Red Hat Enterprise Linux 8

보안 강화

Red Hat Enterprise Linux 8 시스템의 보안 강화

Last Updated: 2024-07-26

Red Hat Enterprise Linux 8 보안 강화

Red Hat Enterprise Linux 8 시스템의 보안 강화

법적 공지

초록

로컬 및 원격 침입, 악용 및 악의적인 활동에 대해 Red Hat Enterprise Linux 서버 및 워크스테이션을 보호하는 프로세스와 관행에 대해 알아보십시오. 이러한 접근 방식과 도구를 사용하면 데이터 센터, 작업 공간 및 가정을 위한 보다 안전한 컴퓨팅 환경을 구축할 수 있습니다.

차례

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견에 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. **요약** 필드에 설명 제목을 입력합니다.
4. **설명** 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

1장. 설치 중 RHEL 보안

Red Hat Enterprise Linux 설치를 시작하기 전에 이미 보안 대응이 시작됩니다. 처음부터 안전하게 시스템을 구성하면 나중에 추가 보안 설정을 더 쉽게 구현할 수 있습니다.

1.1. 디스크 파티션 설정

디스크 파티셔닝에 대한 권장 사례는 베어 메탈 시스템에 설치하고 이미 설치된 운영 체제를 포함하는 가상 디스크 하드웨어 및 파일 시스템 조정을 지원하는 가상화 또는 클라우드 환경의 경우 다릅니다.

베어 메탈 설치에서 데이터를 분리하고 보호하려면 **/boot**, / , **/home**,/ tmp , **/var**/**tmp** / 디렉토리에 대한 별도의 파티션을 만듭니다.

/boot

이 파티션은 부팅 중에 시스템에서 읽은 첫 번째 파티션입니다. 시스템을 Red Hat Enterprise Linux 8로 부팅하는 데 사용되는 부트 로더 및 커널 이미지는 이 파티션에 저장됩니다. 이 파티션은 암호화해서는 안 됩니다. 이 파티션이 / 에 포함되어 있고 해당 파티션을 암호화하거나 사용할 수 없게 되면 시스템을 부팅할 수 없습니다.

/home

사용자 데이터(**/home**)가 별도의 파티션 대신 /에 저장되어 파티션이 채워지면 운영 체제가 불안정해집니다. 또한 시스템을 Red Hat Enterprise Linux 8의 다음 버전으로 업그레이드할 때 데이터를 **/home** 파티션에 덮어쓰지 않으므로 업그레이드가 더 쉽습니다. 루트 파티션(/)이 손상되면 데이터가 영구적으로 손실될 수 있습니다. 별도의 파티션을 사용하면 데이터 손실을 조금 더 완화할 수 있습니다. 이 파티션을 빈번한 백업의 대상으로 지정할 수도 있습니다.

/tmp 및 /var/tmp/

/tmp 및 **/var/tmp/** 디렉터리는 모두 장기간 저장하지 않아도 되는 데이터를 저장하는 데 사용됩니다. 그러나 이러한 디렉토리 중 하나에 많은 데이터가 범람하는 경우 모든 스토리지 공간을 소비할 수 있습니다. 이 경우 이러한 디렉토리가 / 내에 저장되면 시스템이 불안정해 충돌할 수 있습니다. 따라서 이러한 디렉터리를 해당 파티션으로 이동하는 것이 좋습니다.

가상 머신 또는 클라우드 인스턴스의 경우 별도의 **/boot**,**/home**,**/tmp** 및 **/var/tmp** 파티션은 가상 디스크 크기 및 / 파티션을 늘릴 수 있기 때문에 선택 사항입니다. 적절하게 가상 디스크 크기를 늘리기 전에는 가상 디스크 크기를 늘리기 전에는 / 파티션 사용을 정기적으로 점검하도록 모니터링을 설정합니다.

참고

설치 프로세스 중에 파티션을 암호화할 수 있는 옵션이 있습니다. 암호를 제공해야 합니다. 이 암호는 파티션의 데이터를 보호하는 데 사용되는 대량 암호화 키의 잠금을 해제하는 키 역할을 합니다.

1.2. 설치 프로세스 중에 네트워크 연결 제한

Red Hat Enterprise Linux 8을 설치할 때 설치 미디어는 특정 시간에 시스템의 스냅샷을 나타냅니다. 이로 인해 최신 보안 수정 사항이 최신 상태가 아닐 수 있으며 설치 미디어에서 제공한 시스템이 릴리스된 후에만 수정된 특정 문제에 취약해질 수 있습니다.

잠재적으로 취약한 운영 체제를 설치하는 경우 항상 가장 필요한 네트워크 영역으로만 노출을 제한합니다. 가장 안전한 선택은 "네트워크 없음" 영역으로, 설치 프로세스 중에 시스템의 연결이 끊어진 상태로 두는 것을 의미합니다. 인터넷 연결이 가장 위험한 경우에는 LAN 또는 인트라넷 연결만으로도 충분합니다. 최상의 보안 사례를 따르려면 네트워크에서 Red Hat Enterprise Linux 8을 설치하는 동안 리포지토리에서 가장 가까운 영역을 선택하십시오.

1.3. 필요한 최소 패키지 설치

컴퓨터에 있는 각 소프트웨어에 취약점이 있을 수 있으므로 사용할 패키지만 설치하는 것이 좋습니다. DVD 미디어에서 설치하는 경우 설치 중에 설치할 패키지를 정확하게 선택할 수 있습니다. 다른 패키지가 필요한 경우 나중에 시스템에 항상 추가할 수 있습니다.

1.4. 설치 후 절차

다음 단계는 Red Hat Enterprise Linux 8을 설치한 직후 수행해야 하는 보안 관련 절차입니다.

- 시스템을 업데이트합니다. root로 다음 명령을 입력합니다.

```
# yum update
```

- 방화벽 서비스 **firewalld** 는 Red Hat Enterprise Linux 설치를 통해 자동으로 활성화되지만 Kickstart 구성에서는 명시적으로 비활성화할 수 있습니다. 이러한 경우 방화벽을 다시 활성화합니다.
  firewalld를 시작하려면 root로 다음 명령을 입력합니다.

```
# systemctl start firewalld
# systemctl enable firewalld
```

- 보안을 강화하려면 필요하지 않은 서비스를 비활성화합니다. 예를 들어 컴퓨터에 프린터가 설치되어 있지 않은 경우 다음 명령을 사용하여 **cups** 서비스를 비활성화합니다.

```
# systemctl disable cups
```

  활성 서비스를 검토하려면 다음 명령을 입력합니다.

```
$ systemctl list-units | grep service
```

2장. RHEL을 FIPS 모드로 전환

FIPS(Federal Information Processing Standard) 140-2에서 요구하는 암호화 모듈 자체 점검을 활성화하려면 FIPS 모드에서 RHEL 8을 실행해야 합니다. FIPS 규정 준수를 목표로 하는 경우 FIPS 모드에서 설치를 시작하는 것이 좋습니다.

2.1. 연방 정보 처리 표준 140 및 FIPS 모드

FIPS(Federal Information Processing Standards) 발행 140는 암호화 모듈의 품질을 보장하기 위해 NIST(National Institute of Standards and Technology)에서 개발한 일련의 컴퓨터 보안 표준입니다. FIPS 140 표준을 사용하면 암호화 도구가 알고리즘이 올바르게 구현됩니다. 런타임 암호화 알고리즘 및 무결성 자체 테스트는 시스템이 표준 요구 사항을 충족하는 암호화를 사용하도록 하는 몇 가지 메커니즘입니다.

RHEL 시스템이 FIPS 승인 알고리즘에서만 모든 암호화 키를 생성하고 사용하는지 확인하려면 RHEL을 FIPS 모드로 전환해야 합니다.

다음 방법 중 하나를 사용하여 FIPS 모드를 활성화할 수 있습니다.

- FIPS 모드에서 설치 시작
- 설치 후 FIPS 모드로 시스템 전환

FIPS 컴플라이언스를 사용하려면 FIPS 모드에서 설치를 시작합니다. 이렇게 하면 이미 배포된 시스템 변환과 관련된 결과 시스템의 규정 준수를 위한 암호화 주요 자료 재생성 및 재평가를 방지할 수 있습니다.

FIPS 호환 시스템을 작동하려면 FIPS 모드에서 모든 암호화 키 자료를 생성합니다. 또한 암호화 키 자료는 안전하게 래핑되고 비FIPS 환경에서 래핑되지 않는 한 FIPS 환경을 남겨 두지 않아야 합니다.

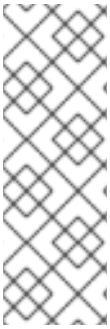
fips-mode-setup 툴을 사용하여 시스템을 FIPS 모드로 전환해도 FIPS 140 표준을 준수하지 않습니다. 시스템을 FIPS 모드로 설정한 후 모든 암호화 키를 다시 생성할 수 없을 수 있습니다. 예를 들어 사용자의 암호화 키가 있는 기존 IdM 영역의 경우 모든 키를 다시 생성할 수 없습니다. FIPS 모드에서 설치를 시작할 수 없는 경우 설치 후 구성 단계를 수행하거나 워크로드를 설치하기 전에 설치 후 첫 번째 단계로 항상 FIPS 모드를 활성화합니다.

fips-mode-setup 툴에서는 내부적으로 **FIPS** 시스템 전체 암호화 정책도 사용합니다. 그러나 **update-crypto-policies --set FIPS** 명령이 수행하는 작업 위에 **fips- finish-install** 도구를 사용하여 FIPS dracut 모듈을 설치할 수 있습니다. **fips=1** 부팅 옵션도 커널 명령줄에 추가하고 초기 RAM 디스크를 다시 생성합니다.

또한 FIPS 모드에서 필요한 제한 적용은 **/proc/sys/crypto/fips_enabled** 파일의 콘텐츠에 따라 다릅니다. 파일에 **1** 개의 암호화 구성 요소가 포함된 경우 RHEL 코어 암호화 구성 요소는 FIPS 승인 암호화 알고리즘 구현만 사용하는 모드로 전환됩니다. **/proc/sys/crypto/fips_enabled** 에 **0** 이 포함된 경우 암호화 구성 요소에서 FIPS 모드를 활성화하지 않습니다.

FIPS 시스템 전체 암호화 정책은 높은 수준의 제한을 구성하는 데 도움이 됩니다. 따라서 암호화 민첩성을 지원하는 통신 프로토콜은 선택한 경우 시스템이 거부하는 암호를 알리지 않습니다. 예를 들어,keCha20 알고리즘은 FIPS 승인되지 않으며 **FIPS** 암호화 정책은 TLS 서버 및 클라이언트가 TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 TLS 암호화 제품군을 발표하지 않도록 합니다.

FIPS 모드에서 RHEL을 작동하고 자체 FIPS 모드 관련 구성 옵션을 제공하는 애플리케이션을 사용하는 경우 이러한 옵션과 해당 애플리케이션 지침을 무시합니다. FIPS 모드에서 실행되는 시스템 및 시스템 전체 암호화 정책은 FIPS 호환 암호화만 적용합니다. 예를 들어, 시스템이 FIPS 모드에서 실행되는 경우 Node.js 구성 옵션 **--enable-fips** 는 무시됩니다. FIPS 모드에서 실행되지 않는 시스템에서 **--enable-fips** 옵션을 사용하는 경우 FIPS-140 규정 준수 요구 사항을 충족하지 않습니다.



참고

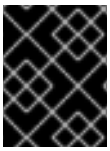
FIPS 모드에서 RHEL 8.6 (및 최신) 커널은 FIPS 140-3을 준수하도록 설계되었지만 NIST (National Institute of Standards and Technology) Cryptographic Module Validation Program (CMVP)의 인증을 받지 않았습니다. 최신 인증된 커널 모듈은 RHSA-2021:4356 권고 업데이트 후 업데이트된 RHEL 8.5 커널입니다. 해당 인증은 FIPS 140-2 표준에 적용됩니다. 암호화 모듈이 FIPS 140-2 또는 140-3를 준수하는지 여부를 선택할 수 없습니다. 규정 준수 활동 및 정부 표준 지식 베이스 문서의 FIPS 140-2 및 FIPS 140-3 섹션에서는 선택한 RHEL 마이너 릴리스의 암호화 모듈에 대한 검증 상태에 대한 개요를 제공합니다.

추가 리소스

- 규정 준수 활동 및 정부 표준 지식 베이스 문서의 FIPS 140-2 및 FIPS 140-3 섹션
- RHEL 시스템 전체 암호화 정책
- NIST Computer Security Resource Center에서 FIPS 발행
- 연방 정보 처리 표준 발행: FIPS 140-3

2.2. FIPS 모드가 활성화된 시스템 설치

FIPS(Federal Information Processing Standard) 140에서 요구하는 암호화 모듈 자체 점검을 활성화하려면 시스템 설치 중에 FIPS 모드를 활성화합니다.



중요

RHEL 설치 중에 FIPS 모드를 활성화하면 시스템이 FIPS 승인 알고리즘 및 지속적인 모니터링 테스트를 사용하여 모든 키를 생성합니다.

주의

FIPS 모드 설정을 완료한 후에는 시스템을 일관되지 않은 상태로 전환하지 않고 FIPS 모드를 전환할 수 없습니다. 시나리오에 이러한 변경이 필요한 경우 유일한 올바른 방법은 시스템을 완전히 다시 설치하는 것입니다.

절차

1. 시스템 설치 중에 커널 명령줄에 **fips=1** 옵션을 추가합니다.
2. 소프트웨어 선택 단계에서는 타사 소프트웨어를 설치하지 마십시오.
3. 설치 후 FIPS 모드에서 시스템이 자동으로 시작됩니다.

검증

- 시스템이 시작된 후 FIPS 모드가 활성화되었는지 확인합니다.

```
$ fips-mode-setup --check
FIPS mode is enabled.
```

추가 리소스

- 부팅 옵션 편집

2.3. 시스템을 FIPS 모드로 전환

시스템 전체 암호화 정책에는 FIPS(Federal Information Processing Standard) 발행 140의 요구 사항에 따라 암호화 알고리즘을 활성화하는 정책 수준이 포함되어 있습니다. FIPS 모드를 활성화하거나 비활성화하는 **fips-mode-setup** 툴은 **FIPS** 시스템 전체 암호화 정책을 내부적으로 사용합니다.

FIPS 시스템 전체 암호화 정책을 사용하여 시스템을 **FIPS** 모드로 전환해도 FIPS 140 표준을 준수하는 것은 아닙니다. 시스템을 FIPS 모드로 설정한 후 모든 암호화 키를 다시 생성할 수 없을 수 있습니다. 예를 들어 사용자의 암호화 키가 있는 기존 IdM 영역의 경우 모든 키를 다시 생성할 수 없습니다.

중요

RHEL 설치 중에 FIPS 모드를 활성화하면 시스템이 FIPS 승인 알고리즘 및 지속적인 모니터링 테스트를 사용하여 모든 키를 생성합니다.

fips-mode-setup 툴은 내부적으로 **FIPS** 정책을 사용합니다. 그러나 **--set FIPS** 옵션을 사용하는 **update-crypto-policies** 명령에서 **fips-mode-setup** 을 사용하면 **fips-finish-install** 도구를 사용하여 FIPS dracut 모듈을 설치할 수 있습니다. 또한 **fips=1** 부팅 옵션도 커널 명령줄에 다시 생성되고 초기 RAM 디스크를 다시 생성합니다.

주의

FIPS 모드 설정을 완료한 후에는 시스템을 일관되지 않은 상태로 전환하지 않고 FIPS 모드를 전환할 수 없습니다. 시나리오에 이러한 변경이 필요한 경우 유일한 올바른 방법은 시스템을 완전히 다시 설치하는 것입니다.

절차

1. 시스템을 FIPS 모드로 전환하려면 다음을 수행합니다.

```
# fips-mode-setup --enable
Kernel initramdisks are being regenerated. This might take some time.
Setting system policy to FIPS
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place.
FIPS mode will be enabled.
Please reboot the system for the setting to take effect.
```

2. 커널이 FIPS 모드로 전환되도록 시스템을 다시 시작하십시오.

```
# reboot
```

검증

- 재시작 후 FIPS 모드의 현재 상태를 확인할 수 있습니다.

```
# fips-mode-setup --check
FIPS mode is enabled.
```

추가 리소스

- **fips-mode-setup(8)** 도움말 페이지
- NIST(National Institute of Standards and Technology) 웹 사이트의 암호화 모듈에 대한 보안 요구 사항.

2.4. 컨테이너에서 FIPS 모드 활성화

FIPS 모드(Federal Information Processing Standard)에서 요구하는 전체 암호화 모듈 자체 점검을 활성화하려면 호스트 시스템 커널이 FIPS 모드에서 실행되어야 합니다. 호스트 시스템의 버전에 따라 컨테이너에서 FIPS 모드를 활성화하는 것은 완전히 자동화되거나 하나의 명령만 필요합니다.

fips-mode-setup 명령은 컨테이너에서 제대로 작동하지 않으며 이 시나리오에서는 FIPS 모드를 활성화하거나 확인하는 데 사용할 수 없습니다.

사전 요구 사항

- 호스트 시스템은 FIPS 모드여야 합니다.

절차

- **RHEL 8.1 및 8.2를 실행하는 호스트에서** 다음을 수행합니다. 다음 명령을 사용하여 컨테이너에서 FIPS 암호화 정책 수준을 설정하고 **fips-mode-setup** 명령을 사용하는 조언을 무시합니다.

```
$ update-crypto-policies --set FIPS
```

- **RHEL 8.4 이상을 실행하는 호스트에서 다음을 수행합니다.** FIPS 모드가 활성화된 시스템에서 **podman** 유틸리티는 지원되는 컨테이너에서 FIPS 모드를 자동으로 활성화합니다.

추가 리소스

- 시스템을 FIPS 모드로 전환합니다.
- FIPS 모드에서 시스템 설치

2.5. FIPS 140-2와 호환되지 않는 암호화를 사용하는 RHEL 애플리케이션 목록

FIPS 140와 같은 모든 관련 암호화 인증을 전달하려면 코어 암호화 구성 요소 세트에서 라이브러리를 사용합니다. **libgcrypt** 를 제외한 이러한 라이브러리는 RHEL 시스템 전체 암호화 정책도 따릅니다.

핵심 암호화 구성 요소에 대한 개요, 선택한 방법, 운영 체제에 어떻게 통합되는지, 하드웨어 보안 모듈 및 스마트 카드를 지원하는 방법, 암호화 인증이 적용되는 방법에 대한 정보는 RHEL 핵심 암호화 구성 요소 문서를 참조하십시오.

다음 표 외에도 일부 RHEL 8 Z-stream 릴리스(예: 8.1.1)에서 Firefox 브라우저 패키지가 업데이트되었으며 NSS 암호화 라이브러리의 별도의 사본이 포함되어 있습니다. 이렇게 하면 Red Hat은 패치 릴리스에서 이

러한 낮은 수준의 구성 요소 재구성이 중단되는 것을 피하고자 합니다. 결과적으로 이러한 Firefox 패키지는 FIPS 140-2 검증 모듈을 사용하지 않습니다.

FIPS 140-2와 호환되지 않는 암호화를 사용하는 RHEL 8 애플리케이션 목록

FreeRADIUS

RADIUS 프로토콜은 MD5를 사용합니다.

Ghostscript

사용자 정의 암호화 구현(MD5, RC4, SHA-2, AES)은 문서를 암호화하고 해독합니다.

iPXE

TLS의 암호화 스택은 컴파일되지만 사용되지 않습니다.

Libica

CPACF 명령어를 통한 RSA 및 ECDH와 같은 다양한 알고리즘에 대한 소프트웨어 폴백입니다.

OVMF (UEFI 펌웨어), Edk2, shim

전체 암호화 스택(OpenSSL 라이브러리의 임베디드 사본).

Perl

HMAC, HMAC-SHA1, HMAC-MD5, SHA-1, SHA-224,...

Pidgin

DES 및 RC4를 구현합니다.

QAT 엔진

암호화 프리미티브의 혼합 하드웨어 및 소프트웨어 구현(RSA, EC, DH, AES,...).

samba [1]

AES, DES 및 RC4를 구현합니다.

SWTPM

OpenSSL 사용에서 FIPS 모드를 명시적으로 비활성화합니다.

valgrind

AES, 해시 [2]

zip

암호를 사용하여 아카이브를 암호화하고 해독하기 위한 사용자 정의 암호화 구현(비보안 PKWARE 암호화 알고리즘).

추가 리소스

- 규정 준수 활동 및 정부 표준 지식 베이스 문서의 FIPS 140-2 및 FIPS 140-3 섹션
- RHEL 핵심 암호화 구성 요소 지식 베이스 문서

[1] RHEL 8.3부터는 FIPS 호환 암호화를 사용합니다.

[2] AES-NI와 같은 소프트웨어 하드웨어 오프로드 작업에 다시 구현합니다.

3장. 시스템 전체 암호화 정책 사용

시스템 전체의 암호화 정책은 TLS, IPsec, SSH, DNSSec 및 Kerberos 프로토콜을 다루는 코어 암호화 하위 시스템을 구성하는 시스템 구성 요소입니다. 관리자가 선택할 수 있는 몇 가지 정책 세트를 제공합니다.

3.1. 시스템 전체 암호화 정책

시스템 전체 정책이 설정되면 RHEL의 애플리케이션은 이를 따르며 애플리케이션을 명시적으로 요청하지 않는 한, 정책을 준수하지 않는 알고리즘과 프로토콜을 사용하지 않습니다. 즉, 이 정책은 시스템 제공 구성으로 실행할 때 애플리케이션의 기본 동작에 적용되지만 필요한 경우 이를 재정의할 수 있습니다.

RHEL 8에는 다음과 같은 사전 정의된 정책이 포함되어 있습니다.

DEFAULT

기본 시스템 전체 암호화 정책 수준은 현재 위협 모델에 대한 보안 설정을 제공합니다. 이 보안 설정은 TLS 1.2 및 1.3 프로토콜과 IKEv2 및 SSH2 프로토콜을 허용합니다. RSA 키와 Diffie-Hellman 매개변수는 2048비트 이상인 경우 허용됩니다.

LEGACY

Red Hat Enterprise Linux 5 및 이전 버전과의 호환성을 극대화하며 공격 면적이 증가하여 보안이 떨어집니다. **DEFAULT** 수준 알고리즘 및 프로토콜 외에도 TLS 1.0 및 1.1 프로토콜 지원이 포함됩니다. 알고리즘 DSA, 3DES, RC4는 사용할 수 있지만 RSA 키와 Diffie-Hellman 매개변수는 최소 1023비트인 경우 허용됩니다.

FUTURE

가능한 향후 정책을 테스트하기 위한 보다 엄격한 미래 지향 보안 수준입니다. 이 정책은 서명 알고리즘에서 SHA-1을 사용할 수 없습니다. 이 보안 설정은 TLS 1.2 및 1.3 프로토콜과 IKEv2 및 SSH2 프로토콜을 허용합니다. RSA 키와 Diffie-Hellman 매개변수는 최소 3072비트인 경우 허용됩니다. 시스템이 공용 인터넷에서 통신할 경우 상호 운용성 문제에 직면할 수 있습니다.

중요

고객 포털 API의 인증서에서 사용하는 암호화 키가 **FUTURE** 시스템 전체 암호화 정책의 요구 사항을 충족하지 않으므로 **redhat-support-tool** 유틸리티는 현재 이 정책 수준에서 작동하지 않습니다.

이 문제를 해결하려면 고객 포털 API에 연결하는 동안 **DEFAULT** 암호화 정책을 사용합니다.

FIPS

FIPS 140 요구 사항을 준수합니다. RHEL 시스템을 FIPS 모드로 전환하는 **fips-mode-setup** 툴은 내부적으로 이 정책을 사용합니다. **FIPS** 정책으로 전환해도 FIPS 140 표준을 준수하지 않습니다. 또한 시스템을 FIPS 모드로 설정한 후 모든 암호화 키를 다시 생성해야 합니다. 많은 경우에서는 이 작업을 수행할 수 없습니다.
RHEL은 CC(Common Criteria) 인증에 필요한 암호화 알고리즘에 대한 추가 제한이 포함된 **FIPS:OSPP** 시스템 전체 하위 정책을 제공합니다. 이 하위 정책을 설정한 후 시스템이 상호 운용성이 떨어집니다. 예를 들어 3072비트, 추가 SSH 알고리즘 및 여러 TLS 그룹보다 짧은 RSA 및 DH 키를 사용할 수 없습니다. **FIPS:OSPP** 를 설정하면 Red Hat CDN(Content Delivery Network) 구조에 연결할 수 없습니다. 또한 **FIPS:OSPP** 를 사용하는 IdM 배포에는 AD(Active Directory)를 통합할 수 없으며 **FIPS:OSPP** 및 AD 도메인을 사용하는 RHEL 호스트 간 통신이 작동하지 않거나 일부 AD 계정이 인증할 수 없을 수 있습니다.

참고

FIPS:OSPP 암호화 하위 정책을 설정한 후에는 시스템이 CC와 호환되지 않습니다. RHEL 시스템을 CC 표준을 준수하는 유일한 방법은 **cc-config** 패키지에 제공된 지침을 따르는 것입니다. 인증된 RHEL 버전, 검증 보고서 및 CC 가이드 링크 목록은 규정 준수 활동 및 정부 표준 지식 베이스 문서의 Common Criteria 섹션을 참조하십시오.

Red Hat은 **LEGACY** 정책을 사용하는 경우를 제외하고 모든 라이브러리가 보안 기본값을 제공하도록 모든 정책 수준을 지속적으로 조정합니다. **LEGACY** 프로필은 보안 기본값을 제공하지 않지만 쉽게 사용할 수 있는 알고리즘은 포함되지 않습니다. 따라서 Red Hat Enterprise Linux의 라이프 사이클 기간 동안 제공되는 정책에서 활성화된 알고리즘이나 사용 가능한 주요 크기 세트가 변경될 수 있습니다.

이러한 변경 사항은 새로운 보안 표준과 새로운 보안 연구를 반영합니다. Red Hat Enterprise Linux의 전체 수명 동안 특정 시스템과의 상호 운용성을 보장해야 하는 경우 시스템과 상호 작용하는 구성 요소에 대한 시스템 전체 암호화 정책을 비활성화하거나 사용자 지정 암호화 정책을 사용하여 특정 알고리즘을 다시 활성화해야 합니다.

정책 수준에서 허용되는 대로 설명된 특정 알고리즘 및 암호는 애플리케이션에서 지원하는 경우에만 사용할 수 있습니다.

표 3.1. 암호화 정책에서 활성화된 암호화 제품군 및 프로토콜

	LEGACY	DEFAULT	FIPS	FUTURE
IKEv1	제공되지 않음	제공되지 않음	제공되지 않음	제공되지 않음
3DES	제공됨	제공되지 않음	제공되지 않음	제공되지 않음
RC4	제공됨	제공되지 않음	제공되지 않음	제공되지 않음
DH	최소 1024비트	최소 2048비트	최소 2048비트[a]	최소 3072비트
RSA	최소 1024비트	최소 2048비트	최소 2048비트	최소 3072비트
DSA	제공됨	제공되지 않음	제공되지 않음	제공되지 않음
TLS v1.0	제공됨	제공되지 않음	제공되지 않음	제공되지 않음
TLS v1.1	제공됨	제공되지 않음	제공되지 않음	제공되지 않음
디지털 서명의 SHA-1	제공됨	제공됨	제공되지 않음	제공되지 않음
CBC 모드 암호	제공됨	제공됨	제공됨	제공되지 않음[b]
키 < 256비트인 대칭 암호	제공됨	제공됨	제공됨	제공되지 않음
인증서의 SHA-1 및 SHA-224 서명	제공됨	제공됨	제공됨	제공되지 않음

LEGACY	DEFAULT	FIPS	FUTURE

[a] RFC 7919 및 RFC 3526에 정의된 Diffie-Hellman 그룹만 사용할 수 있습니다.

[b] CBC 암호가 TLS에 대해 비활성화되어 있습니다. 비 TLS 시나리오에서는 **AES-128-CBC**가 비활성화되지만 **AES-256-CBC**가 활성화됩니다. **AES-256-CBC**도 비활성화하려면 사용자 지정 하위 정책을 적용합니다.

추가 리소스

- 시스템의 **crypto-policies(7)** 및 **update-crypto-policies(8)** 도움말 페이지

3.2. 시스템 전체 암호화 정책 변경

update-crypto-policies 도구를 사용하여 시스템에서 시스템 전체 암호화 정책을 변경하고 시스템을 다시 시작할 수 있습니다.

사전 요구 사항

- 시스템에 대한 root 권한이 있습니다.

절차

1. 선택 사항: 현재 암호화 정책을 표시합니다.

```
$ update-crypto-policies --show
DEFAULT
```

2. 새 암호화 정책을 설정합니다.

```
# update-crypto-policies --set <POLICY>
<POLICY>
```

< ***POLICY*** >를 설정할 정책 또는 하위 정책(예: **FUTURE**,**LEGACY** 또는 **FIPS:OSPP**)으로 바꿉니다.

3. 시스템을 다시 시작하십시오.

```
# reboot
```

검증

- 현재 암호화 정책을 표시합니다.

```
$ update-crypto-policies --show
<POLICY>
```

추가 리소스

- 시스템 전체 암호화 정책에 대한 자세한 내용은 시스템 전체 암호화 정책을 참조하십시오.

3.3. 시스템 전체 암호화 정책을 이전 릴리스와 호환되는 모드로 전환

Red Hat Enterprise Linux 8의 기본 시스템 전체 암호화 정책은 이전의 안전하지 않은 프로토콜을 사용한 통신을 허용하지 않습니다. Red Hat Enterprise Linux 6 및 이전 릴리스와 호환되어야 하는 환경의 경우 **LEGACY** 정책 수준을 사용할 수 있습니다.

주의

LEGACY 정책 수준으로 전환하면 덜 안전한 시스템 및 애플리케이션이 됩니다.

절차

1. 시스템 전체 암호화 정책을 **LEGACY** 수준으로 전환하려면 **root**로 다음 명령을 입력합니다.

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

추가 리소스

- 사용 가능한 암호화 정책 수준 목록은 **update-crypto-policies(8)** 도움말 페이지를 참조하십시오.
- 사용자 지정 암호화 정책을 정의하려면 **update-crypto-policies(8)** 도움말 페이지의 **Custom Policies** 섹션 및 **crypto-policies(7)** 도움말 페이지의 **Crypto Policy Definition Format** 섹션을 참조하십시오.

3.4. 웹 콘솔에서 시스템 전체 암호화 정책 설정

RHEL 웹 콘솔 인터페이스에서 직접 시스템 전체 암호화 정책 및 하위 정책 중 하나를 설정할 수 있습니다. 사전 정의된 시스템 전체 암호화 정책 외에도 그래픽 인터페이스를 통해 다음과 같은 정책 및 하위 정책 조합을 적용할 수도 있습니다.

DEFAULT:SHA1

SHA-1 알고리즘이 활성화된 **DEFAULT** 정책입니다.

LEGACY:AD-SUPPORT

Active Directory 서비스의 상호 운용성을 개선하는 보안 설정이 적은 **LEGACY** 정책입니다.

FIPS:OSPP

정보 기술 보안 평가 표준에 대해 Common Criteria에 필요한 추가 제한이 있는 **FIPS** 정책입니다.

주의

FIPS:OSPP 시스템 전체 하위 정책에는 CC(Common Criteria) 인증에 필요한 암호화 알고리즘에 대한 추가 제한이 포함되어 있으므로 설정한 후 시스템이 상호 운용이 불가능합니다. 예를 들어 3072비트, 추가 SSH 알고리즘 및 여러 TLS 그룹보다 짧은 RSA 및 DH 키를 사용할 수 없습니다. **FIPS:OSPP** 를 설정하면 Red Hat CDN(Content Delivery Network) 구조에 연결할 수 없습니다. 또한 **FIPS:OSPP** 를 사용하는 IdM 배포에는 AD(Active Directory)를 통합할 수 없으며 **FIPS:OSPP** 및 AD 도메인을 사용하는 RHEL 호스트 간 통신이 작동하지 않거나 일부 AD 계정이 인증할 수 없을 수 있습니다.

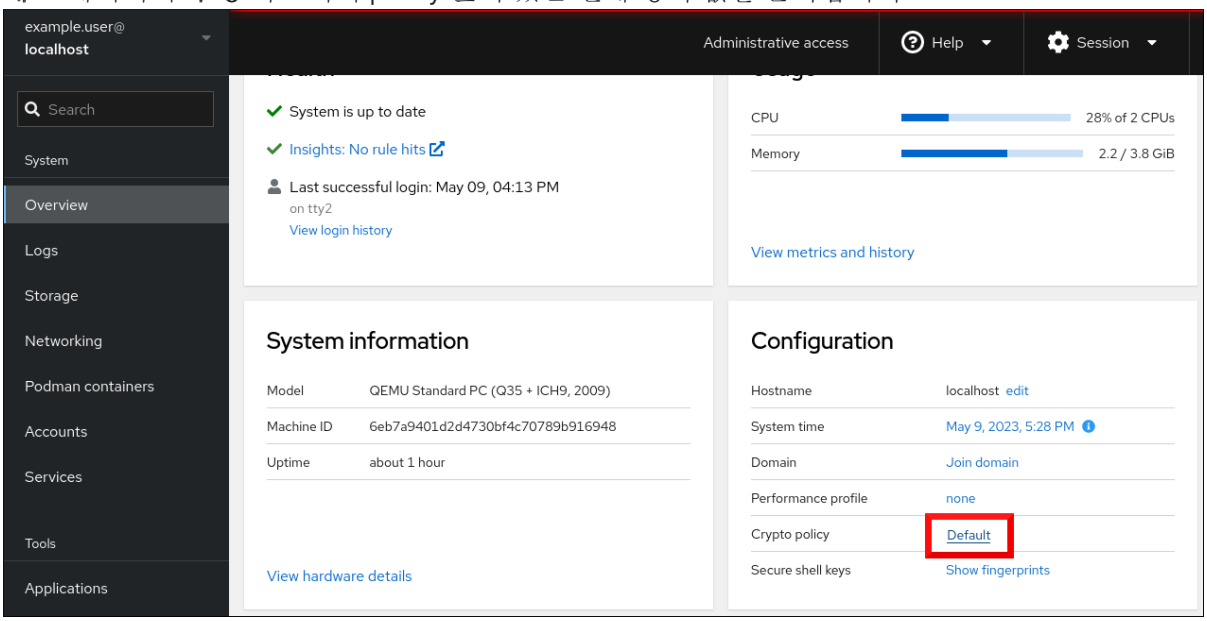
FIPS:OSPP 암호화 하위 정책을 설정한 후에는 **시스템이 CC와 호환되지 않습니다.** RHEL 시스템을 CC 표준을 준수하는 유일한 방법은 **cc-config** 패키지에 제공된 지침을 따르는 것입니다. 인증된 RHEL 버전, 검증 보고서, NIAP(National Information Assurance Partnership) 웹 사이트에서 호스팅되는 CC 가이드 링크 목록은 규정 준수 활동 및 정부 표준 지식 베이스 문서의 Common Criteria 섹션을 참조하십시오.

사전 요구 사항

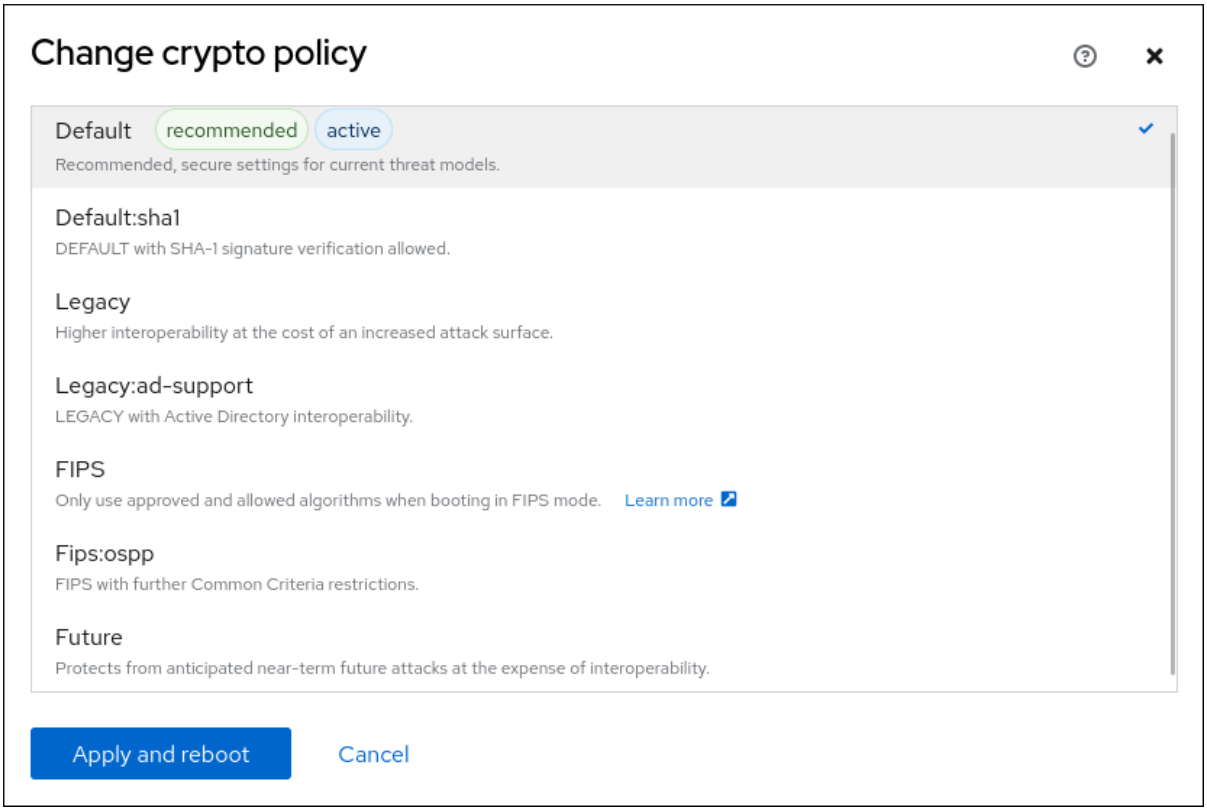
- RHEL 8 웹 콘솔이 설치되었습니다. 자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.
- **sudo** 를 사용하여 관리 명령을 입력할 수 있는 **루트** 권한 또는 권한이 있습니다.

절차

1. 웹 콘솔에 로그인합니다. 자세한 내용은 웹 콘솔에 로그인을 참조하십시오.
2. **개요** 페이지의 **구성** 카드에서 policy 옆에 있는 현재 정책 값을 클릭합니다.



3. **암호화 정책 변경** 대화 상자에서 **시스템에서 사용을 시작할 정책을 클릭합니다.**



4. 적용 및 재부팅 버튼을 클릭합니다.

검증

- 다시 시작한 후 웹 콘솔에 다시 로그인하고 Crypto 정책 값이 선택한 값에 해당하는지 확인합니다. 또는 **update-crypto-policies --show** 명령을 입력하여 현재 시스템 전체 암호화 정책을 터미널에 표시할 수 있습니다.

3.5. 다음 시스템 전체 암호화 정책에서 애플리케이션 제외

애플리케이션에서 직접 지원되는 암호화 제품군 및 프로토콜을 구성하여 애플리케이션에서 사용하는 암호화 설정을 사용자 지정할 수 있습니다.

애플리케이션과 관련된 심볼릭 링크를 **/etc/crypto-policies/back-ends** 디렉터리에서 제거하고 사용자 지정 암호화 설정으로 바꿀 수도 있습니다. 이 구성을 사용하면 제외된 백엔드를 사용하는 애플리케이션의 시스템 전체 암호화 정책을 사용할 수 없습니다. 또한 이러한 수정은 Red Hat에서 지원하지 않습니다.

3.5.1. 시스템 차원의 암호화 정책 옵트아웃의 예

wget

wget 네트워크 다운로드자가 사용하는 암호화 설정을 사용자 지정하려면**--secure-protocol** 및 **--ciphers** 옵션을 사용합니다. 예를 들어 다음과 같습니다.

```
$ wget --secure-protocol=TLSv1_1 --ciphers="SECURE128" https://example.com
```

자세한 내용은 **wget(1)** 도움말 페이지의 HTTPS(SSL/TLS) 옵션 섹션을 참조하십시오.

curl

curl 툴에서 사용하는 암호를 지정하려면**--ciphers** 옵션을 사용하고 콜론으로 구분된 암호 목록을 값으로 제공합니다. 예를 들어 다음과 같습니다.

```
$ curl https://example.com --ciphers '@SECLEVEL=0:DES-CBC3-SHA:RSA-DES-CBC3-SHA'
```

자세한 내용은 **curl(1)** 도움말 페이지를 참조하십시오.

Firefox

Firefox 웹 브라우저에서 시스템 전체 암호화 정책을 비활성화할 수는 없지만 Firefox 의 구성 편집기에서 지원되는 암호 및 TLS 버전을 추가로 제한할 수 있습니다. 주소 표시줄에 **about:config**를 입력하고 필요에 따라 **security.tls.version.min** 옵션의 값을 변경합니다. **security.tls.version.min** 을 **1** 로 설정하면 필요한 최소 TLS 1.0이 허용되며, **security.tls.version.min 2** 는 TLS 1.1을 활성화합니다.

OpenSSH

OpenSSH 서버에 대한 시스템 전체 암호화 정책을 비활성화하려면 **/etc/sysconfig/sshd** 파일에서 **CRYPTO_POLICY=** 변수로 행의 주석을 제거합니다. 이 변경 후 **/etc/ssh/sshd_config** 파일의 **Ciphers, MACs, KexAlgoritms, GSSAPIKexAlgorithms** 섹션에 지정하는 값은 재정의되지 않습니다.

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 5보다 작도록 하여 **05-redhat.conf** 파일 앞에 .conf 접미사와**.conf** 접미사(예: **04-crypto-policy-override.conf**)를 사용합니다.

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

Libreswan

자세한 내용은 보안 네트워크 문서에서 시스템 전체 암호화 정책을 거부하는 IPsec 연결 구성을 참조하십시오.

추가 리소스

- **update-crypto-policies(8)** 도움말 페이지

3.6. 하위 정책을 사용하여 시스템 전체 암호화 정책 사용자 정의

활성화된 암호화 알고리즘 또는 프로토콜 집합을 조정하려면 다음 절차를 사용하십시오.

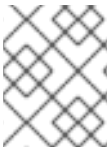
기존 시스템 전체 암호화 정책에 사용자 지정 하위 정책을 적용하거나 이러한 정책을 처음부터 정의할 수 있습니다.

범위가 지정된 정책의 개념은 다양한 백엔드에 대해 다양한 알고리즘 세트를 활성화할 수 있습니다. 각 구성 지시문을 특정 프로토콜, 라이브러리 또는 서비스로 제한할 수 있습니다.

또한 지시문은 와일드카드를 사용하여 여러 값을 지정하는 데 별표를 사용할 수 있습니다.

/etc/crypto-policies/state/CURRENT.pol 파일에는 와일드카드 확장 후 현재 적용되는 시스템 전체 암호화 정책의 모든 설정이 나열됩니다. 암호화 정책을 보다 엄격하게 설정하려면 **/usr/share/crypto-policies/policies/FUTURE.pol** 파일에 나열된 값을 사용하는 것이 좋습니다.

/usr/share/crypto-policies/policies/modules/ 디렉토리에서 예제 하위 정책을 찾을 수 있습니다. 이 디렉터리의 하위 정책 파일에는 주석 처리된 줄에 대한 설명도 포함되어 있습니다.



참고

RHEL 8.2에서는 시스템 전체 암호화 정책의 사용자 지정을 사용할 수 있습니다. 범위가 지정된 정책의 개념과 RHEL 8.5 이상에서 와일드카드를 사용하는 옵션을 사용할 수 있습니다.

절차

1. **/etc/crypto-policies/policies/modules/** 디렉토리로 체크아웃합니다.

```
# cd /etc/crypto-policies/policies/modules/
```

2. 조정을 위한 하위 정책을 생성합니다. 예를 들면 다음과 같습니다.

```
# touch MYCRYPTO-1.pmod
# touch SCOPES-AND-WILDCARDS.pmod
```

중요

정책 모듈의 파일 이름에 대문자를 사용합니다.

3. 선택한 텍스트 편집기에서 정책 모듈을 열고 시스템 전체 암호화 정책을 수정하는 옵션을 삽입합니다. 예를 들면 다음과 같습니다.

```
# vi MYCRYPTO-1.pmod
```

```
min_rsa_size = 3072
hash = SHA2-384 SHA2-512 SHA3-384 SHA3-512
```

```
# vi SCOPES-AND-WILDCARDS.pmod
```

```
# Disable the AES-128 cipher, all modes
cipher = -AES-128-*

# Disable CHACHA20-POLY1305 for the TLS protocol (OpenSSL, GnuTLS, NSS, and
OpenJDK)
cipher@TLS = -CHACHA20-POLY1305

# Allow using the FFDHE-1024 group with the SSH protocol (libssh and OpenSSH)
group@SSH = FFDHE-1024+

# Disable all CBC mode ciphers for the SSH protocol (libssh and OpenSSH)
cipher@SSH = -*-CBC

# Allow the AES-256-CBC cipher in applications using libssh
cipher@libssh = AES-256-CBC+
```

4. 모듈 파일의 변경 사항을 저장합니다.

5. **DEFAULT** 시스템 전체 암호화 정책 수준에 정책 조정을 적용합니다.

```
# update-crypto-policies --set DEFAULT:MYCRYPTO-1:SCOPES-AND-WILDCARDS
```

6. 이미 실행 중인 서비스 및 애플리케이션에 암호화 설정을 적용하려면 시스템을 다시 시작하십시오.

```
# reboot
```

검증

- **/etc/crypto-policies/state/CURRENT.pol** 파일에 변경 사항이 포함되어 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ cat /etc/crypto-policies/state/CURRENT.pol | grep rsa_size
min_rsa_size = 3072
```

추가 리소스

- **update-crypto-policies(8)** 도움말 페이지의 **Custom Policies** 섹션
- **crypto-policies(7)** 도움말 페이지의 **Crypto Policy Definition Format** 섹션
- RHEL 8.2에서 암호화 정책을 사용자 지정하는 방법에 대한 Red Hat 블로그 기사

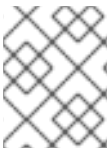
3.7. 시스템 전체 암호화 정책을 사용자 지정하여 SHA-1 비활성화

SHA-1 해시 함수에는 본질적으로 약한 디자인이 있으며 발전된 암호화 기능이 공격에 취약하기 때문에 RHEL 8은 기본적으로 SHA-1을 사용하지 않습니다. 하지만 일부 타사 애플리케이션(예: 공개 서명)은 여전히 SHA-1을 사용합니다. 시스템의 서명 알고리즘에서 SHA-1 사용을 비활성화하려면 **NO-SHA1** 정책 모듈을 사용할 수 있습니다.

중요

NO-SHA1 정책 모듈은 다른 위치에서는 서명에서만 SHA-1 해시 기능을 비활성화합니다. 특히 **NO-SHA1** 모듈은 여전히 SHA-1과 해시 기반 메시지 인증 코드(HMAC)를 사용할 수 있습니다. 이는 HMAC 보안 속성이 해당 해시 기능의 충돌 내성에 의존하지 않으므로 SHA-1에 대한 최근 공격으로 SHA-1의 SHA-1 사용에 미치는 영향이 크게 낮기 때문입니다.

시나리오에 특정 키 교환(KEX) 알고리즘 조합(예: **diffie-hellman-group-exchange-sha1**)을 비활성화해야 하지만 관련 KEX 및 다른 조합의 알고리즘을 모두 사용하려는 경우 SSH에서 diffie-hellman-group-group1-sha1 알고리즘을 비활성화하여 SSH에서 직접 SSH를 구성하는 단계를 참조하십시오.



참고

SHA-1을 비활성화하는 모듈은 RHEL 8.3에서 사용할 수 있습니다. RHEL 8.2에서는 시스템 전체 암호화 정책의 사용자 지정을 사용할 수 있습니다.

절차

1. **DEFAULT** 시스템 전체 암호화 정책 수준에 정책 조정을 적용합니다.

```
# update-crypto-policies --set DEFAULT:NO-SHA1
```

2. 이미 실행 중인 서비스 및 애플리케이션에 암호화 설정을 적용하려면 시스템을 다시 시작하십시오.

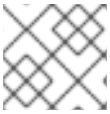
```
# reboot
```

추가 리소스

- **update-crypto-policies(8)** 도움말 페이지의 **Custom Policies** 섹션
- **crypto-policiesECDHE** 매뉴얼 페이지의 암호화 정책 정의 형식 섹션입니다.
- RHEL Red Hat 블로그 문서에서 암호화 정책을 사용자 지정하는 방법

3.8. 사용자 정의 시스템 전체 암호화 정책 생성 및 설정

다음 단계에서는 전체 정책 파일로 시스템 전체 암호화 정책 사용자 지정을 보여줍니다.



참고

RHEL 8.2에서는 시스템 전체 암호화 정책의 사용자 지정을 사용할 수 있습니다.

절차

1. 사용자 지정 정책 파일을 생성합니다.

```
# cd /etc/crypto-policies/policies/
# touch MYPOLICY.pol
```

또는 사전 정의된 4가지 정책 수준 중 하나를 복사하여 시작합니다.

```
# cp /usr/share/crypto-policies/policies/DEFAULT.pol /etc/crypto-
policies/policies/MYPOLICY.pol
```

2. 다음과 같은 요구 사항에 맞게 선택한 텍스트 편집기에서 사용자 지정 암호화 정책으로 파일을 편집합니다.

```
# vi /etc/crypto-policies/policies/MYPOLICY.pol
```

3. 시스템 전체 암호화 정책을 사용자 지정 수준으로 전환합니다.

```
# update-crypto-policies --set MYPOLICY
```

4. 이미 실행 중인 서비스 및 애플리케이션에 암호화 설정을 적용하려면 시스템을 다시 시작하십시오.

```
# reboot
```

추가 리소스

- **update-crypto-policies(8)** 도움말 페이지의 **Custom Policies** 섹션 및 **crypto-policies(7)** 도움말 페이지의 **Crypto Policy Definition Format** 섹션
- RHEL에서 암호화 정책을 사용자 지정하는 방법에 대한 Red Hat 블로그 기사

3.9. CRYPTO_POLICIES RHEL 시스템 역할을 사용하여 사용자 정의 암호화 정책 설정

crypto_policies 시스템 역할을 사용하여 단일 제어 노드에서 많은 수의 관리형 노드를 일관되게 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: Configure crypto policies
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure crypto policies
      ansible.builtin.include_role:
        name: rhel-system-roles.crypto_policies
      vars:
        - crypto_policies_policy: FUTURE
        - crypto_policies_reboot_ok: true
```

예를 들어 *FUTURE* 값을 선호하는 암호화 정책으로 교체할 수 있습니다.**DEFAULT**,**LEGACY** 및 **FIPS:OSPP**.

crypto_policies_reboot_ok: true 설정으로 인해 시스템 역할이 암호화 정책을 변경한 후 시스템이 재부팅됩니다.

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
```

주의

FIPS:OSPP 시스템 전체 하위 정책에는 CC(Common Criteria) 인증에 필요한 암호화 알고리즘에 대한 추가 제한이 포함되어 있으므로 설정한 후 시스템이 상호 운용이 불가능합니다. 예를 들어 3072비트, 추가 SSH 알고리즘 및 여러 TLS 그룹보다 짧은 RSA 및 DH 키를 사용할 수 없습니다. **FIPS:OSPP** 를 설정하면 Red Hat CDN(Content Delivery Network) 구조에 연결할 수 없습니다. 또한 **FIPS:OSPP** 를 사용하는 IdM 배포에는 AD(Active Directory)를 통합할 수 없으며 **FIPS:OSPP** 및 AD 도메인을 사용하는 RHEL 호스트 간 통신이 작동하지 않거나 일부 AD 계정이 인증할 수 없을 수 있습니다.

FIPS:OSPP 암호화 하위 정책을 설정한 후에는시스템이 CC와 호환되지 않습니다. RHEL 시스템을 CC 표준을 준수하는 유일한 방법은 **cc-config** 패키지에 제공된 지침을 따르는 것입니다. 인증된 RHEL 버전, 검증 보고서, NIAP(National Information Assurance Partnership) 웹 사이트에서 호스팅되는 CC 가이드 링크 목록은 규정 준수 활동 및 정부 표준 지식 베이스 문서의 Common Criteria 섹션을 참조하십시오.

검증

1. 제어 노드에서 **verify_playbook.yml:**이라는 다른 플레이북을 생성합니다.

```
---
- name: Verification
  hosts: managed-node-01.example.com
  tasks:
    - name: Verify active crypto policy
      ansible.builtin.include_role:
        name: rhel-system-roles.crypto_policies
    - debug:
        var: crypto_policies_active
```

2. 플레이북 구문을 확인합니다.

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

3. Playbook을 실행합니다.

```
$ ansible-playbook ~/verify_playbook.yml
TASK [debug] ***************************
ok: [host] => {
    "crypto_policies_active": "FUTURE"
}
```

crypto_policies_active 변수는 관리 노드에서 활성 상태인 정책을 표시합니다.

추가 리소스

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

3.10. 추가 리소스

- RHEL 8의 시스템 전체 암호화 정책 및 RHEL 8의 강력한 암호화 기본값 및 약한 암호화 알고리즘 지원 중단 지식 베이스 문서

4장. PKCS #11을 통해 암호화 하드웨어를 사용하도록 애플리케이션 구성

서버 애플리케이션을 위한 스마트 카드 및 암호화 토큰과 같은 전용 암호화 장치에 대한 시크릿 정보의 부분을 분리하면 서버 애플리케이션을 위한 HSM(하드웨어 보안 모듈)이 추가로 제공됩니다. RHEL에서 PKCS #11 API를 통한 암호화 하드웨어 지원은 서로 다른 애플리케이션에서 일관성이 유지되며 암호화 하드웨어에서 시크릿을 격리하는 작업이 복잡하지 않습니다.

4.1. PKCS #11을 통한 하드웨어 지원

PKI(Public-Key Cryptography Standard) #11은 암호화 정보를 유지하고 암호화 기능을 수행하는 장치를 암호화하기 위한 API(애플리케이션 프로그래밍 인터페이스)를 정의합니다.

PKCS #11에는 각 하드웨어 또는 소프트웨어 장치를 통합된 방식으로 애플리케이션에 제공하는 오브젝트인 *암호화 토큰이* 도입되었습니다. 따라서 애플리케이션은 일반적으로 개인에 의해 사용되는 스마트 카드와 같은 장치를 봅니다. 일반적으로 PKCS #11 암호화 토큰으로 컴퓨터에서 사용하는 하드웨어 보안 모듈입니다.

PKCS #11 토큰은 인증서, 데이터 오브젝트 및 공용, 개인 키 또는 시크릿 키를 비롯한 다양한 오브젝트 유형을 저장할 수 있습니다. 이러한 오브젝트는 PKCS #11 URI(Uniform Resource Identifier) 체계를 통해 고유하게 식별할 수 있습니다.

PKCS #11 URI는 개체 특성에 따라 PKCS #11 모듈에서 특정 오브젝트를 식별하는 표준 방법입니다. 이를 통해 URI 형식으로 동일한 구성 문자열로 모든 라이브러리 및 애플리케이션을 구성할 수 있습니다.

RHEL은 기본적으로 스마트 카드용 OpenSC PKCS #11 드라이버를 제공합니다. 그러나 하드웨어 토큰과 HSM에는 시스템에 해당되지 않는 자체 PKCS #11 모듈이 있을 수 있습니다. 시스템에서 등록된 스마트 카드 드라이버를 통해 래퍼 역할을 하는 **p11-kit** 도구로 이러한 PKCS #11 모듈을 등록할 수 있습니다.

고유한 PKCS #11 모듈이 시스템에서 작동하도록 하려면 새 텍스트 파일을 **/etc/pkcs11/modules/** 디렉토리에 추가합니다.

/etc/pkcs11/modules/ 디렉터리에 새 텍스트 파일을 생성하여 자체 PKCS #11 모듈을 시스템에 추가할 수 있습니다. 예를 들어 **p11-kit** 의 OpenSC 구성 파일은 다음과 같습니다.

```
$ cat /usr/share/p11-kit/modules/opensc.module
module: opensc-pkcs11.so
```

추가 리소스

- Red Hat Enterprise Linux 8에서 일관된 PKCS #11 지원
- PKCS #11 URI 스키마
- 스마트 카드에 대한 액세스 제어

4.2. 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

4.3. 스마트 카드에서 인증서를 사용하여 인증을 위한 애플리케이션 구성

애플리케이션에서 스마트 카드를 사용하여 인증하면 보안이 향상되고 자동화가 간소화될 수 있습니다. 다음 방법을 사용하여 PKI(Public Key Cryptography Standard) #11 URI를 애플리케이션에 통합할 수 있습니다.

- **Firefox** 웹 브라우저에서 **p11-kit-proxy** PKCS #11 모듈을 자동으로 로드합니다. 즉, 시스템에서 지원되는 모든 스마트 카드가 자동으로 감지됩니다. TLS 클라이언트 인증을 사용하려면 추가 설정이 필요하지 않으며 서버가 요청할 때 스마트 카드의 키와 인증서가 자동으로 사용됩니다.
- 애플리케이션에서 **GnuTLS** 또는 **NSS** 라이브러리를 사용하는 경우 PKCS #11 URI를 이미 지원합니다. 또한 **OpenSSL** 라이브러리에 의존하는 애플리케이션은 **openssl-pkcs11** 패키지에서 제공하는 **pkcs11** 엔진을 통해 스마트 카드를 포함한 암호화 하드웨어 모듈에 액세스할 수 있습니다.
- 스마트 카드에서 개인 키로 작업해야 하며 **NSS**,**GnuTLS** 또는 **OpenSSL** 을 사용하지 않는 애플리케이션에서는 특정 PKCS #11 모듈의 PKCS #11 API를 사용하는 대신, 스마트 카드를 포함한 암호화 하드웨어 모듈과 함께 작업하는 데 직접 **p11-kit** API를 사용할 수 있습니다.
- **wget** 네트워크 다운로더를 사용하면 로컬에 저장된 개인 키 및 인증서에 대한 경로 대신 PKCS #11 URI를 지정할 수 있습니다. 이렇게 하면 안전하게 저장된 개인 키와 인증서가 필요한 작업의 스크립트 생성이 간소화될 수 있습니다. 예를 들어 다음과 같습니다.

```
$ wget --private-key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --
certificate 'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
```

- **curl** 툴을 사용할 때 PKCS #11 URI를 지정할 수도 있습니다.

```
$ curl --key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --cert
'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
```

추가 리소스

- **curl(1)**, **wget(1)**, **p11-kit(8)** 도움말 페이지

4.4. APACHE에서 HSM을 사용하여 개인 키 보호

Apache HTTP 서버는 HSM(하드웨어 보안 모듈)에 저장된 개인 키로 작동할 수 있으므로 키 공개 및 중간자 공격을 방지하는 데 도움이 됩니다. 이 경우 일반적으로 사용 중인 서버에는 고성능 HSM이 필요합니다.

HTTPS 프로토콜 형식의 보안 통신을 위해 **Apache** HTTP 서버(**httpd**)는 OpenSSL 라이브러리를 사용합니다. OpenSSL은 기본적으로 PKCS #11을 지원하지 않습니다. HSM을 사용하려면 엔진 인터페이스를 통해 PKCS #11 모듈에 대한 액세스를 제공하는 **openssl-pkcs11** 패키지를 설치해야 합니다. 일반 파일 이름 대신 PKCS #11 URI를 사용하여 **/etc/httpd/conf.d/ssl.conf** 구성 파일에 서버 키와 인증서를 지정할 수 있습니다. 예를 들면 다음과 같습니다.

```
SSLCertificateFile    "pkcs11:id=%01;token=softhsm;type=cert"
SSLCertificateKeyFile "pkcs11:id=%01;token=softhsm;type=private?pin-value=111111"
```

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

4.5. NGINX에서 개인 키를 보호하는 HSM 사용

Nginx HTTP 서버는 HSM(하드웨어 보안 모듈)에 저장된 개인 키로 작동할 수 있으므로 키 공개 및 중간자 공격을 방지하는 데 도움이 됩니다. 이 경우 일반적으로 사용 중인 서버에는 고성능 HSM이 필요합니다.

Nginx는 암호화 작업에도 OpenSSL을 사용하므로 PKCS #11에 대한 지원은openssl-pkcs11 엔진을 통과해야 합니다. Nginx는 현재 HSM에서 개인 키 로드만 지원하며 인증서는 일반 파일로 별도로 제공해야 합니다. /etc/nginx/nginx.conf 구성 파일의 server 섹션에서 ssl_certificate 및 ssl_certificate_key 옵션을 수정합니다.

```
ssl_certificate     /path/to/cert.pem
ssl_certificate_key "engine:pkcs11:pkcs11:token=softhsm;id=%01;type=private?pin-value=111111";
```

Nginx 구성 파일의 PKCS #11 URI에는engine:pkcs11: 접두사가 필요합니다. 이는 다른pkcs11 접두사가 엔진 이름을 참조하기 때문입니다.

4.6. 추가 리소스

- **pkcs11.conf(5) 도움말 페이지.**

5장. POLKIT을 사용하여 스마트 카드에 대한 액세스 제어

Pins, PIN pads 및 biometrics와 같이 스마트 카드에 내장된 메커니즘으로 방지할 수 없는 가능한 위협과 보다 세분화된 제어를 위해 RHEL은 스마트 카드에 대한 액세스 제어를 제어하기 위해 **polkit** 프레임워크를 사용합니다.

시스템 관리자는 권한이 없는 사용자 또는 로컬이 아닌 사용자 또는 서비스에 대한 스마트 카드 액세스와 같은 특정 시나리오에 맞게 **polkit**을 구성할 수 있습니다.

5.1. POLKIT을 통한 스마트 카드 액세스 제어

개인 컴퓨터/스마트 카드(PC/SC) 프로토콜은 스마트 카드 및 독자를 컴퓨팅 시스템에 통합하기 위한 표준을 지정합니다. RHEL에서 **pcsc-lite** 패키지는 PC/SC API를 사용하는 스마트 카드에 대한 미들웨어를 제공합니다. 이 패키지의 일부인 **pcscd** (PC/SC Smart Card) 데몬을 사용하면 시스템이 PC/SC 프로토콜을 사용하여 스마트 카드에 액세스할 수 있습니다.

Pins, PIN pads 및 biometrics와 같은 스마트 카드에 내장된 액세스 제어 메커니즘은 가능한 모든 위협을 다루지 않기 때문에 RHEL은 보다 강력한 액세스 제어를 위해 **polkit** 프레임워크를 사용합니다. **polkit** 권한 부여 관리자는 권한 있는 작업에 대한 액세스 권한을 부여할 수 있습니다. 디스크에 대한 액세스 권한을 부여하는 것 외에도 **polkit** 을 사용하여 스마트 카드 보안을 위한 정책을 지정할 수 있습니다. 예를 들어 스마트 카드로 작업을 수행할 수 있는 사용자를 정의할 수 있습니다.

pcsc-lite 패키지를 설치하고 **pcscd** 데몬을 시작한 후 시스템은 **/usr/share/polkit-1/actions/** 디렉터리에 정의된 정책을 적용합니다. 기본 시스템 전체 정책은 **/usr/share/polkit-1/actions/org.debian.pcsc-lite.policy** 파일에 있습니다. polkit 정책 파일은 XML 형식을 사용하며 구문은 **polkit(8)** 도움말 페이지에 설명되어 있습니다.

polkitd 서비스는 **/etc/polkit-1/rules.d/** 및 **/usr/share/polkit-1/rules.d/** 디렉토리를 모니터링하여 이러한 디렉토리에 저장된 규칙 파일의 변경 사항을 모니터링합니다. 파일에는 JavaScript 형식의 권한 부여 규칙이 포함되어 있습니다. 시스템 관리자는 두 디렉토리에 모두 사용자 지정 규칙 파일을 추가할 수 있으며 **polkitd** 는 파일 이름을 기반으로 하여 사전순으로 읽습니다. 두 파일의 이름이 같은 경우 **/etc/polkit-1/rules.d/** 의 파일이 먼저 표시됩니다.

SSSD(시스템 보안 서비스 데몬)가 **root** 로 실행되지 않을 때 스마트 카드 지원을 활성화해야 하는 경우 **sssd-polkit-rules** 패키지를 설치해야 합니다. 패키지는 SSSD와 **polkit** 통합을 제공합니다.

추가 리소스

- 시스템의 **polkit(8)**, **polkitd(8)**, **pcscd(8)** 매뉴얼 페이지

5.2. PC/SC 및 POLKIT 관련 문제 해결

pcsc-lite 패키지를 설치한 후 자동으로 시행되고 **pcscd** 데몬을 시작하면 사용자가 스마트 카드와 직접 상호 작용하지 않더라도 사용자 세션에서 인증을 요청할 수 있습니다. GNOME에는 다음과 같은 오류 메시지가 표시됩니다.

> Authentication is required to access the PC/SC daemon

opensc 와 같은 스마트 카드 관련 다른 패키지를 설치할 때 시스템이 **pcsc-lite** 패키지를 종속성으로 설치할 수 있습니다.

시나리오가 스마트 카드와의 상호 작용을 필요로하지 않고 PC/SC 데몬에 대한 권한 부여 요청을 표시하지 않으려면 **pcsc-lite** 패키지를 제거할 수 있습니다. 필요한 최소 패키지를 유지하는 것은 어쨌든 좋은 보안 관행입니다.

스마트 카드를 사용하는 경우 **/usr/share/polkit-1/actions/org.debian.pcsc-lite.policy**에서 시스템 제공 정책 파일에서 규칙을 확인하여 문제 해결을 시작합니다. **/etc/polkit-1/rules.d/** 디렉터리(예: **03-allow-pcscd.rules**)의 정책에 사용자 지정 규칙 파일을 추가할 수 있습니다. 규칙 파일은 JavaScript 구문을 사용하며 정책 파일은 XML 형식으로 되어 있습니다.

시스템이 표시하는 권한 부여 요청을 이해하려면 저널 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
$ journalctl -b | grep pcsc
...
Process 3087 (user: 1001) is NOT authorized for action: access_pcsc
...
```

이전 로그 항목은 사용자가 정책에 의해 작업을 수행할 수 있는 권한이 없음을 의미합니다. **/etc/polkit-1/rules.d/** 에 해당 규칙을 추가하여 이 거부를 해결할 수 있습니다.

polkitd 유닛과 관련된 로그 항목을 검색할 수도 있습니다. 예를 들면 다음과 같습니다.

```
$ journalctl -u polkit
...
polkitd[NNN]: Error compiling script /etc/polkit-1/rules.d/00-debug-pcscd.rules
...
polkitd[NNN]: Operator of unix-session:c2 FAILED to authenticate to gain authorization for action
org.debian.pcsc-lite.access_pcsc for unix-process:4800:14441 [/usr/libexec/gsd-smartcard] (owned
by unix-user:group)
...
```

이전 출력에서 첫 번째 항목은 규칙 파일에 일부 구문 오류가 있음을 나타냅니다. 두 번째 항목은 사용자가 **pcscd**에 대한 액세스 권한을 얻지 못했음을 의미합니다.

또한 짧은 스크립트로 PC/SC 프로토콜을 사용하는 모든 애플리케이션을 나열할 수 있습니다. 실행 가능한 파일(예: **pcsc-apps.sh**)을 생성하고 다음 코드를 삽입합니다.

```
#!/bin/bash

cd /proc

for p in [0-9]*
do
 if grep libpcsclite.so.1.0.0 $p/maps &> /dev/null
 then
  echo -n "process: "
  cat $p/cmdline
  echo " ($p)"
 fi
done
```

스크립트를 **root**로 실행합니다.

```
# ./pcsc-apps.sh
process: /usr/libexec/gsd-smartcard (3048)
enable-sync --auto-ssl-client-auth --enable-crashpad (4828)
...
```

추가 리소스

- **journalctl,polkit(8), polkitd(8)** 및 **pcscd(8)** 메뉴얼 페이지.

5.3. PC/SC에 대한 POLKIT 권한에 대한 자세한 정보 표시

기본 구성에서 **polkit** 권한 부여 프레임워크는 저널 로그에 제한된 정보만 보냅니다. 새 규칙을 추가하여 PC/SC 프로토콜과 관련된 **polkit** 로그 항목을 확장할 수 있습니다.

사전 요구 사항

- 시스템에 **pcsc-lite** 패키지를 설치했습니다.
- **pcscd** 데몬이 실행 중입니다.

절차

1. **/etc/polkit-1/rules.d/** 디렉토리에 새 파일을 생성합니다.

```
# touch /etc/polkit-1/rules.d/00-test.rules
```

2. 선택한 편집기에서 파일을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi /etc/polkit-1/rules.d/00-test.rules
```

3. 다음 행을 삽입합니다.

```
polkit.addRule(function(action, subject) {
  if (action.id == "org.debian.pcsc-lite.access_pcsc" ||
   action.id == "org.debian.pcsc-lite.access_card") {
 polkit.log("action=" + action);
 polkit.log("subject=" + subject);
  }
});
```

파일을 저장하고 편집기를 종료합니다.

4. **pcscd** 및 **polkit** 서비스를 다시 시작합니다.

```
# systemctl restart pcscd.service pcscd.socket polkit.service
```

검증

1. **pcscd**에 대한 권한 부여 요청을 만듭니다. 예를 들어 Firefox 웹 브라우저를 열거나**opensc** 패키지에서 제공하는 **pkcs11-tool -L** 명령을 사용합니다.
2. 확장 로그 항목을 표시합니다. 예를 들면 다음과 같습니다.

```
# journalctl -u polkit --since "1 hour ago"
polkitd[1224]: <no filename>:4: action=[Action id='org.debian.pcsc-lite.access_pcsc']
polkitd[1224]: <no filename>:5: subject=[Subject pid=2020481 user=user'
groups=user,wheel,mock,wireshark seat=null session=null local=true active=true]
```

추가 리소스

- **polkit(8)** 및 **polkitd(8)** 메뉴얼 페이지.

5.4. 추가 리소스

- 스마트 카드에 대한 액세스 제어 Red Hat 블로그 문서.

6장. 구성 준수 및 취약성에 대한 시스템 검사

규정 준수 감사는 지정된 오브젝트가 규정 준수 정책에 지정된 모든 규칙을 따르는지 여부를 결정하는 프로세스입니다. 규정 준수 정책은 컴퓨팅 환경에서 사용해야 하는 체크리스트 형태로 필요한 설정을 지정하는 보안 전문가가 정의합니다.

규정 준수 정책은 조직마다, 심지어 동일한 조직 내의 여러 시스템에서도 크게 달라질 수 있습니다. 이러한 정책의 차이는 각 시스템의 목적과 조직의 중요성을 기반으로 합니다. 사용자 지정 소프트웨어 설정 및 배포 특성으로 인해 사용자 지정 정책 체크리스트가 필요합니다.

6.1. RHEL의 구성 준수 도구

다음 구성 규정 준수 툴을 사용하여 Red Hat Enterprise Linux에서 완전히 자동화된 규정 준수 감사를 수행할 수 있습니다. 이러한 툴은 SCAP(Security Content Automation Protocol) 표준을 기반으로 하며 규정 준수 정책의 자동화된 조정을 위해 설계되었습니다.

SCAP Workbench

scap-workbench 그래픽 유틸리티는 단일 로컬 또는 원격 시스템에서 구성 및 취약점 검사를 수행하도록 설계되었습니다. 또한 이러한 스캔 및 평가를 기반으로 보안 보고서를 생성하는 데 사용할 수도 있습니다.

OpenSCAP

oscap 명령줄 유틸리티가 포함된 **OpenSCAP** 라이브러리는 로컬 시스템에서 구성 및 취약점 검사를 수행하고 구성 규정 준수 콘텐츠를 검증하고 이러한 검사 및 평가를 기반으로 보고서 및 가이드를 생성하도록 설계되었습니다.

중요

OpenSCAP 을 사용하는 동안 메모리 사용량 문제가 발생할 수 있으므로 프로그램을 조기 중단하고 결과 파일이 생성되지 않을 수 있습니다. 자세한 내용은 OpenSCAP 메모리 사용 문제 지식 베이스 문서를 참조하십시오.

SCAP Security Guide (SSG)

scap-security-guide 패키지는 Linux 시스템에 대한 보안 정책 컬렉션을 제공합니다. 이 지침은 적용 가능한 경우 정부 요구 사항과 연결된 실용적인 강화 조언 카탈로그로 구성됩니다. 이 프로젝트는 일반화된 정책 요구 사항과 특정 구현 지침 간의 격차를 해소합니다.

스크립트 검사 엔진(SCE)

SCAP 프로토콜에 대한 확장인 SCE를 사용하면 관리자가 Bash, Python, Ruby와 같은 스크립팅 언어를 사용하여 보안 콘텐츠를 작성할 수 있습니다. SCE 확장은 **openscap-engine-sce** 패키지에 제공됩니다. SCE 자체는 SCAP 표준의 일부가 아닙니다.

여러 시스템에서 원격으로 자동화된 규정 준수 감사를 수행하려면 Red Hat Satellite에 OpenSCAP 솔루션을 사용할 수 있습니다.

추가 리소스

- **oscap(8), scap-workbench(8)** 및 **scap-security-guide(8)** 도움말 페이지
- Red Hat 보안 데모: 으로 보안 준수 자동화를 위한 사용자 지정된 보안 정책 콘텐츠 생성
- Red Hat 보안 데모: RHEL 보안 기술로 자체 방어
- Red Hat Satellite 관리 가이드의 보안 준수 관리 가이드

6.2. 취약점 검사

6.2.1. Red Hat 보안 공지 OVAL 피드

Red Hat Enterprise Linux 보안 감사 기능은 SCAP(Security Content Automation Protocol) 표준을 기반으로 합니다. SCAP는 자동화된 구성, 취약점 및 패치 검사, 기술 제어 준수 활동 및 보안 측정을 지원하는 다용도 사양 프레임워크입니다.

SCAP 사양은 스캐너 또는 정책 편집기를 구현하지 않아도 보안 콘텐츠 형식이 잘 알려져 표준화되어 있는 에코시스템을 생성합니다. 이를 통해 조직은 채택한 보안 벤더 수에 관계없이 SCC(보안 정책)를 한 번 구축할 수 있습니다.

OVAL(Open Vulnerability Assessment Language)은 SCAP에서 필수적이고 오래된 구성 요소입니다. 다른 툴 및 사용자 지정 스크립트와 달리 OVAL은 선언적 방식으로 필요한 리소스 상태를 설명합니다. OVAL 코드는 직접 실행되지 않지만 scanner라는 OVAL 인터프리터 툴을 사용합니다. OVAL의 선언적 특성은 평가된 시스템의 상태가 실수로 수정되지 않도록 합니다.

다른 모든 SCAP 구성 요소와 마찬가지로, OVAL은 XML을 기반으로 합니다. SCAP 표준은 여러 문서 형식을 정의합니다. 각각 다른 유형의 정보를 포함하며 다른 목적을 제공합니다.

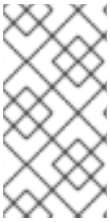
Red Hat Product Security는 Red Hat 고객에게 영향을 미치는 모든 보안 문제를 추적하고 조사하여 고객이 위험을 평가하고 관리할 수 있도록 지원합니다. Red Hat 고객 포털에서 적시에 간결한 패치와 보안 공지를 제공합니다. Red Hat은 OVAL 패치 정의를 생성 및 지원하여 시스템에서 읽을 수 있는 보안 권고 버전을 제공합니다.

플랫폼, 버전 및 기타 요인 간의 차이로 인해 취약점의 Red Hat 제품 보안 질적 심각도 등급은 타사에서 제공하는 CVSS(Common Vulnerability Scoring System) 기준 평가와 직접적으로 일치하지 않습니다. 따라서 타사가 제공하는 정의 대신 RHSA OVAL 정의를 사용하는 것이 좋습니다.

RHSA OVAL 정의는 개별적으로 및 전체 패키지로 사용할 수 있으며 Red Hat 고객 포털에서 사용할 수 있는 새 보안 권고를 1시간 이내에 업데이트합니다.

각 OVAL 패치 정의는 일대일로 Red Hat 보안 권고(RHSA)에 매핑됩니다. RHSA에는 여러 취약점에 대한 수정 사항이 포함될 수 있으므로 각 취약점은 CVE(Common Vulnerabilities and Exposures) 이름으로 별도로 나열되며 공개 버그 데이터베이스에 해당 항목에 대한 링크가 있습니다.

RHSA OVAL 정의는 시스템에 설치된 취약한 버전의 RPM 패키지를 확인하도록 설계되었습니다. 예를 들어 이러한 정의를 확장하여 추가 검사를 포함하여 패키지가 취약한 구성에서 사용 중인지 확인할 수 있습니다. 이러한 정의는 Red Hat이 제공하는 소프트웨어 및 업데이트를 포괄하도록 설계되었습니다. 타사 소프트웨어의 패치 상태를 감지하려면 추가 정의가 필요합니다.



참고

Red Hat Enterprise Linux 규정 준수 서비스를 위한 Red Hat Insights는 IT 보안 및 규정 준수 관리자가 Red Hat Enterprise Linux 시스템의 보안 정책 준수를 평가, 모니터링 및 보고할 수 있도록 지원합니다. 규정 준수 서비스 UI 내에서 SCAP 보안 정책을 완전히 생성하고 관리할 수도 있습니다.

추가 리소스

- Red Hat 및 OVAL 호환성
- Red Hat 및 CVE 호환성
- 제품 보안 개요의 알림 및 공지

- 보안 데이터 지표

6.2.2. 시스템에서 취약점 스캔

oscap 명령줄 유틸리티를 사용하면 로컬 시스템을 스캔하고, 구성 준수 콘텐츠를 검증하고, 이러한 스캔 및 평가를 기반으로 보고서 및 가이드를 생성할 수 있습니다. 이 유틸리티는 OpenSCAP 라이브러리의 프런트 엔드 역할을 하며 처리하는 SCAP 콘텐츠 유형에 따라 해당 기능을 모듈(하위 명령)에 그룹화합니다.

사전 요구 사항

- **openscap-scanner** 및 **0.0/162** 패키지를 설치합니다.

절차

1. 시스템에 대한 최신 RHSA OVAL 정의를 다운로드합니다.

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL8/rhel-8.oval.xml.bz2 | bzip2 --decompress > rhel-8.oval.xml
```

2. 시스템에서 취약점을 스캔하고 결과를 *vulnerability.html* 파일에 저장합니다.

```
# oscap oval eval --report vulnerability.html rhel-8.oval.xml
```

검증

- 선택한 브라우저의 결과를 확인합니다. 예를 들면 다음과 같습니다.

```
$ firefox vulnerability.html &
```

추가 리소스

- **oscap(8)** 도움말 페이지
- Red Hat OVAL 정의
- OpenSCAP 메모리 사용 문제

6.2.3. 원격 시스템에서 취약점 스캔

SSH 프로토콜을 통해 **oscap-ssh** 툴을 사용하여 OpenSCAP 스캐너가 있는 취약점이 원격 시스템에 있는지 확인할 수 있습니다.

사전 요구 사항

- 스캔에 사용하는 시스템에 **openscap-utils** 및 **bzip2** 패키지가 설치되어 있습니다.
- **openscap-scanner** 패키지는 원격 시스템에 설치됩니다.
- SSH 서버는 원격 시스템에서 실행되고 있습니다.

절차

1. 시스템에 대한 최신 RHSA OVAL 정의를 다운로드합니다.

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL8/rhel-8.oval.xml.bz2 | bzip2 --decompress > rhel-8.oval.xml
```

2. 원격 시스템에서 취약점을 스캔하고 결과를 파일에 저장합니다.

```
# oscap-ssh <username>@<hostname> <port> oval eval --report <scan-report.html> rhel-8.oval.xml
```

교체:

- **<username>** @ **<hostname>** 및 원격 시스템의 사용자 이름 및 호스트 이름입니다.
- 원격 시스템에 액세스할 수 있는 포트 번호가 있는 <port>(예: 22)
- *oscap 이 검사 결과를 저장하는 파일 이름이* **<scan-report.html>** *입니다.*

추가 리소스

- ***oscap-ssh(8)***
- ***Red Hat OVAL 정의***
- ***OpenSCAP 메모리 사용 문제***

6.3. 구성 규정 준수 검사

6.3.1. RHEL의 구성 규정 준수

*구성 규정 준수 스캔을 사용하여 특정 조직에서 정의한 기준을 준수할 수 있습니다. 예를 들어, 미국 정부와 협력하는 경우 시스템을 **OSPP(운영 체제 보호 프로필)**에 맞춰야 할 수 있으며 결제 프로세서인 경우 시스템을 **PCI-DSS(Payment Card Industry Data Security Standard)**에 맞춰 조정해야 할 수 있습니다. 구성 준수 스캔을 수행하여 시스템 보안을 강화할 수도 있습니다.*

*영향을 받는 구성 요소에 대한 **Red Hat** 모범 사례에 부합하므로 **SCAP Security Guide** 패키지에 제공된 **SCAP(Security Content Automation Protocol)** 콘텐츠를 따르는 것이 좋습니다.*

***SCAP** 보안 가이드 패키지는 **SCAP 1.2** 및 **SCAP 1.3** 표준을 준수하는 콘텐츠를 제공합니다. **openscap** 스캐너 유틸리티는 **SCAP** 보안 가이드 패키지에 제공된 **SCAP 1.2** 및 **SCAP 1.3** 콘텐츠와 호환됩니다.*

중요

구성 규정 준수 스캔을 수행해도 시스템이 규정을 준수하는 것은 아닙니다.

SCAP 보안 가이드 제품군은 여러 플랫폼의 프로필을 데이터 스트림 문서 형태로 제공합니다. 데이터 스트림은 정의, 벤치마크, 프로필 및 개별 규칙이 포함된 파일입니다. 각 규칙은 규정 준수에 대한 적용 가능성 및 요구 사항을 지정합니다. **RHEL**에서는 보안 정책을 준수하기 위해 여러 프로필을 제공합니다. 업계 표준 외에도 **Red Hat** 데이터 스트림에는 실패한 규칙의 수정에 대한 정보도 포함되어 있습니다.

컴플라이언스 검사 리소스 구조

```
Data stream
├── xccdf
/    ├── benchmark
/        ├── profile
/        /   ├──rule reference
/        /   └──variable
/        ├── rule
/            ├── human readable data
/            ├── oval reference
├── oval         ├── ocil reference
├── ocil         ├── cpe reference
└── cpe          └── remediation
```

프로필은 **OSPP, PCI-DSS** 및 **HIPAA(Health Insurance Portability and Accountability Act)**와 같은 보안 정책을 기반으로 하는 규칙 집합입니다. 이를 통해 보안 표준을 준수하는 자동화된 방식으로 시스템을 감사할 수 있습니다.

프로필을 수정하여 특정 규칙(예: 암호 길이)을 사용자 지정할 수 있습니다. 프로필 맞춤에 대한 자세한 내용은 **SCAP Workbench**를 사용하여 보안 프로필 사용자 지정을 참조하십시오.

6.3.2. OpenSCAP 스캔의 가능한 결과

OpenSCAP 스캔에 적용되는 데이터 스트림 및 프로필과 시스템의 다양한 속성에 따라 각 규칙이 특정 결과를 생성할 수 있습니다. 이러한 결과는 그 의미에 대한 간략한 설명과 함께 가능한 결과입니다.

통과

검사에서 이 규칙과의 충돌을 찾지 못했습니다.

실패

검사에서 이 규칙과 충돌하는 것을 발견했습니다.

확인되지 않음

***OpenSCAP**에서는 이 규칙을 자동으로 평가하지 않습니다. 시스템이 이 규칙을 수동으로 준수하는지 확인합니다.*

해당 없음

이 규칙은 현재 구성에 적용되지 않습니다.

선택되지 않음

*이 규칙은 프로필에 포함되지 않습니다. **OpenSCAP**은 이 규칙을 평가하지 않으며 결과에 이러한 규칙을 표시하지 않습니다.*

오류

*검사에 오류가 발생했습니다. 자세한 내용은 **--verbose DEVEL** 옵션을 사용하여 **oscap** 명령을 입력할 수 있습니다. 버그 보고서를 작성하는 것이 좋습니다.*

알 수 없음

*검사에 예기치 않은 상황이 발생했습니다. 자세한 내용은 '**--verbose DEVEL** 옵션을 사용하여 **oscap** 명령을 입력합니다. 버그 보고서를 작성하는 것이 좋습니다.*

6.3.3. 구성 규정 준수 프로필 보기

*검사 또는 수정을 위해 프로필을 사용하기 전에 나열한 후 **oscap info** 하위 명령을 사용하여 자세한 설명을 확인할 수 있습니다.*

사전 요구 사항

- ***openscap-scanner** 및 **scap-security-guide** 패키지가 설치됩니다.*

절차

1. ***SCAP** 보안 가이드 프로젝트에서 제공하는 보안 준수 프로필이 있는 사용 가능한 모든 파일을 나열합니다.*

```
$ ls /usr/share/xml/scap/ssg/content/
ssg-firefox-cpe-dictionary.xml  ssg-rhel6-ocil.xml
ssg-firefox-cpe-oval.xml        ssg-rhel6-oval.xml
…
ssg-rhel6-ds-1.2.xml          ssg-rhel8-oval.xml
ssg-rhel8-ds.xml              ssg-rhel8-xccdf.xml
…
```

2. ***oscap info** 하위 명령을 사용하여 선택한 데이터 스트림에 대한 세부 정보를 표시합니다. 데이터 스트림을 포함하는 **XML** 파일은 이름에 **-ds** 문자열로 표시됩니다. **Profiles** 섹션에서 사용 가능한 프로필 및 해당 **ID** 목록을 찾을 수 있습니다.*

```
$ oscap info /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
Profiles:
…
  Title: Health Insurance Portability and Accountability Act (HIPAA)
    Id: xccdf_org.ssgproject.content_profile_hipaa
  Title: PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8
    Id: xccdf_org.ssgproject.content_profile_pci-dss
  Title: OSPP - Protection Profile for General Purpose Operating Systems
    Id: xccdf_org.ssgproject.content_profile_ospp
…
```

3. *데이터 스트림 파일에서 프로필을 선택하고 선택한 프로필에 대한 추가 세부 정보를 표시합니다. 이렇게 하려면 **--profile** 옵션 다음에 이전 명령의 출력에 표시된 **ID**의 마지막 섹션과 함께 **oscap info** 를 사용합니다. 예를 들어 **HIPPA** 프로파일의 **ID**는 **xccdf_org.ssgproject.content_profile_hipaa** 이며 **--profile** 옵션의 값은 **hipaa** 입니다.*

```
$ oscap info --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
…
Profile
 Title: Health Insurance Portability and Accountability Act (HIPAA)

 Description: The HIPAA Security Rule establishes U.S. national standards to protect
individuals' electronic personal health information that is created, received, used, or
maintained by a covered entity.
…
```

추가 리소스

- ***scap-security-guide(8)** 도움말 페이지*
- *OpenSCAP 메모리 사용 문제*

6.3.4. 특정 기준의 구성 준수 평가

*시스템 또는 원격 시스템이 특정 기준을 준수하는지 여부를 확인하고 **oscap** 명령줄 도구를 사용하여 결과를 보고서에 저장할 수 있습니다.*

사전 요구 사항

- ***openscap-scanner** 및 **scap-security-guide** 패키지가 설치됩니다.*
- *시스템이 준수해야 하는 기준 내에서 프로필의 **ID**를 알고 있습니다. **ID**를 찾으려면 구성 규정 준수에 대한 프로필 보기 섹션을 참조하십시오.*

절차

1. *로컬 시스템에서 선택한 프로필을 준수하는지 스캔하여 검사 결과를 파일에 저장합니다.*

```
$ oscap xccdf eval --report <scan-report.html> --profile <profileID>
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

교체:

- ***oscap** 이 검사 결과를 저장하는 파일 이름이 **<scan-report.html >**입니다.*
- *시스템이 준수해야 하는 프로파일 **ID** 가 있는 **<profileID>(예: hipaa).***

2. *선택 사항: 원격 시스템에서 선택한 프로필을 준수하는지 스캔하여 검사 결과를 파일에 저장합니다.*

```
$ oscap-ssh <username>@<hostname> <port> xccdf eval --report <scan-report.html> --
profile <profileID> /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

교체:

- ***<username> @ <hostname >** 및 원격 시스템의 사용자 이름 및 호스트 이름입니다.*
- *원격 시스템에 액세스할 수 있는 포트 번호가 있는 **<port>**입니다.*

- *oscap 이 검사 결과를 저장하는 파일 이름이 <scan-report.html >입니다.*
- *시스템이 준수해야 하는 프로파일 ID 가 있는 <profileID>(예: hipaa).*

추가 리소스

- *scap-security-guide(8) 도움말 페이지*
- */usr/share/doc/scap-security-guide/ 디렉터리에 있는 SCAP 보안 가이드 문서*
- */usr/share/doc/scap-security-guide/guides/ssg-rhel8-guide-index.html - scap-security-guide-doc 패키지와 함께 설치된 [Red Hat Enterprise Linux 8의 보안 구성 가이드]*
- *OpenSCAP 메모리 사용 문제*

6.4. 특정 기준선에 맞게 시스템 수정

특정 기준선에 맞게 RHEL 시스템을 수정할 수 있습니다. SCAP 보안 가이드에서 제공하는 모든 프로필에 맞게 시스템을 교정할 수 있습니다. 사용 가능한 프로필 나열에 대한 자세한 내용은 구성 규정 준수에 대한 프로필 보기 섹션을 참조하십시오.

주의

신중하게 사용하지 않는 경우 Remediate 옵션을 활성화하여 시스템 평가를 실행하면 시스템에 작동하지 않을 수 있습니다. Red Hat은 보안 강화 수정으로 인한 변경 사항을 되돌릴 수 있는 자동화된 방법을 제공하지 않습니다. 수정은 기본 구성의 RHEL 시스템에서 지원됩니다. 설치 후 시스템이 변경된 경우 수정을 실행하여 필요한 보안 프로필을 준수하지 못할 수 있습니다.

사전 요구 사항

- *scap-security-guide 패키지가 설치됩니다.*

절차

1. *--remediate 옵션과 함께 oscap 명령을 사용하여 시스템을 교정합니다.*

```
# oscap xccdf eval --profile <profileID> --remediate /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

& lt;profileID >를 시스템이 준수해야 하는 프로필 ID로 바꿉니다(예: hipaa).

2. *시스템을 다시 시작합니다.*

검증

1. *프로필로 시스템 규정 준수를 평가하고 검사 결과를 파일에 저장합니다.*

```
$ oscap xccdf eval --report <scan-report.html> --profile <profileID> /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

교체:

- *oscap 이 검사 결과를 저장하는 파일 이름이 <scan-report.html >입니다.*
- *시스템이 준수해야 하는 프로파일 ID 가 있는 <profileID>(예: hipaa).*

추가 리소스

- *scap-security-guide(8) 및 oscap(8) 도움말 페이지*
- *SSG 콘텐츠 지식 베이스 문서를 사용하여 DISA 벤치마크 보완*

6.5. SSG ANSIBLE 플레이북을 사용하여 특정 기준과 일치하도록 시스템 수정

SCAP 보안 가이드 프로젝트의 Ansible 플레이북 파일을 사용하여 특정 기준과 일치하도록 시스템을 교정할 수 있습니다. 이 예에서는 HIPAA(Health Insurance Portability and Accountability Act) 프로필

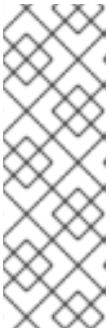
*을 사용하지만 **SCAP** 보안 가이드에서 제공하는 다른 프로필과 일치하도록 수정할 수 있습니다. 사용 가능한 프로필 나열에 대한 자세한 내용은 구성 규정 준수에 대한 프로필 보기 섹션을 참조하십시오.*

주의

*신중하게 사용하지 않는 경우 **Remediate** 옵션을 활성화하여 시스템 평가를 실행하면 시스템에 작동하지 않을 수 있습니다. **Red Hat**은 보안 강화 수정으로 인한 변경 사항을 되돌릴 수 있는 자동화된 방법을 제공하지 않습니다. 수정은 기본 구성의 **RHEL** 시스템에서 지원됩니다. 설치 후 시스템이 변경된 경우 수정을 실행하여 필요한 보안 프로필을 준수하지 못할 수 있습니다.*

사전 요구 사항

- ***scap-security-guide** 패키지가 설치됩니다.*
- ***ansible-core** 패키지가 설치되어 있습니다. 자세한 내용은 **Ansible** 설치 가이드를 참조하십시오.*
- ***RHEL 8.6** 이상이 설치되어 있어야 합니다. **RHEL** 설치에 대한 자세한 내용은 설치 미디어에서 **RHEL** 상호 작용 설치를 참조하십시오.*



참고

***RHEL 8.5** 및 이전 버전에서 **Ansible** 패키지는 **Ansible Core**가 아닌 다른 수준의 지원을 통해 **Ansible Engine**을 통해 제공되었습니다. 패키지가 **RHEL 8.6** 이상에서 **Ansible** 자동화 콘텐츠와 호환되지 않을 수 있으므로 **Ansible Engine**을 사용하지 마십시오. 자세한 내용은 **RHEL 9** 및 **RHEL 8.6** 이상 **AppStream** 리포지토리에 포함된 **Ansible Core** 패키지에 대한 지원 범위를 참조하십시오.*

절차

1. ***Ansible**을 사용하여 **HIPAA**에 맞게 시스템을 교정합니다.*

```
# ansible-playbook -i localhost, -c local /usr/share/scap-security-guide/ansible/rhel8-playbook-hipaa.yml
```

2. 시스템을 다시 시작합니다.

검증

1. HIPAA 프로필을 사용하여 시스템의 규정 준수를 평가하고 검사 결과를 파일에 저장합니다.

```
# oscap xccdf eval --profile hipaa --report <scan-report.html>
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

& lt;scan-report.html& gt;을 oscap 이 검사 결과를 저장하는 파일 이름으로 바꿉니다.

추가 리소스

- scap-security-guide(8) 및 oscap(8) 도움말 페이지
- Ansible 설명서

6.6. 시스템을 특정 기준과 정렬하도록 수정 ANSIBLE 플레이북 생성

시스템을 특정 기준과 조정하는 데 필요한 수정 사항만 포함하는 Ansible 플레이북을 생성할 수 있습니다. 이 예에서는 HIPAA(Health Insurance Portability and Accountability Act) 프로필을 사용합니다. 이 절차를 통해 이미 충족된 요구 사항을 다루지 않는 작은 플레이북을 생성합니다. 이러한 단계를 수행하면 시스템을 어떤 식으로든 수정하지 않으며 이후 애플리케이션을 위한 파일만 준비합니다.

참고

RHEL 8.6에서는 Ansible Engine이 기본 제공 모듈만 포함된 ansible-core 패키지로 교체됩니다. 많은 Ansible 수정에서는 기본 제공 모듈에 포함되지 않은 커뮤니티 및 이식 가능한 운영 체제 인터페이스(POSIX) 컬렉션의 모듈을 사용합니다. 이 경우 Bash 수정을 Ansible 수정을 대신 사용할 수 있습니다. RHEL 8.6의 Red Hat Connector에는 Ansible Core와 함께 작동하는 수정 플레이북을 활성화하는 데 필요한 Ansible 모듈이 포함되어 있습니다.

사전 요구 사항

- scap-security-guide 패키지가 설치됩니다.

절차

1. 시스템을 스캔하고 결과를 저장합니다.

```
# oscap xccdf eval --profile hipaa --results <hipaa-results.xml>
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

2. 결과를 사용하여 파일에서 결과 **ID** 값을 찾습니다.

```
# oscap info <hipaa-results.xml>
```

3. **1**단계에서 생성된 파일을 기반으로 **Ansible** 플레이북을 생성합니다.

```
# oscap xccdf generate fix --fix-type ansible --result-id <xccdf_org.open-
scap_testresult_xccdf_org.ssgproject.content_profile_hipaa> --output <hipaa-
remediations.yml> <hipaa-results.xml>
```

4. **1**단계에서 수행한 검사 중에 실패한 규칙에 대한 **Ansible** 수정이 포함된 생성된 파일을 검토합니다. 생성된 이 파일을 검토한 후 **ansible-playbook < hipaa-remediations.yml >** 명령을 사용하여 적용할 수 있습니다.

검증

- 선택한 텍스트 편집기에서 생성된 **< hipaa-remediations.yml >** 파일에 **1**단계에서 수행한 검사에 실패한 규칙이 포함되어 있는지 검토합니다.

추가 리소스

- **scap-security-guide(8)** 및 **oscap(8)** 도움말 페이지
- ***Ansible* 설명서**

6.7. 이후 애플리케이션에 대한 해결 BASH 스크립트 생성

이 절차를 사용하여 시스템을 **HIPAA**와 같은 보안 프로필에 정렬하는 수정이 포함된 **Bash** 스크립트를 생성합니다. 다음 단계를 사용하여 시스템을 수정하지 않고 이후 애플리케이션을 위한 파일만 준비합니다.

사전 요구 사항

- **scap-security-guide** 패키지가 **RHEL** 시스템에 설치되어 있습니다.

절차

1. **oscap** 명령을 사용하여 시스템을 스캔하고 결과를 **XML** 파일에 저장합니다. 다음 예에서 **oscap**은 **hipaa** 프로필에 대해 시스템을 평가합니다.

```
# oscap xccdf eval --profile hipaa --results <hipaa-results.xml>
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

2. 결과를 사용하여 파일에서 결과 **ID** 값을 찾습니다.

```
# oscap info <hipaa-results.xml>
```

3. **1**단계에서 생성된 결과 파일을 기반으로 **Bash** 스크립트를 생성합니다.

```
# oscap xccdf generate fix --fix-type bash --result-id <xccdf_org.open-
scap_testresult_xccdf_org.ssgproject.content_profile_hipaa> --output <hipaa-
remediations.sh> <hipaa-results.xml>
```

4. **& lt;hipaa-remediations.sh >** 파일에는 **1**단계에서 수행한 검사 중에 실패한 규칙에 대한 수정이 포함되어 있습니다. 생성된 이 파일을 검토한 후 이 파일과 동일한 디렉터리에 있는 경우 **./ <hipaa-remediations.sh >** 명령으로 적용할 수 있습니다.

검증

- 선택한 텍스트 편집기에서 **< hipaa-remediations.sh >** 파일에 **1**단계에서 수행한 검사에 실패한 규칙이 포함되어 있는지 검토합니다.

추가 리소스

- **scap-security-guide(8), oscap(8)** 및 **bash(1)** 도움말 페이지

6.8. SCAP WORKBENCH를 사용하여 사용자 지정 프로필로 시스템 검사

scap-workbench 패키지에 포함된 **SCAP Workbench**는 사용자가 단일 로컬 또는 원격 시스템에서 구성 및 취약점 검사를 수행하고 시스템 수정을 수행하고 스캔 평가를 기반으로 보고서를 생성하는 그래픽

유틸리티입니다. **SCAP Workbench**에는 **oscap** 명령줄 유틸리티에 비해 기능이 제한되어 있습니다. **SCAP Workbench** 는 데이터 스트림 파일 형식으로 보안 콘텐츠를 처리합니다.

6.8.1. SCAP Workbench를 사용하여 시스템 검사 및 수정

선택한 보안 정책에 대해 시스템을 평가하려면 다음 절차를 사용합니다.

사전 요구 사항

- **scap-workbench** 패키지가 시스템에 설치되어 있습니다.

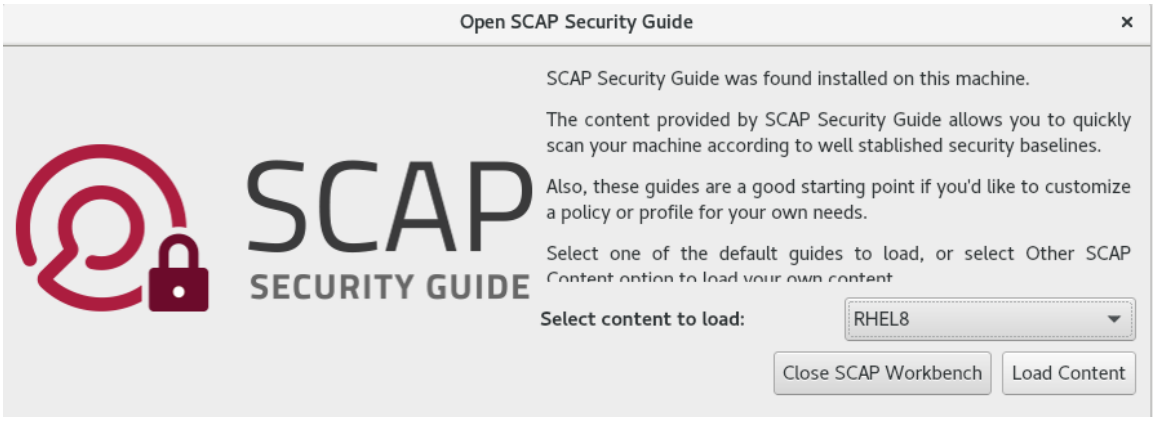
절차

1. **GNOME Classic** 데스크탑 환경에서 **SCAP Workbench**를 실행하려면 **Super** 키를 눌러 **Activities Overview**를 입력하고 **scap-workbench**를 입력한 다음 **Enter** 키를 누릅니다. 또는 다음을 사용합니다.

```
$ scap-workbench &
```

2. 다음 옵션 중 하나를 사용하여 보안 정책을 선택합니다.

- 시작 창에서 콘텐츠 로드 버튼
- **SCAP** 보안 가이드에서 콘텐츠 열기
- **File** (파일) 메뉴에서 기타 콘텐츠를 열고 해당 **XCCDF, SCAP RPM** 또는 데이터 스트림 파일을 검색합니다.



3.

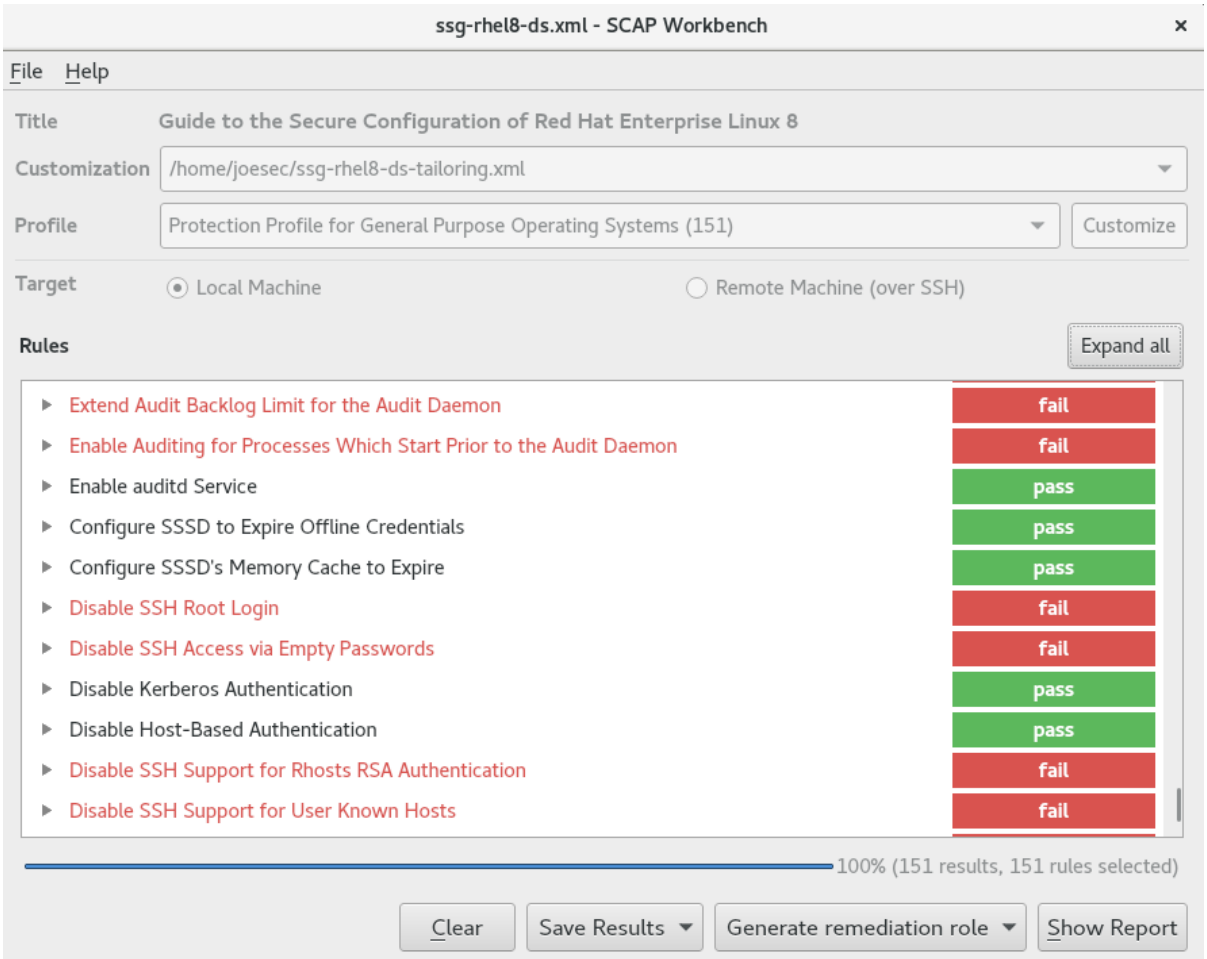
Remediate 확인란을 선택하여 시스템 구성을 자동으로 수정할 수 있습니다. 이 옵션을 활성화하면 SCAP Workbench는 정책에서 적용하는 보안 규칙에 따라 시스템 구성을 변경합니다. 이 프로세스는 시스템 검사 중에 실패하는 관련 검사를 수정해야 합니다.

주의

신중하게 사용하지 않는 경우 Remediate 옵션을 활성화하여 시스템 평가를 실행하면 시스템에 작동하지 않을 수 있습니다. Red Hat은 보안 강화 수정으로 인한 변경 사항을 되돌릴 수 있는 자동화된 방법을 제공하지 않습니다. 수정은 기본 구성의 RHEL 시스템에서 지원됩니다. 설치 후 시스템이 변경된 경우 수정을 실행하여 필요한 보안 프로필을 준수하지 못할 수 있습니다.

4.

Scan 버튼을 클릭하여 선택한 프로필로 시스템을 스캔합니다.



5.

검사 결과를 XCCDF, ARF 또는 HTML 파일의 형식으로 저장하려면 Save Results 콤보 상자를 클릭합니다. HTML Report 옵션을 선택하여 사용자가 읽을 수 있는 형식으로 스캔 보고서를

*생성합니다. **XCCDF** 및 **ARF**(데이터 스트림) 형식은 추가 자동 처리에 적합합니다. 세 가지 옵션을 모두 반복적으로 선택할 수 있습니다.*

6. *결과 기반 수정을 파일로 내보내려면 **Generate remediation** 역할 팝업 메뉴를 사용합니다.*

6.8.2. SCAP Workbench를 사용하여 보안 프로필 사용자 정의

특정 규칙(예: 최소 암호 길이)에서 매개 변수를 변경하고, 다른 방식으로 적용되는 규칙을 제거하고 추가 규칙을 선택하여 내부 정책을 구현하여 보안 프로필을 사용자 지정할 수 있습니다. 프로필을 사용자 지정하여 새 규칙을 정의할 수 없습니다.

*다음 절차에서는 프로필을 사용자 지정하는 데 **SCAP Workbench**를 사용하는 방법을 보여줍니다. **oscap** 명령줄 유틸리티와 함께 사용할 맞춤형 프로필을 저장할 수도 있습니다.*

사전 요구 사항

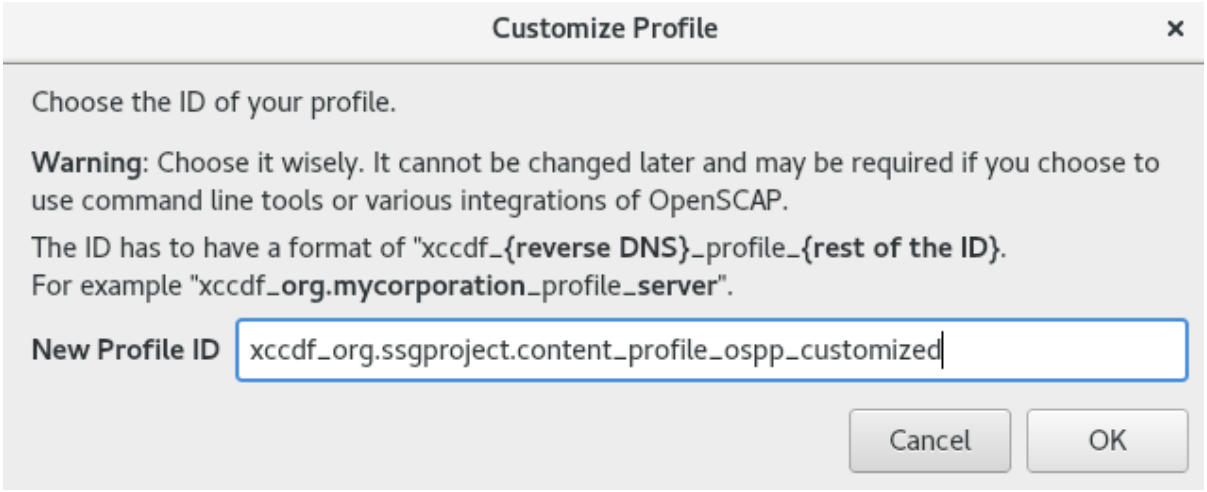
- ***scap-workbench** 패키지가 시스템에 설치되어 있습니다.*

절차

1. ***SCAP Workbench**를 실행하고 **File** 메뉴에서 **Open content from SCAP Security Guide** 또는 **Open Other Content**를 열어 사용자 지정할 프로필을 선택합니다.*

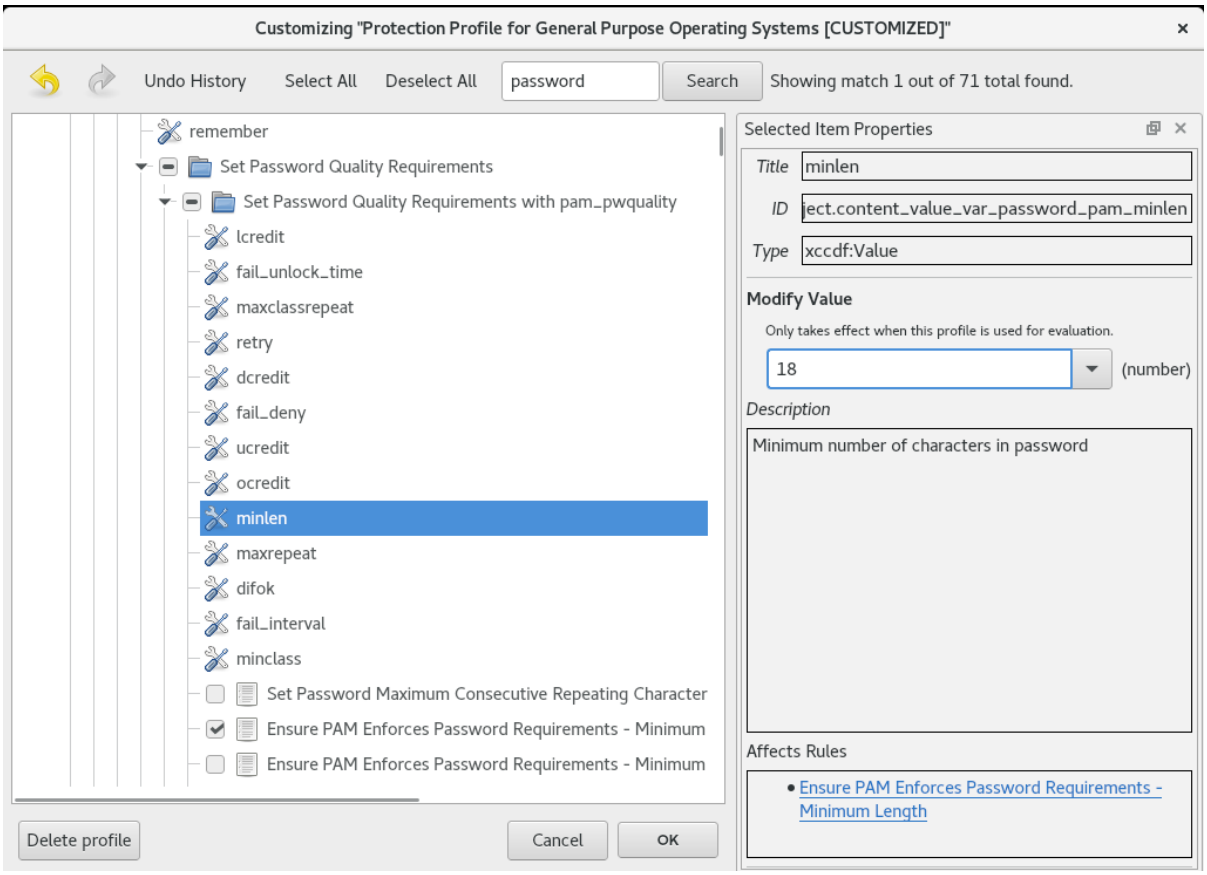
2. *요구 사항에 따라 선택한 보안 프로필을 조정하려면 사용자 지정 버튼을 클릭합니다.*

*그러면 원래 데이터 스트림 파일을 변경하지 않고 현재 선택한 프로필을 수정할 수 있는 새 사용자 지정 창이 열립니다. 새 프로필 **ID**를 선택합니다.*



3. *논리 그룹 또는 **Search** (검색) 필드로 구성된 규칙과 함께 트리 구조를 사용하여 수정하는 규칙을 찾습니다.*

4. *트리 구조의 확인란을 사용하여 규칙을 포함하거나 제외하거나 해당하는 규칙의 값을 수정합니다.*



5. ***OK** (확인) 버튼을 클릭하여 변경 사항을 확인합니다.*

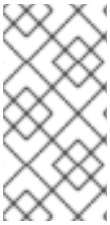
6. *변경 사항을 영구적으로 저장하려면 다음 옵션 중 하나를 사용합니다.*

- ***File** 메뉴에서 **Save Customization only**을 사용하여 사용자 지정 파일을 별도로 저장합니다.*

- ***File** 메뉴에서 **Save All**을 사용하여 한 번에 모든 보안 콘텐츠를 저장합니다.*

***Into a directory** 옵션을 선택하면 **SCAP Workbench**는 데이터 스트림 파일과 사용자 지정 파일을 지정된 위치에 모두 저장합니다. 이를 백업 솔루션으로 사용할 수 있습니다.*

As RPM 옵션을 선택하면 SCAP Workbench에 데이터 스트림 파일과 사용자 지정 파일이 포함된 RPM 패키지를 생성하도록 지시할 수 있습니다. 이 기능은 원격으로 스캔할 수 없는 시스템에 보안 콘텐츠를 배포하고 추가 처리를 위해 콘텐츠를 전달하는 데 유용합니다.



참고

SCAP Workbench 는 맞춤형 프로필의 결과 기반 수정을 지원하지 않으므로 oscap 명령줄 유틸리티와 함께 내보낸 수정을 사용합니다.

6.8.3. 추가 리소스

- scap-workbench(8) 도움말 페이지
- scap-workbench 패키지에서 제공하는 /usr/share/doc/scap-workbench/user_manual.html 파일
- Satellite 6.x를 사용하여 사용자 정의 SCAP 정책 배포 KCS 문서

6.9. 설치 직후 보안 프로필과 호환되는 시스템 배포

OpenSCAP 제품군을 사용하여 설치 프로세스 직후에 OSPP, PCI-DSS 및 HIPAA 프로필과 같은 보안 프로필과 호환되는 RHEL 시스템을 배포할 수 있습니다. 이 배포 방법을 사용하여 나중에 해결 스크립트를 사용하여 적용할 수 없는 특정 규칙을 적용할 수 있습니다 (예: 암호 보안 강도 및 파티셔닝 규칙).

6.9.1. GUI에서 서버와 호환되지 않는 프로파일

SCAP 보안 가이드의 일부로 제공되는 특정 보안 프로필은 GUI 기본 환경에서 서버에 포함된 확장 패키지 세트와 호환되지 않습니다. 따라서 다음 프로필 중 하나와 호환되는 시스템을 설치할 때 GUI를 사용하여 서버를 선택하지 마십시오.

표 6.1. GUI에서 서버와 호환되지 않는 프로파일

프로파일 이름	프로파일 ID	이유	참고

프로파일 이름	프로파일 ID	이유	참고
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - 서버	**xccdf_org.ssgproject.content_profile_cis**	패키지 **xorg-x11-server-Xorg**,**xorg-x11-server-server-common**,**xorg-x11-server-server -utils** 및 **xorg-x11-server-Xwayland** 는 **GUI 패키지 세트를 사용하는** 서버의 일부입니다. 그러나 정책에는 제거가 필요합니다.	
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - 서버	**xccdf_org.ssgproject.content_profile_cis_server_l1**	패키지 **xorg-x11-server-Xorg**,**xorg-x11-server-server-common**,**xorg-x11-server-server -utils** 및 **xorg-x11-server-Xwayland** 는 **GUI 패키지 세트를 사용하는** 서버의 일부입니다. 그러나 정책에는 제거가 필요합니다.	
비연방 정보 시스템 및 조직의 분류되지 않은 정보 (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	**nfs-utils** 패키지는 **GUI** 패키지가 설정된 서버의 일부이지만 정책을 제거해야 합니다.	
일반적인 용도 운영 체제를 위한 보안 프로필	**xccdf_org.ssgproject.content_profile_ospp**	**nfs-utils** 패키지는 **GUI** 패키지가 설정된 서버의 일부이지만 정책을 제거해야 합니다.	
DISA STIG for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	패키지 **xorg-x11-server-Xorg**,**xorg-x11-server-server-common**,**xorg-x11-server-server -utils** 및 **xorg-x11-server-Xwayland** 는 **GUI 패키지 세트를 사용하는** 서버의 일부입니다. 그러나 정책에는 제거가 필요합니다.	RHEL 버전 8.4 이상에서 DISA STIG **와 일치하는 GUI** 가 있는 서버로 RHEL 시스템을 설치하려면 **DISA STIG with GUI** 프로필을 사용할 수 있습니다.

6.9.2. 그래픽 설치를 사용하여 기준으로 호환되는 RHEL 시스템 배포

*특정 기준과 일치하는 **RHEL** 시스템을 배포하려면 다음 절차를 사용하십시오. 이 예에서는*

OSDPP(General Purpose Operating System)에 보안 프로필을 사용합니다.

주의

SCAP 보안 가이드의 일부로 제공되는 특정 보안 프로필은 GUI 기본 환경에서 서버에 포함된 확장 패키지 세트와 호환되지 않습니다. 자세한 내용은 GUI 서버와 호환되지 않는 프로필을 참조하십시오.

사전 요구 사항

- *graphical 설치 프로그램으로 부팅되었습니다. OSCAP Anaconda 애드온은 대화형 텍스트 전용 설치를 지원하지 않습니다.*
- *Installation Summary 창에 액세스했습니다.*

절차

1. *Installation Summary 창에서 Software Selection을 클릭합니다. Software Selection 창이 열립니다.*
2. *Base Environment 창에서 Server 환경을 선택합니다. 기본 환경 하나만 선택할 수 있습니다.*
3. *Done을 클릭하여 설정을 적용하고 Installation Summary 창으로 돌아갑니다.*
4. *OSPP에는 충족해야 하는 엄격한 파티셔닝 요구 사항이 있으므로 /boot,/home, /var ,/tmp,/var /log,/var/tmp, /var/log/audit 에 대해 별도의 파티션을 생성하십시오.*
5. *Security Policy를 클릭합니다. Security Policy 창이 열립니다.*
6. *시스템에서 보안 정책을 활성화하려면 Apply security policy을 ON으로 전환합니다.*

7. *프로필 창에서 **General Purpose Operating Systems**의 **Protection Profile for General Purpose Operating Systems**를 선택합니다.*

8. ***Select Profile**을 클릭하여 선택을 확인합니다.*

9. ***Changes that were done or need to be done**을 확인합니다. 나머지 수동 변경 사항을 완료합니다.*

10. *그래픽 설치 프로세스를 완료합니다.*

참고

*성공적인 설치 후 그래픽 설치 프로그램은 해당 **Kickstart** 파일을 자동으로 생성합니다. **/root/anaconda-ks.cfg** 파일을 사용하여 **OSPP** 호환 시스템을 자동으로 설치할 수 있습니다.*

검증

- *설치가 완료된 후 시스템의 현재 상태를 확인하려면 시스템을 재부팅하고 새 검사를 시작합니다.*

```
# oscap xccdf eval --profile ospp --report eval_postinstall_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

추가 리소스

- *수동 파티션 설정*

6.9.3. Kickstart를 사용하여 기준 준수 RHEL 시스템 배포

*특정 기준과 일치하는 **RHEL** 시스템을 배포할 수 있습니다. 이 예에서는 **OSDPP(General Purpose Operating System)**에 보안 프로필을 사용합니다.*

사전 요구 사항

- ***scap-security-guide** 패키지는 **RHEL 8** 시스템에 설치됩니다.*

절차

1. 선택한 편집기에서 **/usr/share/scap-security-guide/kickstart/ssg-rhel8-ospp-ks.cfg Kickstart** 파일을 엽니다.

2. 구성 요구 사항에 맞게 파티션 구성표를 업데이트합니다. **OSPP** 규정 준수의 경우 **/boot /home,/var,/tmp,/var/log,/var/tmp, /var/log/audit** 에 대한 별도의 파티션을 유지해야 하며 파티션 크기만 변경할 수 있습니다.

3. **Kickstart**를 사용하여 자동 설치를 수행하는 방법에 설명된 대로 **Kickstart** 설치를 시작합니다.

중요

Kickstart 파일의 암호는 **OSPP** 요구 사항에 대해 확인되지 않습니다.

검증

- 설치가 완료된 후 시스템의 현재 상태를 확인하려면 시스템을 재부팅하고 새 검사를 시작합니다.

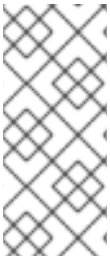
```
# oscap xccdf eval --profile ospp --report eval_postinstall_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

추가 리소스

- **OSCAP Anaconda** 애드온
- **Kickstart** 명령 및 옵션 참조: **%addon org_fedora_oscap**

6.10. 컨테이너 및 컨테이너 이미지에서 취약점 스캔

컨테이너 또는 컨테이너 이미지에서 보안 취약점을 찾으려면 다음 절차를 사용하십시오.



참고

oscap-podman 명령은 **RHEL 8.2**에서 사용할 수 있습니다. **RHEL 8.1** 및 **8.0**의 경우 **RHEL 8 Knowledgebase**의 컨테이너 스캔용 **OpenSCAP** 사용에 설명된 해결방법을 사용하십시오.

사전 요구 사항

- **openscap-utils** 및 **bzip2** 패키지가 설치됩니다.

절차

1. 시스템에 대한 최신 **RHSA OVAL** 정의를 다운로드합니다.

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL8/rhel-8.oval.xml.bz2 | bzip2 --decompress > rhel-8.oval.xml
```

2. 컨테이너 또는 컨테이너 이미지의 **ID**를 가져옵니다. 예를 들면 다음과 같습니다.

```
# podman images
REPOSITORY                              TAG      IMAGE ID       CREATED       SIZE
registry.access.redhat.com/ubi8/ubi     latest   096cae65a207   7 weeks ago   239 MB
```

3. 컨테이너 또는 컨테이너 이미지에서 취약점을 검사하고 결과를 **vulnerability.html** 파일에 저장합니다.

```
# oscap-podman 096cae65a207 oval eval --report vulnerability.html rhel-8.oval.xml
```

oscap-podman 명령에는 루트 권한이 필요하며 컨테이너 **ID**는 첫 번째 인수입니다.

검증

- 선택한 브라우저의 결과를 확인합니다. 예를 들면 다음과 같습니다.

```
$ firefox vulnerability.html &
```

추가 리소스

- 자세한 내용은 **oscap-podman(8)** 및 **oscap(8)** 도움말 페이지를 참조하십시오.

6.11. 특정 기준에서 컨테이너 또는 컨테이너 이미지의 보안 준수 평가

OSP(운영 체제 보호 프로필), PCI-DSS(Payment Card Industry Data Security Standard) 및 **HIPAA(Health Insurance Portability and Accountability Act)**와 같은 특정 보안 기준이 있는 컨테이너 또는 컨테이너 이미지의 규정 준수를 평가할 수 있습니다.

참고

oscap-podman 명령은 **RHEL 8.2**에서 사용할 수 있습니다. **RHEL 8.1** 및 **8.0**의 경우 **RHEL 8 Knowledgebase**의 컨테이너 스캔용 **OpenSCAP** 사용에 설명된 해결방법을 사용하십시오.

사전 요구 사항

- **openscap-utils** 및 **scap-security-guide** 패키지가 설치되어 있습니다.
- 시스템에 대한 **root** 액세스 권한이 있습니다.

절차

1. 컨테이너 또는 컨테이너 이미지의 **ID**를 찾습니다. 예를 들면 다음과 같습니다.

```
# podman images
REPOSITORY                         TAG      IMAGE ID       CREATED       SIZE
registry.access.redhat.com/ubi8/ubi   latest   096cae65a207   7 weeks ago   239 MB
```

2. 프로필로 컨테이너 이미지의 규정 준수를 평가하고 검사 결과를 파일에 저장합니다.

```
# oscap-podman <imageID> xccdf eval --report <scan-report.html> --profile <profileID>
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

교체:

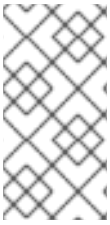
- 컨테이너 이미지의 **ID**가 있는 **<ImageId>**

- *oscap 이 검사 결과를 저장하는 파일 이름이 < scan-report.html >*
- *<profileID > 시스템이 준수해야 하는 프로필 ID(예: hipaa,ospp 또는 pci-dss) 사용*

검증

- *선택한 브라우저의 결과를 확인합니다. 예를 들면 다음과 같습니다.*

```
$ firefox <scan-report.html> &amp;
```



참고

적용되지 않음 으로 표시된 규칙은 컨테이너화된 시스템에 적용되지 않습니다. 이러한 규칙은 베어 메탈 및 가상화 시스템에만 적용됩니다.

추가 리소스

- *oscap-podman(8) 및 scap-security-guide(8) 도움말 페이지.*
- */usr/share/doc/scap-security-guide/ 디렉터리.*

6.12. RHEL 8에서 지원되는 SCAP 보안 가이드 프로필

RHEL의 특정 마이너 릴리스에서 제공된 SCAP 콘텐츠만 사용합니다. 강화에 참여하는 구성 요소가 새로운 기능으로 업데이트되는 경우가 있기 때문입니다. 이러한 업데이트를 반영하기 위해 SCAP 콘텐츠 변경이 필요하지만 이전 버전과 항상 호환되지는 않습니다.

다음 표에서는 프로필이 정렬되는 정책 버전과 함께 RHEL의 각 마이너 버전에서 제공되는 프로필을 찾을 수 있습니다.

표 6.2. RHEL 8.10에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	2.0

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	2.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	3.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	4.0

프로파일 이름	프로파일 ID	정책 버전
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.10.0:V1R13 RHEL 8.10.1 이상:V1R14
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.10.0:V1R13 RHEL 8.10.1 이상:V1R14

표 6.3. RHEL 8.9에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	2.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	RHEL 8.9.0 및 RHEL 8.9.2:2.0.0 RHEL 8.9.3:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	RHEL 8.9.0 및 RHEL 8.9.2:2.0.0 RHEL 8.9.3:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	RHEL 8.9.0 및 RHEL 8.9.2:2.0.0 RHEL 8.9.3:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	RHEL 8.9.0 및 RHEL 8.9.2:2.0.0 RHEL 8.9.3:3.0.0

프로파일 이름	프로파일 ID	정책 버전
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	RHEL 8.9.0 및 RHEL 8.9.2:3.2.1 RHEL 8.9.3:4.0
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.9.0 및 RHEL 8.9.2:V1R11 RHEL 8.9.3:V1R13
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.9.0 및 RHEL 8.9.2:V1R11 RHEL 8.9.3:V1R13

표 6.4. RHEL 8.8에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	2.0

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	2.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	RHEL 8.8.0 및 RHEL 8.8.5:2.0.0 RHEL 8.8.6:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	RHEL 8.8.0 및 RHEL 8.8.5:2.0.0 RHEL 8.8.6:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	RHEL 8.8.0 및 RHEL 8.8.5:2.0.0 RHEL 8.8.6:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	RHEL 8.8.0 및 RHEL 8.8.5:2.0.0 RHEL 8.8.6:3.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	RHEL 8.8.0 및 RHEL 8.8.5:3.2.1 RHEL 8.8.6:4.0
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.8.0 및 RHEL 8.8.5:V1R9 RHEL 8.8.6 및 RHEL 8.8.7:V1R13 RHEL 8.8.8.8 이상:V1R14

프로파일 이름	프로파일 ID	정책 버전
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.8.0 및 RHEL 8.8.5:V1R9 RHEL 8.8.6 및 RHEL 8.8.7:V1R13 RHEL 8.8.8.8 이상:V1R14

표 6.5. RHEL 8.7에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	1.2
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	2.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	2.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	2.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	2.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음

프로파일 이름	프로파일 ID	정책 버전
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.7.0 및 RHEL 8.7.1:V1R7 RHEL 8.7.2 이상:V1R9
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.7.0 및 RHEL 8.7.1:V1R7 RHEL 8.7.2 이상:V1R9

표 6.6. RHEL 8.6에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	RHEL 8.6.0에서 8.6.10:1.2 RHEL 8.6.11 이상:2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	RHEL 8.6.0에서 8.6.10:1.2 RHEL 8.6.11 이상:2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	RHEL 8.6.0에서 8.6.10:1.2 RHEL 8.6.11 이상:2.0
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	RHEL 8.6.0에서 8.6.10:1.2 RHEL 8.6.11 이상:2.0

프로파일 이름	프로파일 ID	정책 버전
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	RHEL 8.6.0에서 RHEL 8.6.2:1.0.0 RHEL 8.6.6.3에서 RHEL 8.6.15:2.0.0 RHEL 8.6.16 이상:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	RHEL 8.6.0에서 RHEL 8.6.2:1.0.0 RHEL 8.6.6.3에서 RHEL 8.6.15:2.0.0 RHEL 8.6.16 이상:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	RHEL 8.6.0에서 RHEL 8.6.2:1.0.0 RHEL 8.6.6.3에서 RHEL 8.6.15:2.0.0 RHEL 8.6.16 이상:3.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	RHEL 8.6.0에서 RHEL 8.6.2:1.0.0 RHEL 8.6.6.3에서 RHEL 8.6.15:2.0.0 RHEL 8.6.16 이상:3.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.6.0:V1R5 RHEL 8.6.1 및 RHEL 8.6.2:V1R6 RHEL 8.6.3에서 RHEL 8.6.6:V1R7 RHEL 8.6.7에서 RHEL 8.6.10:V1R9 RHEL 8.6.15:V1R11 RHEL 8.6.15 및 RHEL 8.6.16 및 RHEL 8.6.16 및 RHEL 8.6.16.8.8.6.1v1

프로파일 이름	프로파일 ID	정책 버전
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.6.0:V1R5 RHEL 8.6.1 및 RHEL 8.6.2:V1R6 RHEL 8.6.3에서 RHEL 8.6.6:V1R7 RHEL 8.6.7에서 RHEL 8.6.10:V1R9 RHEL 8.6.15:V1R11 RHEL 8.6.15 및 RHEL 8.6.16 및 RHEL 8.6.16 및 RHEL 8.6.16.8.8.6.1v1

표 6.7. RHEL 8.5에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	1.2
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	1.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	1.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	1.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	1.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1

프로파일 이름	프로파일 ID	정책 버전
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.5.0에서 RHEL 8.5.3:V1R3 RHEL 8.5.4 이상:V1R5
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.5.0에서 RHEL 8.5.3:V1R3 RHEL 8.5.4 이상:V1R5

표 6.8. RHEL 8.4에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
French National Agency for the Security of Information Systems (ANSSI) BP-028 Enhanced Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 High Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	RHEL 8.4.4 이상:1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 Intermediary Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	1.2
French National Agency for the Security of Information Systems (ANSSI) BP-028 Minimal Level	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	1.2

프로파일 이름	프로파일 ID	정책 버전
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	RHEL 8.4.3 이상:1.0.0 RHEL 8.4.4에서 RHEL 8.4.10:1.0.1 RHEL 8.4.11 이상:2.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	RHEL 8.4.4에서 RHEL 8.4.10:1.0.1 RHEL 8.4.11 이상:2.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	RHEL 8.4.4에서 RHEL 8.4.10:1.0.1 RHEL 8.4.11 이상:2.0.0
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	RHEL 8.4.4에서 RHEL 8.4.10:1.0.1 RHEL 8.4.11 이상:2.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
Australian Cyber Security Centre (ACSC) ISM Official	**xccdf_org.ssgproject.content_profile_ism_o**	RHEL 8.4.4 이상: 버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	RHEL 8.4.3 및 이전:V1R1 RHEL 8.4.4에서 RHEL 8.4.7:V1R3 RHEL 8.4.8:V1R5 RHEL 8.4.10:V1R6 RHEL 8.4.14:V1R7 RHEL 8.4.14:V1R7 RHEL 8.4.15 이상:V1R9
The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) with GUI for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 8.4.4에서 RHEL 8.4.7:V1R3 RHEL 8.4.8:V1R5 RHEL 8.4.9에서 RHEL 8.4.10:V1R6 RHEL 8.4.14:V1R7 RHEL 8.4.15 이상:V1R9

표 6.9. RHEL 8.3에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
CIS Red Hat Enterprise Linux 8 Benchmark	**xccdf_org.ssgproject.content_profile_cis**	1.0.0
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r1
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
HIPAA(Health Insurance Portability and Accountability Act)	**xccdf_org.ssgproject.content_profile_hipaa**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
[DRAFT] The Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	초안

표 6.10. RHEL 8.2에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	버전이 지정되지 않음
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
[DRAFT] DISA STIG for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	초안

표 6.11. RHEL 8.1에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1

표 6.12. RHEL 8.0에서 지원되는 SCAP 보안 가이드 프로필

프로파일 이름	프로파일 ID	정책 버전
OSPP - Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	초안
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1

6.13. 추가 리소스

- *RHEL에서 지원되는 SCAP 보안 가이드 버전*
- *OpenSCAP 프로젝트 페이지는 oscap 유틸리티 및 SCAP와 관련된 기타 구성 요소 및 프로젝트에 대한 자세한 정보를 제공합니다.*
- *SCAP Workbench 프로젝트 페이지는 scap-workbench 애플리케이션에 대한 자세한 정보를 제공합니다.*
- *SCAP Security Guide (SSG) 프로젝트 페이지에서 는 Red Hat Enterprise Linux에 최신 보안 컨텐츠를 제공합니다.*
- *보안 컴플라이언스 및 취약점 스캔에 OpenSCAP 사용 - RHEL의 규정 준수 및 취약점 스캔을 위해 SCAP(Security Content Automation Protocol) 표준을 기반으로 하는 툴 실행에 대한 실습 랩입니다.*
- *Red Hat 보안 데모: 사용자 지정 보안 정책 콘텐츠 Automate Security Compliance - 업계 표준 보안 정책 및 사용자 지정 보안 정책을 준수하기 위해 RHEL에 포함된 툴을 사용하여 보안 준수 자동화에 대한 초기 경험을 얻을 수 있습니다. 팀을 위해 교육을 받거나 이러한 실습에 액세스하려면 Red Hat 계정 팀에 자세한 내용을 문의하십시오.*

- *Red Hat 보안 데모: RHEL Security Technologies - OpenSCAP을 포함하여 RHEL에서 사용 가능한 주요 보안 기술을 사용하여 RHEL 시스템의 모든 수준에서 보안을 구현하는 방법을 배울 수 있는 실습 랩입니다. 팀을 위해 교육을 받거나 이러한 실습에 액세스하려면 Red Hat 계정 팀에 자세한 내용을 문의하십시오.*

- *NSS (National research of Standards and Technology) SCAP 페이지에 는 SCAP 발행물, 사양 및 SCAP 검증 프로그램을 포함한 광범위한 SCAP 관련 자료 컬렉션이 있습니다.*

- *NVD(National Vulnerability Database) 는 가장 큰 SCAP 콘텐츠 저장소와 기타 SCAP 표준 기반 취약점 관리 데이터를 보유하고 있습니다.*

- *Red Hat OVAL 콘텐츠 리포지토리에 는 RHEL 시스템의 취약점에 대한 OVAL 정의가 포함되어 있습니다. 권장되는 취약점 콘텐츠 소스입니다.*

- *MITRE CVE - MITRE 기업에서 제공하는 알려진 보안 취약점의 데이터베이스입니다. RHEL의 경우 Red Hat에서 제공하는 OVAL CVE 콘텐츠를 사용하는 것이 좋습니다.*

- *MITRE OVAL - 이는 MITRE 기업이 제공하는 OVAL 관련 프로젝트입니다. 다른 OVAL 관련 정보 중 하나로 이러한 페이지에는 OVAL 언어 및 수천 개의 OVAL 정의가 포함된 OVAL 콘텐츠 리포지토리가 포함되어 있습니다. RHEL 스캔에는 Red Hat에서 제공하는 OVAL CVE 콘텐츠를 사용하는 것이 좋습니다.*

- *Red Hat Satellite에서 보안 규정 준수 관리 - 이 가이드 세트는 OpenSCAP을 사용하여 여러 시스템에서 시스템 보안을 유지하는 방법을 설명합니다.*

7장. AIDE로 무결성 확인

AIDE(Advanced Intrusion Detection Environment)는 시스템에서 파일 데이터베이스를 만든 다음 해당 데이터베이스를 사용하여 파일 무결성을 보장하고 시스템 침입을 감지하는 유틸리티입니다.

7.1. AIDE 설치

AIDE를 사용하여 **file-integrity** 검사를 시작하려면 해당 패키지를 설치하고 **AIDE** 데이터베이스를 시작해야 합니다.

사전 요구 사항

- **AppStream** 리포지토리가 활성화되어 있어야 합니다.

절차

1. **aide** 패키지를 설치합니다.

```
# yum install aide
```

2. 초기 데이터베이스를 생성합니다.

```
# aide --init
Start timestamp: 2024-07-08 10:39:23 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries: 55856

---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db.new.gz
…
  SHA512   : mZaWoGzL2m6ZcyyZ/AXTIowliEXWSZqx
             IFYImY4f7id4u+Bq8WeuSE2jasZur/A4
             FPBFaBkoCFHdoE/FW/V94Q==
```

3. 선택 사항: 기본 구성에서 **aide --init** 명령은 **/etc/aide.conf** 파일에 정의된 디렉토리와 파일 집합만 확인합니다. **AIDE** 데이터베이스에 추가 디렉터리 또는 파일을 포함시키고 감시된 매개 변수를 변경하려면 그에 따라 **/etc/aide.conf** 를 편집합니다.

4. *데이터베이스 사용을 시작하려면 초기 데이터베이스 파일 이름에서 **.new** 하위 문자열을 제거합니다.*

```
# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

5. *선택 사항: **AIDE** 데이터베이스의 위치를 변경하려면 **/etc/aide.conf** 파일을 편집하고 **DBDIR** 값을 수정합니다. 보안을 강화하기 위해 데이터베이스, 구성 및 **/usr/sbin/aide** 바이너리 파일을 읽기 전용 미디어와 같은 보안 위치에 저장하십시오.*

7.2. AIDE를 사용하여 무결성 검사 수행

*__crond__ 서비스를 사용하여 **AIDE**에서 일반 **file-integrity** 검사를 예약할 수 있습니다.*

사전 요구 사항

- *__AIDE__가 올바르게 설치되고 데이터베이스가 초기화됩니다. **AIDE** 설치 참조*

절차

1. *수동 검사를 시작하려면 다음을 수행합니다.*

```
# aide --check
Start timestamp: 2024-07-08 10:43:46 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!!

Summary:
  Total number of entries: 55856
  Added entries:  0
  Removed entries:  0
  Changed entries:  1

---------------------------------------------------
Changed entries:
---------------------------------------------------

f   ...      ..S : /root/.viminfo

---------------------------------------------------
Detailed information about changes:
---------------------------------------------------

File: /root/.viminfo
```

```
  SELinux  : system_u:object_r:admin_home_t:s | unconfined_u:object_r:admin_home
             0                               | _t:s0
…
```

2. 최소한 **AIDE**가 매주 **AIDE**를 실행하도록 시스템을 구성합니다. 최적으로 **AIDE**를 매일 실행합니다. 예를 들어 **cron** 명령을 사용하여 매일 **AIDE** 실행을 **04:05 a.m**로 예약하려면 **/etc/crontab** 파일에 다음 행을 추가합니다.

```
05 4 * * * root /usr/sbin/aide --check
```

추가 리소스

- *시스템에 **cron(8)** 도움말 페이지*

7.3. AIDE 데이터베이스 업데이트

*패키지 업데이트 또는 구성 파일 조정과 같은 시스템 변경 사항을 확인한 후 기본 **AIDE** 데이터베이스도 업데이트합니다.*

사전 요구 사항

- ***AIDE**가 올바르게 설치되고 데이터베이스가 초기화됩니다. **AIDE 설치** 참조*

절차

1. *기준 **AIDE** 데이터베이스를 업데이트합니다.*

```
# aide --update
```

   ***aide --update** 명령은 **/var/lib/aide/aide.db.new.gz** 데이터베이스 파일을 만듭니다.*

2. *업데이트된 데이터베이스를 사용하여 무결성 검사를 시작하려면 파일 이름에서 **.new** 하위 문자열을 제거합니다.*

7.4. 파일 통합 툴: AIDE 및 IMA

***Red Hat Enterprise Linux**는 시스템에서 파일 및 디렉토리의 무결성을 확인하고 유지하기 위한 몇 가지 툴을 제공합니다. 다음 표는 시나리오에 가장 적합한 도구를 결정하는 데 도움이 됩니다.*

표 7.1. AIDE와 IMA 간 비교

질문	AIDE(Advanced Intrusion Detection Environment)	무결성 측정 아키텍처(IMA)
내용	AIDE는 시스템에 파일 및 디렉터리의 데이터베이스를 생성하는 유틸리티입니다. 이 데이터베이스는 파일 무결성을 확인하고 침입을 감지하는 데 사용됩니다.	IMA는 이전에 저장된 확장 속성에 비해 파일 측정(해시 값)을 확인하여 파일이 변경되었는지 여부를 감지합니다.
방법	AIDE에서는 규칙을 사용하여 파일과 디렉터리의 무결성 상태를 비교합니다.	IMA는 파일 해시 값을 사용하여 침입을 감지합니다.
이유	감지 - AIDE에서 규칙을 확인하여 파일이 수정되는지 여부를 감지합니다.	감지 및 예방 - IMA는 파일의 확장 속성을 교체하여 공격을 감지하고 차단합니다.
사용법	파일 또는 디렉터리가 수정되면 AIDE에서 위협을 감지합니다.	누군가가 전체 파일을 변경하려고 할 때 위협을 감지합니다.
확장	AIDE는 로컬 시스템에서 파일 및 디렉터리의 무결성을 확인합니다.	IMA는 로컬 및 원격 시스템에 대한 보안을 보장합니다.

7.5. 추가 리소스

- ***AIDE(1) 도움말 페이지***
- *커널 무결성 하위 시스템*

8장. 커널 무결성 하위 시스템으로 보안 강화

커널 무결성 하위 시스템의 구성 요소를 사용하여 시스템 보호를 개선할 수 있습니다. 관련 구성 요소 및 해당 구성에 대해 자세히 알아보십시오.

8.1. 커널 무결성 하위 시스템

무결성 하위 시스템은 시스템 데이터의 전체 무결성을 유지 관리하는 커널의 일부입니다. 이 하위 시스템은 시스템 상태를 빌드한 시점과 동일하게 유지하는 데 도움이 됩니다. 이 하위 시스템을 사용하면 특정 시스템 파일의 바람직하지 않은 수정을 방지할 수 있습니다.

커널 무결성 하위 시스템은 다음 두 가지 주요 구성 요소로 구성됩니다.

무결성 측정 아키텍처(IMA)

- IMA는 암호화 방식으로 해시 또는 암호화 키를 사용하여 서명을 통해 실행되거나 열 때마다 파일 콘텐츠를 측정합니다. 키는 커널 키링 하위 시스템에 저장됩니다.
- IMA는 측정된 값을 커널의 메모리 공간에 배치합니다. 이렇게 하면 시스템의 사용자가 측정 값을 수정하지 못하도록 합니다.
- IMA를 사용하면 로컬 및 원격 당사자가 측정 값을 확인할 수 있습니다.
- IMA는 커널 메모리 내의 측정 목록에 이전에 저장된 값에 대해 현재 파일 콘텐츠에 대한 로컬 유효성 검사를 제공합니다. 이 확장은 현재 및 이전 측정값이 일치하지 않는 경우 특정 파일에서 모든 작업을 수행하는 것을 금지합니다.

EVM(Extended Verification Module)

- EVM은 IMA 측정 및 SELinux 속성과 같은 시스템 보안과 관련된 파일의 확장된 속성(xattr라고도 함)을 보호합니다. EVM은 해당 값을 암호화 방식으로 해시하거나 암호화 키를 사용하여 서명합니다. 키는 커널 키링 하위 시스템에 저장됩니다.

커널 무결성 하위 시스템은 신뢰할 수 있는 플랫폼 모듈(TPM)을 사용하여 시스템 보안을 강화할 수 있습니다.

TPM은 중요한 암호화 기능을 위해 신뢰할 수 있는 컴퓨팅 그룹(TCG)의 TPM 사양에 따라 빌드되는 통합 암호화 키가 있는 하드웨어, 펌웨어 또는 가상 구성 요소입니다. TPMS는 일반적으로 플랫폼의 마더보드에 연결된 전용 하드웨어로 구축됩니다. 하드웨어 칩의 보호 및 변조 방지 영역에서 암호화 기능을 제공하여 TPM은 소프트웨어 기반 공격으로부터 보호됩니다. TPMS는 다음 기능을 제공합니다.

- random-number 생성기
- 암호화 키 생성기 및 보안 스토리지
- 해시 생성기
- 원격 인증

추가 리소스

- 보안 강화
- SELinux(Security- Enhanced Linux)의 기본 및 고급 구성

8.2. 신뢰할 수 있는 암호화된 키

신뢰할 수 있는 키와 암호화된 키는 시스템 보안을 강화하는 데 중요한 부분입니다.

신뢰할 수 있고 암호화된 키는 커널 키링 서비스를 사용하는 커널에서 생성되는 변수 길이 대칭 키입니다. 키의 무결성을 확인할 수 있습니다. 즉, 실행 중인 시스템의 무결성을 확인하고 확인하기 위해 확장 검증 모듈(EVM)에서 사용할 수 있습니다. 사용자 수준 프로그램은 암호화된 Blob 형식의 키만 액세스할 수 있습니다.

신뢰할 수 있는 키

신뢰할 수 있는 키에는 키를 생성하고 암호화(seal)하는 데 사용되는 신뢰할 수 있는 플랫폼 모듈(TPM) 칩이 필요합니다. 각 TPM에는 TPM 자체에 저장된 스토리지 루트 키라는 마스터 래핑 키가 있습니다.

참고

Red Hat Enterprise Linux 8은 TPM 1.2 및 TPM 2.0을 모두 지원합니다. 자세한 내용은 Red Hat에서 지원하는 Is Trusted Platform Module (TPM) 솔루션을 참조하십시오.

다음 명령을 입력하여 TPM 2.0 칩이 활성화되어 있는지 확인할 수 있습니다.

```
$ cat /sys/class/tpm/tpm0/tpm_version_major
2
```

TPM 2.0 칩을 활성화하고 머신 펌웨어의 설정을 통해 TPM 2.0 장치를 관리할 수도 있습니다.

또한 신뢰할 수 있는 키를 TPM의 플랫폼 구성 레지스터 (PCR) 값 세트로 봉인할 수 있습니다. PCR에는 펌웨어, 부트 로더 및 운영 체제를 반영하는 무결성 관리 값 세트가 포함되어 있습니다. 즉, PCRsealed 키는 암호화된 동일한 시스템에서 TPM에서만 해독할 수 있습니다. 그러나 PCRsealed 신뢰할 수 있는 키가 로드되고(인증 키에 추가) 연결된 PCR 값이 확인되면 새 커널(예: 새 커널)을 사용하도록 새 (또는 향후) PCR 값으로 업데이트할 수 있습니다. 단일 키를 각각 다른 PCR 값을 가진 여러 Blob으로 저장할 수 있습니다.

암호화된 키

암호화된 키에는 AES(커널 고급 암호화 표준)를 사용하므로 TPM이 필요하지 않으므로 신뢰할 수 있는 키보다 더 빠릅니다. 암호화된 키는 커널 생성 임의 번호를 사용하여 생성되고 마스터 키로 사용자 공간 Blob으로 내보낼 때 암호화됩니다.

마스터 키는 신뢰할 수 있는 키 또는 사용자 키입니다. 마스터 키를 신뢰할 수 없는 경우 암호화 키는 암호화에 사용된 사용자 키만큼만 보안됩니다.

8.3. 신뢰할 수 있는 키로 작업

keyctl 유틸리티를 사용하여 신뢰할 수 있는 키를 생성, 내보내기, 로드 및 업데이트할 수 있습니다.

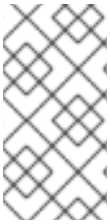
사전 요구 사항

- 64비트 ARM 아키텍처 및 IBM Z의 경우 trusted 커널 모듈이 로드됩니다.

```
# modprobe trusted
```

커널 모듈을 로드하는 방법에 대한 자세한 내용은 시스템 런타임 시 커널 모듈 로드 를 참조하십시오.

- 신뢰할 수 있는 플랫폼 모듈(TPM)이 활성화되어 활성화되어 있습니다. 커널 무결성 하위 시스템 및 신뢰할 수 있고 암호화된 키를 참조하십시오.



참고

Red Hat Enterprise Linux 8은 TPM 1.2 및 TPM 2.0을 모두 지원합니다. TPM 1.2를 사용하는 경우 1단계를 건너뜁니다.

절차

1. 다음 유틸리티 중 하나를 사용하여 영구 처리(예: 81000001)가 있는 SHA-256 기본 스토리지 키로 2048비트 RSA 키를 생성합니다.

   a. tss2 패키지를 사용하여 다음을 수행합니다.

```
# TPM_DEVICE=/dev/tpm0 tsscreateprimary -hi o -st
Handle 80000000
# TPM_DEVICE=/dev/tpm0 tssevictcontrol -hi o -ho 80000000 -hp 81000001
```

   b. tpm2-tools 패키지를 사용하여 다음을 수행합니다.

```
# tpm2_createprimary --key-algorithm=rsa2048 --key-context=key.ctxt
name-alg:
  value: sha256
  raw: 0xb
...
sym-keybits: 128
rsa: xxxxxx...

# tpm2_evictcontrol -c key.ctxt 0x81000001
persistentHandle: 0x81000001
action: persisted
```

2. 신뢰할 수 있는 키를 생성합니다.

   a. keyctl 구문으로 TPM 2.0을 사용하면 trusted < NAME > "new < KEY_LENGTH > keyhandle= <PERSISTENT-HANDLE > [options]" < KEYRING > . 이 예에서 영구 처리는

81000001 입니다.

```
# keyctl add trusted kmk "new 32 keyhandle=0x81000001" @u
642500861
```

이 명령은 32 바이트(256비트) 길이로 kmk 라는 신뢰할 수 있는 키를 생성하여 사용자 인증 키(@u)에 배치합니다. 키는 32~128바이트(256비트에서 1024비트)의 길이를 가질 수 있습니다.

b. keyctl 구문으로 TPM 1.2를 사용하면 신뢰할 수 있는 < NAME > "new <KEY_LENGTH>" <KEYRING > :

```
# keyctl add trusted kmk "new 32" @u
```

3. 커널 인증 키의 현재 구조를 나열합니다.

```
# keyctl show
Session Keyring
     -3 --alswrv    500   500  keyring: ses 97833714 --alswrv 500 -1 \ keyring: uid.1000
642500861 --alswrv 500 500 \ trusted: kmk
```

4. 신뢰할 수 있는 키의 일련 번호를 사용하여 사용자 공간 Blob으로 키를 내보냅니다.

```
# keyctl pipe 642500861 > kmk.blob
```

명령은 pipe 하위 명령과 kmk 의 일련 번호를 사용합니다.

5. 사용자 공간 Blob에서 신뢰할 수 있는 키를 로드합니다.

```
# keyctl add trusted kmk "load `cat kmk.blob`" @u
268728824
```

6. TPM-sealed 신뢰할 수 있는 키(kmk)를 사용하는 안전한 암호화된 키를 만듭니다. 다음 구문을 따르십시오. keyctl add encrypted <NAME> "new [FORMAT] <KEY_TYPE>:<PRI Cryostat_KEY_NAME> <KEY_LENGTH>" <KEYRING > :

```
# keyctl add encrypted encr-key "new trusted:kmk 32" @u
159771175
```

추가 리소스

- keyctl(1) 매뉴얼 페이지
- 신뢰할 수 있는 암호화된 키
- 커널 키 보존 서비스
- 커널 무결성 하위 시스템

8.4. 암호화된 키 작업

암호화된 키를 관리하여 신뢰할 수 있는 플랫폼 모듈(TPM)을 사용할 수 없는 시스템에서 시스템 보안을 개선할 수 있습니다.

사전 요구 사항

- 64비트 ARM 아키텍처 및 IBM Z의 경우 encrypted-keys 커널 모듈이 로드됩니다.

```
# modprobe encrypted-keys
```

커널 모듈을 로드하는 방법에 대한 자세한 내용은 시스템 런타임 시 커널 모듈 로드 를 참조하십시오.

절차

1. 임의의 숫자 시퀀스를 사용하여 사용자 키를 생성합니다.

```
# keyctl add user kmk-user "$(dd if=/dev/urandom bs=1 count=32 2>/dev/null)" @u
427069434
```

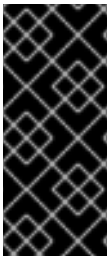
명령은 기본 키 역할을 하고 실제 암호화된 키를 봉인하는 데 사용되는 kmk-user 라는 사용자 키를 생성합니다.

2. 이전 단계의 기본 키를 사용하여 암호화된 키를 생성합니다.

```
# keyctl add encrypted encr-key "new user:kmk-user 32" @u
1012412758
```

3. 선택적으로 지정된 사용자 인증 키의 모든 키를 나열합니다.

```
# keyctl list @u
2 keys in keyring:
427069434: --alswrv  1000  1000 user: kmk-user
1012412758: --alswrv  1000  1000 encrypted: encr-key
```



중요

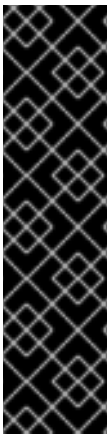
신뢰할 수 있는 기본 키로 봉인되지 않은 암호화된 키는 암호화에 사용된 사용자 기본 키(임의 숫자 키)만큼 안전합니다. 따라서 기본 사용자 키를 최대한 안전하게 로드하고 부팅 프로세스 중 조기에 로드됩니다.

추가 리소스

- **keyctl(1)** 매뉴얼 페이지
- 커널 키 보존 서비스

8.5. IMA 및 EVM 활성화

무결성 측정 아키텍처(**IMA**) 및 **EVM**(확장 확인 모듈)을 활성화하고 구성하여 운영 체제의 보안을 개선할 수 있습니다.



중요

항상 **IMA**와 함께 **EVM**을 활성화합니다.

EVM만 활성화할 수 있지만 **EVM** 평가는 **IMA** 평가 규칙에 의해서만 트리거됩니다. 따라서 **EVM**은 **SELinux** 속성과 같은 파일 메타데이터를 보호하지 않습니다. 파일 메타데이터가 오프라인으로 변조되는 경우 **EVM**은 파일 메타데이터 변경만 방지할 수 있습니다. 파일 실행과 같은 파일 액세스를 차단하지 않습니다.

사전 요구 사항

- *Secure Boot는 일시적으로 비활성화되어 있습니다.*

참고

Secure Boot가 활성화되면 ima_appraise=fix 커널 명령줄 매개변수가 작동하지 않습니다.

- *securityfs 파일 시스템은 /sys/kernel/security/ 디렉터리에 마운트되고 /sys/kernel/security/integrity/ima/ 디렉터리가 있습니다. mount 명령을 사용하여 securityfs 가 마운트된 위치를 확인할 수 있습니다.*

```
# mount
...
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
...
```

- *systemd 서비스 관리자는 부팅 시 IMA 및 EVM을 지원하도록 패치됩니다. 다음 명령을 사용하여 확인합니다.*

```
# grep <options> pattern <files>
```

예를 들어 다음과 같습니다.

```
# dmesg | grep -i -e EVM -e IMA -w
[ 0.598533] ima: No TPM chip found, activating TPM-bypass!
[ 0.599435] ima: Allocated hash algorithm: sha256
[ 0.600266] ima: No architecture policies found
[ 0.600813] evm: Initialising EVM extended attributes:
[ 0.601581] evm: security.selinux
[ 0.601963] evm: security.ima
[ 0.602353] evm: security.capability
[ 0.602713] evm: HMAC attrs: 0x1
[ 1.455657] systemd[1]: systemd 239 (239-74.el8_8) running in system mode. (+PAM
+AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP
+GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -
IDN +PCRE2 default-hierarchy=legacy)
[ 2.532639] systemd[1]: systemd 239 (239-74.el8_8) running in system mode. (+PAM
+AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP
+GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -
IDN +PCRE2 default-hierarchy=legacy)
```

절차

1.

*현재 부팅 항목에 대한 수정 모드에서 **IMA** 및 **EVM**을 활성화하고 사용자가 다음 커널 명령줄 매개변수를 추가하여 **IMA** 측정값을 수집하고 업데이트할 수 있습니다.*

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --
args="ima_policy=appraise_tcb ima_appraise=fix evm=fix"
```

*이 명령을 사용하면 현재 부팅 항목에 대한 수정 모드에서 **IMA** 및 **EVM**을 활성화하고 사용자가 **IMA** 측정값을 수집하고 업데이트할 수 있습니다.*

***ima_policy=appraise_tcb** 커널 명령줄 매개 변수를 사용하면 커널이 기본 **trusted Computing Base(ECDHEB)** 측정 정책 및 평가 단계를 사용합니다. 평가 단계는 이전 및 현재 측정값이 일치하지 않는 파일에 대한 액세스를 금지합니다.*

2. *재부팅하여 변경 사항을 적용합니다.*

3. *선택 사항: 매개변수가 커널 명령줄에 추가되었는지 확인합니다.*

```
# cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-167.el8.x86_64 root=/dev/mapper/rhel-
root ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet ima_policy=appraise_tcb ima_appraise=fix evm=fix
```

4. *커널 마스터 키를 생성하여 **EVM** 키를 보호합니다.*

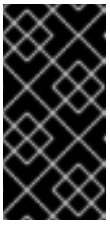
```
# keyctl add user kmk "$(dd if=/dev/urandom bs=1 count=32 2> /dev/null)" @u
748544121
```

***kmk** 는 커널 공간 메모리에 전적으로 유지됩니다. **kmk** 의 **32**바이트 긴 값은 **/dev/urandom** 파일에서 임의의 바이트에서 생성되고 사용자(**@u**) 인증 키에 배치됩니다. 키 일련 번호는 이전 출력의 첫 번째 행에 있습니다.*

5. ***kmk** 를 기반으로 암호화된 **EVM** 키를 만듭니다.*

```
# keyctl add encrypted evm-key "new user:kmk 64" @u
641780271
```

*명령은 **kmk** 를 사용하여 **64**바이트 긴 사용자 키(**evm-key**)를 생성하고 암호화하여 사용자(**@u**) 인증 키에 배치합니다. 키 일련 번호는 이전 출력의 첫 번째 행에 있습니다.*



중요

*이 이름은 **EVM** 하위 시스템 이름이 필요하며 작업 중이므로 사용자 키의 이름을 **evm-key** 로 지정해야 합니다.*

6. *내보낸 키에 사용할 디렉토리를 생성합니다.*

```
# mkdir -p /etc/keys/
```

7. *kmk 를 검색하고 암호화되지 않은 값을 새 디렉터리로 내보냅니다.*

```
# keyctl pipe $(keyctl search @u user kmk) > /etc/keys/kmk
```

8. *evm-key 를 검색하고 암호화된 값을 새 디렉터리로 내보냅니다.*

```
# keyctl pipe $(keyctl search @u encrypted evm-key) > /etc/keys/evm-key
```

evm-key 는 이전에 커널 마스터 키로 암호화되었습니다.

9. *선택 사항: 새로 생성된 키를 확인합니다.*

```
# keyctl show
Session Keyring
974575405   --alswrv     0       0      keyring: ses 299489774 --alswrv 0 65534 \
keyring: uid.0 748544121 --alswrv 0 0 \ user: kmk
641780271   --alswrv     0       0           \_ encrypted: evm-key

# ls -l /etc/keys/
total 8
-rw-r--r--. 1 root root 246 Jun 24 12:44 evm-key
-rw-r--r--. 1 root root  32 Jun 24 12:43 kmk
```

10. *선택 사항: 예를 들어 시스템을 재부팅한 후 인증 키에서 키가 제거된 경우 새 키를 생성하는 대신 이미 내보낸 **kmk** 및 **evm-key** 를 가져올 수 있습니다.*

    a. *kmk 를 가져옵니다.*

```
# keyctl add user kmk "$(cat /etc/keys/kmk)" @u
451342217
```

b. *evm-key 를 가져옵니다.*

```
# keyctl add encrypted evm-key "load $(cat /etc/keys/evm-key)" @u
924537557
```

11. *EVM을 활성화합니다.*

```
# echo 1 > /sys/kernel/security/evm
```

12. *전체 시스템의 레이블을 다시 지정합니다.*

```
# find / -fstype xfs -type f -uid 0 -exec head -n 1 '{}' >/dev/null \;
```

주의

시스템의 레이블을 다시 지정하지 않고 IMA 및 EVM을 활성화하면 시스템의 대부분의 파일에 액세스할 수 없게 될 수 있습니다.

검증

- *EVM이 초기화되었는지 확인합니다.*

```
# dmesg | tail -1
[...] evm: key initialized
```

추가 리소스

- *grep(1) 도움말 페이지*
- *커널 무결성 하위 시스템*

- 신뢰할 수 있는 암호화된 키

8.6. 무결성 측정 아키텍처를 사용하여 파일 해시 수집

측정 단계에서는 파일 해시를 생성하여 해당 파일의 확장 속성(xattrs)으로 저장할 수 있습니다. 파일 해시를 사용하면 RSA 기반 디지털 서명 또는 HMAC-SHA1(HMAC-SHA1)을 생성하여 확장된 속성에 대한 오프라인 변조 공격을 방지할 수 있습니다.

사전 요구 사항

- IMA 및 EVM이 활성화됩니다. 자세한 내용은 무결성 측정 아키텍처 활성화 및 확장 검증 모듈을 참조하십시오.
- 신뢰할 수 있는 유효한 키 또는 암호화된 키는 커널 인증 키에 저장됩니다.
- ima-evm-utils,attr 및 keyutils 패키지가 설치됩니다.

절차

1. 테스트 파일을 생성합니다.

```
# echo <Test_text> > test_file
```

IMA 및 EVM은 test_file 예제 파일에 확장 속성으로 저장된 해시 값을 할당했는지 확인합니다.

2. 파일의 확장 속성을 검사합니다.

```
# getfattr -m . -d test_file
# file: test_file
security.evm=0sAnDIy4VPA0HArpPO/EqiutnNyBql
security.ima=0sAQOEDeuUnWzwwKYk+n66h/vby3eD
```

예제 출력에는 IMA 및 EVM 해시 값과 SELinux 컨텍스트가 포함된 확장된 속성이 표시되어 있습니다. EVM은 다른 속성과 관련된 security.evm 확장 속성을 추가합니다. 이 시점에서 security.evm 에서 evmctl 유틸리티를 사용하여 RSA 기반 디지털 서명 또는 HMAC-SHA1(Hash-based Message Authentication Code)을 생성할 수 있습니다.

추가 리소스

- *보안 강화*

9장. LUKS를 사용하여 블록 장치 암호화

디스크 암호화를 사용하면 블록 장치의 데이터를 암호화하여 보호할 수 있습니다. 장치의 암호 해독된 콘텐츠에 액세스하려면 암호 또는 키를 인증으로 입력합니다. 이는 시스템에서 물리적으로 제거된 경우에도 장치의 콘텐츠를 보호하는 데 도움이 되므로 이동식 미디어와 이동식 미디어에 중요합니다. **LUKS** 형식은 **Red Hat Enterprise Linux**에서 블록 장치 암호화의 기본 구현입니다.

9.1. LUKS 디스크 암호화

Linux Unified Key Setup-on-disk-format (LUKS)은 암호화된 장치 관리를 단순화하는 도구 세트를 제공합니다. **LUKS**를 사용하면 블록 장치를 암호화하고 여러 사용자 키를 활성화하여 마스터 키를 해독할 수 있습니다. 파티션의 대규모 암호화의 경우 이 마스터 키를 사용합니다.

Red Hat Enterprise Linux는 **LUKS**를 사용하여 블록 장치 암호화를 수행합니다. 기본적으로 블록 장치를 암호화하는 옵션은 설치 중에 선택되지 않습니다. 디스크를 암호화할 옵션을 선택하면 시스템을 부팅할 때마다 시스템에서 암호를 입력하라는 메시지가 표시됩니다. 이 암호는 파티션을 해독하는 대규모 암호화 키의 잠금을 해제합니다. 기본 파티션 테이블을 수정하려면 암호화할 파티션을 선택할 수 있습니다. 이는 파티션 테이블 설정에서 설정됩니다.

암호화

LUKS에 사용되는 기본 암호는 **aes-xts-plain64** 입니다. **LUKS**의 기본 키 크기는 **512**비트입니다. **Anaconda XTS** 모드를 사용하는 **LUKS**의 기본 키 크기는 **512**비트입니다. 다음은 사용 가능한 암호입니다.

- **Advanced Encryption Standard(AES)**
- **Twofish**
- **Serpent**

LUKS에서 수행하는 작업

- **LUKS**는 전체 블록 장치를 암호화하므로 이동식 스토리지 미디어 또는 랩톱 디스크 드라이브와 같은 모바일 장치의 콘텐츠를 보호하는 데 적합합니다.
- 암호화된 블록 장치의 기본 내용은 임의이므로 스왑 장치를 암호화하는 데 유용합니다. 이는 데이터 저장을 위해 특별히 포맷된 블록 장치를 사용하는 특정 데이터베이스에서도 유용할 수 있

습니다.

- **LUKS**는 기존 장치 매퍼 커널 하위 시스템을 사용합니다.
- **LUKS**는 사전 공격으로부터 보호하는 암호 강화를 제공합니다.
- **LUKS** 장치에는 여러 개의 키 슬롯이 포함되어 있으므로 백업 키 또는 암호를 추가할 수 있습니다.

중요

다음 시나리오에는 **LUKS**를 사용하지 않는 것이 좋습니다.

- **LUKS**와 같은 디스크 암호화 솔루션은 시스템이 꺼진 경우에만 데이터를 보호합니다. 시스템이 있고 **LUKS**가 디스크의 암호를 해독한 후 해당 디스크의 파일은 액세스 권한이 있는 모든 사용자가 사용할 수 있습니다.
- 여러 사용자가 동일한 장치에 별도의 액세스 키를 사용해야 하는 시나리오입니다. **LUKS1** 형식은 **8**개의 키 슬롯을 제공하며 **LUKS2**는 최대 **32**개의 키 슬롯을 제공합니다.
- 파일 수준 암호화가 필요한 애플리케이션입니다.

추가 리소스

- **LUKS** 프로젝트 홈 페이지
- **LUKS On-Disk** 형식 사양
- **FIPS 197: Advanced Encryption Standard(AES)**

9.2. RHEL의 LUKS 버전

Red Hat Enterprise Linux에서 LUKS 암호화의 기본 형식은 LUKS2입니다. 이전 LUKS1 형식은 완전히 지원되며 이전 Red Hat Enterprise Linux 릴리스와 호환되는 형식으로 제공됩니다. LUKS2 재암호화는 LUKS1 재암호화와 비교하여 보다 강력하고 안전한 것으로 간주됩니다.

LUKS2 형식을 사용하면 바이너리 구조를 수정할 필요 없이 다양한 부분을 나중에 업데이트할 수 있습니다. 내부적으로는 메타데이터에 JSON 텍스트 형식을 사용하고, 메타데이터 중복을 제공하고, 메타데이터 손상을 감지하며, 메타데이터 복사본에서 자동으로 복구합니다.

중요

LUKS2 및 LUKS1은 디스크를 암호화하기 위해 다른 명령을 사용하므로 LUKS1만 지원하는 시스템에서 LUKS2를 사용하지 마십시오. LUKS 버전에 잘못된 명령을 사용하면 데이터가 손실될 수 있습니다.

표 9.1. LUKS 버전에 따른 암호화 명령

LUKS 버전	암호화 명령
LUKS2	**cryptsetup reencrypt**
LUKS1	**cryptsetup-reencrypt**

온라인 재암호화

LUKS2 형식은 장치가 사용 중인 동안 암호화된 장치 재암호화를 지원합니다. 예를 들어 다음 작업을 수행하기 위해 장치에서 파일 시스템을 마운트 해제할 필요가 없습니다.

- *볼륨 키 변경*
- *암호화 알고리즘 변경*

암호화되지 않은 장치를 암호화할 때 파일 시스템을 마운트 해제해야 합니다. 암호화를 간단히 초기화한 후 파일 시스템을 다시 마운트할 수 있습니다.

LUKS1 형식은 온라인 재암호화 기능을 지원하지 않습니다.

변환

특정 상황에서 LUKS1을 LUKS2로 변환할 수 있습니다. 다음 시나리오에서는 변환이 특히 불가능합니다.

- *LUKS1 장치는 PBD(Policy-Based Decryption) Clevis 솔루션이 사용하는 것으로 표시됩니다. cryptsetup 툴은 일부 luksmeta 메타데이터가 감지되면 장치를 변환하지 않습니다.*

- *장치가 활성 상태입니다. 변환이 가능하려면 장치가 비활성 상태여야 합니다.*

9.3. LUKS2 재암호화 중에 데이터 보호 옵션

LUKS2는 재암호화 프로세스 중에 성능 또는 데이터 보호 우선 순위를 지정하는 몇 가지 옵션을 제공합니다. 복구 옵션에 대한 다음 모드를 제공하며 cryptsetup reencrypt --resilience resilience-mode /dev/sdx명령을 사용하여 이러한 모드를 선택할 수 있습니다.

checksum

기본 모드입니다. 데이터 보호 및 성능 균형을 유지합니다.

이 모드에서는 재암호화 영역에 섹터의 개별 체크섬을 저장하므로, LUKS2에서 다시 암호화한 섹터를 복구 프로세스에서 감지할 수 있습니다. 이 모드에서는 블록 장치 섹터 쓰기가 atomic이어야 합니다.

journal

가장 안전한 모드이지만 가장 느린 모드이기도 합니다. 이 모드는 바이너리 영역에서 재암호화 영역을 저널링하므로 LUKS2는 데이터를 두 번 씁니다.

none

none 모드는 성능에 우선 순위를 지정하고 데이터 보호를 제공하지 않습니다. SIGTERM 신호 또는 Ctrl+C 키를 누른 사용자와 같은 안전한 프로세스 종료로부터만 데이터를 보호합니다. 예기치 않은 시스템 오류 또는 애플리케이션 오류로 인해 데이터가 손상될 수 있습니다.

LUKS2 재암호화 프로세스가 강제 종료되면 LUKS2는 다음 방법 중 하나로 복구를 수행할 수 있습니다.

automatically

다음 작업 중 하나를 수행하면 다음 LUKS2 장치 열기 작업 중에 자동 복구 작업이 트리거됩니다.

- *cryptsetup open 명령 실행*

- ***systemd-cryptsetup*** *명령을 사용하여 장치를 연결합니다.*

Manual

LUKS2 *장치에서* ***cryptsetup repair /dev/sdx*** *명령을 사용합니다.*

추가 리소스

- ***cryptsetup-reencrypt(8)*** *및* ***cryptsetup-repair(8)*** *매뉴얼 페이지*

9.4. LUKS2를 사용하여 블록 장치의 기존 데이터 암호화

LUKS2 *형식을 사용하여 아직 암호화되지 않은 장치에서 기존 데이터를 암호화할 수 있습니다. 새* ***LUKS*** *헤더가 장치의 헤드에 저장됩니다.*

사전 요구 사항

- *블록 장치에는 파일 시스템이 있습니다.*
- *데이터를 백업했습니다.*

주의

하드웨어, 커널 또는 사람의 오류로 인해 암호화 프로세스 중에 데이터가 손실될 수 있습니다. 데이터 암호화를 시작하기 전에 신뢰할 수 있는 백업이 있는지 확인합니다.

절차

1. *암호화하려는 장치에서 모든 파일 시스템을 마운트 해제합니다. 예를 들면 다음과 같습니다.*

```
# umount /dev/mapper/vg00-lv00
```

2.

LUKS 헤더 저장에 사용 가능한 공간을 만듭니다. 시나리오에 맞는 다음 옵션 중 하나를 사용합니다.

- *논리 볼륨을 암호화하는 경우 파일 시스템의 크기를 조정하지 않고 논리 볼륨을 확장할 수 있습니다. 예를 들어 다음과 같습니다.*

```
# lvextend -L+32M /dev/mapper/vg00-lv00
```

- *parted 와 같은 파티션 관리 도구를 사용하여 파티션을 확장하십시오.*

- *장치의 파일 시스템을 축소합니다. ext 2, ext3 또는 ext4 파일 시스템에 resize2fs 유틸리티를 사용할 수 있습니다. XFS 파일 시스템을 축소할 수 없습니다.*

3. *암호화를 초기화합니다.*

```
# cryptsetup reencrypt --encrypt --init-only --reduce-device-size 32M /dev/mapper/vg00-lv00 lv00_encrypted

/dev/mapper/lv00_encrypted is now active and ready for online encryption.
```

4. *장치를 마운트합니다.*

```
# mount /dev/mapper/lv00_encrypted /mnt/lv00_encrypted
```

5. */etc/crypttab 파일에 영구 매핑 항목을 추가합니다.*

   a. *luksUUID 를 찾습니다.*

```
# cryptsetup luksUUID /dev/mapper/vg00-lv00

a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325
```

   b. *선택한 텍스트 편집기에서 /etc/crypttab 을 열고 이 파일에 장치를 추가합니다.*

```
$ vi /etc/crypttab

lv00_encrypted UUID=a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325 none
```

a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325 를 장치의 luksUUID 로 바꿉니다.

c. *dracut 을 사용하여 initramfs 새로 고침 :*

```
$ dracut -f --regenerate-all
```

6. *영구 마운트 항목을 /etc/fstab 파일에 추가합니다.*

a. *활성 LUKS 블록 장치의 파일 시스템의 UUID를 찾습니다.*

```
$ blkid -p /dev/mapper/lv00_encrypted

/dev/mapper/lv00-encrypted: UUID="37bc2492-d8fa-4969-9d9b-bb64d3685aa9"
BLOCK_SIZE="4096" TYPE="xfs" USAGE="filesystem"
```

b. *선택한 텍스트 편집기에서 /etc/fstab 를 열고 이 파일에 장치를 추가합니다. 예를 들면 다음과 같습니다.*

```
$ vi /etc/fstab

UUID=37bc2492-d8fa-4969-9d9b-bb64d3685aa9 /home auto rw,user,auto 0
```

37bc2492-d8fa-4969-9d9b-bb64d3685aa9 를 파일 시스템의 UUID로 바꿉니다.

7. *온라인 암호화를 다시 시작하십시오.*

```
# cryptsetup reencrypt --resume-only /dev/mapper/vg00-lv00

Enter passphrase for /dev/mapper/vg00-lv00:
Auto-detected active dm device 'lv00_encrypted' for data device /dev/mapper/vg00-lv00.
Finished, time 00:31.130, 10272 MiB written, speed 330.0 MiB/s
```

검증

1. *기존 데이터가 암호화되었는지 확인합니다.*

```
# cryptsetup luksDump /dev/mapper/vg00-lv00
```

```
LUKS header information
Version: 2
Epoch: 4
Metadata area: 16384 [bytes]
Keyslots area: 16744448 [bytes]
UUID: a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325
Label: (no label)
Subsystem: (no subsystem)
Flags: (no flags)

Data segments:
  0: crypt
 offset: 33554432 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
[...]
```

2. 암호화된 빈 블록 장치의 상태를 확인합니다.

```
# cryptsetup status lv00_encrypted

/dev/mapper/lv00_encrypted is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/mapper/vg00-lv00
```

추가 리소스

- **cryptsetup(8), cryptsetup-reencrypt(8), lvextend(8), resize2fs(8), parted(8)** 매뉴얼 페이지

9.5. 분리된 헤더로 LUKS2를 사용하여 블록 장치에서 기존 데이터 암호화

LUKS 헤더를 저장하기 위한 여유 공간을 생성하지 않고 블록 장치의 기존 데이터를 암호화할 수 있습니다. 헤더는 분리된 위치에 저장되며 추가 보안 계층 역할을 합니다. 절차에서는 **LUKS2** 암호화 형식을 사용합니다.

사전 요구 사항

- 블록 장치에는 파일 시스템이 있습니다.
- 데이터를 백업했습니다.

주의

하드웨어, 커널 또는 사람의 오류로 인해 암호화 프로세스 중에 데이터가 손실될 수 있습니다. 데이터 암호화를 시작하기 전에 신뢰할 수 있는 백업이 있는지 확인합니다.

절차

1. *장치의 모든 파일 시스템을 마운트 해제합니다. 예를 들면 다음과 같습니다.*

```
# umount /dev/nvme0n1p1
```

2. *암호화를 초기화합니다.*

```
# cryptsetup reencrypt --encrypt --init-only --header /home/header /dev/nvme0n1p1
nvme_encrypted

WARNING!
========
Header file does not exist, do you want to create it?

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /home/header:
Verify passphrase:
/dev/mapper/nvme_encrypted is now active and ready for online encryption.
```

*/**home**/**header** 를 파일 경로로 교체하고 분리된 **LUKS** 헤더로 바꿉니다. 분리된 **LUKS** 헤더는 나중에 암호화된 장치를 잠금 해제하려면 액세스할 수 있어야 합니다.*

3. *장치를 마운트합니다.*

```
# mount /dev/mapper/nvme_encrypted /mnt/nvme_encrypted
```

4. *온라인 암호화를 다시 시작하십시오.*

```
# cryptsetup reencrypt --resume-only --header /home/header /dev/nvme0n1p1

Enter passphrase for /dev/nvme0n1p1:
```

```
Auto-detected active dm device 'nvme_encrypted' for data device /dev/nvme0n1p1.
Finished, time 00m51s,   10 GiB written, speed 198.2 MiB/s
```

검증

1. *분리된 헤더와 함께 **LUKS2**를 사용하는 블록 장치의 기존 데이터가 암호화되었는지 확인합니다.*

```
# cryptsetup luksDump /home/header

LUKS header information
Version:        2
Epoch:          88
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           c4f5d274-f4c0-41e3-ac36-22a917ab0386
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
 offset: 0 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
 sector: 512 [bytes]
[...]
```

2. *암호화된 빈 블록 장치의 상태를 확인합니다.*

```
# cryptsetup status nvme_encrypted

/dev/mapper/nvme_encrypted is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/nvme0n1p1
```

추가 리소스

- ***cryptsetup(8)** 및 **cryptsetup-reencrypt(8)** 매뉴얼 페이지*

9.6. LUKS2를 사용하여 빈 블록 장치 암호화

***LUKS2** 형식을 사용하여 암호화된 스토리지에 사용할 수 있는 빈 블록 장치를 암호화할 수 있습니다.*

사전 요구 사항

- *빈 블록 장치입니다. **lsblk** 와 같은 명령을 사용하여 해당 장치에 실제 데이터(예: 파일 시스템)가 없는지 확인할 수 있습니다.*

절차

1. *파티션을 암호화된 **LUKS** 파티션으로 설정합니다.*

```
# cryptsetup luksFormat /dev/nvme0n1p1

WARNING!
========
This will overwrite data on /dev/nvme0n1p1 irrevocably.
Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/nvme0n1p1:
Verify passphrase:
```

2. *암호화된 **LUKS** 파티션을 엽니다.*

```
# cryptsetup open /dev/nvme0n1p1 nvme0n1p1_encrypted

Enter passphrase for /dev/nvme0n1p1:
```

*이렇게 하면 파티션 잠금을 해제하고 장치 매퍼를 사용하여 새 장치에 매핑합니다. 암호화된 데이터를 덮어쓰지 않으려면 이 명령은 **/dev/mapper/device_mapped_name**경로를 사용하여 장치가 암호화된 장치 임을 커널에 경고하고 **LUKS**를 통해 처리합니다.*

3. *암호화된 데이터를 파티션에 쓰도록 파일 시스템을 만들고, 장치 매핑된 이름을 통해 액세스해야 합니다.*

```
# mkfs -t ext4 /dev/mapper/nvme0n1p1_encrypted
```

4. *장치를 마운트합니다.*

```
# mount /dev/mapper/nvme0n1p1_encrypted mount-point
```

검증

1. *빈 블록 장치가 암호화되었는지 확인합니다.*

-

```
# cryptsetup luksDump /dev/nvme0n1p1

LUKS header information
Version:        2
Epoch:          3
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           34ce4870-ffdf-467c-9a9e-345a53ed8a25
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
 offset: 16777216 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
 sector: 512 [bytes]
[...]
```

2. 암호화된 빈 블록 장치의 상태를 확인합니다.

```
# cryptsetup status nvme0n1p1_encrypted

/dev/mapper/nvme0n1p1_encrypted is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/nvme0n1p1
  sector size:  512
  offset:  32768 sectors
  size:    20938752 sectors
  mode:    read/write
```

추가 리소스

- **cryptsetup(8), cryptsetup-openECDHE, cryptsetup-lusFormat(8)** 매뉴얼 페이지

9.7. 스토리지 RHEL 시스템 역할을 사용하여 LUKS2 암호화된 볼륨 생성

storage 역할을 사용하여 **Ansible** 플레이북을 실행하여 **LUKS**로 암호화된 볼륨을 생성하고 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: Create and configure a volume encrypted with LUKS
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
         - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: <password>
```

encryption_key ,encryption_cipher, encryption_key _size , encryption_ luks 와 같은 다른 암호화 매개변수를 플레이북 파일에 추가할 수도 있습니다.

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
```

검증

1. *암호화 상태를 확인합니다.*

```
# cryptsetup status sdb

/dev/mapper/sdb is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/sdb
...
```

2. *생성된 LUKS 암호화된 볼륨을 확인합니다.*

```
# cryptsetup luksDump /dev/sdb

Version:        2
Epoch:          6
Metadata area:  16384 [bytes]
Keyslots area:  33521664 [bytes]
UUID:           a4c6be82-7347-4a91-a8ad-9479b72c9426
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          allow-discards

Data segments:
  0: crypt
        offset: 33554432 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 4096 [bytes]
...
```

추가 리소스

- */usr/share/ansible/roles/rhel-system-roles.storage/README.md* 파일
- */usr/share/doc/rhel-system-roles/storage/* 디렉터리
- *LUKS를 사용하여 블록 장치 암호화*

10장. 정책 기반 암호 해독을 사용하여 암호화된 볼륨의 자동 잠금 해제 구성

정책 기반 암호 해독(Policy-Based Decryption)은 물리적 및 가상 머신에서 암호화된 루트 및 보조 드라이브의 하드 드라이브의 잠금을 해제할 수 있는 기술 컬렉션입니다. PBD는 사용자 암호, 신뢰할 수 있는 플랫폼 모듈(TPM) 장치, 시스템에 연결된 PKCS #11 장치(예: 스마트 카드 또는 특수 네트워크 서버)와 같은 다양한 잠금 해제 방법을 사용합니다.

PBD를 사용하면 다른 잠금 해제 방법을 정책에 결합하여 동일한 볼륨을 다른 방식으로 잠금 해제할 수 있습니다. RHEL에서 PBD의 현재 구현은 Clevis 프레임워크와 pins라는 플러그인으로 구성되어 있습니다. 각 핀은 별도의 잠금 해제 기능을 제공합니다. 현재 다음 핀을 사용할 수 있습니다.

Tang

네트워크 서버를 사용하여 볼륨을 잠금 해제할 수 있습니다.

tpm2

TPM2 정책을 사용하여 볼륨 잠금을 허용합니다.

sss

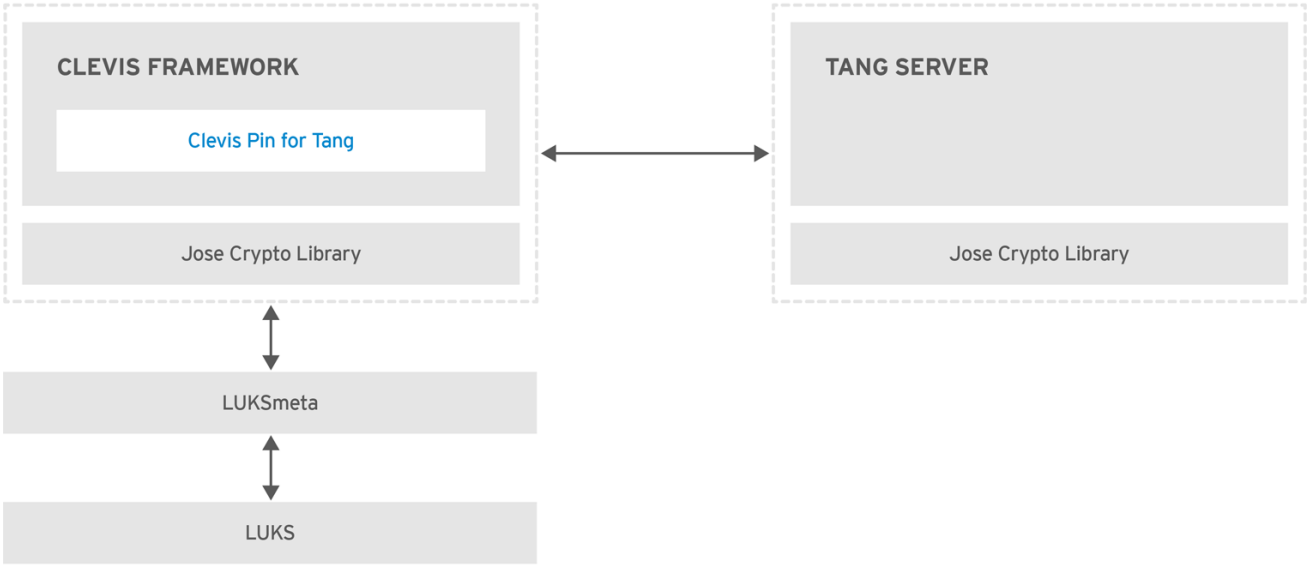
Shamir's Secret Sharing (SSS) 암호화 체계를 사용하여 고가용성 시스템을 배포할 수 있습니다.

10.1. 네트워크 바인딩 디스크 암호화

NBDE(Network Bound Disc Encryption)는 암호화된 볼륨을 특수 네트워크 서버에 바인딩할 수 있는 정책 기반 암호 해독(PBD)의 하위 범주입니다. Clevis의 현재 구현에는 Tang 서버 및 Tang 서버 자체에 대한 Clevis 핀이 포함되어 있습니다.

RHEL에서 NBDE는 다음과 같은 구성 요소 및 기술을 통해 구현됩니다.

그림 10.1. LUKS1 암호화 볼륨을 사용할 때의 NBDE 스키마입니다 luksmeta 패키지는 LUKS2 볼륨에 사용되지 않습니다.



Tang은 데이터를 네트워크 상에 바인딩하는 서버입니다. 시스템이 특정 보안 네트워크에 바인딩될 때 데이터를 포함하는 시스템을 사용할 수 있습니다. **Tang**은 상태 비저장이며 **TLS** 또는 인증이 필요하지 않습니다. 서버가 모든 암호화 키를 저장하고 사용된 모든 키에 대한 지식이 있는 에스크로 기반 솔루션과 달리 **Tang**은 클라이언트 키와 상호 작용하지 않으므로 클라이언트로부터 식별 정보를 얻을 수 없습니다.

Clevis는 자동 암호 해독을 위한 플러그형 프레임워크입니다. **KnativeServing**에서 **Clevis**는 **LUKS** 볼륨의 자동 잠금을 해제하는 기능을 제공합니다. **clevis** 패키지는 기능의 클라이언트 측을 제공합니다.

Clevis PIN은 **Clevis** 프레임워크의 플러그인입니다. 이러한 핀 중 하나는 **DASD** 서버 **- Tang**과의 상호 작용을 구현하는 플러그인입니다.

Clevis 및 **Tang**은 네트워크 바인딩 암호화를 제공하는 일반 클라이언트 및 서버 구성 요소입니다. **RHEL**에서는 **LUKS**와 함께 사용하여 루트 및 루트가 아닌 스토리지 볼륨을 암호화하고 암호 해독하여 **Network-Bound Disk Encryption**을 수행합니다.

클라이언트 및 서버 측 구성 요소 모두 **José** 라이브러리를 사용하여 암호화 및 암호 해독 작업을 수행합니다.

Clevis 프로비저닝을 시작할 때 **Tang** 서버의 **Clevis** 핀은 **Tang** 서버의 공개된 **symmetric** 키 목록을 가져옵니다. 또는 키가 **symmetric**이므로 **Tang**의 공개 키 목록을 대역에서 배포할 수 있으므로 클라이언트가 **Tang** 서버에 액세스하지 않고도 작동할 수 있습니다. 이 모드를 오프라인 프로비저닝 이라고 합니다.

Tang의 Clevis 핀은 공개 키 중 하나를 사용하여 고유하고 암호화 방식으로 암호화 키를 생성합니다. 이 키를 사용하여 데이터를 암호화하면 키가 삭제됩니다. Clevis 클라이언트는 이 프로비저닝 작업에서 생성한 상태를 편리한 위치에 저장해야 합니다. 데이터를 암호화하는 이 프로세스는 프로비저닝 단계입니다.

LUKS 버전 2(LUKS2)는 RHEL의 기본 disk-encryption 형식입니다. 따라서 Clevis의 프로비저닝 상태는 LUKS2 헤더에 토큰으로 저장됩니다. luksmeta 패키지를 통해 NBDE의 프로비저닝 상태를 활용하는 것은 LUKS1로 암호화된 볼륨에만 사용됩니다.

Tang의 Clevis 핀은 사양 없이 LUKS1 및 LUKS2를 모두 지원합니다. Clevis는 일반 텍스트 파일을 암호화할 수 있지만 block 장치를 암호화하기 위해 cryptsetup 툴을 사용해야 합니다. 자세한 내용은 LUKS를 사용하여 블록 장치 암호화를 참조하십시오.

클라이언트가 데이터에 액세스할 준비가 되면 프로비저닝 단계에서 생성된 메타데이터를 로드하고 암호화 키를 복구할 수 있습니다. 이 과정은 회복 단계입니다.

Clevis는 자동으로 잠금 해제될 수 있도록 핀을 사용하여 LUKS 볼륨을 바인딩합니다. 바인딩 프로세스가 성공적으로 완료되면 제공된 Dracut unlocker를 사용하여 디스크의 잠금을 해제할 수 있습니다.

참고

kdump 커널 크래시 덤프 메커니즘이 시스템 메모리의 콘텐츠를 LUKS 암호화 장치에 저장하도록 설정된 경우 두 번째 커널 부팅 중에 암호를 입력하라는 메시지가 표시됩니다.

추가 리소스

- NBDE(Network-Bound Disk Encryption) 기술 지식베이스 문서
- Tang(8), clevis(1), jose(1), clevis-luks-unlockers(7) 매뉴얼 페이지
- 여러 LUKS 장치(Clevis + Tang 잠금) 지식 베이스 문서로 네트워크-Bound 디스크 암호화를 설정하는 방법

10.2. 암호화 클라이언트 설치 - CLEVIS

암호화된 볼륨의 자동 잠금 해제를 구성하려면 먼저 클라이언트 시스템에 Clevis 플러그형 프레임워크를 배포해야 합니다.

절차

1. *암호화된 볼륨이 있는 시스템에 **Clevis** 및 해당 핀을 설치하려면 다음을 수행합니다.*

```
# yum install clevis
```

2. *데이터의 암호를 해독하려면 **clevis decrypt** 명령을 사용하고 **JSON Web Encryption(JWE)** 형식의 암호화 텍스트를 제공합니다. 예를 들면 다음과 같습니다.*

```
$ clevis decrypt < secret.jwe
```

추가 리소스

- ***Clevis(1)** 도움말 페이지*
- *인수 없이 **clevis** 명령을 입력한 후 기본 제공 **CLI** 도움말:*

```
$ clevis
Usage: clevis COMMAND [OPTIONS]

clevis decrypt      Decrypts using the policy defined at encryption time
clevis encrypt sss  Encrypts using a Shamir's Secret Sharing policy
clevis encrypt tang Encrypts using a Tang binding server policy
clevis encrypt tpm2 Encrypts using a TPM2.0 chip binding policy
clevis luks bind    Binds a LUKS device using the specified policy
clevis luks edit    Edit a binding from a clevis-bound slot in a LUKS device
clevis luks list    Lists pins bound to a LUKSv1 or LUKSv2 device
clevis luks pass    Returns the LUKS passphrase used for binding a particular slot.
clevis luks regen   Regenerate clevis binding
clevis luks report  Report tang keys' rotations
clevis luks unbind  Unbinds a pin bound to a LUKS volume
clevis luks unlock  Unlocks a LUKS volume
```

10.3. 강제 모드에서 SELINUX를 사용하여 TANG 서버 배포

***Tang** 서버를 사용하여 **Clevis** 지원 클라이언트에서 **LUKS** 암호화 볼륨을 자동으로 잠금 해제할 수 있습니다. 최소 시나리오에서는 **tang** 패키지를 설치하고 **systemctl enable tangd.socket --now** 명령을 입력하여 포트 **80**에 **Tang** 서버를 배포합니다. 다음 예제 절차에서는 사용자 지정 포트에서 실행되는 **Tang** 서버를 **SELinux** 강제 모드에서 제한된 서비스로 배포하는 방법을 보여줍니다.*

사전 요구 사항

- ***policycoreutils-python-utils** 패키지 및 해당 종속 항목이 설치됩니다.*

- ***firewalld** 서비스가 실행 중입니다.*

절차

1. ***tang** 패키지 및 해당 종속 항목을 설치하려면 **root**로 다음 명령을 입력합니다.*

```
# yum install tang
```

2. ***(예: 7500/tcp) .occupied** 포트를 선택하고 **tangd** 서비스가 해당 포트에 바인딩되도록 허용합니다.*

```
# semanage port -a -t tangd_port_t -p tcp 7500
```

*포트는 한 번에 하나의 서비스에서만 사용할 수 있으므로 이미 사용되고 있는 포트를 사용하려는 경우 **ValueError**를 의미합니다. 포트가 이미 정의된 오류 메시지입니다.*

3. *방화벽에서 포트를 엽니다.*

```
# firewall-cmd --add-port=7500/tcp
# firewall-cmd --runtime-to-permanent
```

4. ***tangd** 서비스를 활성화합니다.*

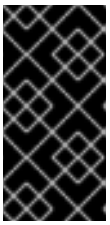
```
# systemctl enable tangd.socket
```

5. *덮어쓰기 파일을 생성합니다.*

```
# systemctl edit tangd.socket
```

6. *다음 편집기 화면에서 **/etc/systemd/system/tangd.socket.d/** 디렉터리에 있는 빈 **override.conf** 파일을 열고 다음 행을 추가하여 **Tang** 서버의 기본 포트를 **80**에서 이전에 선택한 숫자로 변경합니다.*

```
[Socket]
ListenStream=
ListenStream=7500
```



중요

이 아래에 있는 줄과 # 줄 사이에 이전 코드 조각을 삽입합니다. 그렇지 않으면 시스템은 변경 사항을 삭제합니다.

7. **Ctrl+O** 누른 후 를 입력하여 변경 사항을 저장합니다. **Ctrl+X** 를 눌러 편집기를 종료합니다.

8. 변경된 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

9. 구성이 작동하는지 확인합니다.

```
# systemctl show tangd.socket -p Listen
Listen=[::]:7500 (Stream)
```

10. **tangd** 서비스를 시작합니다.

```
# systemctl restart tangd.socket
```

tangd는 **systemd** 소켓 활성화 메커니즘을 사용하므로 첫 번째 연결이 들어오는 즉시 서버가 시작됩니다. 새로 생성된 암호화 키 세트는 처음 시작할 때 자동으로 생성됩니다. 수동 키 생성과 같은 암호화 작업을 수행하려면 **jose** 유틸리티를 사용합니다.

추가 리소스

- **Tang(8), semanage(8), firewall-cmd(1), jose(1), systemd.unit(5)** 및 **systemd.socket(5)** 도움말 페이지.

10.4. 클라이언트에서 TANG 서버 키 교체 및 바인딩 업데이트

보안상의 이유로 **Tang** 서버 키를 교체하고 클라이언트에서 기존 바인딩을 정기적으로 업데이트합니다. 교체해야하는 정확한 간격은 애플리케이션, 키 크기 및 기관 정책에 따라 다릅니다.

또는 **nbde_server RHEL** 시스템 역할을 사용하여 **Tang** 키를 교체할 수 있습니다. 자세한 내용은 **nbde_server** 시스템 역할을 사용하여 여러 **Tang** 서버 설정을 참조하십시오.

사전 요구 사항

- **Tang** 서버가 실행 중입니다.
- **clevis** 및 **clevis-luks** 패키지가 클라이언트에 설치됩니다.
- **clevis luks list,clevis luks report, clevis luks regen** 은 **RHEL 8.2**에서 도입되었습니다.

절차

1. **/var/db/tang** 키 데이터베이스 디렉터리의 모든 키 이름을 앞에 **.** 로 바꿔서 광고에서 숨길 수 있습니다. 다음 예제의 파일 이름은 **Tang** 서버의 키 데이터베이스 디렉터리에 있는 고유한 파일 이름과 다릅니다.

```
# cd /var/db/tang
# ls -l
-rw-r--r--. 1 root root 349 Feb  7 14:55 UV6dqXSwe1bRKG3KbJmdiR020hY.jwk
-rw-r--r--. 1 root root 354 Feb  7 14:55 y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk
# mv UV6dqXSwe1bRKG3KbJmdiR020hY.jwk .UV6dqXSwe1bRKG3KbJmdiR020hY.jwk
# mv y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk .y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk
```

2. 이름이 변경되었고, 따라서 **Tang** 서버 광고에서 모든 키를 숨겼는지 확인합니다.

```
# ls -l
total 0
```

3. **Tang** 서버의 **/var/db/tang**에서 **/usr/libexec/tangd-keygen** 명령을 사용하여 새 키를 생성합니다.

```
# /usr/libexec/tangd-keygen /var/db/tang
# ls /var/db/tang
3ZWS6-cDrCG61UPJS2BMmPU4I54.jwk zyLuX6hijUy_PSeUEFDi7hi38.jwk
```

4. **Tang** 서버가 새 키 쌍에서 서명 키를 알리는지 확인합니다. 예를 들면 다음과 같습니다.

```
# tang-show-keys 7500
3ZWS6-cDrCG61UPJS2BMmPU4I54
```

5. **EgressIP** 클라이언트에서 **clevis luks report** 명령을 사용하여 **Tang** 서버에서 광고하는 키가 동일하게 남아 있는지 확인합니다. 다음과 같이 **clevis luks list** 명령을 사용하여 관련 바인딩으로 슬롯을 식별할 수 있습니다.

```
# clevis luks list -d /dev/sda2
1: tang '{"url":"http://tang.srv"}'
# clevis luks report -d /dev/sda2 -s 1
...
Report detected that some keys were rotated.
Do you want to regenerate luks metadata with "clevis luks regen -d /dev/sda2 -s 1"? [ynYN]
```

6. 새 키에 대해 **LUKS** 메타데이터를 다시 생성하려면 이전 명령의 프롬프트에 **y** 를 누르거나 **clevis luks regen** 명령을 사용합니다.

```
# clevis luks regen -d /dev/sda2 -s 1
```

7. 모든 이전 클라이언트가 새 키를 사용하도록 확신하면 **Tang** 서버에서 이전 키를 제거할 수 있습니다. 예를 들면 다음과 같습니다.

```
# cd /var/db/tang
# rm .*.jwk
```

주의

클라이언트가 계속 사용하는 동안 이전 키를 제거하면 데이터 손실이 발생할 수 있습니다. 실수로 이러한 키를 제거하는 경우 클라이언트에서 **clevis luks regen** 명령을 사용하고 수동으로 **LUKS** 암호를 제공하십시오.

추가 리소스

- **tang-show-keys(1), clevis-luks-list(1), clevis-luks-report(1), and clevis-luks-regen(1)** 메뉴얼 페이지

10.5. 웹 콘솔에서 TANG 키를 사용하여 자동 잠금 해제 구성

Tang 서버에서 제공하는 키를 사용하여 LUKS 암호화 스토리지 장치의 자동 잠금 해제를 구성할 수 있습니다.

사전 요구 사항

- *RHEL 8 웹 콘솔이 설치되어 있습니다. 자세한 내용은 웹 콘솔 설치를 참조하십시오.*
- *cockpit-storaged 및 clevis-luks 패키지가 시스템에 설치됩니다.*
- *cockpit.socket 서비스는 포트 9090에서 실행됩니다.*
- *Tang 서버를 사용할 수 있습니다. 자세한 내용은 강제 모드에서 SELinux를 사용하여 Tang 서버 배포를 참조하십시오.*

절차

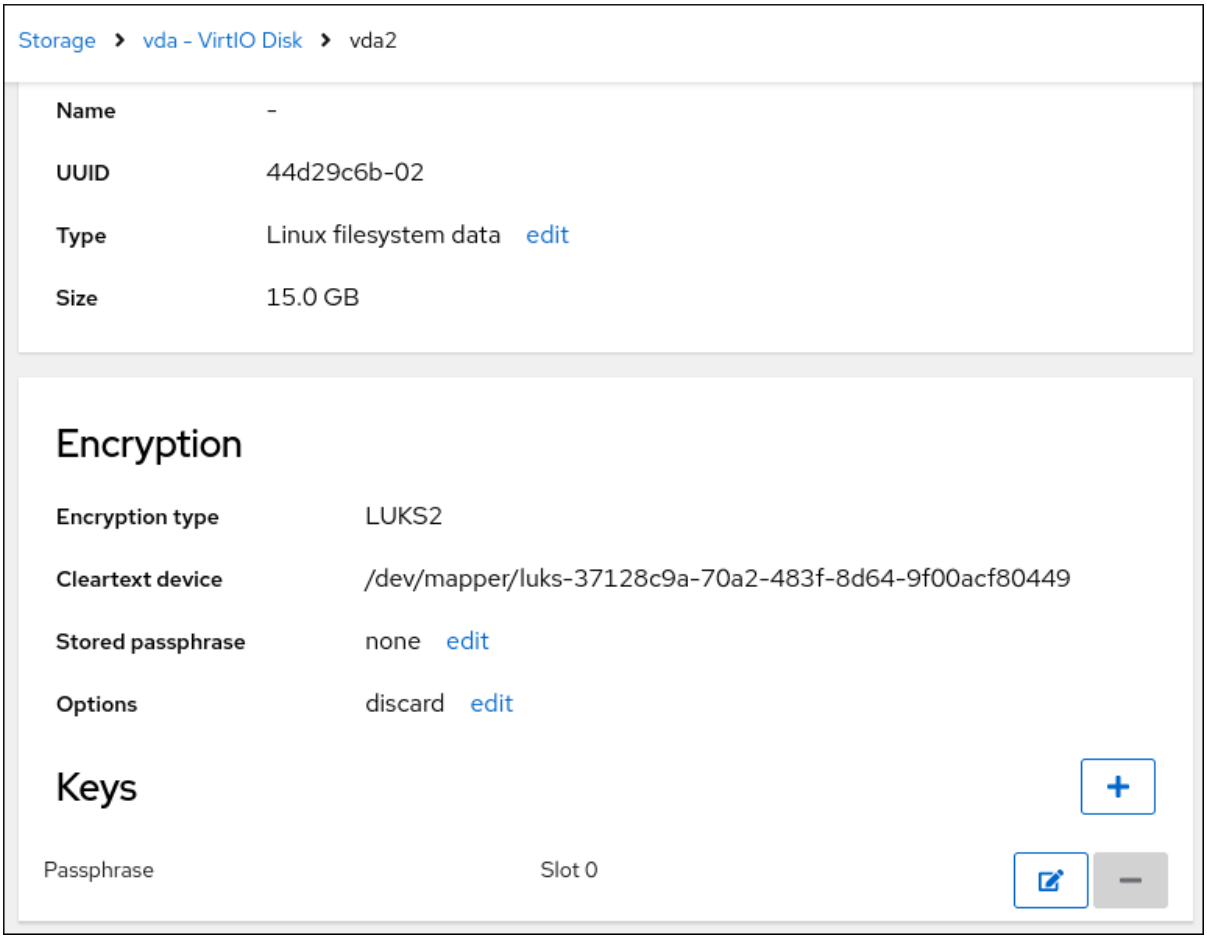
1. *웹 브라우저에 다음 주소를 입력하여 RHEL 웹 콘솔을 엽니다.*

   ```
   https://<localhost>:9090
   ```

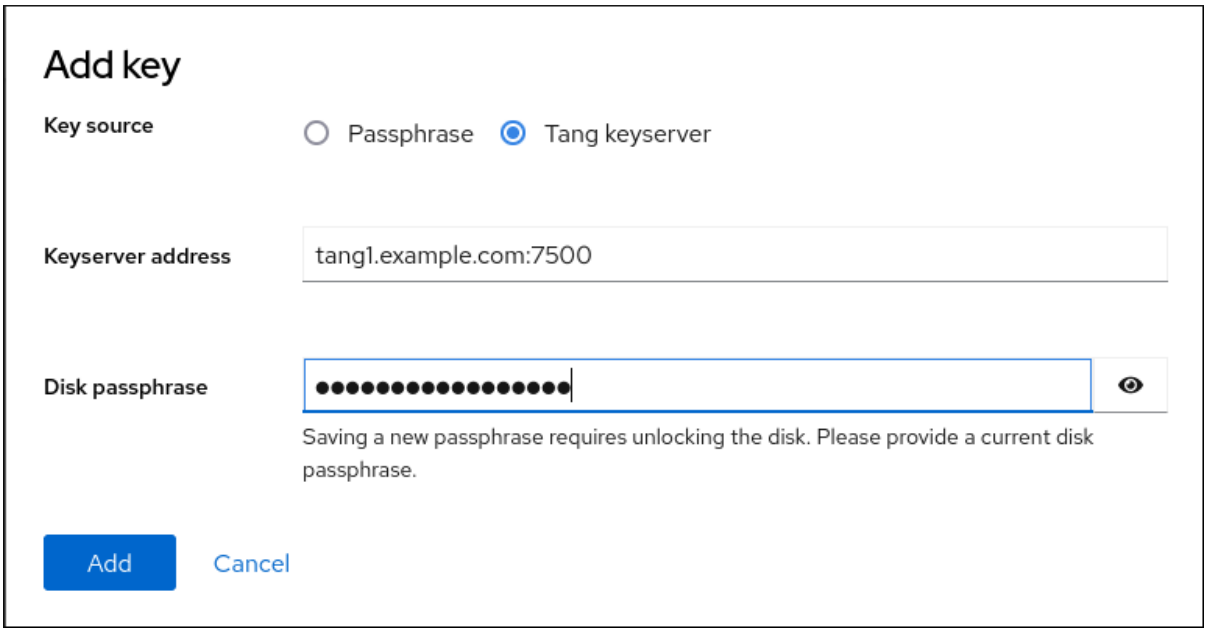
   원격 시스템에 연결할 때 < localhost > 부분을 원격 서버의 호스트 이름 또는 IP 주소로 바꿉니다.

2. *인증 정보를 제공하고 스토리지를 클릭합니다. 스토리지 테이블에서 잠금 해제를 위해 추가할 암호화된 볼륨이 포함된 디스크를 클릭합니다.*

3. *선택한 디스크에 대한 세부 정보가 있는 다음 페이지에서 Keys 섹션에서 + 를 클릭하여 Tang 키를 추가합니다.*



4. *Tang 키 서버 를 키 소스로 선택하고 Tang 서버의 주소 및 LUKS 암호화 장치를 잠금 해제하는 암호를 제공합니다. 추가를 클릭하여 확인합니다.*



다음 대화 상자 창에서 키 해시가 일치하는지 확인하는 명령을 제공합니다.

5. *Tang 서버의 터미널에서 tang-show-keys 명령을 사용하여 비교할 키 해시를 표시합니다. 이 예에서 Tang 서버는 포트 7500에서 실행되고 있습니다.*

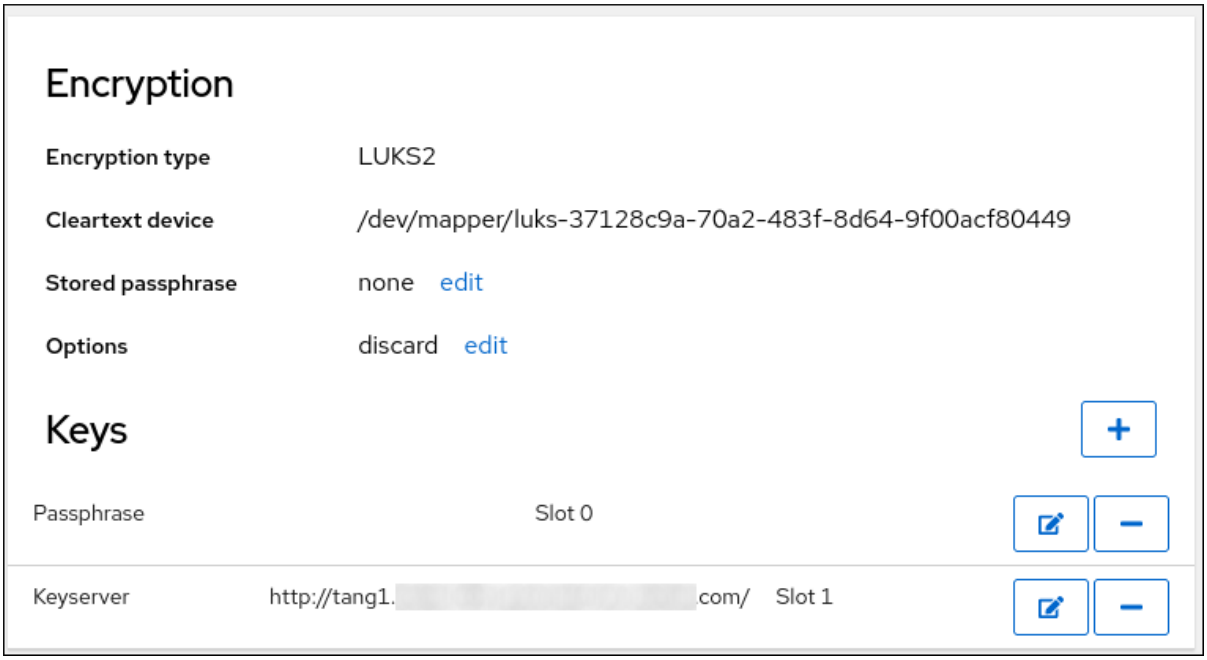
```
# tang-show-keys 7500
x100_1k6GPiDOaMlL3WbpCjHOy9ul1bSfdhl3M08wO0
```

6. *웹 콘솔의 키 해시와 이전에 나열된 명령의 출력에서와 이전에 나열된 명령의 출력에서 신뢰 키를 클릭하면 신뢰 키를 클릭합니다.*

7. ***RHEL 8.8** 이상에서는 암호화된 루트 파일 시스템 및 **Tang** 서버를 선택한 후 커널 명령줄에 **rd.neednet=1** 매개변수 추가를 건너뛰고 **clevis-dracut** 패키지를 설치하고 초기 **RAM** 디스크 **(Initrd)**를 다시 생성할 수 있습니다. 루트가 아닌 파일 시스템의 경우 웹 콘솔에서 **remote-cryptsetup.target** 및 **clevis-luks-akspass.path systemd** 장치를 활성화하고 **clevis-systemd** 패키지를 설치하고 **_netdev** 매개 변수를 **fstab** 및 **crypttab** 구성 파일에 추가합니다.*

검증

1. *새로 추가된 **Tang** 키가 **Keyserver** 유형의 **Keys** 섹션에 나열되어 있는지 확인합니다.*



2. *초기 부팅에 바인딩을 사용할 수 있는지 확인합니다. 예를 들면 다음과 같습니다.*

```
# lsinitrd | grep clevis-luks
lrwxrwxrwx   1 root     root          48 Jan  4 02:56
etc/systemd/system/cryptsetup.target.wants/clevis-luks-askpass.path ->
/usr/lib/systemd/system/clevis-luks-askpass.path
...
```

추가 리소스

- ***RHEL 웹 콘솔을 사용하여 시작하기***

10.6. 기본 CLEVIS 및 TPM2 암호화-클라이언트 작업

***Clevis** 프레임워크는 일반 텍스트 파일을 암호화하고 **JSON** 웹 암호화(JWE) 형식과 **LUKS** 암호화 블록 장치의 암호화 텍스트를 모두 해독할 수 있습니다. **Clevis** 클라이언트는 암호화 작업을 위해 **Tang** 네트워크 서버 또는 신뢰할 수 있는 **Platform Module 2.0(TPM 2.0)** 칩을 사용할 수 있습니다.*

*다음 명령은 일반 텍스트 파일을 포함하는 예제의 **Clevis**에서 제공하는 기본 기능을 보여줍니다. **Clevis** 또는 **Clevis+TPM** 배포 문제를 해결하는 데도 사용할 수 있습니다.*

***Tang** 서버에 바인딩된 암호화 클라이언트*

- ***Clevis** 암호화 클라이언트가 **Tang** 서버에 바인딩되는지 확인하려면 **clevis encrypt tang** 하위 명령을 사용합니다.*

-

```
$ clevis encrypt tang '{"url":"http://tang.srv:port"}' < input-plain.txt > secret.jwe
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
```

위 예제의 **http://tang.srv:port URL**을 **tang** 이 설치된 서버의 **URL**과 일치하도록 변경합니다. **secret.jwe** 출력 파일에는 **JWE** 형식의 암호화된 암호 텍스트가 포함되어 있습니다. 이 암호화 방식 텍스트는 **input-plain.txt** 입력 파일에서 읽습니다.

또는 구성에 **SSH** 액세스 권한이 없는 **Tang** 서버와의 비대화형 통신이 필요한 경우 광고를 다운로드하여 파일에 저장할 수 있습니다.

```
$ curl -sfg http://tang.srv:port/adv -o adv.jws
```

파일 또는 메시지의 암호화와 같은 다음 작업에 대해 **adv.jws** 파일의 광고를 사용합니다.

```
$ echo 'hello' | clevis encrypt tang '{"url":"http://tang.srv:port","adv":"adv.jws"}'
```

- 데이터의 암호를 해독하려면 **clevis decrypt** 명령을 사용하여 **JWE(암호화 텍스트)**를 제공합니다.

```
$ clevis decrypt < secret.jwe > output-plain.txt
```

TPM 2.0을 사용하는 암호화 클라이언트

- **TPM 2.0** 칩을 사용하여 암호화하려면 **JSON** 구성 개체 형식의 유일한 인수와 함께 **clevis encrypt tpm2** 하위 명령을 사용하십시오.

```
$ clevis encrypt tpm2 '{}' < input-plain.txt > secret.jwe
```

다른 계층 구조, 해시 및 키 알고리즘을 선택하려면 구성 속성을 지정합니다.

```
$ clevis encrypt tpm2 '{"hash":"sha256","key":"rsa"}' < input-plain.txt > secret.jwe
```

- 데이터의 암호를 해독하려면 **JSON** 웹 암호화**(JWE)** 형식으로 암호화 텍스트를 제공합니다.

```
$ clevis decrypt < secret.jwe > output-plain.txt
```

또한 핀은 **PCR(Platform Configuration Registers)** 상태에 대한 데이터 잠금을 지원합니다. 이렇게 하면 **PCR** 해시 값이 봉인할 때 사용된 정책과 일치하는 경우에만 데이터를 봉인 해제할 수 있습니다.

예를 들어 **SHA-256** 뱅크의 인덱스 **0**과 **7**로 **PCR**에 데이터를 봉인하려면 다음과 같이 하십시오.

```
$ clevis encrypt tpm2 '{"pcr_bank":"sha256","pcr_ids":"0,7"}' < input-plain.txt > secret.jwe
```

주의

PCR의 해시는 다시 작성할 수 있으며 더 이상 암호화된 볼륨을 잠금 해제할 수 없습니다. 따라서 **PCR**의 값이 변경된 경우에도 암호화된 볼륨을 수동으로 잠금 해제할 수 있는 강력한 암호를 추가합니다.

shim-x64 패키지를 업그레이드한 후 시스템이 암호화된 볼륨 잠금을 자동으로 해제할 수 없는 경우, **KCS**를 다시 시작한 후 **Clevis TPM2**의 단계에 따라 **LUKS** 장치의 암호를 해독 하지 않습니다.

추가 리소스

- **clevis-encrypt-tang(1), clevis-luks-unlockers(7), clevis(1), and clevis-encrypt-tpm2(1)** 도움말 페이지
- **clevis, clevis decrypt, clevis encrypt tang** 명령은 인수 없이 내장 **CLI** 도움말을 보여줍니다. 예를 들면 다음과 같습니다.

```
$ clevis encrypt tang
Usage: clevis encrypt tang CONFIG < PLAINTEXT > JWE
...
```

10.7. LUKS 암호화 볼륨 수동 등록 구성

Clevis 프레임워크를 사용하면 선택한 **Tang** 서버를 사용할 수 있을 때 **LUKS** 암호화 볼륨의 자동 잠금 해제를 위해 클라이언트를 구성할 수 있습니다. 그러면 **NBDE(Network-Bound Disk Encryption)** 배포가 생성됩니다.

사전 요구 사항

- ***Tang** 서버가 실행 중이고 사용 가능합니다.*

절차

1. *기존 **LUKS** 암호화된 볼륨의 잠금을 자동으로 해제하려면 **clevis-luks** 하위 패키지를 설치합니다.*

```
# yum install clevis-luks
```

2. ***PBD**의 **LUKS** 암호화 볼륨을 식별합니다. 다음 예에서 블록 장치는 **/dev/sda2** 라고 합니다.*

```
# lsblk
NAME                                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                             8:0    0    12G  0 disk
├─sda1                                          8:1    0     1G  0 part  /boot
└─sda2                                          8:2    0    11G  0 part
  └─luks-40e20552-2ade-4954-9d56-565aa7994fb6 253:0    0    11G  0 crypt
    ├─rhel-root                               253:0    0   9.8G  0 lvm   /
    └─rhel-swap                               253:1    0   1.2G  0 lvm   [SWAP]
```

3. ***clevis luks bind** 명령을 사용하여 볼륨을 **Tang** 서버에 바인딩합니다.*

```
# clevis luks bind -d /dev/sda2 tang '{"url":"http://tang.srv"}'
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
You are about to initialize a LUKS device for metadata storage.
Attempting to initialize it may result in data loss if data was
already written into the LUKS header gap in a different format.
A backup is advised before initialization is performed.

Do you wish to initialize /dev/sda2? [yn] y
Enter existing LUKS password:
```

이 명령은 다음 네 가지 단계를 수행합니다.

a. ***LUKS** 마스터 키와 동일한 엔트로피를 사용하여 새 키를 만듭니다.*

b.

Clevis를 사용하여 새 키를 암호화합니다.

c.

LUKS2 헤더에 Clevis JWE 오브젝트를 저장하거나 기본이 아닌 LUKS1 헤더가 사용되는 경우 LUKSMeta를 사용합니다.

d.

LUKS에 사용할 새 키를 활성화합니다.

참고

바인딩 절차에서는 사용 가능한 LUKS 암호 슬롯이 하나 이상 있다고 가정합니다. clevis luks bind 명령은 슬롯 중 하나를 사용합니다.

이제 Clevis 정책과 함께 기존 암호를 사용하여 볼륨을 잠금 해제할 수 있습니다.

4.

초기 부팅 시스템이 디스크 바인딩을 처리할 수 있도록 하려면 이미 설치된 시스템에서 dracut 툴을 사용합니다.

```
# yum install clevis-dracut
```

RHEL에서 Clevis는 호스트별 구성 옵션 없이 일반 initrd (초기 RAM 디스크)를 생성하고 커널 명령줄에 rd.neednet=1 과 같은 매개변수를 자동으로 추가하지 않습니다. 구성이 초기 부팅 중에 네트워크가 필요한 Tang 핀을 사용하는 경우 --hostonly-cmdline 인수를 사용하고 dracut 은 Tang 바인딩을 감지할 때 rd.neednet=1 을 추가합니다.

```
# dracut -fv --regenerate-all --hostonly-cmdline
```

또는 /etc/dracut.conf.d/ 에 .conf 파일을 생성하고 hostonly_cmdline=yes 옵션을 파일에 추가합니다. 예를 들면 다음과 같습니다.

```
# echo "hostonly_cmdline=yes" > /etc/dracut.conf.d/clevis.conf
```

참고

Clevis가 설치된 시스템에서 **grubby** 툴을 사용하여 초기 부팅 시 **Tang** 핀의 네트워킹을 사용할 수 있는지 확인할 수도 있습니다.

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

그런 다음 **--hostonly-cmdline** 없이 **dracut** 을 사용할 수 있습니다.

```
# dracut -fv --regenerate-all
```

검증

1. **Clevis JWE** 오브젝트가 **LUKS** 헤더에 성공적으로 배치되었는지 확인하려면 **clevis luks list** 명령을 사용합니다.

```
# clevis luks list -d /dev/sda2
1: tang '{"url":"http://tang.srv:port"}'
```

중요

고정 **IP** 구성(**DHCP** 제외)이 있는 클라이언트에 대해 **DASD**를 사용하려면 네트워크 구성을 수동으로 **dracut** 툴에 전달합니다. 예를 들면 다음과 같습니다.

```
# dracut -fv --regenerate-all --kernel-cmdline
"ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none nameserver=192.0.2.100"
```

또는 정적 네트워크 정보를 사용하여 **/etc/dracut.conf.d/** 디렉터리에 **.conf** 파일을 만듭니다. 예를 들어 다음과 같습니다.

```
# cat /etc/dracut.conf.d/static_ip.conf
kernel_cmdline="ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none
nameserver=192.0.2.100"
```

초기 **RAM** 디스크 이미지를 다시 생성합니다.

```
# dracut -fv --regenerate-all
```

추가 리소스

- *Clevis-luks-bind(1) 및 dracut.cmdline(7) 도움말 페이지*
- *초기 램디스크 (initrd)에서 Linux 네트워크 설정 예상*

10.8. TPM 2.0 정책을 사용하여 LUKS 암호화 볼륨 수동 등록 구성

신뢰할 수 있는 플랫폼 모듈 2.0(TPM 2.0) 정책을 사용하여 LUKS 암호화 볼륨의 잠금을 구성할 수 있습니다.

사전 요구 사항

- *액세스 가능한 TPM 2.0 호환 장치.*
- *64비트 Intel 또는 64비트 AMD 아키텍처가 있는 시스템.*

절차

1. *기존 LUKS 암호화된 볼륨의 잠금을 자동으로 해제하려면 clevis-luks 하위 패키지를 설치합니다.*

```
# yum install clevis-luks
```

2. *PBD의 LUKS 암호화 볼륨을 식별합니다. 다음 예에서 블록 장치는 /dev/sda2 라고 합니다.*

```
# lsblk
NAME                                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                             8:0    0    12G  0 disk
├─sda1                                          8:1    0     1G  0 part  /boot
└─sda2                                          8:2    0    11G  0 part
  └─luks-40e20552-2ade-4954-9d56-565aa7994fb6 253:0    0    11G  0 crypt
    ├─rhel-root                               253:0    0   9.8G  0 lvm   /
    └─rhel-swap                               253:1    0   1.2G  0 lvm   [SWAP]
```

3. *clevis luks bind 명령을 사용하여 TPM 2.0 장치에 볼륨을 바인딩합니다. 예를 들면 다음과 같습니다.*

```
# clevis luks bind -d /dev/sda2 tpm2 '{"hash":"sha256","key":"rsa"}'
...
```

```
Do you wish to initialize /dev/sda2? [yn] y
Enter existing LUKS password:
```

이 명령은 다음 네 가지 단계를 수행합니다.

a. **LUKS** 마스터 키와 동일한 엔트로피를 사용하여 새 키를 만듭니다.

b. **Clevis**를 사용하여 새 키를 암호화합니다.

c. **LUKS2** 헤더에 **Clevis JWE** 오브젝트를 저장하거나 기본이 아닌 **LUKS1** 헤더가 사용되는 경우 **LUKSMeta**를 사용합니다.

d. **LUKS**에 사용할 새 키를 활성화합니다.

참고

바인딩 절차에서는 사용 가능한 **LUKS** 암호 슬롯이 하나 이상 있다고 가정합니다. **clevis luks bind** 명령은 슬롯 중 하나를 사용합니다.

또는 데이터를 특정 플랫폼 구성 등록 **(PCR)** 상태로 전환하려는 경우 **pcr_bank** 및 **pcr_ids** 값을 **clevis luks bind** 명령에 추가합니다.

```
# clevis luks bind -d /dev/sda2 tpm2
'{"hash":"sha256","key":"rsa","pcr_bank":"sha256","pcr_ids":"0,1"}'
```

중요

PCR 해시 값이 밀봉 및 해시를 다시 작성할 때 사용되는 정책과 일치하는 경우에만 데이터가 음소거될 수 있으므로 **PCR**의 값이 변경될 때 암호화된 볼륨을 수동으로 잠금 해제할 수 있는 강력한 암호를 추가합니다.

shim-x64 패키지를 업그레이드한 후 시스템이 암호화된 볼륨 잠금을 자동으로 해제할 수 없는 경우, **KCS**를 다시 시작한 후 **Clevis TPM2**의 단계에 따라 **LUKS** 장치의 암호를 해독 하지 않습니다.

4. 이제 **Clevis** 정책과 함께 기존 암호를 사용하여 볼륨을 잠금 해제할 수 있습니다.

5. 초기 부팅 시스템이 디스크 바인딩을 처리할 수 있도록 하려면 이미 설치된 시스템에서 **dracut** 툴을 사용합니다.

```
# yum install clevis-dracut
# dracut -fv --regenerate-all
```

검증

1. **Clevis JWE** 오브젝트가 **LUKS** 헤더에 성공적으로 배치되었는지 확인하려면 **clevis luks list** 명령을 사용합니다.

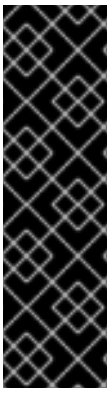
```
# clevis luks list -d /dev/sda2
1: tpm2 '{"hash":"sha256","key":"rsa"}'
```

추가 리소스

- **clevis-luks-bind(1), clevis-encrypt-tpm2(1),** 및 **dracut.cmdline(7)** 도움말 페이지

10.9. LUKS 암호화된 볼륨에서 CLEVIS 핀 제거 수동으로 제거

clevis luks bind 명령으로 생성된 메타데이터를 수동으로 제거하고 **Clevis**에서 추가한 암호가 포함된 키 슬롯을 삭제하려면 다음 절차를 사용하십시오.



중요

LUKS 암호화 볼륨에서 **Clevis** 핀을 제거하는 권장 방법은 **clevis luks unbind** 명령을 사용하는 것입니다. **clevis luks unbind**를 사용하는 제거 절차는 하나의 단계로 구성되며 **LUKS1** 및 **LUKS2** 볼륨 모두에 대해 작동합니다. 다음 예제 명령은 바인딩 단계에서 생성한 메타데이터를 제거하고 **/dev/sda2** 장치의 키 슬롯 **1** 을 지웁니다.

```
# clevis luks unbind -d /dev/sda2 -s 1
```

사전 요구 사항

- **Clevis** 바인딩이 있는 **LUKS** 암호화 볼륨.

절차

1. 볼륨(예: /dev/sda2)이 암호화된 LUKS 버전을 확인하고 Clevis에 바인딩된 슬롯과 토큰을 식별합니다.

```
# cryptsetup luksDump /dev/sda2
LUKS header information
Version:        2
...
Keyslots:
  0: luks2
...
1: luks2
      Key:        512 bits
      Priority:   normal
      Cipher:     aes-xts-plain64
...
      Tokens:
        0: clevis
              Keyslot:  1
...
```

이전 예에서 Clevis 토큰은 0 으로 식별되고 연결된 키 슬롯은 1 입니다.

2. LUKS2 암호화의 경우 토큰을 제거합니다.

```
# cryptsetup token remove --token-id 0 /dev/sda2
```

3. 장치가 LUKS1에 의해 암호화된 경우 Version으로 표시됩니다. 1 string in the output of the cryptsetup luksDump 명령 출력의 문자열 luksmeta wipe 명령을 사용하여 다음 추가 단계를 수행합니다.

```
# luksmeta wipe -d /dev/sda2 -s 1
```

4. Clevis 암호가 포함된 키 슬롯을 지웁니다.

```
# cryptsetup luksKillSlot /dev/sda2 1
```

추가 리소스

- Clevis-luks-unbind(1), cryptsetup(8) 및 luksmeta(8) 도움말 페이지

10.10. KICKSTART를 사용하여 LUKS 암호화 볼륨 자동 등록 구성

*이 절차의 단계에 따라 **LUKS** 암호화 볼륨 등록에 **Clevis**를 사용하는 자동화된 설치 프로세스를 구성합니다.*

절차

1. *Kickstart에 임시 암호로 **/boot** 이외의 모든 마운트 지점에 대해 **LUKS** 암호화가 활성화되도록 디스크를 파티션하도록 지시합니다. 암호는 이 등록 프로세스 단계에서 임시적입니다.*

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --grow --encrypted --passphrase=temppass
```

*예를 들어 **OSPP** 호환 시스템에는 더 복잡한 구성이 필요합니다.*

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --size=2048 --encrypted --passphrase=temppass
part /var --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /tmp --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /home --fstype="xfs" --ondisk=vda --size=2048 --grow --encrypted --
passphrase=temppass
part /var/log --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /var/log/audit --fstype="xfs" --ondisk=vda --size=1024 --encrypted --
passphrase=temppass
```

2. *%packages 섹션에 나열하여 관련 **Clevis** 패키지를 설치합니다.*

```
%packages
clevis-dracut
clevis-luks
clevis-systemd
%end
```

3. *필요한 경우 암호화된 볼륨을 수동으로 잠금 해제하려면 임시 암호를 제거하기 전에 강력한 암호를 추가합니다. 자세한 내용은 기존 **LUKS** 장치에 암호, 키 또는 키 파일을 추가하는 방법 문서를 참조하십시오.*

4. *%post 섹션에서 바인딩을 수행하기 위해 **clevis luks bind**를 호출합니다. 임시 암호를 삭제합니다.*

```
%post
clevis luks bind -y -k - -d /dev/vda2 \
tang '{"url":"http://tang.srv"}' <<< "temppass"
```

```
cryptsetup luksRemoveKey /dev/vda2 <<< "temppass"
dracut -fv --regenerate-all
%end
```

구성이 초기 부팅 중 네트워크가 필요한 **Tang** 핀을 사용하거나 고정 **IP** 구성과 함께 **EgressIP** 클라이언트를 사용하는 경우 **LUKS** 암호화 볼륨 수동 등록 구성에 설명된 대로 **dracut** 명령을 수정해야 합니다.

RHEL 8.3에서 **clevis luks bind** 명령의 **-y** 옵션을 사용할 수 있습니다. **RHEL 8.2** 이상에서는 **-y** 를 **clevis luks bind** 명령에서 **-f** 로 바꾸고 **Tang** 서버에서 광고를 다운로드합니다.

```
%post
curl -sfg http://tang.srv/adv -o adv.jws
clevis luks bind -f -k - -d /dev/vda2 \
tang '{"url":"http://tang.srv","adv":"adv.jws"}' <<< "temppass"
cryptsetup luksRemoveKey /dev/vda2 <<< "temppass"
dracut -fv --regenerate-all
%end
```

주의

cryptsetup luksRemoveKey 명령은 적용한 **LUKS2** 장치의 추가 관리를 방지합니다. **LUKS1** 장치에 대해서만 **dmsetup** 명령을 사용하여 제거된 마스터 키를 복구할 수 있습니다.

Tang 서버 대신 **TPM 2.0** 정책을 사용할 때 유사한 절차를 사용할 수 있습니다.

추가 리소스

- **Clevis(1), clevis-luks-bind(1), cryptsetup(8)** 및 **dmsetup(8)** 도움말 페이지
- **RHEL** 자동 설치

10.11. LUKS 암호화 이동식 스토리지 장치의 자동 잠금 해제 구성

LUKS 암호화 USB 스토리지 장치의 자동 잠금 해제 프로세스를 설정할 수 있습니다.

절차

1. *USB 드라이브와 같은 LUKS로 암호화된 이동식 스토리지 장치의 잠금을 자동으로 해제하려면 clevis-udisks2 패키지를 설치합니다.*

```
# yum install clevis-udisks2
```

2. *시스템을 재부팅한 다음 LUKS 암호화 볼륨의 수동 등록 구성에 설명된 대로 clevis luks bind 명령을 사용하여 바인딩 단계를 수행합니다. 예를 들면 다음과 같습니다.*

```
# clevis luks bind -d /dev/sdb1 tang '{"url":"http://tang.srv"}'
```

3. *이제 GNOME 데스크탑 세션에서 LUKS로 암호화된 이동식 장치의 잠금을 자동으로 해제할 수 있습니다. Clevis 정책에 바인딩된 장치는 clevis luks unlock 명령으로 잠금 해제할 수도 있습니다.*

```
# clevis luks unlock -d /dev/sdb1
```

Tang 서버 대신 TPM 2.0 정책을 사용할 때 유사한 절차를 사용할 수 있습니다.

추가 리소스

- *clevis-luks-unlockers(7) 매뉴얼 페이지*

10.12. 고가용성 NBDE 시스템 배포

Tang은 고가용성 배포를 구축하기 위한 두 가지 방법을 제공합니다.

클라이언트 중복(권장)

클라이언트를 여러 Tang 서버에 바인딩할 수 있는 기능으로 구성해야 합니다. 이 설정에서 각 Tang 서버에는 자체 키가 있으며 클라이언트는 이러한 서버의 하위 집합에 연결하여 암호를 해독할 수 있습니다. Clevis는 이미 sss 플러그인을 통해 이 워크플로를 지원합니다. Red Hat은 고가용성 배포에 이 방법을 권장합니다.

키 공유

중복성을 위해 둘 이상의 **Tang** 인스턴스를 배포할 수 있습니다. 두 번째 또는 후속 인스턴스를 설정하려면 **tang** 패키지를 설치하고 **SSH** 를 통해 **rsync** 를 사용하여 키 디렉터리를 새 호스트에 복사합니다. 키를 공유하면 키 손상 위험이 증가하고 추가 자동화 인프라가 필요하므로 **Red Hat**은 이 방법을 권장하지 않습니다.

Shamir의 **Secret Sharing**을 사용하여 **high-available EgressIP**

Shamir의 **SDS(Secret Sharing)**는 시크릿을 여러 가지 고유한 부분으로 나누는 암호화 체계입니다. 시크릿을 복원하려면 여러 부분이 필요합니다. 이 수를 임계값이라고 하며 **SSS**를 임계값 지정 체계라고도 합니다.

Clevis는 **SSS** 구현을 제공합니다. 키를 생성하고 여러 조각으로 나눕니다. 각 조각은 **SSS**를 포함하여 다른 핀을 사용하여 암호화됩니다. 또한 **t** 임계값을 정의합니다. **TPM** 배포가 최소 **t** 조각을 암호 해독하는 경우 암호화 키를 복구하고 암호 해독 프로세스가 성공합니다. **Clevis**가 임계값에 지정된 것보다 적은 수의 부분을 감지하면 오류 메시지를 출력합니다.

예 1: 두 개의 **Tang** 서버를 통한 이중화

다음 명령은 두 개의 **Tang** 서버 중 하나를 사용할 수 있는 경우 **LUKS** 암호화 장치의 암호를 해독합니다.

```
# clevis luks bind -d /dev/sda1 sss '{"t":1,"pins":{"tang":[{"url":"http://tang1.srv"},
{"url":"http://tang2.srv"}]}}'
```

이전 명령에서는 다음 구성 스키마를 사용했습니다.

```
{
    "t":1,
    "pins":{
        "tang":[
            {
                "url":"http://tang1.srv"
            },
            {
                "url":"http://tang2.srv"
            }
        ]
    }
}
```

이 구성에서 **SSS** 임계값 **t**는 **1**로 설정되고 **clevis luks bind** 명령은 나열된 두 개 이상의 **tang** 서버가 사용 가능한 경우 시크릿을 성공적으로 재구성합니다.

예 2: Tang 서버 및 **TPM** 장치의 공유 시크릿

다음 명령은 **tang** 서버와 **tpm2** 장치를 모두 사용할 수 있을 때 **LUKS** 암호화 장치를 성공적으로 해독

합니다.

```
# clevis luks bind -d /dev/sda1 sss '{"t":2,"pins":{"tang":[{"url":"http://tang1.srv"}], "tpm2":
{"pcr_ids":"0,7"}}}'
```

SSS 임계값 **'t'**가 **'2'**로 설정된 구성 스키마는 이제 다음과 같습니다.

```
{
    "t":2,
    "pins":{
        "tang":[
            {
                "url":"http://tang1.srv"
            }
        ],
        "tpm2":{
            "pcr_ids":"0,7"
        }
    }
}
```

추가 리소스

- **tang(8) (section High Availability), clevis(1) (섹션 Shamir's Secret Sharing), clevis-encrypt-sss(1)** 매뉴얼 페이지

10.13. NBDE 네트워크에 가상 머신 배포

clevis luks bind 명령은 **LUKS** 마스터 키를 변경하지 않습니다. 즉, 가상 머신 또는 클라우드 환경에서 사용하기 위해 **LUKS**로 암호화된 이미지를 생성하면 이 이미지를 실행하는 모든 인스턴스가 마스터 키를 공유합니다. 이는 매우 안전하지 않으며 항상 피해야 합니다.

이는 **Clevis**의 제한 사항은 아니지만 **LUKS**의 설계 원칙입니다. 클라우드에 암호화된 루트 볼륨이 필요한 경우 클라우드에서 **Red Hat Enterprise Linux**의 각 인스턴스에 대해 설치 프로세스(일반적으로 **Kickstart** 사용)를 수행합니다. **LUKS** 마스터 키도 공유하지 않고 이미지를 공유할 수 없습니다.

가상화 환경에서의 자동 잠금 해제를 배포하려면 **lorax** 또는 **virt-install** 과 같은 시스템을 **Kickstart** 파일과 함께 사용하십시오(**Kickstart**를 사용하여 **LUKS** 암호화 볼륨 자동 등록 구성 참조) 또는 다른 자동화된 프로비저닝 툴을 사용하여각 암호화된 **VM**에 고유한 마스터 키가 있는지 확인합니다.

추가 리소스

- *clevis-luks-bind(1) 매뉴얼 페이지*

10.14. NBDE를 사용하여 클라우드 환경에 대해 자동으로 등록할 수 있는 VM 이미지 빌드

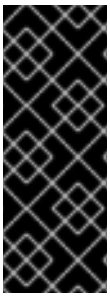
클라우드 환경에 자동으로 표시될 수 있는 암호화된 이미지를 배포하면 고유한 문제를 제공할 수 있습니다. 다른 가상화 환경과 마찬가지로 LUKS 마스터 키를 공유하지 않도록 단일 이미지에서 시작된 인스턴스 수를 줄이는 것이 좋습니다.

따라서 공용 리포지토리에서 공유되지 않고 제한된 인스턴스 배포에 대한 기반을 제공하는 사용자 지정 이미지를 생성하는 것이 좋습니다. 생성할 정확한 인스턴스 수는 배포의 보안 정책에 따라 정의되어야 하며 LUKS 마스터 키 공격 벡터와 관련된 위험 허용 오차를 기반으로 합니다.

LUKS 지원 자동 배포를 빌드하려면 Lorax 또는 virt-install과 같은 시스템을 Kickstart 파일과 함께 사용하여 이미지 빌드 프로세스 중 마스터 키의 고유성을 보장해야 합니다.

클라우드 환경에서는 여기에서 고려할 두 가지 Tang 서버 배포 옵션을 사용할 수 있습니다. 먼저 Tang 서버를 클라우드 환경 자체에 배포할 수 있습니다. 두 번째, Tang 서버는 두 인프라 간에 VPN 링크를 사용하여 독립 인프라에 클라우드 외부에 배포할 수 있습니다.

Tang을 클라우드에 기본적으로 배포하면 쉽게 배포할 수 있습니다. 그러나 인프라가 다른 시스템의 암호 텍스트의 데이터 지속성 계층과 공유되므로 Tang 서버의 개인 키와 Clevis 메타데이터가 동일한 물리적 디스크에 저장될 수 있습니다. 이 물리적 디스크에 대한 액세스는 암호화 텍스트 데이터를 완전히 손상시킬 수 있습니다.



중요

항상 데이터가 저장된 위치와 Tang이 실행 중인 시스템을 물리적으로 분리합니다. 클라우드와 Tang 서버를 분리하면 Tang 서버의 개인 키를 실수로 Clevis 메타데이터와 결합할 수 없습니다. 또한 클라우드 인프라가 위험한 경우 Tang 서버의 로컬 제어 기능을 제공합니다.

10.15. TANG을 컨테이너로 배포

tang 컨테이너 이미지는 OpenShift Container Platform (OCP) 클러스터 또는 별도의 가상 시스템에서 실행되는 Clevis 클라이언트에 대해 Tang-server 암호 해독 기능을 제공합니다.

사전 요구 사항

- **podman** 패키지 및 해당 종속 항목은 시스템에 설치됩니다.

- **podman login registry.redhat.io** 명령을 사용하여 **registry.redhat.io** 컨테이너 카탈로그에 로그인했습니다. 자세한 내용은 **Red Hat Container Registry Authentication**을 참조하십시오.

- **Clevis** 클라이언트는 **Tang** 서버를 사용하여 자동으로 잠금 해제하려는 **LUKS** 암호화된 볼륨이 포함된 시스템에 설치됩니다.

절차

1. **registry.redhat.io** 레지스트리에서 **tang** 컨테이너 이미지를 가져옵니다.

```
# podman pull registry.redhat.io/rhel8/tang
```

2. 컨테이너를 실행하고 해당 포트를 지정하고 **Tang** 키의 경로를 지정합니다. 이전 예제에서는 **tang** 컨테이너를 실행하고 포트 **7500**을 지정하고 **/var/db/tang** 디렉터리의 **Tang** 키의 경로를 나타냅니다.

```
# podman run -d -p 7500:7500 -v tang-keys:/var/db/tang --name tang
registry.redhat.io/rhel8/tang
```

Tang은 기본적으로 포트 **80**을 사용하지만 이는 **Apache HTTP** 서버와 같은 다른 서비스와 충돌할 수 있습니다.

3. [선택 사항] 보안을 강화하기 위해 **Tang** 키를 주기적으로 회전합니다. **tangd-rotate-keys** 스크립트를 사용할 수 있습니다. 예를 들면 다음과 같습니다.

```
# podman run --rm -v tang-keys:/var/db/tang registry.redhat.io/rhel8/tang tangd-rotate-keys -
v -d /var/db/tang
Rotated key 'rZAMKAseaXBe0rcKXL1hCCIq-DY.jwk' -> .'rZAMKAseaXBe0rcKXL1hCCIq-
DY.jwk'
Rotated key 'x1AIpc6WmnCU-CabD8_4q18vDuw.jwk' -> .'x1AIpc6WmnCU-
CabD8_4q18vDuw.jwk'
Created new key GrMMX_WfdqomIU_4RyjpcdlXb0E.jwk
Created new key _dTTfn17sZZqVAp80u3ygFDHtjk.jwk
Keys rotated successfully.
```

검증

- **Tang** 서버의 존재로 자동 잠금 해제를 위한 **LUKS** 암호화된 볼륨이 포함된 시스템에서

Clevis 클라이언트가 **Tang**을 사용하여 일반 텍스트 메시지를 암호화 및 암호 해독할 수 있는지 확인합니다.

```
# echo test | clevis encrypt tang '{"url":"http://localhost:7500"}' | clevis decrypt
The advertisement contains the following signing keys:

x1Alpc6WmnCU-CabD8_4q18vDuw

Do you wish to trust these keys? [ynYN] y
test
```

위 예제 명령은 **localhost URL**에서 **Tang** 서버를 사용할 수 있고 포트 **7500** 을 통해 통신할 때 출력 끝에 **test** 문자열을 표시합니다.

추가 리소스

- **podman(1), clevis(1), tang(8)** 매뉴얼 페이지

10.16. NBDE_CLIENT 및 NBDE_SERVER RHEL 시스템 역할 소개(CLEVIS 및 TANG)

RHEL 시스템 역할은 여러 **RHEL** 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 **Ansible** 역할 및 모듈의 컬렉션입니다.

RHEL 8.3에서는 **Clevis** 및 **Tang**을 사용하여 **PBD(Policy-Based Decryption)** 솔루션의 자동 배포를 위해 **Ansible** 역할을 도입했습니다. **rhel-system-roles** 패키지에는 이러한 시스템 역할, 관련 예제 및 참조 문서가 포함되어 있습니다.

nbde_client 시스템 역할을 사용하면 자동화된 방식으로 여러 **Clevis** 클라이언트를 배포할 수 있습니다. **nbde_client** 역할은 **Tang** 바인딩만 지원하며 현재 **TPM2** 바인딩에는 사용할 수 없습니다.

nbde_client 역할에는 **LUKS**를 사용하여 이미 암호화된 볼륨이 필요합니다. 이 역할은 **LUKS** 암호화 볼륨을 하나 이상의 **Network-Bound(NBDE)** 서버 - **Tang** 서버에 바인딩하도록 지원합니다. 암호를 사용하여 기존 볼륨 암호화를 보존하거나 제거할 수 있습니다. 암호를 제거한 후 **Clevis**를 사용하여 볼륨의 잠금을 해제할 수 있습니다. 이 기능은 시스템을 프로비저닝한 후 제거해야 하는 임시 키 또는 암호를 사용하여 볼륨을 처음 암호화할 때 유용합니다.

암호와 키 파일을 둘 다 제공하는 경우 역할은 먼저 제공한 항목을 사용합니다. 이러한 유효한 항목이 없는 경우 기존 바인딩에서 암호를 검색하려고 합니다.

PBD는 장치를 슬롯에 매핑하는 것으로 바인딩을 정의합니다. 즉, 동일한 장치에 대한 여러 바인딩을 사용할 수 있습니다. 기본 슬롯은 슬롯 1입니다.

nbde_client 역할은 상태 변수도 제공합니다. 새 바인딩을 생성하거나 기존 바인딩을 업데이트하려면 present 값을 사용합니다. clevis luks bind 명령과 반대로 state: present 를 사용하여 장치 슬롯의 기존 바인딩을 덮어쓸 수도 있습니다. absent 값은 지정된 바인딩을 제거합니다.

nbde_client 시스템 역할을 사용하여 Tang 서버를 자동화된 디스크 암호화 솔루션의 일부로 배포하고 관리할 수 있습니다. 이 역할은 다음 기능을 지원합니다.

- Tang 키 교체
- Tang 키 배포 및 백업

추가 리소스

- /usr/share/ansible/roles/rhel-system-roles.nbde_server/README.md 파일
- /usr/share/ansible/roles/rhel-system-roles.nbde_client/README.md 파일
- /usr/share/doc/rhel-system-roles/nbde_server/ directory
- /usr/share/doc/rhel-system-roles/nbde_client/ directory

10.17. NBDE_SERVER RHEL 시스템 역할을 사용하여 여러 TANG 서버를 설정

Tang 서버 설정이 포함된 Ansible 플레이북을 준비하고 적용할 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.*

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_server
  vars:
    nbde_server_rotate_keys: yes
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true
```

*이 예제 플레이북은 **Tang** 서버와 키 교체의 배포를 확인합니다.*

__nbde_server_manage_firewall__ 및 __nbde_server_manage_selinux__ 가 모두 __true__ 로 설정된 경우 __nbde_server__ 역할은 __firewall__ 및 __selinux__ 역할을 사용하여 __nbde_server__ 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
```

검증

- ***Clevis**가 설치된 시스템에서 **grubby** 툴을 사용하여 초기 부팅 시 **Tang** 핀의 네트워킹을 사용할 수 있도록 하려면 다음을 입력합니다.*

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

추가 리소스

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

10.18. NBDE_CLIENT RHEL 시스템 역할을 사용하여 여러 CLEVIS 클라이언트 설정

nbde_client RHEL 시스템 역할을 사용하면 여러 시스템에서 Clevis 클라이언트 설정이 포함된 Ansible 플레이북을 준비하고 적용할 수 있습니다.

참고

nbde_client 시스템 역할은 Tang 바인딩만 지원합니다. 따라서 TPM2 바인딩에는 사용할 수 없습니다.

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.

```
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_client
  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
      - device: /dev/rhel/swap
```

```
      encryption_key_src: /etc/luks/keyfile
      servers:
        - http://server1.example.com
        - http://server2.example.com
```

*이 예제 플레이북은 두 개 이상의 **Tang** 서버 중 하나를 사용할 수 있을 때 두 개의 **LUKS** 암호화 볼륨의 잠금 해제를 자동화하도록 **Clevis** 클라이언트를 구성합니다.*

*nbde_client 시스템 역할은 **DHCP(Dynamic Host Configuration Protocol)**가 있는 시나리오만 지원합니다. 고정 **IP** 구성이 있는 클라이언트에 **NBDE**를 사용하려면 다음 플레이북을 사용합니다.*

```
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_client
  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
      - device: /dev/rhel/swap
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
  tasks:
    - name: Configure a client with a static IP address during early boot
      ansible.builtin.command:
        cmd: grubby --update-kernel=ALL --
args='GRUB_CMDLINE_LINUX_DEFAULT="ip={{ <ansible_default_ipv4.address> }}::{{
<ansible_default_ipv4.gateway> }}:{{ <ansible_default_ipv4.netmask> }}::{{
<ansible_default_ipv4.alias> }}:none"'
```

*이 플레이북에서 **< ansible_default_ipv4.* >** 문자열을 네트워크의 **IP** 주소로 바꿉니다(예: **ip={{ 192.0.2.10 }}::{{ 192.0.2.1 }}:{{ 255.255.255.0 }}::{{ ens3 }}:none).***

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
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.nbde_client/README.md 파일**
- **/usr/share/doc/rhel-system-roles/nbde_client/ directory**
- 초기 램디스크 (initrd) 문서에서 Linux 네트워크 구성을 예상

11장. 시스템 감사

감사는 시스템에 추가 보안을 제공하지 않습니다. 대신 시스템에서 사용되는 보안 정책 위반을 검색하는 데 사용할 수 있습니다. 이러한 위반은 SELinux와 같은 추가 보안 조치로 인해 추가로 방지할 수 있습니다.

11.1. LINUX 감사

Linux 감사 시스템은 시스템에 대한 보안 관련 정보를 추적하는 방법을 제공합니다. 사전 구성된 규칙에 따라 감사는 로그 항목을 생성하여 시스템에서 발생하는 이벤트에 대한 정보를 가능한 한 많이 기록합니다. 이 정보는 미션 크리티컬한 환경과 보안 정책의 위반 및 수행 조치를 결정하는 데 중요합니다.

다음 목록에는 감사가 로그 파일에 기록할 수 있는 몇 가지 정보가 요약되어 있습니다.

- 이벤트 날짜 및 시간, 유형 및 결과
- 주체 및 오브젝트의 민감도 레이블
- 이벤트를 트리거한 사용자의 ID와 이벤트 연결
- 감사 구성에 대한 모든 수정 사항 및 감사 로그 파일에 액세스 시도
- SSH, Kerberos 등과 같은 인증 메커니즘의 모든 사용
- 신뢰할 수 있는 데이터베이스(예: /etc/passwd)에 대한 변경 사항
- 시스템 내 또는 시스템으로 정보를 가져오거나 내보내려는 시도
- 사용자 ID, 주체 및 오브젝트 라벨 및 기타 특성을 기반으로 이벤트를 포함하거나 제외

감사 시스템을 사용하는 것도 여러 보안 관련 인증에 대한 요구 사항입니다. 감사는 다음 인증 또는 컴플라이언스 가이드의 요구사항을 충족하거나 초과하도록 설계되었습니다.

- ***Controled Access Protection Profile (CAPP)***
- *레이블이 지정된 보안 보호 프로필(**LSPP**)*
- *규칙 세트 기본 액세스 제어(**RSBAC**)*
- *국제 산업 보안 프로그램 운영 설명서 (**NISPOM**)*
- *연방 정보 보안 관리 법률 (**FISMA**)*
- *결제 카드 산업 - **PCI-DSS(Data Security Standard)***
- ***STIG(Security Technical Implementation Guides)***

감사 또한 다음과 같습니다.

- ***National Information Assurance Partnership (NIAP) 및 Best Security Industries (BSI)에** 의해 평가*
- ***Red Hat Enterprise Linux 5에서 LSPP/CAPP/RSBAC/EAL4+** 인증*
- ***Red Hat Enterprise Linux 6에서 운영 체제 보호 프로필/평가 보장 레벨 4 이상(OSPP/EAL4** 이상) 인증*

사용 사례

파일 액세스 감시

감사에서는 파일 또는 디렉터리에 액세스, 수정, 실행 또는 파일의 속성이 변경되었는지 추적할 수 있습니다. 예를 들어 중요한 파일에 대한 액세스를 감지하고 이러한 파일 중 하나가 손상된 경우 감사 추적을 사용할 수 있는 데 유용합니다.

시스템 호출 모니터링

특정 시스템 호출을 사용할 때마다 로그 항목을 생성하도록 감사를 구성할 수 있습니다. 예를 들어 **settimeofday,clock_adjtime** 및 기타 시간 관련 시스템 호출을 모니터링하여 시스템 시간 변경 사항을 추적하는 데 사용할 수 있습니다.

사용자가 실행한 명령 기록

감사는 파일이 실행되었는지 여부를 추적하므로 특정 명령의 모든 실행을 기록하도록 규칙을 정의할 수 있습니다. 예를 들어 **/bin** 디렉터리의 모든 실행 파일에 대해 규칙을 정의할 수 있습니다. 그런 다음 결과 로그 항목을 사용자 **ID**로 검색하여 사용자당 실행된 명령의 감사 추적을 생성할 수 있습니다.

시스템 경로의 실행 기록

규칙 호출 시 경로를 **inode**로 변환하는 파일 액세스를 감시하는 것 외에도 감사에서는 규칙 호출 시 존재하지 않는 경로 실행 또는 규칙 호출 후 파일을 교체한 경우에도 경로 실행을 조사할 수 있습니다. 이를 통해 프로그램 실행 파일을 업그레이드하거나 설치되기 전에 규칙이 계속 작동할 수 있습니다.

보안 이벤트 기록

pam_ceilometer 인증 모듈은 실패한 로그인 시도를 기록할 수 있습니다. 감사는 실패한 로그인 시도를 기록하도록 설정하고 로그인을 시도한 사용자에 대한 추가 정보를 제공합니다.

이벤트 검색

감사에서는 로그 항목을 필터링하고 여러 조건에 따라 전체 감사 추적을 제공하는 **ausearch** 유틸리티를 제공합니다.

요약 보고서 실행

aureport 유틸리티는 기록된 이벤트의 일일 보고서를 생성하는 데 사용할 수 있습니다. 그런 다음 시스템 관리자는 이러한 보고서를 분석하고 의심스런 활동을 추가로 조사할 수 있습니다.

네트워크 액세스 모니터링

시스템 관리자가 네트워크 액세스를 모니터링할 수 있도록 **nftables,iptables** 및 **ebtables** 유틸리티를 구성하여 감사 이벤트를 트리거할 수 있습니다.

참고

감사에서 수집하는 정보의 양에 따라 시스템 성능이 영향을 받을 수 있습니다.

11.2. 감사 시스템 아키텍처

감사 시스템은 사용자 공간 애플리케이션과 유틸리티와 커널 측 시스템 호출 처리의 두 가지 주요 부분으로 구성됩니다. 커널 구성 요소는 사용자 공간 애플리케이션에서 시스템 호출을 수신하고 사용자,작

업,**fstype** 또는 종료 필터 중 하나를 통해 필터링합니다.

시스템 호출이 **exclude** 필터를 통과하면 감사 규칙 구성에 따라 앞서 언급한 필터 중 하나를 통해 전송되며, 추가 처리를 위해 감사 데몬으로 전송됩니다.

사용자 공간 감사 데몬은 커널에서 정보를 수집하고 로그 파일에 항목을 생성합니다. 기타 감사 사용자 공간 유틸리티는 감사 데몬, 커널 감사 구성 요소 또는 감사 로그 파일과 상호 작용합니다.

- **auditctl** 감사 제어 유틸리티는 커널 감사 구성 요소와 상호 작용하여 규칙을 관리하고 이벤트 생성 프로세스의 많은 설정 및 매개 변수를 제어합니다.
- 나머지 감사 유틸리티는 감사 로그 파일의 내용을 입력으로 사용하고 사용자 요구 사항에 따라 출력을 생성합니다. 예를 들어 **aureport** 유틸리티는 기록된 모든 이벤트에 대한 보고서를 생성합니다.

RHEL 8에서 감사 디스패치 데몬(**audisp**) 기능이 감사 데몬(**auditd**)에 통합되어 있습니다. 감사 이벤트와 실시간 분석 프로그램의 상호 작용을 위한 플러그인의 구성 파일은 기본적으로 **/etc/audit/plugins.d/** 디렉토리에 있습니다.

11.3. 보안 환경을 위해 AUDITD 구성

기본 **auditd** 구성은 대부분의 환경에 적합해야 합니다. 그러나 환경이 엄격한 보안 정책을 충족해야 하는 경우 **/etc/audit/auditd.conf** 파일에서 감사 데몬 구성에 대해 다음 설정을 변경할 수 있습니다.

log_file

감사 로그 파일(일반적으로 **/var/log/audit/**)이 있는 디렉터리는 별도의 마운트 지점에 있어야 합니다. 이렇게 하면 다른 프로세스가 이 디렉터리에서 공간을 소비하지 않으며 감사 데몬의 나머지 공간을 정확하게 감지할 수 있습니다.

max_log_file

감사 로그 파일이 있는 파티션에서 사용 가능한 공간을 완전히 사용하려면 단일 감사 로그 파일의 최대 크기를 지정해야 합니다. **max_log_file'** 매개 변수는 최대 파일 크기를 메가바이트 단위로 지정합니다. 지정된 값은 숫자여야 합니다.

max_log_file_action

max_log_file 에 설정된 제한에 도달하면 감사 로그 파일을 덮어쓰지 않도록 **keep_logs** 로 설정해야 하는 작업을 결정합니다.

space_left

space_left_action *매개 변수에 설정된 작업이 트리거되는 디스크에 남은 여유 공간의 양을 지정합니다. 관리자에게 충분한 시간을 제공하고 디스크 공간을 확보할 수 있는 번호로 설정해야 합니다.* ***space_left*** *값은 감사 로그 파일이 생성되는 비율에 따라 달라집니다.* ***space_left*** *값이 정수로 지정되면 절대 크기(MiB)로 해석됩니다. 값이* ***1***에서 ***99*** *사이의 숫자로 지정되고 백분율 기호(예:* ***5%****)가 있으면 감사 데몬은* ***log_file*** *을 포함하는 파일 시스템의 크기에 따라 절대 크기를 메가바이트 단위로 계산합니다.*

space_left_action

space_left_action *매개 변수를 이메일 또는 적절한 알림 방법을 사용하여* ***exec*** *로 설정하는 것이 좋습니다.*

admin_space_left

admin_space_left_action *매개변수에 설정된 작업이 트리거되는 절대 최소 공간 크기를 관리자가 수행하는 작업을 로깅할 충분한 공간을 남겨 두는 값으로 설정해야 합니다. 이 매개변수의 숫자 값은* ***space_left****의 숫자보다 작아야 합니다. 또한 감사 데몬이 디스크 파티션 크기에 따라 숫자를 계산하도록 숫자에 백분율 기호(예:* ***1%****)를 추가할 수도 있습니다.*

admin_space_left_action

단일 사용자 모드로 시스템을 배치하려면 ***single -user****로 설정하고 관리자가 일부 디스크 공간을 확보할 수 있도록 해야 합니다.*

disk_full_action

감사 로그 파일을 보유한 파티션에서 사용 가능한 공간이 없는 경우 트리거되는 작업을 중지 또는 단일 로 설정해야 합니다. 이렇게 하면 감사가 더 이상 이벤트를 로깅할 수 없는 경우 시스템이 단일 사용자 모드로 종료되거나 작동합니다.

disk_error_action

감사 로그 파일을 보유하는 파티션에서 오류가 감지되는 경우, 하드웨어 오작동의 처리와 관련된 로컬 보안 정책에 따라 ***syslog****,단일 또는 중단 으로 설정해야 하는 경우 트리거되는 작업을 지정합니다.*

flush

incremental_async *로 설정해야 합니다. 하드 드라이브와 하드 동기화를 강제하기 전에 디스크에 보낼 수 있는 레코드 수를 결정하는* ***freq*** *매개변수와 함께 작동합니다.* ***freq*** *매개변수는* ***100*** *으로 설정해야 합니다. 이러한 매개 변수를 사용하면 활동 버스트에 적합한 성능을 유지하면서 감사 이벤트 데이터가 디스크의 로그 파일과 동기화됩니다.*

나머지 구성 옵션은 로컬 보안 정책에 따라 설정해야 합니다.

11.4. AUDITD 시작 및 제어

auditd 가 구성된 후 서비스를 시작하여 감사 정보를 수집하여 로그 파일에 저장합니다. **auditd** 를 시작하려면 **root** 사용자로 다음 명령을 사용합니다.

```
# service auditd start
```

부팅 시 시작하도록 **auditd** 를 구성하려면 다음을 수행합니다.

```
# systemctl enable auditd
```

 # auditctl -e 0 명령을 사용하여 **auditd** 를 일시적으로 비활성화하고 **# auditctl -e 1** 을 사용하여 다시 활성화할 수 있습니다.

service auditd < action> 명령을 사용하여 **auditd** 에서 다른 작업을 수행할 수 있습니다. 여기서 **< action >**은 다음 중 하나일 수 있습니다.

중지

auditd 를 중지합니다.

재시작

auditd 를 다시 시작합니다.

다시 로드 또는 강제 로드

/etc/audit/auditd.conf 파일에서 **auditd** 구성을 다시 로드합니다.

rotate

/var/log/audit/ 디렉터리에서 로그 파일을 순환합니다.

resume

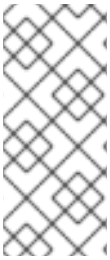
감사 로그 파일을 보유하는 디스크 파티션에 사용 가능한 공간이 충분하지 않은 경우와 같이 이전에 일시 중지된 후 감사 이벤트의 로깅을 재개합니다.

condrestart 또는 **try-restart**

auditd 가 이미 실행 중인 경우에만 다시 시작합니다.

status

auditd 의 실행 상태를 표시합니다.



참고

service 명령은 **auditd** 데몬과 올바르게 상호 작용하는 유일한 방법입니다. **auid** 값이 올바르게 기록되도록 **service** 명령을 사용해야 합니다. **systemctl** 명령은 두 가지 작업인 **enable** 및 **status** 에만 사용할 수 있습니다.

11.5. 로그 파일 감사 이해

기본적으로 감사 시스템은 **/var/log/audit/audit.log** 파일에 로그 항목을 저장합니다. 로그 순환이 활성화된 경우 순환된 **audit.log** 파일이 동일한 디렉터리에 저장됩니다.

/etc/ssh/sshd_config 파일을 읽거나 수정하기 위해 모든 시도를 기록하려면 다음 감사 규칙을 추가합니다.

```
# auditctl -w /etc/ssh/sshd_config -p warx -k sshd_config
```

예를 들어 다음 명령을 사용하여 **auditd** 데몬이 실행 중인 경우 감사 로그 파일에 새 이벤트가 생성됩니다.

```
$ cat /etc/ssh/sshd_config
```

audit.log 파일의 이 이벤트는 다음과 같습니다.

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2 success=no exit=-13
a0=7fffd19c5592 a1=0 a2=7fffd19c4b50 a3=a items=1 ppid=2686 pid=3538 auid=1000 uid=1000
gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1
comm="cat" exe="/bin/cat" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key="sshd_config"
type=CWD msg=audit(1364481363.243:24287):  cwd="/home/shadowman"
type=PATH msg=audit(1364481363.243:24287): item=0 name="/etc/ssh/sshd_config" inode=409248
dev=fd:00 mode=0100600 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0
nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0
type=PROCTITLE msg=audit(1364481363.243:24287) :
proctitle=636174002F6574632F7373682F737368645F636F6E666967
```

위의 이벤트는 동일한 타임스탬프와 일련 번호를 공유하는 네 개의 레코드로 구성됩니다. 레코드는 항상 **type=** 키워드로 시작합니다. 각 레코드는 공백이나 쉼표로 구분된 여러 **name=값** 쌍으로 구성됩니다. 위 이벤트의 자세한 분석은 다음과 같습니다.

첫 번째 레코드

type=SYSCALL

type 필드에는 레코드의 유형이 포함됩니다. 이 예에서 **SYSCALL** 값은 이 레코드가 커널에 대한 시스템 호출에 의해 트리거되었음을 지정합니다.

msg=audit(1364481363.243:24287):

msg 필드는 다음과 같이 기록합니다.

- **audit(time_stamp: ID)** 형식의 타임스탬프와 고유한 레코드**ID** 입니다. 동일한 감사 이벤트의 일부로 생성된 경우 여러 레코드가 동일한 타임스탬프 및 **ID**를 공유할 수 있습니다. 타임 스탬프는 **1970**년 **1**월 **1**일 **00:00:00**부터 **UTC** 이후의 **Unix** 시간 형식을 사용합니다.
- 커널 또는 사용자 공간 애플리케이션에서 제공하는 다양한 이벤트별 이름=값 쌍입니다.

arch=c000003e

arch 필드에는 시스템의 **CPU** 아키텍처에 대한 정보가 포함되어 있습니다. 값 **c000003e** 은 **16**진수 표기법으로 인코딩됩니다. **ausearch** 명령으로 감사 레코드를 검색할 때 **-i** 또는 **--interpret** 옵션을 사용하여 **16**진수 값을 사람이 읽을 수 있는 동등한 값으로 자동 변환합니다. **c000003e** 값은 **x86_64** 로 해석됩니다.

syscall=2

syscall 필드는 커널에 전송된 시스템 호출 유형을 기록합니다. 값 **2** 는 **/usr/include/asm/unistd_64.h** 파일에서 사람이 읽을 수 있는 것과 일치시킬 수 있습니다. 이 경우 **2** 는 공개 시스템 호출입니다. **ausyscall** 유틸리티를 사용하면 시스템 호출 번호를 사람이 읽을 수 있는 해당 번호로 변환할 수 있습니다. **ausyscall --dump** 명령을 사용하여 숫자와 함께 모든 시스템 호출 목록을 표시합니다. 자세한 내용은 **ausyscall(8)** 매뉴얼 페이지를 참조하십시오.

success=no

success 필드는 특정 이벤트에 기록된 시스템 호출이 성공 또는 실패인지를 기록합니다. 이 경우 전화가 성공하지 못했습니다.

exit=-13

exit 필드에는 시스템 호출에서 반환된 종료 코드를 지정하는 값이 포함되어 있습니다. 이 값은 다른 시스템 호출에 따라 다릅니다. 다음 명령을 사용하여 사람이 읽을 수 있는 해당 값을 해석할 수 있습니다.

```
# ausearch --interpret --exit -13
```

*이전 예에서는 감사 로그에 종료 코드 **-13** 로 실패한 이벤트가 포함되어 있다고 가정합니다.*

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

*a0 필드의 **a 0** 필드는 이 이벤트에서 시스템 호출의 **16**진수 표기법으로 인코딩된 처음 네 개의 인수를 기록합니다. 이러한 인수는 사용되는 시스템 호출에 따라 다릅니다. **ausearch** 유틸리티에서 해석할 수 있습니다.*

items=1

*items 필드에는 **syscall** 레코드를 따르는 **PATH** 보조 레코드 수가 포함되어 있습니다.*

ppid=2686

*ppid 필드는 **PPID(Parent Process ID)**를 기록합니다. 이 경우 **2686** 은 상위 프로세스의 **PPID(예: bash)**였습니다.*

pid=3538

*pid 필드는 **PID(프로세스 ID)**를 기록합니다. 이 경우 **3538** 은 **cat** 프로세스의 **PID**입니다.*

auid=1000

*auid 필드는 **loginuid**인 감사 사용자 **ID**를 기록합니다. 이 **ID**는 로그인 시 사용자에게 할당되며, 예를 들어 사용자 계정을 **su - john** 명령으로 전환하여 모든 프로세스에서 상속됩니다.*

uid=1000

*uid 필드는 분석 프로세스를 시작한 사용자의 사용자 **ID**를 기록합니다. 사용자 **ID**는 다음 명령을 사용하여 사용자 이름으로 해석될 수 있습니다. **ausearch -i --uid UID.***

gid=1000

*gid 필드는 분석 프로세스를 시작한 사용자의 그룹 **ID**를 기록합니다.*

euid=1000

*euid 필드는 분석 프로세스를 시작한 사용자의 유효한 사용자 **ID**를 기록합니다.*

suid=1000

*suid 필드는 분석 프로세스를 시작한 사용자의 설정된 사용자 **ID**를 기록합니다.*

fsuid=1000

*fsuid 필드는 분석 프로세스를 시작한 사용자의 파일 시스템 사용자 **ID**를 기록합니다.*

egid=1000

egid 필드는 분석 프로세스를 시작한 사용자의 유효한 그룹 ID를 기록합니다.

sgid=1000

sgid 필드는 분석 프로세스를 시작한 사용자의 세트 그룹 ID를 기록합니다.

fsgid=1000

fsgid 필드는 분석 프로세스를 시작한 사용자의 파일 시스템 그룹 ID를 기록합니다.

tty=pts0

tty 필드는 분석 프로세스가 호출된 터미널을 기록합니다.

ses=1

ses 필드는 분석 프로세스가 호출된 세션의 세션 ID를 기록합니다.

comm="cat"

comm 필드는 분석 프로세스를 호출하는 데 사용된 명령의 명령줄 이름을 기록합니다. 이 경우 cat 명령을 사용하여 이 감사 이벤트를 트리거했습니다.

exe="/bin/cat"

exe 필드는 분석 프로세스를 호출하는 데 사용된 실행 파일의 경로를 기록합니다.

subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

subj 필드는 분석 프로세스가 실행 시 레이블이 지정된 SELinux 컨텍스트를 기록합니다.

key="sshd_config"

key 필드는 감사 로그에서 이 이벤트를 생성한 규칙과 연관된 관리자 정의 문자열을 기록합니다.

두 번째 레코드

type=CWD

두 번째 레코드에서 type 필드 값은 CWD - 현재 작업 디렉터리입니다. 이 유형은 첫 번째 레코드에 지정된 시스템 호출을 호출한 프로세스가 실행된 작업 디렉터리를 기록하는 데 사용됩니다.

이 레코드의 목적은 상대 경로가 연결된 PATH 레코드에 캡처되는 경우 현재 프로세스의 위치를 기록하는 것입니다. 이렇게 하면 절대 경로를 재구성할 수 있습니다.

msg=audit(1364481363.243:24287)

msg 필드는 첫 번째 레코드의 값과 동일한 타임스탬프 및 **ID** 값을 보유합니다. 타임 스탬프는 **1970**년 **1**월 **1**일 **00:00:00**부터 **UTC** 이후의 **Unix** 시간 형식을 사용합니다.

cwd="/home/user_name"

cwd 필드에는 시스템 호출이 호출된 디렉터리의 경로가 포함됩니다.

세 번째 레코드

type=PATH

세 번째 레코드에서 **type** 필드 값은 **PATH** 입니다. 감사 이벤트에는 시스템 호출에 인수로 전달되는 모든 경로에 대한 **PATH-type** 레코드가 포함됩니다. 이 감사 이벤트에서는 하나의 경로(**/etc/ssh/sshd_config**)만 인수로 사용되었습니다.

msg=audit(1364481363.243:24287):

msg 필드는 첫 번째 및 두 번째 레코드의 값과 동일한 타임스탬프 및 **ID** 값을 보유합니다.

item=0

item 필드는 **SYSCALL** 유형 레코드에서 참조되는 총 항목 수, 현재 레코드를 나타냅니다. 이 숫자는 **0**을 기반으로 하며, 값이 **0** 이면 첫 번째 항목임을 의미합니다.

name="/etc/ssh/sshd_config"

name 필드는 시스템 호출에 전달된 파일 또는 디렉터리의 경로를 인수로 기록합니다. 이 경우 **/etc/ssh/sshd_config** 파일이었습니다.

inode=409248

inode 필드에는 이 이벤트에 기록된 파일 또는 디렉터리와 연결된 **inode** 번호가 포함됩니다. 다음 명령은 **409248 inode** 번호와 연결된 파일 또는 디렉터리를 표시합니다.

```
# find / -inum 409248 -print
/etc/ssh/sshd_config
```

dev=fd:00

dev 필드는 이 이벤트에 기록된 파일 또는 디렉터리가 포함된 장치의 마이너 및 주요 **ID**를 지정합니다. 이 경우 값은 **/dev/fd/0** 장치를 나타냅니다.

mode=0100600

mode 필드는 **st_mode** 필드의 **stat** 명령에서 반환된 대로 숫자 표기법으로 인코딩된 파일 또는 디렉터리 권한을 기록합니다. 자세한 내용은 **stat(2)** 매뉴얼 페이지를 참조하십시오. 이 경우 **0100600**

은 **-rw-------** 로 해석될 수 있습니다. 즉, **root** 사용자만 **/etc/ssh/sshd_config** 파일에 대한 읽기 및 쓰기 권한을 갖습니다.

ouid=0

O uid 필드는 오브젝트 소유자의 사용자 **ID**를 기록합니다.

ogid=0

ogid 필드는 오브젝트 소유자의 그룹 **ID**를 기록합니다.

rdev=00:00

rdev 필드에는 특수 파일에 대한 기록된 장치 식별자만 포함됩니다. 이 경우 기록된 파일이 일반 파일이므로 사용되지 않습니다.

obj=system_u:object_r:etc_t:s0

obj 필드는 실행 시 기록된 파일 또는 디렉터리의 레이블이 지정된 **SELinux** 컨텍스트를 기록합니다.

nametype=NORMAL

nametype 필드는 지정된 **syscall**의 컨텍스트에서 각 경로 레코드의 작업의 의도를 기록합니다.

cap_fp=none

cap_fp 필드는 파일 또는 디렉터리 오브젝트의 허용된 파일 시스템 기반 기능의 설정과 관련된 데이터를 기록합니다.

cap_fi=none

cap_fi 필드는 파일 또는 디렉터리 오브젝트의 상속된 파일 시스템 기반 기능의 설정과 관련된 데이터를 기록합니다.

cap_fe=0

cap_fe 필드는 파일 또는 디렉터리 오브젝트의 유효 시스템 기반 기능의 설정을 기록합니다.

cap_fver=0

cap_fver 필드는 파일 또는 디렉터리 오브젝트의 파일 시스템 기반 기능의 버전을 기록합니다.

네 번째 레코드

type=PROCTITLE

type 필드에는 레코드의 유형이 포함됩니다. 이 예제에서 **PROCTITLE** 값은 이 레코드가 커널에

대한 시스템 호출에 의해 트리거되는 이 감사 이벤트를 트리거한 전체 명령줄을 제공하도록 지정합니다.

proctitle=636174002F6574632F7373682F737368645F636F6E666967

proctitle 필드는 분석 프로세스를 호출하는 데 사용된 명령의 전체 명령줄을 기록합니다. 이 필드는 사용자가 감사 로그 구문 분석기에 영향을 미치지 않도록 **16**진수 표기법으로 인코딩됩니다. 텍스트는 이 감사 이벤트를 트리거한 명령에 대해 디코딩합니다. **ausearch** 명령으로 감사 레코드를 검색할 때 **-i** 또는 **--interpret** 옵션을 사용하여 **16**진수 값을 사람이 읽을 수 있는 동등한 값으로 자동 변환합니다. **636174002F6574632F7373682F736F636F6E666967** 값은 **cat /etc/ssh/sshd_config** 로 해석됩니다.

11.6. 감사 규칙 정의 및 실행에 AUDITCTL 사용

감사 시스템은 로그 파일에 캡처된 항목을 정의하는 일련의 규칙에서 작동합니다. **auditctl** 유틸리티를 사용하여 명령줄에서 또는 **/etc/audit/rules.d/** 디렉터리에서 감사 규칙을 설정할 수 있습니다.

auditctl 명령을 사용하면 감사 시스템의 기본 기능을 제어하고 기록된 감사 이벤트를 결정하는 규칙을 정의할 수 있습니다.

파일 시스템 규칙 예

1. **/etc/passwd** 파일의 모든 쓰기 액세스 권한을 로깅하는 규칙을 정의하기 위해 **/etc/passwd** 파일의 모든 속성 변경 사항:

```
# auditctl -w /etc/passwd -p wa -k passwd_changes
```

2. **/etc/selinux/** 디렉터리에 있는 모든 파일에 대한 모든 쓰기 액세스 권한을 로깅하는 규칙을 정의하기 위해 다음을 수행하십시오.

```
# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

시스템 호출 규칙 예

1. **adjtimex** 또는 **settimeofofday** 시스템 호출이 프로그램에서 사용될 때마다 로그 항목을 생성하는 규칙을 정의하기 위해 **64**비트 아키텍처를 사용합니다.

```
# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
```

2. 파일이 삭제될 때마다 로그 항목을 생성하는 규칙을 정의하거나 **ID**가 **1000** 이상인 시스템 사용자로 이름을 변경하려면 다음을 수행합니다.

```
# auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=1000 -F
auid!=4294967295 -k delete
```

-F auid!=4294967295 옵션은 로그인 **UID**가 설정되지 않은 사용자를 제외하는 데 사용됩니다.

실행 파일 규칙

/bin/id 프로그램의 모든 실행을 로깅하는 규칙을 정의하려면 다음 명령을 실행합니다.

```
# auditctl -a always,exit -F exe=/bin/id -F arch=b64 -S execve -k execution_bin_id
```

추가 리소스

- **auditctl(8)** 도움말 페이지.

11.7. 영구 감사 규칙 정의

재부팅 후에도 지속되는 감사 규칙을 정의하려면 **/etc/audit/rules.d/audit.rules** 파일에 직접 포함하거나 **/etc/audit/rules.d/** 디렉터리에 있는 규칙을 읽는 **augenrules** 프로그램을 사용해야 합니다.

auditd 서비스가 시작될 때마다 **/etc/audit/audit.rules** 파일이 생성됩니다. **/etc/audit/rules.d/** 의 파일은 동일한 **auditctl** 명령줄 구문을 사용하여 규칙을 지정합니다. 해시 기호(**#**) 뒤에 오는 빈 줄과 텍스트는 무시됩니다.

또한 **auditctl** 명령을 사용하여 **-R** 옵션을 사용하여 지정된 파일에서 규칙을 읽을 수 있습니다. 예를 들면 다음과 같습니다.

```
# auditctl -R /usr/share/audit/sample-rules/30-stig.rules
```

11.8. 표준을 준수하기 위해 사전 구성된 감사 규칙 파일

OSPP, PCI DSS 또는 **STIG**와 같은 특정 인증 표준 준수에 대한 감사를 구성하려면 감사 패키지와 함께 설치된 사전 구성된 규칙 파일 집합을 시작점으로 사용할 수 있습니다. 샘플 규칙은 **/usr/share/audit/sample-rules** 디렉터리에 있습니다.

주의

*보안 표준이 동적이고 변경될 수 있으므로 **sample-rules** 디렉터리의 감사 샘플 규칙은 완전히 또는 최신 상태가 아닙니다. 이러한 규칙은 감사 규칙을 구성하고 작성할 수 있는 방법을 증명하기 위해서만 제공됩니다. 최신 보안 표준을 즉시 준수하지는 않습니다. 특정 보안 지침에 따라 시스템이 최신 보안 표준을 준수하도록 하려면 SCAP 기반 보안 규정 준수 툴 을 사용합니다.*

30-nispom.rules

국가 산업 보안 프로그램 운영 설명서의 정보 시스템 보안 장에 지정된 요구 사항을 충족하는 감사 규칙 구성.

30-ospp-v42*.rules

***OSPP(Protection Profile for General Purpose Operating Systems)** 프로파일 버전 **4.2**에 정의된 요구 사항을 충족하는 감사 규칙 구성.*

30-pci-dss-v31.rules

***PCIDSS(Payment Card Industry Data Security Standard) v3.1**로 설정된 요구 사항을 충족하는 감사 규칙 구성.*

30-stig.rules

***STIG(Security Technical Implementation Guides)**에 의해 설정된 요구 사항을 충족하는 감사 규칙 구성.*

*이러한 구성 파일을 사용하려면 **/etc/audit/rules.d/** 디렉터리에 복사하고 **augenrules --load** 명령을 사용합니다. 예를 들면 다음과 같습니다.*

```
# cd /usr/share/audit/sample-rules/
# cp 10-base-config.rules 30-stig.rules 31-privileged.rules 99-finalize.rules /etc/audit/rules.d/
# augenrules --load
```

*번호가 매겨진 스키마를 사용하여 감사 규칙을 정렬할 수 있습니다. 자세한 내용은 **/usr/share/audit/sample-rules/README-rules** 파일을 참조하십시오.*

추가 리소스

- *audit.rules(7) 도움말 페이지.*

11.9. AUGENRULES를 사용하여 영구 규칙 정의

augenrules 스크립트는 /etc/audit/rules.d/ 디렉터리에 있는 규칙을 읽고 audit.rules 파일로 컴파일합니다. 이 스크립트는 자연적인 정렬 순서에 따라 .rules 로 끝나는 모든 파일을 특정 순서로 처리합니다. 이 디렉터리의 파일은 다음과 같은 의미를 사용하여 그룹으로 구성됩니다.

10

kernel 및 auditctl 구성

20

일반적인 규칙과 일치할 수 있지만 다른 일치를 원하는 규칙

30

주요 규칙

40

선택적 규칙

50

서버별 규칙

70

시스템 로컬 규칙

90

종료 가능(immutable)

이 규칙은 한 번에 모두 사용할 수 없습니다. 이러한 정책은 간주해야 하는 정책의 일부이며, 개별 파일을 /etc/audit/rules.d/ 로 복사합니다. 예를 들어 STIG 구성에서 시스템을 설정하려면 규칙 10-base-config,30-stig,31-privileged, 99-finalize 를 복사합니다.

/etc/audit/rules.d/ 디렉터리에 규칙이 있으면 --load 지시문을 사용하여 augenrules 스크립트를 실행하여 로드합니다.

```
# augenrules --load
/sbin/augenrules: No change
No rules
enabled 1
failure 1
pid 742
rate_limit 0
...
```

추가 리소스

- **audit.rules(8)** 및 **augenrules(8)** 도움말 페이지.

11.10. AUGENRULES 비활성화

다음 단계를 사용하여 **augenrules** 유틸리티를 비활성화합니다. 이 스위치는 감사를 수행하여 **/etc/audit/audit.rules** 파일에 정의된 규칙을 사용합니다.

절차

1. **/usr/lib/systemd/system/auditd.service** 파일을 **/etc/systemd/system/** 디렉터리에 복사합니다.

```
# cp -f /usr/lib/systemd/system/auditd.service /etc/systemd/system/
```

2. 선택한 텍스트 편집기에서 **/etc/systemd/system/auditd.service** 파일을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi /etc/systemd/system/auditd.service
```

3. **augenrules** 가 포함된 행을 주석 처리하고 **auditctl -R** 명령이 포함된 행의 주석을 제거합니다.

```
#ExecStartPost=-/sbin/augenrules --load
ExecStartPost=-/sbin/auditctl -R /etc/audit/audit.rules
```

4. **systemd** 데몬을 다시 로드하여 **auditd.service** 파일의 변경 사항을 가져옵니다.

```
# systemctl daemon-reload
```

5. **auditd** 서비스를 다시 시작하십시오.

```
# service auditd restart
```

추가 리소스

- **augenrules(8)** 및 **audit.rules(8)** 도움말 페이지.
- **auditd** 서비스를 다시 시작해도 **/etc/audit/audit.rules**에 대한 변경 사항을 덮어씁니다.

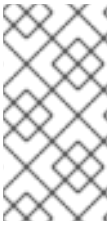
11.11. 소프트웨어 업데이트 모니터링을 위한 감사 설정

RHEL 8.6 이상 버전에서는 사전 구성된 규칙 **44-installers.rules** 를 사용하여 소프트웨어를 설치하는 다음 유틸리티를 모니터링하도록 감사를 구성할 수 있습니다.

- **dnf** [3]
- **yum**
- **pip**
- **npm**
- **cpan**
- **gem**
- **luarocks**

기본적으로 **rpm** 은 패키지를 설치하거나 업데이트할 때 감사 **SORE_UPDATE** 이벤트를 이미 제공합니다. 명령줄에서 **ausearch -m SOFTWARE_UPDATE** 를 입력하여 나열할 수 있습니다.

RHEL 8.5 및 이전 버전에서는 수동으로 규칙을 추가하여 **/etc/audit/rules.d/** 디렉터리 내의 **.rules** 파일에 소프트웨어를 설치하는 유틸리티를 모니터링할 수 있습니다.



참고

ppc64le 및 **aarch64** 아키텍처가 있는 시스템에서는 사전 구성된 규칙 파일을 사용할 수 없습니다.

사전 요구 사항

- **auditd** 는 보안 환경에 대해 **auditd** 구성에 제공된 설정에 따라 구성됩니다.

절차

1. **RHEL 8.6** 이상에서 **/usr/share/audit/sample-rules/** 디렉터리에서 **/etc/audit/rules.d/** 디렉터리에 사전 구성된 규칙 파일 **44-installers.rules** 를 복사합니다.

```
# cp /usr/share/audit/sample-rules/44-installers.rules /etc/audit/rules.d/
```

RHEL 8.5 및 이전 버전에서 **44-installers.rules** 라는 **/etc/audit/rules.d/** 디렉터리에 새 파일을 생성하고 다음 규칙을 삽입합니다.

```
-a always,exit -F perm=x -F path=/usr/bin/dnf-3 -F key=software-installer
-a always,exit -F perm=x -F path=/usr/bin/yum -F
```

동일한 구문을 사용하여 소프트웨어를 설치하는 다른 유틸리티에 대한 규칙을 추가할 수 있습니다(예: **pip** 및 **npm**).

2. 감사 규칙을 로드합니다.

```
# augenrules --load
```

검증

1. 로드된 규칙을 나열합니다.

```
# auditctl -l
-p x-w /usr/bin/dnf-3 -k software-installer
```

```
-p x-w /usr/bin/yum -k software-installer
-p x-w /usr/bin/pip -k software-installer
-p x-w /usr/bin/npm -k software-installer
-p x-w /usr/bin/cpan -k software-installer
-p x-w /usr/bin/gem -k software-installer
-p x-w /usr/bin/luarocks -k software-installer
```

2. *설치를 수행합니다. 예를 들면 다음과 같습니다.*

```
# yum reinstall -y vim-enhanced
```

3. *최근 설치 이벤트의 감사 로그를 검색합니다. 예를 들면 다음과 같습니다.*

```
# ausearch -ts recent -k software-installer
----
time->Thu Dec 16 10:33:46 2021
type=PROCTITLE msg=audit(1639668826.074:298):
proctitle=2F7573722F6C6962657865632F706C6174666F726D2D707974686F6E002F75737
22F62696E2F646E66007265696E7374616C6C002D790076696D2D656E68616E636564
type=PATH msg=audit(1639668826.074:298): item=2 name="/lib64/ld-linux-x86-64.so.2"
inode=10092 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:ld_so_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=1 name="/usr/libexec/platform-python"
inode=4618433 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=0 name="/usr/bin/dnf" inode=6886099
dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:rpm_exec_t:s0
nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1639668826.074:298): cwd="/root"
type=EXECVE msg=audit(1639668826.074:298): argc=5 a0="/usr/libexec/platform-python"
a1="/usr/bin/dnf" a2="reinstall" a3="-y" a4="vim-enhanced"
type=SYSCALL msg=audit(1639668826.074:298): arch=c000003e syscall=59 success=yes
exit=0 a0=55c437f22b20 a1=55c437f2c9d0 a2=55c437f2aeb0 a3=8 items=3 ppid=5256
pid=5375 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=3
comm="dnf" exe="/usr/libexec/platform-python3.6"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="software-installer"
```

11.12. 감사를 사용하여 사용자 로그인 시간 모니터링

*특정 시간에 로그인한 사용자를 모니터링하려면 특별한 방법으로 감사를 구성할 필요가 없습니다. 동일한 정보를 제공하는 다양한 방법을 제공하는 **ausearch** 또는 **aureport** 도구를 사용할 수 있습니다.*

사전 요구 사항

- **auditd** 는 보안 환경에 대해 **auditd** 구성에 제공된 설정에 따라 구성됩니다.

절차

사용자 로그인 시간을 표시하려면 다음 명령 중 하나를 사용합니다.

- **USER_LOGIN** 메시지 유형의 감사 로그를 검색합니다.

```
# ausearch -m USER_LOGIN -ts '12/02/2020' '18:00:00' -sv no
time->Mon Nov 22 07:33:22 2021
type=USER_LOGIN msg=audit(1637584402.416:92): pid=1939 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login acct="
(unknown)" exe="/usr/sbin/sshd" hostname=? addr=10.37.128.108 terminal=ssh res=failed'
```

  - **ts** 옵션을 사용하여 날짜와 시간을 지정할 수 있습니다. 이 옵션을 사용하지 않는 경우 **ausearch** 는 오늘부터 결과를 제공하며 시간을 생략하면 **ausearch** 는 자정에서 결과를 제공합니다.

  - **-sv yes** 옵션을 사용하여 성공적인 로그인 시도를 필터링하고 실패한 로그인 시도에는 **-sv no** 를 사용할 수 있습니다.

- **ausearch** 명령의 원시 출력을 **aulast** 유틸리티로 파이프하여 마지막 명령의 출력과 유사한 형식으로 출력을 표시합니다. 예를 들어 다음과 같습니다.

```
# ausearch --raw | aulast --stdin
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.22.16.106     Mon Nov 22 07:40 - 07:40  (00:00)
reboot   system boot  4.18.0-348.6.el8 Mon Nov 22 07:33
```

- **aureport** 명령을 **--login -i** 옵션과 함께 사용하여 로그인 이벤트 목록을 표시합니다.

```
# aureport --login -i

Login Report
============================================
# date time auid host term exe success event
============================================
1. 11/16/2021 13:11:30 root 10.40.192.190 ssh /usr/sbin/sshd yes 6920
2. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6925
3. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6930
```

```
4. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6935
5. 11/16/2021 13:11:33 root 10.40.192.190 ssh /usr/sbin/sshd yes 6940
6. 11/16/2021 13:11:33 root 10.40.192.190 /dev/pts/0 /usr/sbin/sshd yes 6945
```

추가 리소스

- ausearch(8) 도움말 페이지.
- aulast(8) 도움말 페이지.
- aureport(8) 도움말 페이지.

11.13. 추가 리소스

- RHEL 감사 시스템 참조 지식 베이스 문서.
- 컨테이너 지식베이스 문서의 감사된 실행 옵션.
- Linux 감사 설명서 프로젝트 페이지.
- audit 패키지는 /usr/share/doc/audit/ 디렉터리에 있는 문서를 제공합니다.
- auditd(8), auditctl(8), ausearch(8), audit.rules(7), audispd.conf(5), audispd(8), auditd.conf(5), ausearch-expression(5), aulast(8), aulastlog(8), aureport(8), ausyscall(8), autrace(8) 및 auvirt(8) 도움말 페이지.

---

[3] dnf 는 RHEL의 심볼릭 링크이므로 dnf 감사 규칙의 경로에 symlink의 대상이 포함되어야 합니다. 올바른 감사 이벤트를 수신하려면 path=/usr/bin/dnf path를 /usr/bin/dnf-3 으로 변경하여 44-installers.rules 파일을 수정합니다.

12장. FAPOLICYD를 사용하여 애플리케이션 차단 및 허용

규칙 집합에 따라 애플리케이션 실행을 허용하거나 거부하는 정책을 설정 및 적용하면 알 수 없고 잠재적으로 악의적인 소프트웨어가 실행되지 않습니다.

12.1. FAPOLICYD 소개

fapolicyd 소프트웨어 프레임워크는 사용자 정의 정책을 기반으로 애플리케이션 실행을 제어합니다. 이는 시스템에서 신뢰할 수 없고 잠재적으로 악성 애플리케이션을 실행하는 것을 방지하는 가장 효율적인 방법 중 하나입니다.

fapolicyd 프레임워크는 다음 구성 요소를 제공합니다.

- **fapolicyd** 서비스
- **fapolicyd** 명령줄 유틸리티
- **fapolicyd RPM** 플러그인
- **fapolicyd** 규칙 언어
- **fagenrules** 스크립트

관리자는 경로, 해시, **MIME** 유형 또는 신뢰에 따라 감사의 가능성을 사용하여 모든 애플리케이션에 대해 허용 및 거부 실행 규칙을 정의할 수 있습니다.

fapolicyd 프레임워크는 신뢰 개념을 소개합니다. 애플리케이션은 시스템 패키지 관리자가 적절하게 설치할 때 신뢰할 수 있으므로 시스템 **RPM** 데이터베이스에 등록됩니다. **fapolicyd** 데몬은 **RPM** 데이터베이스를 신뢰할 수 있는 바이너리 및 스크립트 목록으로 사용합니다. **fapolicyd RPM** 플러그인은 **YUM** 패키지 관리자 또는 **RPM** 패키지 관리자에서 처리하는 시스템 업데이트를 등록합니다. 플러그인은 이 데이터베이스의 변경 사항에 대해 **fapolicyd** 데몬에 알립니다. 애플리케이션을 추가하는 다른 방법으로는 사용자 지정 규칙을 생성하고 **fapolicyd** 서비스를 다시 시작해야 합니다.

fapolicyd 서비스 구성은 다음 구조의 /etc/fapolicyd/ 디렉터리에 있습니다.

- */etc/fapolicyd/fapolicyd.trust 파일에는 신뢰할 수 있는 파일 목록이 포함되어 있습니다. /etc/fapolicyd/trust.d/ 디렉토리에서 여러 신뢰 파일을 사용할 수도 있습니다.*

- *허용 및 거부 실행 규칙이 포함된 파일의 /etc/fapolicyd/rules.d/ 디렉터리입니다. fagenrules 스크립트는 이러한 구성 요소 규칙 파일을 /etc/fapolicyd/ECDHE.rules 파일에 병합합니다.*

- *fapolicyd.conf 파일에는 데몬 구성 옵션이 포함되어 있습니다. 이 파일은 주로 성능 튜닝 목적에 유용합니다.*

/etc/fapolicyd/rules.d/ 의 규칙은 여러 파일로 구성되며 각각 다른 정책 목표를 나타냅니다. 해당 파일 이름의 시작 부분에 있는 숫자는 /etc/fapolicyd/ECDHE.rules에서 순서를 결정합니다.

10

언어 규칙.

20

dracut 관련 규칙.

21

업데이트에 대한 규칙입니다.

30

패턴.

40

ELF 규칙입니다.

41

공유 오브젝트 규칙.

42

신뢰할 수 있는 ELF 규칙

70

신뢰할 수 있는 언어 규칙입니다.

72

쉘 규칙.

90

실행 규칙을 거부합니다.

95

열린 규칙을 허용합니다.

***fapolicyd** 무결성 검사를 위해 다음 방법 중 하나를 사용할 수 있습니다.*

- *파일 크기 검사*
- ***SHA-256** 해시 비교*
- ***IMA(Integrity Measurement Architecture)** 하위 시스템*

*기본적으로 **fapolicyd** 는 무결성 검사를 수행하지 않습니다. 파일 크기에 따른 무결성 검사는 빠르지만 공격자는 파일의 내용을 교체하고 바이트 크기를 유지할 수 있습니다. **SHA-256** 체크섬을 계산하고 확인하는 것은 더 안전하지만 시스템의 성능에 영향을 미칩니다. **fapolicyd.conf** 의 **integrity = ima** 옵션은 실행 파일이 포함된 모든 파일 시스템에서 파일 확장 속성(**xattr**라고도 함)을 지원해야 합니다.*

추가 리소스

- ***fapolicyd(8), fapolicyd.rules(5), fapolicyd.conf(5), fapolicyd.trust ECDHE , fagenrules(8), fapolicyd-cli(1)** 매뉴얼 페이지.*
- *커널 무결성 하위 시스템을 사용한 보안 강화 장은 커널 문서의 관리, 모니터링 및 업데이트 장입니다.*
- ***/usr/share/doc/fapolicyd/** 디렉터리에 **fapolicyd** 패키지와 **/usr/share/fapolicyd/sample-***

rules/README-rules 파일에 설치된 문서입니다.

12.2. FAPOLICYD 배포

RHEL에 fapolicyd 프레임워크를 배포하려면 다음을 수행합니다.

절차

1. *fapolicyd 패키지를 설치합니다.*

```
# yum install fapolicyd
```

2. *fapolicyd 서비스를 활성화하고 시작합니다.*

```
# systemctl enable --now fapolicyd
```

검증

1. *fapolicyd 서비스가 올바르게 실행되고 있는지 확인합니다.*

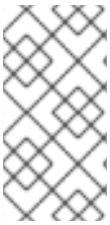
```
# systemctl status fapolicyd
● fapolicyd.service - File Access Policy Daemon
   Loaded: loaded (/usr/lib/systemd/system/fapolicyd.service; enabled; vendor p>
   Active: active (running) since Tue 2019-10-15 18:02:35 CEST; 55s ago
  Process: 8818 ExecStart=/usr/sbin/fapolicyd (code=exited, status=0/SUCCESS)
 Main PID: 8819 (fapolicyd)
    Tasks: 4 (limit: 11500)
   Memory: 78.2M
   CGroup: /system.slice/fapolicyd.service
           └─8819 /usr/sbin/fapolicyd

Oct 15 18:02:35 localhost.localdomain systemd[1]: Starting File Access Policy D>
Oct 15 18:02:35 localhost.localdomain fapolicyd[8819]: Initialization of the da>
Oct 15 18:02:35 localhost.localdomain fapolicyd[8819]: Reading RPMDB into memory
Oct 15 18:02:35 localhost.localdomain systemd[1]: Started File Access Policy Da>
Oct 15 18:02:36 localhost.localdomain fapolicyd[8819]: Creating database
```

2. *root 권한이 없는 사용자로 로그인하고 fapolicyd 가 작동하는지 확인합니다. 예를 들면 다음과 같습니다.*

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

12.3. 추가 신뢰 소스를 사용하여 파일을 신뢰할 수 있음으로 표시

fapolicyd 프레임워크는 **RPM** 데이터베이스에 포함된 파일을 신뢰합니다. **/etc/fapolicyd/fapolicyd.trust plain-text** 파일 또는 신뢰할 수 있는 파일 목록을 더 많은 파일로 분리하는 **/etc/fapolicyd/trust.d/** 디렉터리에 해당 항목을 추가하여 추가 파일을 신뢰할 수 있습니다. 텍스트 편집기 또는 **fapolicyd-cli** 명령을 사용하여 직접 **/etc/fapolicyd/trust.d** 의 **fapolicyd.trust** 또는 파일을 수정할 수 있습니다.



참고

fapolicyd.trust 또는 **trust.d/** 를 사용하여 파일을 신뢰할 수 있는 것으로 표시하는 것이 성능상의 이유로 사용자 정의 **fapolicyd** 규칙을 작성하는 것보다 더 좋습니다.

사전 요구 사항

- **fapolicyd** 프레임워크는 시스템에 배포됩니다.

절차

1. 사용자 지정 바이너리를 필수 디렉터리에 복사합니다. 예를 들면 다음과 같습니다.

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

2. 사용자 정의 바이너리를 신뢰할 수 있음으로 표시하고 해당 항목을 **/etc/fapolicyd/trust.d/** 의 **myapp** 파일에 저장합니다.

```
# fapolicyd-cli --file add /tmp/ls --trust-file myapp
```

- **--trust-file** 옵션을 건너뛰면 이전 명령은 해당 행을 **/etc/fapolicyd/fapolicyd.trust** 에 추가합니다.

- 디렉토리의 기존 파일을 신뢰할 수 있음으로 표시하려면 디렉터리 경로를 **--file** 옵션의 인수로 제공합니다(예: **fapolicyd-cli --file add /tmp/my_bin_dir/ --trust-file myapp**).

3. **fapolicyd** 데이터베이스를 업데이트합니다.

```
# fapolicyd-cli --update
```

참고

신뢰할 수 있는 파일 또는 디렉터리의 내용을 변경하면 체크섬이 변경되어 **fapolicyd** 에서 더 이상 신뢰할 수 있는 것으로 간주하지 않습니다.

새 콘텐츠를 다시 신뢰할 수 있도록 **fapolicyd-cli --file update** 명령을 사용하여 파일 신뢰 데이터베이스를 새로 고칩니다. 인수를 제공하지 않으면 전체 데이터베이스가 새로 고쳐집니다. 또는 특정 파일 또는 디렉터리의 경로를 지정할 수 있습니다. 다음으로 **fapolicyd-cli --update** 를 사용하여 데이터베이스를 업데이트합니다.

검증

1. 사용자 정의 바이너리를 실행할 수 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ /tmp/ls
ls
```

추가 리소스

- **fapolicyd.trustgradle** 매뉴얼 페이지.

12.4. FAPOLICYD에 대한 사용자 정의 허용 및 거부 규칙 추가

fapolicyd 패키지의 기본 규칙 세트는 시스템 기능에 영향을 미치지 않습니다. 표준이 아닌 디렉터리에 바이너리 및 스크립트를 저장하거나 **yum** 또는 **rpm** 설치 프로그램이 없는 애플리케이션 추가와 같은 사용자 지정 시나리오의 경우 추가 파일을 신뢰할 수 있음으로 표시하거나 새 사용자 지정 규칙을 추가해야 합니다.

기본 시나리오 의 경우 추가 신뢰 소스를 사용하여 파일을 신뢰할 수 있는 것으로 표시하는 것이 좋습니다. 특정 사용자 및 그룹 식별자에만 사용자 정의 바이너리를 실행하도록 허용하는 것과 같은 고급 시나리오에서는 **/etc/fapolicyd/rules.d/** 디렉터리에 새 사용자 지정 규칙을 추가합니다.

다음 단계에서는 사용자 지정 바이너리를 허용하는 새 규칙을 추가하는 방법을 보여줍니다.

사전 요구 사항

- ***fapolicyd** 프레임워크는 시스템에 배포됩니다.*

절차

1. *사용자 지정 바이너리를 필수 디렉터리에 복사합니다. 예를 들면 다음과 같습니다.*

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

2. ***fapolicyd** 서비스를 중지합니다.*

```
# systemctl stop fapolicyd
```

3. *디버그 모드를 사용하여 해당 규칙을 식별합니다. **fapolicyd --debug** 명령의 출력은 상세하며 **Ctrl+C** 누르거나 해당 프로세스를 종료하여만 중지할 수 있으므로 오류 출력을 파일로 리디렉션합니다. 이 경우 **--debug:** 대신 **--debug-deny** 옵션을 사용하여 거부에 액세스하도록 출력만 제한할 수 있습니다.*

```
# fapolicyd --debug-deny 2> fapolicy.output &
[1] 51341
```

*또는 다른 터미널에서 **fapolicyd debug** 모드를 실행할 수 있습니다.*

4. ***fapolicyd denied** 명령을 반복합니다.*

```
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

5. ***foreground**에서 다시 시작하고 **Ctrl+C** 눌러 디버그 모드를 중지합니다.*

```
# fg
fapolicyd --debug 2> fapolicy.output
^C
...
```

*또는 **fapolicyd debug** 모드 프로세스를 종료합니다.*

```
# kill 51341
```

6. *애플리케이션 실행을 거부하는 규칙을 찾습니다.*

```
# cat fapolicy.output | grep 'deny_audit'
...
rule=13 dec=deny_audit perm=execute auid=0 pid=6855 exe=/usr/bin/bash : path=/tmp/ls
ftype=application/x-executable trust=0
```

7. *사용자 지정 바이너리를 실행할 수 없는 규칙이 포함된 파일을 찾습니다. 이 경우 **deny_audit perm=execute** 규칙은 **90-deny-execute.rules** 파일에 속합니다.*

```
# ls /etc/fapolicyd/rules.d/
10-languages.rules  40-bad-elf.rules    72-shell.rules
20-dracut.rules     41-shared-obj.rules    90-deny-execute.rules
21-updaters.rules   42-trusted-elf.rules   95-allow-open.rules
30-patterns.rules   70-trusted-lang.rules


# cat /etc/fapolicyd/rules.d/90-deny-execute.rules
# Deny execution for anything untrusted

deny_audit perm=execute all : all
```

8. */**etc/fapolicyd/rules.d**/ 디렉토리에서 사용자 지정 바이너리의 실행을 거부한 규칙이 포함된 규칙 파일 앞에 새 허용 규칙을 추가합니다.*

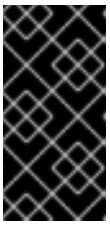
```
# touch /etc/fapolicyd/rules.d/80-myapps.rules
# vi /etc/fapolicyd/rules.d/80-myapps.rules
```

***80-myapps.rules** 파일에 다음 규칙을 삽입합니다.*

```
allow perm=execute exe=/usr/bin/bash trust=1 : path=/tmp/ls ftype=application/x-executable
trust=0
```

*또는 **/etc/fapolicyd/rules.d/**:의 규칙 파일에 다음 규칙을 추가하여 **/tmp** 디렉토리의 모든 바이너리 실행을 허용할 수 있습니다.*

```
allow perm=execute exe=/usr/bin/bash trust=1 : dir=/tmp/ trust=0
```



중요

*지정된 디렉토리 아래의 모든 디렉토리에서 규칙을 반복적으로 적용하려면 규칙에서 **dir=** 매개변수 값에 슬래시를 추가합니다(이전 예에서/**tmp**/).*

9. *사용자 정의 바이너리의 콘텐츠가 변경되지 않도록 하려면 **SHA-256** 체크섬을 사용하여 필요한 규칙을 정의합니다.*

```
$ sha256sum /tmp/ls
780b75c90b2d41ea41679fcb358c892b1251b68d1927c80fbc0d9d148b25e836  ls
```

규칙을 다음 정의로 변경합니다.

```
allow perm=execute exe=/usr/bin/bash trust=1 :
sha256hash=780b75c90b2d41ea41679fcb358c892b1251b68d1927c80fbc0d9d148b25e836
```

10. *컴파일된 목록이 **/etc/fapolicyd/rules.d/** 의 규칙 세트와 다른지 확인하고 / **etc/fapolicyd/ECDHE.rules** 파일에 저장된 목록을 업데이트합니다.*

```
# fagenrules --check
/usr/sbin/fagenrules: Rules have changed and should be updated
# fagenrules --load
```

11. *실행을 중단한 규칙보다 먼저 사용자 지정 규칙이 **fapolicyd** 규칙 목록에 있는지 확인합니다.*

```
# fapolicyd-cli --list
...
13. allow perm=execute exe=/usr/bin/bash trust=1 : path=/tmp/ls ftype=application/x-
executable trust=0
14. deny_audit perm=execute all : all
...
```

12. ***fapolicyd** 서비스를 시작합니다.*

```
# systemctl start fapolicyd
```

검증

1. *사용자 정의 바이너리를 실행할 수 있는지 확인합니다. 예를 들면 다음과 같습니다.*

```
$ /tmp/ls
ls
```

추가 리소스

- **fapolicyd.rules(5)** 및 **fapolicyd-cli(1)** 도움말 페이지
- **/usr/share/fapolicyd/sample-rules/README-rules** 파일에 **fapolicyd** 패키지로 설치된 문서입니다.

12.5. FAPOLICYD 무결성 검사 활성화

기본적으로 **fapolicyd** 는 무결성 검사를 수행하지 않습니다. 파일 크기 또는 **SHA-256** 해시를 비교하여 무결성 검사를 수행하도록 **fapolicyd** 를 구성할 수 있습니다. **Integrity Measurement Architecture (IMA)** 하위 시스템을 사용하여 무결성 검사를 설정할 수도 있습니다.

사전 요구 사항

- **fapolicyd** 프레임워크는 시스템에 배포됩니다.

절차

1. 선택한 텍스트 편집기에서 **/etc/fapolicyd/fapolicyd.conf** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vi /etc/fapolicyd/fapolicyd.conf
```

2. 무결성 옵션의 값을 **none** 에서 **sha256** 으로 변경하고 파일을 저장한 다음 편집기를 종료합니다.

```
integrity = sha256
```

3. **fapolicyd** 서비스를 다시 시작합니다.

```
# systemctl restart fapolicyd
```

검증

1. *확인에 사용되는 파일을 백업합니다.*

```
# cp /bin/more /bin/more.bak
```

2. */bin/more 바이너리의 내용을 변경합니다.*

```
# cat /bin/less > /bin/more
```

3. *변경된 바이너리를 일반 사용자로 사용합니다.*

```
# su example.user
$ /bin/more /etc/redhat-release
bash: /bin/more: Operation not permitted
```

4. *변경 사항을 되돌립니다.*

```
# mv -f /bin/more.bak /bin/more
```

12.6. FAPOLICYD 관련 문제 해결

다음 섹션에서는 rpm 명령을 사용하여 애플리케이션을 추가하기 위한 fapolicyd 애플리케이션 프레임워크의 기본 문제 해결 및 지침을 제공합니다.

rpm을 사용하여 애플리케이션 설치

- *rpm 명령을 사용하여 애플리케이션을 설치하는 경우 fapolicyd RPM 데이터베이스를 수동으로 새로 고침해야 합니다.*

  1. *애플리케이션 을 설치합니다.*

  ```
  # rpm -i application.rpm
  ```

  2. *데이터베이스를 새로 고칩니다.*

  ```
  # fapolicyd-cli --update
  ```

이 단계를 건너뛰면 시스템이 정지될 수 있으며 시스템을 다시 시작해야 합니다.

서비스 상태

- ***fapolicyd** 가 올바르게 작동하지 않으면 서비스 상태를 확인합니다.*

```
# systemctl status fapolicyd
```

***fapolicyd-cli** 검사 및 목록*

- ***--check-config,--check-watch_fs** 및 **--check-trustdb** 옵션은 구문 오류, **not-yet-watched** 파일 시스템 및 파일 불일치를 찾는 데 도움이 됩니다.*

```
# fapolicyd-cli --check-config
Daemon config is OK

# fapolicyd-cli --check-trustdb
/etc/selinux/targeted/contexts/files/file_contexts miscompares: size sha256
/etc/selinux/targeted/policy/policy.31 miscompares: size sha256
```

- ***--list** 옵션을 사용하여 현재 규칙 목록과 해당 순서를 확인합니다.*

```
# fapolicyd-cli --list
...
9. allow perm=execute all : trust=1
10. allow perm=open all : ftype=%languages trust=1
11. deny_audit perm=any all : ftype=%languages
12. allow perm=any all : ftype=text/x-shellscript
13. deny_audit perm=execute all : all
...
```

디버그 모드

- *디버그 모드는 일치하는 규칙, 데이터베이스 상태 등에 대한 자세한 정보를 제공합니다. **fapolicyd** 를 디버그 모드로 전환하려면 다음을 수행합니다.*

1. ***fapolicyd** 서비스를 중지합니다.*

```
# systemctl stop fapolicyd
```

2. **debug** 모드를 사용하여 해당 규칙을 확인합니다.

```
# fapolicyd --debug
```

fapolicyd --debug 명령의 출력이 상세 정보이므로 오류 출력을 파일로 리디렉션할 수 있습니다.

```
# fapolicyd --debug 2> fapolicy.output
```

또는 **fapolicyd** 에서 액세스를 거부하는 경우에만 출력을 항목으로 제한하려면 **--debug-deny** 옵션을 사용합니다.

```
# fapolicyd --debug-deny
```

fapolicyd 데이터베이스 제거

- **fapolicyd** 데이터베이스와 관련된 문제를 해결하려면 데이터베이스 파일을 제거하십시오.

```
# systemctl stop fapolicyd
# fapolicyd-cli --delete-db
```

주의

/var/lib/fapolicyd/ 디렉터리를 제거하지 마십시오. **fapolicyd** 프레임워크는 이 디렉터리의 데이터베이스 파일만 자동으로 복원합니다.

fapolicyd 데이터베이스 덤프

- **fapolicyd** 에는 활성화된 모든 신뢰 소스의 항목이 포함됩니다. 데이터베이스를 덤프한 후 항목을 확인할 수 있습니다.

```
# fapolicyd-cli --dump-db
```

애플리케이션 파이프

- 드문 경우지만 **fapolicyd** 파이프 파일을 제거하면 잠금을 해결할 수 있습니다.

```
# rm -f /var/run/fapolicyd/fapolicyd.fifo
```

추가 리소스

- **fapolicyd-cli(1)** 매뉴얼 페이지.

12.7. FAPOLICYD RHEL 시스템 역할을 사용하여 사용자가 신뢰할 수 없는 코드를 실행하지 못하도록 방지

fapolicyd RHEL 시스템 역할을 사용하여 **fapolicyd** 서비스의 설치 및 구성을 자동화할 수 있습니다. 이 역할을 사용하면 사용자가 **RPM** 데이터베이스와 허용 목록에 나열된 신뢰할 수 있는 애플리케이션만 실행할 수 있도록 서비스를 원격으로 구성할 수 있습니다. 또한 서비스는 허용된 애플리케이션을 실행하기 전에 무결성 검사를 수행할 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: Configuring fapolicyd
  hosts: managed-node-01.example.com
  tasks:
    - name: Allow only executables installed from RPM database and specific files
      ansible.builtin.include_role:
        name: rhel-system-roles.fapolicyd
      vars:
        fapolicyd_setup_permissive: false
        fapolicyd_setup_integrity: sha256
        fapolicyd_setup_trust: rpmdb,file
```

```
    fapolicyd_add_trusted_file:
      - <path_to_allowed_command>
      - <path_to_allowed_service>
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

fapolicyd_setup_permissive: <true|false>

*적용을 위해 커널에 정책 결정 전송을 활성화하거나 비활성화합니다. 디버깅 및 테스트 목적으로 이 변수를 **false** 로 설정합니다.*

fapolicyd_setup_integrity: <type_type>

무결성 검사 방법을 정의합니다. 다음 값 중 하나를 설정할 수 있습니다.

- ***none** (기본값): 무결성 검사를 비활성화합니다.*
- *크기: 서비스는 허용된 애플리케이션의 파일 크기만 비교합니다.*
- ***i MA**: 서비스는 **SHA256** 해시에서 파일의 확장된 속성에 저장된 커널의 무결성 측정 아키텍처(**IMA**)를 확인합니다. 또한 서비스는 크기 검사를 수행합니다. 역할은 **IMA** 커널 하위 시스템을 구성하지 않습니다. 이 옵션을 사용하려면 **IMA** 하위 시스템을 수동으로 구성해야 합니다.*
- ***sha256**: 서비스는 허용된 애플리케이션의 **SHA256** 해시를 비교합니다.*

fapolicyd_setup_trust: <trust_backends>

*신뢰 백엔드 목록을 정의합니다. 파일 백엔드를 포함하는 경우 **fapolicyd_add_trusted_file** 목록에 허용되는 실행 파일을 지정합니다.*

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

2. *플레이북 구문을 확인합니다.*

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

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. *Playbook을 실행합니다.*

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

검증

- *허용 목록에 없는 바이너리 애플리케이션을 사용자로 실행합니다.*

```
$ ansible managed-node-01.example.com -m command -a 'su -c
"/bin/not_authorized_application " <user_name>'
bash: line 1: /bin/not_authorized_application: Operation not permitted non-zero return
code
```

추가 리소스

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

12.8. 추가 리소스

- *fapolicyd- man -k fapolicyd 명령을 사용하여 나열된 관련 도움말 페이지.*
- *FOSDEM 2020 fapolicyd presentation.*

13장. INTRUSIVE USB 장치로부터 시스템 보호

USB 장치는gradle, 악성 코드 또는 트로이 목마와 함께 로드할 수 있으므로 데이터를 손상 시키거나 시스템을 손상시킬 수 있습니다. Red Hat Enterprise Linux 관리자는 USBGuard 를 사용한 이러한 USB 공격을 방지할 수 있습니다.

13.1. USBGUARD

USBGuard 소프트웨어 프레임워크를 사용하면 커널의 USB 장치 권한 부여 기능을 기반으로 허용되는 기본 장치 목록을 사용하여 시스템을 중단 없이 보호할 수 있습니다.

USBGuard 프레임워크는 다음 구성 요소를 제공합니다.

- 동적 상호 작용 및 정책 적용을 위한 프로세스 간 통신(IPC) 인터페이스가 있는 시스템 서비스 구성 요소
- 실행 중인 usbguard 시스템 서비스와 상호 작용하는 명령줄 인터페이스
- USB 장치 권한 부여 정책을 작성하는 규칙 언어
- 공유 라이브러리에 구현된 시스템 서비스 구성 요소와 상호 작용하기 위한 C++ API

usbguard 시스템 서비스 구성 파일(/etc/usbguard/usbguard-daemon.conf)에는 IPC 인터페이스를 사용할 수 있는 사용자 및 그룹에 권한을 부여하는 옵션이 포함되어 있습니다.

중요

시스템 서비스는 **USBGuard** 공용 **IPC** 인터페이스를 제공합니다. **Red Hat Enterprise Linux**에서 이 인터페이스에 대한 액세스는 기본적으로 **root** 사용자로만 제한됩니다.

IPCAccessControlFiles 옵션(권장) 또는 **IPCAllowedUsers** 및 **IPCAllowedGroups** 옵션을 설정하여 **IPC** 인터페이스에 대한 액세스를 제한하는 것이 좋습니다.

IPC 인터페이스를 모든 로컬 사용자에게 노출하고 **USB** 장치의 권한 부여 상태를 조작하고 **USBGuard** 정책을 수정할 수 있으므로 **ACL(**액세스 제어 목록**)**을 구성하지 않도록 하십시오.

13.2. USBGUARD 설치

USBGuard 프레임워크를 설치하고 시작하려면 다음 절차를 사용하십시오.

절차

1. **usbguard** 패키지를 설치합니다.

```
# yum install usbguard
```

2. 초기 규칙 세트를 생성합니다.

```
# usbguard generate-policy > /etc/usbguard/rules.conf
```

3. **usbguard** 데몬을 시작하고 부팅 시 자동으로 시작되는지 확인합니다.

```
# systemctl enable --now usbguard
```

검증

1. **usbguard** 서비스가 실행 중인지 확인합니다.

```
# systemctl status usbguard
● usbguard.service - USBGuard daemon
   Loaded: loaded (/usr/lib/systemd/system/usbguard.service; enabled; vendor preset:
disabled)
```

```
   Active: active (running) since Thu 2019-11-07 09:44:07 CET; 3min 16s ago
     Docs: man:usbguard-daemon(8)
 Main PID: 6122 (usbguard-daemon)
   Tasks: 3 (limit: 11493)
  Memory: 1.2M
  CGroup: /system.slice/usbguard.service
          └─6122 /usr/sbin/usbguard-daemon -f -s -c /etc/usbguard/usbguard-daemon.conf

Nov 07 09:44:06 localhost.localdomain systemd[1]: Starting USBGuard daemon...
Nov 07 09:44:07 localhost.localdomain systemd[1]: Started USBGuard daemon.
```

2. **USBGuard**에서 인식하는 **USB** 장치를 나열합니다.

```
# usbguard list-devices
4: allow id 1d6b:0002 serial "0000:02:00.0" name "xHCI Host Controller" hash...
```

추가 리소스

- **usbguard(1)** 및 **usbguard-daemon.conf(5)** 도움말 페이지.

13.3. CLI를 사용하여 USB 장치 차단 및 승인

터미널에서 **usbguard** 명령을 사용하여 **USB** 장치를 인증하고 차단하도록 **USBGuard**를 설정할 수 있습니다.

사전 요구 사항

- **usbguard** 서비스가 설치되어 실행 중입니다.

절차

1. **USBGuard**에서 인식하는 **USB** 장치를 나열합니다. 예를 들면 다음과 같습니다.

```
# usbguard list-devices
1: allow id 1d6b:0002 serial "0000:00:06.7" name "EHCI Host Controller" hash
"JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" parent-hash
"4PHGcaDKWtPjKDwYpIRG722cB9SlGz9l9lea93+Gt9c=" via-port "usb1" with-interface
09:00:00
...
6: block id 1b1c:1ab1 serial "000024937962" name "Voyager" hash
"CrXgiaWIf2bZAU+5WkzOE7y0rdSO82XMzubn7HDb95Q=" parent-hash
"JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" via-port "1-3" with-interface
08:06:50
```

2. *시스템과 상호 작용하도록 장치 **<6& gt;**에 권한을 부여합니다.*

```
# usbguard allow-device <6>
```

3. *장치 인증 해제 및 제거 **&lt ;6>:***

```
# usbguard reject-device <6>
```

4. *장치 인증 해제 및 유지 **&lt ;6>:***

```
# usbguard block-device <6>
```

참고

***usbguard**는 **term** 블록을 사용하고 다음 의미를 사용하여 거부합니다.*

block

지금은 이 장치와 상호 작용하지 마십시오.

거부

이 장치가 없는 것처럼 무시합니다.

추가 리소스

- ***usbguard(1)** 도움말 페이지*
- ***usbguard --help** 명령*

13.4. USB 장치를 영구적으로 차단 및 승인

***-p** 옵션을 사용하여 **USB** 장치를 영구적으로 차단하고 인증할 수 있습니다. 이를 통해 장치별 규칙이 현재 정책에 추가됩니다.*

사전 요구 사항

- ***usbguard** 서비스가 설치되어 실행 중입니다.*

절차

1. ***usbguard** 데몬이 규칙을 작성할 수 있도록 **SELinux**를 구성합니다.*

   a. ***usbguard** 와 관련된 **semanage** 부울을 표시합니다.*

   ```
   # semanage boolean -l | grep usbguard
   usbguard_daemon_write_conf     (off  ,  off)  Allow usbguard to daemon write conf
   usbguard_daemon_write_rules    (on   ,   on)  Allow usbguard to daemon write rules
   ```

   b. *선택 사항: **usbguard_daemon_write_rules** 부울이 꺼져 있는 경우 해당 부울을 켭니다.*

   ```
   # semanage boolean -m --on usbguard_daemon_write_rules
   ```

2. ***USBGuard**에서 인식하는 **USB** 장치를 나열합니다.*

   ```
   # usbguard list-devices
   1: allow id 1d6b:0002 serial "0000:00:06.7" name "EHCI Host Controller" hash
   "JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" parent-hash
   "4PHGcaDKWtPjKDwYpIRG722cB9SlGz9l9Iea93+Gt9c=" via-port "usb1" with-interface
   09:00:00
   ...
   6: block id 1b1c:1ab1 serial "000024937962" name "Voyager" hash
   "CrXgiaWIf2bZAU+5WkzOE7y0rdSO82XMzubn7HDb95Q=" parent-hash
   "JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" via-port "1-3" with-interface
   08:06:50
   ```

3. *장치 **6** 가 시스템과 상호 작용하도록 영구적으로 승인하십시오.*

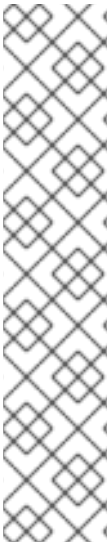
   ```
   # usbguard allow-device 6 -p
   ```

4. *영구적으로 장치 **6** 인증 해제 및 제거:*

   ```
   # usbguard reject-device 6 -p
   ```

5. *영구적으로 장치 **6** 인증 해제 및 유지 :*

```
# usbguard block-device 6 -p
```



참고

***usbguard** 는 **term** 블록을 사용하고 다음 의미를 사용하여 거부합니다.*

block

지금은 이 장치와 상호 작용하지 마십시오.

거부

이 장치가 없는 것처럼 무시합니다.

검증

1. ***USBGuard** 규칙에 변경 사항이 포함되어 있는지 확인합니다.*

```
# usbguard list-rules
```

추가 리소스

- ***usbguard(1)** 도움말 페이지.*
- ***usbguard --help** 명령을 사용하여 나열된 기본 도움말입니다.*

13.5. USB 장치에 대한 사용자 지정 정책 생성

*다음 절차에서는 시나리오의 요구 사항을 반영하는 **USB** 장치에 대한 규칙 세트를 생성하는 단계를 설명합니다.*

사전 요구 사항

- ***usbguard** 서비스가 설치되어 실행 중입니다.*
-

/etc/usbguard/rules.conf 파일에는 **usbguard generate-policy** 명령에서 생성된 초기 규칙 세트가 포함되어 있습니다.

절차

1. 현재 연결된 **USB** 장치를 인증하고 생성된 규칙을 **rules.conf** 파일에 저장하는 정책을 생성합니다.

```
# usbguard generate-policy --no-hashes > ./rules.conf
```

--no-hashes 옵션은 장치에 대한 해시 속성을 생성하지 않습니다. 구성 설정에서 해시 속성이 유지되지 않을 수 있습니다.

2. 선택한 텍스트 편집기를 사용하여 **rules.conf** 파일을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi ./rules.conf
```

3. 필요에 따라 규칙을 추가, 제거 또는 편집합니다. 예를 들어 다음 규칙은 단일 대용량 스토리지 인터페이스가 있는 장치만 시스템과 상호 작용할 수 있습니다.

```
allow with-interface equals { 08:*:* }
```

자세한 규칙 언어 설명 및 자세한 예제는 **usbguard-rules.conf(5)** 매뉴얼 페이지를 참조하십시오.

4. 업데이트된 정책을 설치합니다.

```
# install -m 0600 -o root -g root rules.conf /etc/usbguard/rules.conf
```

5. **usbguard** 데몬을 재시작하여 변경 사항을 적용합니다.

```
# systemctl restart usbguard
```

검증

1. 사용자 지정 규칙이 활성 정책에 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
# usbguard list-rules
...
4: allow with-interface 08:*:*
...
```

추가 리소스

- **usbguard-rules.conf(5)** 도움말 페이지.

13.6. USB 장치에 대한 구조화된 사용자 지정 정책 생성

/etc/usbguard/rules.d/ 디렉터리 내의 여러 **.conf** 파일에서 사용자 지정 **USBGuard** 정책을 구성할 수 있습니다. 그런 다음 **usbguard-daemon** 은 기본 **rules.conf** 파일을 디렉터리 내의 **.conf** 파일과 알파벳 순으로 결합합니다.

사전 요구 사항

- **usbguard** 서비스가 설치되어 실행 중입니다.

절차

1. 현재 연결된 **USB** 장치를 인증하고 생성된 규칙을 새 **.conf** 파일에 저장하는 정책을 생성합니다(예: **policy.conf**).

```
# usbguard generate-policy --no-hashes > ./policy.conf
```

--no-hashes 옵션은 장치에 대한 해시 속성을 생성하지 않습니다. 구성 설정에서 해시 속성이 유지되지 않을 수 있습니다.

2. 선택한 텍스트 편집기로 **policy.conf** 파일을 표시합니다. 예를 들면 다음과 같습니다.

```
# vi ./policy.conf
...
allow id 04f2:0833 serial "" name "USB Keyboard" via-port "7-2" with-interface { 03:01:01
03:00:00 } with-connect-type "unknown"
...
```

3. 선택한 행을 별도의 **.conf** 파일로 이동합니다.

참고

파일 이름의 시작 부분에 있는 두 자리 숫자로 데몬이 구성 파일을 읽는 순서를 지정합니다.

예를 들어 키보드의 규칙을 새 **.conf** 파일에 복사합니다.

```
# grep "USB Keyboard" ./policy.conf > ./10keyboards.conf
```

4. 새 정책을 **/etc/usbguard/rules.d/** 디렉터리에 설치합니다.

```
# install -m 0600 -o root -g root 10keyboards.conf /etc/usbguard/rules.d/10keyboards.conf
```

5. 나머지 행을 기본 **rules.conf** 파일로 이동합니다.

```
# grep -v "USB Keyboard" ./policy.conf > ./rules.conf
```

6. 나머지 규칙을 설치합니다.

```
# install -m 0600 -o root -g root rules.conf /etc/usbguard/rules.conf
```

7. **usbguard** 데몬을 다시 시작하여 변경 사항을 적용합니다.

```
# systemctl restart usbguard
```

검증

1. 모든 활성 **USBGuard** 규칙을 표시합니다.

```
# usbguard list-rules
...
15: allow id 04f2:0833 serial "" name "USB Keyboard" hash
"kxM/iddRe/WSCocgiuQlVs6Dn0VEza7KiHoDeTz0fyg=" parent-hash
"2i6ZBJfTl5BakXF7Gba84/Cp1gslnNc1DM6vWQpie3s=" via-port "7-2" with-interface {
03:01:01 03:00:00 } with-connect-type "unknown"
...
```

2. */etc/usbguard/rules.d/* 디렉토리에 **rules.conf** 파일 및 모든 **.conf** 파일을 표시합니다.

```
# cat /etc/usbguard/rules.conf /etc/usbguard/rules.d/*.conf
```

3. 활성 규칙에 파일의 모든 규칙이 포함되어 있고 올바른 순서로 되어 있는지 확인합니다.

추가 리소스

- **usbguard-rules.conf(5)** 도움말 페이지.

13.7. USBGUARD IPC 인터페이스를 사용하도록 사용자 및 그룹 인증

다음 절차를 사용하여 **USBGuard** 공용 **IPC** 인터페이스를 사용하도록 특정 사용자 또는 그룹에 권한을 부여합니다. 기본적으로 **root** 사용자만 이 인터페이스를 사용할 수 있습니다.

사전 요구 사항

- **usbguard** 서비스가 설치되어 실행 중입니다.

- **/etc/usbguard/rules.conf** 파일에는 **usbguard generate-policy** 명령에서 생성된 초기 규칙 세트가 포함되어 있습니다.

절차

1. 선택한 텍스트 편집기를 사용하여 **/etc/usbguard/usbguard-daemon.conf** 파일을 편집합니다.

```
# vi /etc/usbguard/usbguard-daemon.conf
```

2. 예를 들어, **wheel** 그룹의 모든 사용자가 **IPC** 인터페이스를 사용할 수 있도록 허용하는 규칙이 포함된 행을 추가하고 파일을 저장합니다.

```
IPCAllowGroups=wheel
```

3. **usbguard** 명령을 사용하여 사용자 또는 그룹을 추가할 수 있습니다. 예를 들어 다음 명령을 사용하면 **joesec** 사용자가 **Devices** 및 **Exceptions** 섹션에 대한 전체 액세스 권한을 가질 수 있

습니다. 또한 **joesec** 은 현재 정책을 나열하고 수정할 수 있습니다.

```
# usbguard add-user joesec --devices ALL --policy modify,list --exceptions ALL
```

joesec 사용자에 대해 부여된 권한을 제거하려면 **usbguard remove-user joesec** 명령을 사용합니다.

4. **usbguard** 데몬을 재시작하여 변경 사항을 적용합니다.

```
# systemctl restart usbguard
```

추가 리소스

- **usbguard(1)** 및 **usbguard-rules.conf(5)** 도움말 페이지.

13.8. LINUX 감사 로그에 USBGUARD 인증 이벤트 로깅

다음 단계를 사용하여 **USBguard** 권한 부여 이벤트의 로깅을 표준 **Linux** 감사 로그에 통합합니다. 기본적으로 **usbguard** 데몬은 이벤트를 **/var/log/usbguard/usbguard-audit.log** 파일에 기록합니다.

사전 요구 사항

- **usbguard** 서비스가 설치되어 실행 중입니다.
- **auditd** 서비스가 실행 중입니다.

절차

1. 선택한 텍스트 편집기를 사용하여 **usbguard-daemon.conf** 파일을 편집합니다.

```
# vi /etc/usbguard/usbguard-daemon.conf
```

2. **AuditBackend** 옵션을 **FileAudit** 에서 **LinuxAudit** 로 변경합니다.

```
AuditBackend=LinuxAudit
```

3. **usbguard 데몬을 재시작하여 구성 변경 사항을 적용합니다.**

```
# systemctl restart usbguard
```

검증

1. **USB 권한 부여 이벤트의 감사 데몬 로그를 쿼리합니다. 예를 들면 다음과 같습니다.**

```
# ausearch -ts recent -m USER_DEVICE
```

추가 리소스

- **usbguard-daemon.conf(5) 도움말 페이지.**

13.9. 추가 리소스

- **usbguard(1), usbguard-rules.conf(5), usbguard-daemon(8) 및 usbguard-daemon.conf(5) 도움말 페이지.**

- **USBGuard 홈 페이지.**