일에 액세스하기 위해 RPCGSS 보안 플레이버를 선택합니다. 유효한 보안 플레이버는 없음,sys,krb5,krb5i, krb5p 입니다.

중요

krb5-libs 패키지에서 제공하는 MIT Kerberos 라이브러리는 새 배포에서 DES(Data Encryption Standard) 알고리즘을 지원하지 않습니다. 보안 및 호환성 때문에 DES는 더 이상 사용되지 않으며 Kerberos 라이브러리에서 기본적으로 비활성화되어 있습니다. 호환성 이유로 환경에서 DES가 필요하지 않은 최신 보안 알고리즘을 사용하십시오.

추가 리소스

- *자주 사용하는 NFS 마운트 옵션*

6.3.3. 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 설치됩니다.*
- *firewalld 패키지가 설치되어 실행 중입니다.*

절차

- *NFSv4에서 방화벽이 TCP 포트 2049 를 열어야 합니다.*
- *NFSv3에서 2049 로 4개의 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. *port = <value> 형식의 [mountd] 섹션에 mountd 의 TCP 및 UDP 포트를 설정합니다.*

   b. *port = <value> 형식의 [statd] 섹션에서 statd(framework.statd)에 대한 TCP 및 UDP 포트를 설정합니다.*

2. */etc/nfs.conf 파일에서 NFS 잠금 관리자(nlockmgr)의 TCP 및 UDP 포트를 설정합니다.*

   a. *port=value 형식의 [lockd]섹션에서 nlockmgr (authorization.stat d)의 TCP 포트를 설정합니다. 또는 /etc/modprobe.d/lockd.conf 파일에서 nlm_tcpport 옵션을 사용*

할 수 있습니다.

b. udp-port=value 형식의 [lockd]섹션에서 nlockmgr (authorization.stat d)에 대해 UDP 포트를 설정합니다. 또는 /etc/modprobe.d/lockd.conf 파일에서 nlm_udpport 옵션을 사용할 수 있습니다.

검증

- NFS 서버의 활성 포트 및 RPC 프로그램을 나열합니다.

```
$ rpcinfo -p
```

추가 리소스

- Red Hat Identity Management 사용 시 IdM에서 자동 사용
- exports(5) 및 nfs(5) 도움말 페이지

6.4. FTP 서비스 보안

FTP(파일 전송 프로토콜)를 사용하여 네트워크를 통해 파일을 전송할 수 있습니다. 사용자 인증을 포함하여 서버와의 모든 FTP 트랜잭션은 암호화되지 않으므로 안전하게 구성되었는지 확인하십시오.

RHEL 9는 두 개의 FTP 서버를 제공합니다.

Red Hat Content Accelerator (tux)

FTP 기능이 있는 커널 공간 웹 서버.

매우 안전한 FTP 데몬 (vsftpd)

FTP 서비스의 독립 실행형 보안 지향 구현입니다.

다음 보안 지침은 vsftpd FTP 서비스를 설정하기 위한 것입니다.

6.4.1. FTP 인사 배너 보안

*사용자가 **FTP** 서비스에 연결하면 **FTP**는 기본적으로 버전 정보를 포함하는 인사말 배너를 표시합니다. 공격자는 이 정보를 사용하여 시스템의 취약점을 식별할 수 있습니다. 기본 배너를 변경하여 이 정보를 숨길 수 있습니다.*

*단일 줄 메시지를 직접 포함하거나 여러 줄 메시지를 포함할 수 있는 별도의 파일을 참조하도록 **/etc/banners/ftp.msg** 파일을 편집하여 사용자 지정 배너를 정의할 수 있습니다.*

절차

- *한 줄 메시지를 정의하려면 다음 옵션을 **/etc/COMPLETE/COMPLETE.conf** 파일에 추가하십시오.*

```
ftpd_banner=Hello, all activity on ftp.example.com is logged.
```

- *별도의 파일에 메시지를 정의하려면 다음을 수행합니다.*
  - *배너 메시지가 포함된 **.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.*

  - */etc/COMPLETE/COMPLETE.conf* 파일의 **banner_file** 옵션에 배너 파일 경로를 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.
```

6.4.2. FTP에서 익명 액세스 및 업로드 방지

기본적으로 **vsftpd** 패키지를 설치하면 디렉터리에 대한 읽기 전용 권한이 있는 익명 사용자의 경우 **/var/ftp/** 디렉터리와 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 익명의 사용자가 데이터를 읽지 못하도록 **FTP** 서버를 구성할 수 있습니다. 다음 절차에서는 익명 사용자가 **root** 사용자가 소유한 디렉터리에 파일을 업로드할 수 있지만 변경할 수는 없어야 합니다.

절차

- **/var/ftp/pub/** 디렉토리에 쓰기 전용 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/COMPLETE/vsftpd.conf** 파일에 추가합니다.

```
anon_upload_enable=YES
anonymous_enable=YES
```

- 선택 사항: 시스템에 **SELinux**를 활성화 및 적용하는 경우 **SELinux** 부울 속성 **allow_ftpd_anon_write** 및 **allow_ftpd_full_access** 를 활성화합니다.

주의

익명의 사용자가 디렉토리를 읽고 쓸 수 있도록 허용하면 서버가 손상되는 소프트웨어를 위한 저장소가 될 수 있습니다.

6.4.3. FTP용 사용자 계정 보안

FTP는 인증을 위해 안전하지 않은 네트워크를 통해 사용자 이름과 암호를 전송합니다. 사용자 계정에서 서버에 대한 시스템 사용자 액세스를 거부하여 **FTP**의 보안을 개선할 수 있습니다.

구성에 적용할 수 있는 만큼 다음 단계를 수행합니다.

절차

- /etc/COMPLETE/COMPLETE.conf 파일에 다음 행을 추가하여 vsftpd 서버의 모든 사용자 계정을 비활성화합니다.

```
local_enable=NO
```

- 사용자 이름을 /etc/pam.d/ COMPLETE PAM 구성 파일에 추가하여 특정 계정 또는 특정 계정 그룹(예: root 사용자 및 sudo 권한이 있는 사용자)에 대한 FTP 액세스를 비활성화합니다.

- /etc/COMPLETE/ftpusers 파일에 사용자 계정을 추가하여 사용자 계정을 비활성화합니다.

6.4.4. 추가 리소스

- ftpd_selinux(8) 도움말 페이지

6.5. HTTP 서버 보안

6.5.1. httpd.conf의 보안 강화

/etc/httpd/conf/httpd.conf 파일에 보안 옵션을 구성하여 Apache HTTP 서버의 보안을 강화할 수 있습니다.

프로덕션에 배치하기 전에 시스템에서 실행 중인 모든 스크립트가 올바르게 작동하는지 항상 확인하십시오.

root 사용자만 스크립트 또는 공통 게이트웨이 인터페이스(CGI)가 포함된 디렉터리에 쓰기 권한이 있는지 확인합니다. 쓰기 권한이 있는 디렉터리 소유권을 root 로 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 디렉터리에 있는 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. 사용자 디렉터리 검색 /**root**/ 가 아닌 모든 사용자 디렉터리를 검색하려면 **UserDir**이 활성화된 상태에서 **UserDir** 이 비활성화된 **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** 모듈과 같은 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.

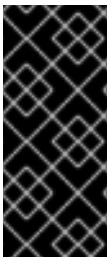
```
Apache/2.4.37
```

ServerTokens OS

웹 서버 릴리스 버전 및 운영 체제를 제공합니다. 예를 들면 다음과 같습니다.

```
Apache/2.4.37 (Red Hat Enterprise Linux)
```

ServerTokens Prod 옵션을 사용하여 공격자가 시스템에 대한 중요한 정보를 얻을 위험을 줄일 수 있습니다.



중요

IncludesNoExec 지시문을 제거하지 마십시오. 기본적으로 **Server Side Includes (SSI)** 모듈은 명령을 실행할 수 없습니다. 이를 변경하면 공격자가 시스템의 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 지정*

6.5.2. 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 옵션을 사용하면 서버 버전 번호와 같은 추가 세부 정보가 표시되지 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** 파일의 특정 알려진 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 이전 브라우저의 **MIME** 유형 스니핑을 방지합니다.

```
add_header X-Content-Type-Options nosniff;
```

  - 예를 들어 **X-XSS-Protection** 헤더는 **XSS(Cross-Site Scripting)** 필터링을 활성화하여 브라우저에서 **Nginx**에 의해 응답에 포함된 잠재적으로 악의적인 콘텐츠를 렌더링하지 못하도록 합니다.

```
add_header X-XSS-Protection "1; mode=block";
```

-

일반 대중에게 노출되는 서비스를 제한하고 자신이하는 것을 제한하고 방문자로부터 수락 할 수 있습니다. 예를 들면 다음과 같습니다.

```
limit_except GET {
    allow 192.168.1.0/32;
    deny  all;
}
```

스니펫은 **GET** 및 **HEAD** 를 제외한 모든 메서드에 대한 액세스를 제한합니다.

- **HTTP** 메서드를 비활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
# 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 있으며 **HTTPS**를 통해서만 서비스를 제공하는 것이 좋습니다. 또한 **Mozilla SSL** 구성 생성기를 사용하여 **Nginx** 서버에서 **SSL**을 활성화하기 위한 보안 구성 프로파일을 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

6.6. 인증된 로컬 사용자에 대한 액세스를 제한하여 POSTGRESQL 보안

PostgreSQL은 개체 관계형 데이터베이스 관리 시스템(**DBMS**)입니다. **Red Hat Enterprise Linux**에서 **PostgreSQL**은 **postgresql-server** 패키지에서 제공합니다.

클라이언트 인증을 구성하여 공격의 위험을 줄일 수 있습니다. 데이터베이스 클러스터의 데이터 디렉터리에 저장된 **pg_hba.conf** 구성 파일은 클라이언트 인증을 제어합니다. 호스트 기반 인증을 위해 **PostgreSQL**을 구성하려면 절차를 따르십시오.

절차

1. ***PostgreSQL*을 설치합니다.**

```
# yum install postgresql-server
```

2. **다음 옵션 중 하나를 사용하여 데이터베이스 스토리지 영역을 초기화합니다.**

   a. ***initdb* 유틸리티 사용:**

   ```
   $ initdb -D /home/postgresql/db1/
   ```

   ***-D* 옵션이 있는 *initdb* 명령은 해당 디렉터리가 아직 존재하지 않는 경우 지정한 디렉터리를 생성합니다(예: */home/postgresql/db1/*). 그러면 이 디렉터리에는 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 시스템 관리자에게 기본 데이터베이스 클러스터 관리를 지원합니다.

3. **인증된 모든 로컬 사용자가 사용자 이름이 있는 모든 데이터베이스에 액세스할 수 있도록 *pg_hba.conf* 파일에서 다음 행을 수정합니다.**

```
local   all             all                                     trust
```

이 문제는 데이터베이스 사용자를 생성하고 로컬 사용자가 없는 계층화된 애플리케이션을 사용할 때 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.

6.7. MEMCACHED 서비스 보안

Memcached는 오픈 소스 고성능 분산 메모리 개체 캐싱 시스템입니다. 데이터베이스 로드를 줄여 동적 웹 애플리케이션의 성능을 향상시킬 수 있습니다.

Memcached는 데이터베이스 호출, **API** 호출 또는 페이지 렌더링 결과에서 문자열 및 오브젝트와 같은 임의의 데이터의 작은 청크를 위한 메모리 내 키-값 저장소입니다. **Memcached**를 사용하면 활용도가 낮은 영역에서 더 많은 메모리가 필요한 애플리케이션에 메모리를 할당할 수 있습니다.

2018년, 공개 인터넷에 노출된 **Memcached** 서버를 악용하여 **DDoS** 개정 공격 취약점을 발견했습니다. 이러한 공격은 전송에 **UDP** 프로토콜을 사용하여 **Memcached** 통신을 활용합니다. 이 공격은 수 백바이트 크기의 요청이 몇 메가바이트 또는 수백 메가바이트 크기의 응답을 생성할 수 있는 높은 배율 비율로 인해 효과가 있었습니다.

대부분의 경우 **memcached** 서비스를 공용 인터넷에 노출할 필요가 없습니다. 이러한 노출에는 원격 공격자가 **Memcached**에 저장된 정보를 유출하거나 수정할 수 있는 자체 보안 문제가 있을 수 있습니다.

6.7.1. DDoS에 대한 Memcached 강화

보안 위험을 완화하려면 구성에 적용 가능한 한 많은 다음 단계를 수행하십시오.

절차

- **LAN**에서 방화벽을 구성합니다. **Memcached** 서버에 로컬 네트워크에서만 액세스할 수 있어야 하는 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 수신 대기합니다. **/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 대한 데이터베이스에 액세스할 수 있는지 확인합니다.*

```
# chown memcached:memcached /path.to/memcached.sasldb
```

4. ***/etc/sysconfig/memcached** 파일의 **OPTIONS** 매개변수에 **-S** 값을 추가하여 **Memcached**에서 **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** 클라이언트와 서버 간 통신을 암호화합니다.*

1. ***/etc/sysconfig/memcached** 파일의 **OPTIONS** 매개변수에 **-Z** 값을 추가하여 **Memcached** 클라이언트와 서버 간 암호화된 통신을 활성화합니다.*

```
OPTIONS="-Z"
```

2. ***-o ssl_chain_cert** 옵션을 사용하여 인증서 체인 파일 경로를 **PEM** 형식으로 추가합니다.*

3. ***-o ssl_key** 옵션을 사용하여 개인 키 파일 경로를 추가합니다.*

7장. MACSEC을 사용하여 동일한 물리적 네트워크에서 LAYER-2 트래픽 암호화

MACsec을 사용하여 두 장치 간(point-to-point) 간의 통신을 보호할 수 있습니다. 예를 들어 분기 사무실은 중앙 사무실과 함께 Metro-Ethernet 연결을 통해 연결되며, 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(Dynamic Host Configuration Protocol)
- ARP(Address resolution protocol)
- 인터넷 프로토콜 버전 4 / 6 (IPv4 / IPv6)
- TCP 또는 UDP와 같은 IP를 통한 모든 트래픽

MACsec은 APP-AES-128 알고리즘으로 기본적으로 LAN의 모든 트래픽을 암호화 및 인증하고 사전 공유 키를 사용하여 참가자 호스트 간의 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 IP 구성을 설정하여 암호화된 연결만 사용하여 다른 호스트와 통신하거나 상위 장치에 IP 구성을 설정할 수도 있습니다. 후자의 경우 상위 장치를 사용하여 암호화되지 않은 연결 및 암호화된 연결에 MACsec 장치를 사용하여 다른 호스트와 통신할 수 있습니다.

MACsec은 특별한 하드웨어가 필요하지 않습니다. 예를 들어 호스트와 스위치 간에만 트래픽을 암호화하려는 경우를 제외하고 모든 스위치를 사용할 수 있습니다. 이 시나리오에서는 스위치가 MACsec도 지원해야 합니다.

즉, MACsec을 구성하는 두 가지 일반적인 방법이 있습니다.

- 호스트 및 호스트

- 호스트를 전환한 후 다른 호스트로 전환

중요

동일한 (물리적 또는 가상) **LAN**에 있는 호스트 간에만 **MACsec**을 사용할 수 있습니다.

7.1. NMCLI를 사용하여 MACSEC 연결 구성

nmcli 유틸리티를 사용하여 **MACsec**을 사용하도록 이더넷 인터페이스를 구성할 수 있습니다. 예를 들어 이더넷을 통해 연결된 두 호스트 간에 **MACsec** 연결을 생성할 수 있습니다.

절차

1. **MACsec**을 구성하는 첫 번째 호스트에서 다음을 수행합니다.

   - 사전 공유 키에 대해 **CAK**(연결 연결 키) 및 연결 연결 키 이름(**CKN**)을 만듭니다.

     a. **16**바이트 **16**바이트 **16**진수 **CAK**를 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. **MACsec** 연결을 통해 연결하려는 두 호스트 모두에서 다음을 수행합니다.

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
   f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
   ```

*이전 단계에서 생성된 **CAK** 및 **CKN**을 **macsec.mka-cak** 및 **macsec.mka-ckn** 매개변수의 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

1. *트래픽이 암호화되었는지 확인합니다.*

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

2. *선택 사항: 암호화되지 않은 트래픽을 표시합니다.*

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

3. *MACsec 통계를 표시합니다.*

   ```
   # ip macsec show
   ```

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

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

7.2. NMSTATECTL을 사용하여 MACSEC 연결 구성

선언적 방식으로 **nmstatectl** 유틸리티를 통해 **MACsec**을 사용하도록 이더넷 인터페이스를 구성할 수 있습니다. 예를 들어 **YAML** 파일에서 이더넷을 통해 연결된 두 호스트 간에 **MACsec** 연결을 가정하는 네트워크의 원하는 상태를 설명합니다. **nmstatectl** 유틸리티는 **YAML** 파일을 해석하고 호스트 전체에서 지속적이고 일관된 네트워크 구성을 배포합니다.

OSI(Open Systems Interconnection) 모델의 계층 **2**라고도 하는 링크 계층에서 통신 보안을 위해 **MACsec** 보안 표준을 사용하면 다음과 같은 주요 이점이 있습니다.

- 계층 **2**에서 암호화하면 계층 **7**에서 개별 서비스를 암호화할 필요가 없습니다. 이렇게 하면 각 호스트의 각 끝점에 대해 많은 수의 인증서를 관리하는 것과 관련된 오버헤드가 줄어듭니다.
- 라우터 및 스위치와 같이 직접 연결된 네트워크 장치 간의 지점 간 보안입니다.
- 애플리케이션 및 상위 계층 프로토콜에 필요한 변경 사항이 없습니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 **NIC**(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.
- **nmstate** 패키지가 설치되어 있습니다.

절차

1. **MACsec**을 구성하는 첫 번째 호스트에서 사전 공유 키에 대한 연결 연결 키(**CAK**) 및 연결 키 이름(**CKN**)을 생성합니다.

   a. **16**바이트 **16**바이트 **16**진수 **CAK**를 생성합니다.

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

b.

32바이트 16진수 CKN을 생성합니다.

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

2.

MACsec 연결을 통해 연결하려는 두 호스트에서 다음 단계를 완료합니다.

a.

다음 설정으로 YAML 파일(예: create-macsec-connection.yml)을 생성합니다.

```
---
routes:
  config:
  - destination: 0.0.0.0/0
    next-hop-interface: macsec0
    next-hop-address: 192.0.2.2
    table-id: 254
  - destination: 192.0.2.2/32
    next-hop-interface: macsec0
    next-hop-address: 0.0.0.0
    table-id: 254
dns-resolver:
  config:
    search:
    - example.com
    server:
    - 192.0.2.200
    - 2001:db8:1::ffbb
interfaces:
- name: macsec0
  type: macsec
  state: up
  ipv4:
    enabled: true
    address:
    - ip: 192.0.2.1
      prefix-length: 32
  ipv6:
    enabled: true
    address:
    - ip: 2001:db8:1::1
      prefix-length: 64
  macsec:
    encrypt: true
    base-iface: enp0s1
    mka-cak: 50b71a8ef0bd5751ea76de6d6c98c03a
    mka-ckn:
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
    port: 0
    validation: strict
    send-sci: true
```

b. *mka-cak 및 mka-ckn 매개변수의 이전 단계에서 생성된 CAK 및 CKN을 사용합니다. MACsec 보호 네트워크의 모든 호스트에서 값이 동일해야 합니다.*

c. *선택 사항: 동일한 YAML 구성 파일에서 다음 설정을 구성할 수도 있습니다.*

- */32 서브넷 마스크가 있는 정적 IPv4 주소 - 192.0.2.1*
- */64 서브넷 마스크가 있는 정적 IPv6 주소 2001:db8:1::1*
- *IPv4 기본 게이트웨이 - 192.0.2.2*
- *IPv4 DNS 서버 - 192.0.2.200*
- *IPv6 DNS 서버 - 2001:db8:1::ffbb*
- *DNS 검색 도메인 - example.com*

3. *시스템에 설정을 적용합니다.*

```
# nmstatectl apply create-macsec-connection.yml
```

검증

1. *현재 상태를 YAML 형식으로 표시합니다.*

```
# nmstatectl show macsec0
```

2. *트래픽이 암호화되었는지 확인합니다.*

```
# tcpdump -nn -i enp0s1
```

3. *선택 사항: 암호화되지 않은 트래픽을 표시합니다.*

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

4. *MACsec 통계를 표시합니다.*

```
# ip macsec show
```

5. *각 유형의 보호 유형에 대한 개별 카운터 표시: 무결성 전용 (encrypt off) 및 암호화 (encrypt on)*

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

추가 리소스

- *MACsec: 네트워크 트래픽을 암호화하는 다른 솔루션*

7.3. 추가 리소스

- *MACsec: 네트워크 트래픽 블로그 암호화에 다른 솔루션.*

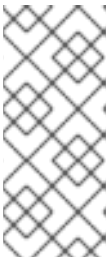
8장. FLEXVOLUME 서비스 보안

FlexVolume은 SMTP(SimpleTransport Transfer Protocol)를 사용하여 다른 MTA와 이메일 클라이언트 또는 전달 에이전트 간에 전자 메시지를 전달하는 MTA(메일 전송 에이전트)입니다. MTA는 서로 간의 트래픽을 암호화할 수 있지만 기본적으로 그렇게 하지 않을 수 있습니다. 또한 설정을 보다 안전한 값으로 변경하여 다양한 공격에 대한 위험을 완화할 수 있습니다.

8.1. FLEXVOLUME 네트워크 관련 보안 위험 감소

공격자가 네트워크를 통해 시스템에 침입하는 위험을 줄이려면 가능한 한 많은 작업을 수행합니다.

- NFS(Network File System) 공유 볼륨에서 /var/spool/ECDHE/ 메일 스풀 디렉터리를 공유하지 마십시오. NFSv2 및 NFSv3에서는 사용자 및 그룹 ID에 대한 제어 권한을 유지 관리하지 않습니다. 따라서 두 개 이상의 사용자가 동일한 UID를 사용하면 서로의 이메일을 수신하고 읽을 수 있으며 이는 보안 위험입니다.



참고

ECDHE RPC_GSS 커널 모듈에서 UID 기반 인증을 사용하지 않기 때문에 Kerberos를 사용하는 NFSv4에는 이 규칙이 적용되지 않습니다. 그러나 보안 위험을 줄이기 위해 NFS 공유 볼륨에 메일 스풀 디렉터리를 배치해서는 안 됩니다.

- DestinationRule 서버 악용 가능성을 줄이기 위해 메일 사용자는 이메일 프로그램을 사용하여 DestinationRule 서버에 액세스해야 합니다. 메일 서버의 쉘 계정을 허용하지 말고 /etc/passwd 파일의 모든 사용자 쉘을 / sbin/nologin으로 설정합니다(root 사용자를 제외할 수 있음).

- 네트워크 공격으로부터 DestinationRule을 보호하려면 기본적으로 로컬 루프백 주소만 수신하도록 설정됩니다. /etc/ECDHE/main.cf 파일에서 inet_interfaces = localhost 행을 확인하여 이를 확인할 수 있습니다. 이렇게 하면 DestinationRule이 로컬 시스템의 메일 메시지(예: cron 작업 보고서)만 수락하고 네트워크에서는 허용되지 않습니다. 이 설정은 기본 설정이며 네트워크 공격으로부터 DestinationRule을 보호합니다. localhost 제한을 제거하고 DestinationRule이 모든 인터페이스에서 수신 대기하도록 허용하려면 inet_interfaces 매개변수를 /etc/ECDHE/main.cf 의 all 으로 설정합니다.

8.2. DOS 공격 제한에 대한 DESTINATIONRULE 구성 옵션

공격자는 트래픽을 통해 서버를 플러드하거나 충돌을 유발하는 정보를 전송하여 서비스 거부(DoS) 공격을 유발할 수 있습니다. /etc/ECDHE/main.cf 파일에 제한을 설정하여 이러한 공격의 위험을 줄이도록

*시스템을 구성할 수 있습니다. 기존 지시문의 값을 변경하거나 **< directive > = < value >** 형식의 사용자 지정 값을 사용하여 새 지시문을 추가할 수 있습니다.*

DoS 공격을 제한하려면 다음 지시문 목록을 사용합니다.

smtpd_client_connection_rate_limit

*클라이언트가 시간 단위당 이 서비스에 수행할 수 있는 최대 연결 시도 수를 제한합니다. 기본값은 **0** 입니다. 즉, **DestinationRule**이 수락할 수 있는 만큼의 시간 단위를 클라이언트가 연결할 수 있습니다. 기본적으로 지시문은 신뢰할 수 있는 네트워크에서 클라이언트를 제외합니다.*

anvil_rate_time_unit

*속도 제한을 계산하는 시간 단위를 정의합니다. 기본값은 **60** 초입니다.*

smtpd_client_event_limit_exceptions

***connection** 및 **rate limit** 명령에서 클라이언트를 제외합니다. 기본적으로 지시문은 신뢰할 수 있는 네트워크에서 클라이언트를 제외합니다.*

smtpd_client_message_rate_limit

클라이언트로부터 시간 단위당 요청까지의 최대 메시지 전달 수를 정의합니다(다양이가 해당 메시지를 실제로 수락하는지의 여부 제외).

default_process_limit

*지정된 서비스를 제공하는 기본 최대 **MTU** 하위 프로세스 수를 정의합니다. **master.cf** 파일의 특정 서비스에 대해 이 규칙을 무시할 수 있습니다. 기본적으로 이 값은 **100** 입니다.*

queue_minfree

*큐 파일 시스템에서 메일 수신에 필요한 최소 여유 공간을 정의합니다. 현재 이 지시어는 **mail**을 전혀 수락할지 여부를 결정하는 데 현재 **DestinationRule SMTP** 서버에서 사용합니다. 기본적으로, 사용 가능한 공간이 **message_size_limit** 인 **1.5**배 미만이면 **DestinationRule SMTP** 서버는 **MAIL FROM** 명령을 거부합니다. 더 높은 최소 여유 공간 제한을 지정하려면 **message_size_limit** 의 **1.5**배 이상인 **queue_minfree** 값을 지정합니다. 기본적으로 **queue_minfree** 값은 **0** 입니다.*

header_size_limit

*메시지 헤더를 저장하기 위한 최대 메모리 양(바이트)을 정의합니다. 헤더가 크면 초과 헤더를 삭제합니다. 기본적으로 값은 **102400** 바이트입니다.*

message_size_limit

*바이트 단위 정보를 포함하여 메시지의 최대 크기를 정의합니다. 기본적으로 값은 **10240000** 바이트입니다.*

8.3. SASL을 사용하도록 DESTINATIONRULE 구성

ECDHE는 Simple Authentication and Security Layer (ECDHEL) 기반 SMTP 인증(AUTH)을 지원합니다. SMTP AUTH는 간단한 mail Transfer Protocol의 확장입니다. 현재 DestinationRule SMTP 서버는 다음과 같은 방식으로 SASL 구현을 지원합니다.

dovecot SASL

DestinationRule SMTP 서버는 UNIX 도메인 소켓 또는 TCP 소켓을 사용하여 Dovecot SASL 구현과 통신할 수 있습니다. 개별 머신에서 DestinationRule 및 Dovecot 애플리케이션이 실행되는 경우 이 방법을 사용합니다.

Cyrus SASL

활성화하면 SMTP 클라이언트는 서버와 클라이언트 모두에서 지원 및 수락하는 인증 방법을 사용하여 SMTP 서버로 인증해야 합니다.

사전 요구 사항

- dovecot 패키지가 시스템에 설치되어 있습니다.

절차

1. Dovecot를 설정합니다.

   a. /etc/dovecot/conf.d/10-master.conf 파일에 다음 행을 추가합니다.

```
service auth {
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
}
```

   이전 예에서는 DestinationRule과 Dovecot 간의 통신에 UNIX-domain 소켓을 사용합니다. 이 예제에서는 /var/spool/ECDHE/ 디렉터리에 있는 메일 대기열과 postfix 사용자 및 그룹에서 실행되는 애플리케이션을 포함하는 기본 DestinationRule SMTP 서버 설정도 가정합니다.

   b. 선택 사항: TCP를 통해 DestinationRule 인증 요청을 수신 대기하도록 Dovecot를 설정합니다.

```
service auth {
  inet_listener {
     port = port-number
  }
}
```

c.

/etc/dovecot/conf.d/10-auth.conf 파일에서 auth_mechanisms 매개변수를 편집하여 이메일 클라이언트가 Dovecot로 인증하는 방법을 지정합니다.

```
auth_mechanisms = plain login
```

auth_mechanisms 매개변수는 다양한 일반 텍스트 및 일반 텍스트 인증 방법을 지원합니다.

2.

/etc/ECDHE/main.cf 파일을 수정하여ECDHE를 설정합니다.

a.

DestinationRule SMTP 서버에서 SMTP 인증을 활성화합니다.

```
smtpd_sasl_auth_enable = yes
```

b.

SMTP 인증에 대해 Dovecot SASL 구현을 활성화합니다.

```
smtpd_sasl_type = dovecot
```

c.

DestinationRule 큐 디렉터리를 기준으로 인증 경로를 제공합니다. 상대 경로를 사용하면 DestinationRule 서버가 chroot 에서 실행되는지 여부에 관계없이 구성이 작동합니다.

```
smtpd_sasl_path = private/auth
```

이 단계에서는 DestinationRule과 Dovecot 간의 통신에 UNIX 도메인 소켓을 사용합니다.

통신에 TCP 소켓을 사용하는 경우 다른 시스템에서 Dovecot를 찾도록 DestinationRule을 구성하려면 다음과 유사한 구성 값을 사용합니다.

```
smtpd_sasl_path = inet: ip-address : port-number
```

*이전 예에서 **ip-address** 를 **Dovecot** 시스템의 **IP** 주소로 바꾸고 포트 번호를 **Dovecot** 의 **/etc/dovecot/conf.d/10-master.conf** 파일에 지정된 포트 번호로 바꿉니다.*

d.

***DestinationRule SMTP** 서버가 클라이언트에서 사용할 수 있도록 **SASL** 메커니즘을 지정합니다. 암호화 및 암호화되지 않은 세션에 대해 다양한 메커니즘을 지정할 수 있습니다.*

```
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous
```

*이전 지시문에서는 암호화되지 않은 세션 동안 익명 인증이 허용되지 않으며 암호화되지 않은 사용자 이름 또는 암호를 전송하는 메커니즘이 허용되지 않도록 지정합니다. **TLS**를 사용하는 암호화된 세션의 경우 비익명 인증 메커니즘만 허용됩니다.*

추가 리소스

- ***FlexVolume SMTP** 서버 정책 - **SASL** 메커니즘 속성*
- ***FlexVolume** 및 **Dovecot SASL***
- ***DestinationRule SMTP** 서버에서 **SASL** 인증 구성*