Red Hat Enterprise Linux 9

보안 강화

Red Hat Enterprise Linux 9 보안

Last Updated: 2022-08-18

Red Hat Enterprise Linux 9 보안 강화

Red Hat Enterprise Linux 9 보안

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

법적 공지

초록

이 문서는 사용자 및 관리자가 로컬 및 원격 침입, 악용 및 악의적인 활동을 방지하기 위한 워크스테이션 및 서버 보안 프로세스 및 관행을 배우는데 사용할 수 있습니다. 이는 Red Hat Enterprise Linux를 대상으로 하지만 개념과 기술은 모든 Linux 시스템에 적용할 수 있으며, 데이터 센터, 직장 및 가정을 위한 안전한 컴퓨팅 환경을 만드는 데 관련된 계획과 툴에 대해 자세히 설명합니다. 적절한 관리 지식, 경계 시스템 및 툴을 통해 Linux를 실행하는 시스템의 기능을 최대한 활용하여 가장 일반적인 침입과 악용 기술로 부터 시스템 보안을 유지할 수 있습니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

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

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 이를 개선할 수 있는지 알려 주십시오.

- 특정 문구에 대한 간단한 의견 작성 방법은 다음과 같습니다.
  1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **피드백** 버튼이 있는지 확인합니다.
  2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
  3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
  4. 표시된 지침을 따릅니다.
- Bugzilla를 통해 피드백을 제출하려면 새 티켓을 생성하십시오.
  1. Bugzilla 웹 사이트로 이동하십시오.
  2. 구성 요소로 **문서**를 사용합니다.
  3. **설명** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
  4. **버그 제출**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. BIOS 및 UEFI 보안

BIOS(또는 이에 상응하는 BIOS) 및 부트 로더에 대한 암호 보호는 시스템에 대한 물리적 액세스 권한이 없는 사용자가 이동식 미디어를 사용하여 부팅하거나 단일 사용자 모드를 통해 root 권한을 얻는 것을 방지할 수 있습니다. 이러한 공격으로부터 보호하기 위해 수행해야 하는 보안 조치는 워크스테이션에 있는 정보의 민감도와 시스템의 위치에 따라 달라집니다.

예를 들어, 시스템이 무역 박람회에 사용되며 중요한 정보가 포함되지 않은 경우 이러한 공격을 방지하는 것이 중요하지 않을 수 있습니다. 그러나 회사 네트워크에 대해 암호화되지 않은 개인용 SSH 키가 있는 직원의 랩톱이 동일한 무역 박람회에서 그대로 유지되는 경우 회사 전체에 심각한 보안 침해가 발생할 수 있습니다.

반면에 권한이 있거나 신뢰할 수 있는 사람만 액세스할 수 있는 장소에 워크스테이션이 있는 경우 BIOS 또는 부트 로더 보안이 필요하지 않을 수 있습니다.

1.1.1. BIOS 암호

컴퓨터의 BIOS를 암호로 보호하는 두 가지 주요 이유는 다음과 같습니다.[1]:

1. **BIOS 설정 변경 방지** - 침입자가 BIOS에 액세스할 수 있는 경우 CD-ROM 또는 플래시 드라이브에서 부팅하도록 설정할 수 있습니다. 이를 통해 복구 모드 또는 단일 사용자 모드로 전환할 수 있으므로 시스템에서 임의의 프로세스를 시작하거나 중요한 데이터를 복사할 수 있습니다.

2. **시스템 부팅 방지** - 일부 BIOS는 부팅 과정의 암호 보호를 허용합니다. 활성화되면 공격자는 BIOS가 부트 로더를 시작하기 전에 암호를 입력해야 합니다.

BIOS 암호를 설정하는 방법은 컴퓨터 제조업체마다 다르기 때문에 특정 지침은 컴퓨터 설명서를 참조하십시오.

BIOS 암호를 잊어버린 경우 마더보드의 점퍼를 사용하거나 CMOS 배터리의 연결을 해제하여 재설정할 수 있습니다. 따라서 가능한 경우 컴퓨터 케이스를 잠그는 것이 좋습니다. 그러나 CMOS 배터리의 연결을 해제하기 전에 컴퓨터 또는 마더보드에 대한 설명서를 참조하십시오.

1.1.2. 비BIOS 기반 시스템 보안

다른 시스템 및 아키텍처는 서로 다른 프로그램을 사용하여 x86 시스템의 BIOS와 거의 동일한 수준의 작업을 수행합니다. 예를 들어*UEFI*(*Unified Extensible Firmware Interface*) 쉘이 있습니다.

BIOS와 같은 프로그램을 보호하는 암호에 대한 지침은 제조업체의 지침을 참조하십시오.

1.2. 디스크 파티션 설정

Red Hat은**/boot**, /, **/home**, **/tmp**, **/var/tmp/** 디렉토리에 대해 별도의 파티션을 만드는 것을 권장합니다.

/boot

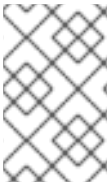
이 파티션은 부팅 중에 시스템에서 읽은 첫 번째 파티션입니다. 시스템을 Red Hat Enterprise Linux 9로 부팅하는 데 사용되는 부트 로더 및 커널 이미지는 이 파티션에 저장됩니다. 이 파티션은 암호화해서는 안 됩니다. 이 파티션이 / 에 포함되어 있고 해당 파티션을 암호화하거나 사용할 수 없게 되면 시스템을 부팅할 수 없습니다.

/home

사용자 데이터(**/home**)가 별도의 파티션 대신 /에 저장되어 파티션이 채워지면 운영 체제가 불안정해집니다. 또한 시스템을 Red Hat Enterprise Linux 9의 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.



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.

1.3. 설치 프로세스 중에 네트워크 연결 제한

Red Hat Enterprise Linux 9를 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 9를 설치하는 동안 리포지토리에서 가장 가까운 영역을 선택하십시오.

1.4. 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.

1.5. 설치 후 절차

다음 단계는 Red Hat Enterprise Linux 9를 설치한 직후 수행해야 하는 보안 관련 절차입니다.

- 시스템을 업데이트합니다. root로 다음 명령을 입력합니다.

```
# dnf update
```

- 방화벽 서비스인 **firewalld**는 Red Hat Enterprise Linux를 설치하여 자동으로 활성화되어 있지만, 예를 들어 Kickstart 구성에서는 명시적으로 비활성화될 수 있는 시나리오가 있습니다. 이러한 경우 방화벽을 다시 활성화하는 것이 좋습니다.
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
  ```

[1] 시스템 BIOS는 제조 업체마다 다르기 때문에 암호 보호 유형 중 하나를 지원하지 않을 수 있지만 다른 유형은 지원하지 않을 수 있습니다.

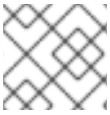
2장. FIPS 모드에서 시스템 설치

연방 정보 처리 표준(FIPS) 140-3에서 요구하는 암호화 모듈 자체 점검을 활성화하려면 FIPS 모드에서 RHEL 9을 실행해야 합니다.

다음을 통해 이를 달성할 수 있습니다.

- FIPS 모드에서 설치를 시작합니다.
- 설치 후 FIPS 모드로 시스템을 전환합니다.

이미 배포된 시스템을 변환하는 것과 연관된 결과 시스템의 규정 준수 및 주요 자료의 재생성 및 재평가를 방지하려면 FIPS 모드에서 설치를 시작하는 것이 좋습니다.



참고

RHEL 9의 암호화 모듈은 FIPS 140-3 요구 사항에 대해 아직 인증되지 않았습니다.

2.1. 연방 정보 처리 표준 (FIPS)

FIPS(Federal Information Processing Standard) 발행 140-3는 미국에서 개발한 컴퓨터 보안 표준입니다. 정부 및 업계 실무 그룹이 암호화 모듈의 품질을 검증합니다. NIST Computer Security Resource Center 에서 공식 FIPS 발행물을 참조하십시오.

FIPS 140-3 표준을 사용하면 암호화 도구가 알고리즘이 올바르게 구현됩니다. 이에 대한 메커니즘 중 하나는 런타임 자체 검사입니다. 자세한 내용 및 FIPS 표준의 기타 사양은 FIPS PUB 140-3 의 전체 FIPS 140-3 표준을 참조하십시오.

규정 준수 요구 사항에 대한 자세한 내용은 Red Hat 정부 표준 페이지를 참조하십시오.

2.2. FIPS 모드가 활성화된 시스템 설치

FIPS(Federal Information Processing Standard) 발행 140-3에서 요구하는 암호화 모듈 자체 점검을 활성화하려면 시스템 설치 중에 FIPS 모드를 활성화합니다.

중요

나중에 FIPS 모드를 활성화하는 대신 FIPS 모드가 활성화된 RHEL을 설치하는 것이 좋습니다. 설치 중에 FIPS 모드를 활성화하면 시스템이 FIPS 승인 알고리즘 및 지속적인 모니터링 테스트로 모든 키를 생성합니다.

절차

- 시스템 설치 중에 커널 명령줄에 **fips=1** 옵션을 추가합니다.
  소프트웨어 선택 단계에서는 타사 소프트웨어를 설치하지 마십시오.

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

- 고급 RHEL 설치 수행에서 부팅 옵션 편집 섹션

2.3. 추가 리소스

- 시스템을 FIPS 모드로 전환
- 컨테이너에서 FIPS 모드 활성화

3장. 시스템 전체 암호화 정책 사용

시스템 전체 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

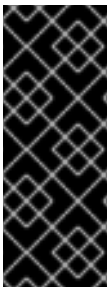
시스템 전체 정책이 설정되면 RHEL의 애플리케이션은 이를 따르며 애플리케이션을 명시적으로 요청하지 않는 한, 정책을 준수하지 않는 알고리즘과 프로토콜을 사용하지 않습니다. 즉, 이 정책은 시스템 제공 구성으로 실행할 때 애플리케이션의 기본 동작에 적용되지만 필요한 경우 이를 재정의할 수 있습니다.

RHEL 9에는 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	이 정책은 Red Hat Enterprise Linux 6 및 이전 버전과의 호환성을 극대화하며 공격 면적이 증가하여 보안이 떨어집니다. SHA-1은 TLS 해시, 서명 및 알고리즘으로 사용할 수 있습니다. CBC-mode 암호는 SSH와 함께 사용할 수 있습니다. GnuTLS를 사용하는 애플리케이션에서는 SHA-1로 서명된 인증서를 허용합니다. 이 보안 설정은 TLS 1.2 및 1.3 프로토콜과 IKEv2 및 SSH2 프로토콜을 허용합니다. RSA 키와 Diffie-Hellman 매개변수는 2048비트 이상인 경우 허용됩니다.
FUTURE	가까운 미래의 공격에도 견딜 수 있는 보수적인 보안 수준입니다. 이 수준은 DNSSec에서 SHA-1을 사용하거나 TPM로 사용할 수 없습니다. SHA2-224 및 SHA3-224 해시는 비활성화되어 있습니다. 128비트 암호가 비활성화되어 있습니다. Kerberos를 제외한 CBC 모드 암호는 비활성화되어 있습니다. 이 보안 설정은 TLS 1.2 및 1.3 프로토콜과 IKEv2 및 SSH2 프로토콜을 허용합니다. RSA 키와 Diffie-Hellman 매개변수는 최소 3072비트인 경우 허용됩니다.
FIPS	FIPS 140-2 요구 사항을 준수하는 정책 수준입니다. 이는 RHEL 시스템을 FIPS 모드로 전환하는 **fips-mode-setup** 도구에서 내부적으로 사용됩니다.

Red Hat은 LEGACY 정책을 사용하는 경우를 제외하고 모든 라이브러리가 안전한 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 라이프사이클 동안 특정 시스템과의 상호 운용성을 보장해야 하는 경우, 해당 시스템과 상호 작용하는 구성 요소의 암호화 정책에서 제외하거나 사용자 지정 정책을 사용하여 특정 알고리즘을 다시 활성화해야 합니다.



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 사용하십시오.

참고

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

암호화 정책 관리를 위한 툴
현재 시스템 전체 암호화 정책을 보거나 변경하려면 **update-crypto-policies** 도구를 사용합니다. 예를 들면 다음과 같습니다.

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

암호화 정책 변경이 적용되었는지 확인하려면 시스템을 다시 시작합니다.

안전하지 않은 암호화 제품군 및 프로토콜을 제거하여 강력한 암호화 기본값
다음 목록에는 Red Hat Enterprise Linux 9의 코어 암호화 라이브러리에서 제거된 암호화 제품군 및 프로토콜이 포함되어 있습니다. 소스에는 존재하지 않거나 빌드 중에 지원이 비활성화되므로 애플리케이션은 사용할 수 없습니다.

- DES (RHEL 7 이후)
- 모든 내보내기 등급 암호화 제품군 (RHEL 7 이후)
- 서명된 MD5 (RHEL 7 이후)
- SSLv2 (RHEL 7 이후)
- SSLv3 (RHEL 8 이후)
- 모든 ECC 곡선 < 224bit (RHEL 6 이후)
- 모든 바이너리 필드 ECC 곡선 (RHEL 6 이후)

모든 정책 수준에서 사용되지 않는 알고리즘
다음은 RHEL 9에 포함된 **LEGACY**,**DEFAULT**,**FUTURE** 및 **FIPS** 암호화 정책에서 비활성화되는 알고리즘은 다음과 같습니다. 사용자 지정 암호화 정책을 적용하거나 개별 애플리케이션의 명시적 구성을 통해서만 활성화할 수 있지만 결과 구성은 지원되는 것으로 간주되지 않습니다.

- 버전 1.2가 지난 TLS(RHEL 9의 경우)는 <RHEL 8의 1.0임)
- 버전 1.2보다 오래된 DTLS(RHEL 9의 경우)는 <RHEL 8의 1.0이었습니다)
- 매개 변수가 있는 DH < 2048bit (since RHEL 9)은 < 1024비트로 RHEL 8의 경우)
- RSA의 키 크기 < 2048bit (since RHEL 9)은 < 1024bit in RHEL 8)
- DSA(RHEL 9의 경우)는 <RHEL 8의 1024비트임)
- 3DES (RHEL 9 이후)
- RC4 (RHEL 9 이후)
- FFDHE-1024 (RHEL 9 이후)

- DHE-DSS (RHEL 9 이후)
- Camellia(RHEL 9 이후)
- ARIA
- IKEv1 (RHEL 8 이후)

crypto-policies 수준에서 활성화되는 알고리즘
다음 표에서는 선택한 알고리즘과 관련하여 네 가지 암호화 정책 수준을 모두 비교합니다.

	LEGACY	DEFAULT	FIPS	FUTURE
IKEv1	제공되지 않음	제공되지 않음	제공되지 않음	제공되지 않음
3DES	제공되지 않음	제공되지 않음	제공되지 않음	제공되지 않음
RC4	제공되지 않음	제공되지 않음	제공되지 않음	제공되지 않음
DH	최소 2048비트	최소 2048비트	최소 2048비트	최소 3072비트
RSA	최소 2048비트	최소 2048비트	최소 2048비트	최소 3072비트
DSA	제공되지 않음	제공되지 않음	제공되지 않음	제공되지 않음
TLS v1.1 이상	제공되지 않음	제공되지 않음	제공되지 않음	제공되지 않음
TLS v1.2 이상	제공됨	제공됨	제공됨	제공됨
디지털 서명 및 인증서의 SHA-1	제공됨	제공되지 않음	제공되지 않음	제공되지 않음
CBC 모드 암호	제공됨	제공되지 않음[a]	제공되지 않음[b]	제공되지 않음[c]
키 < 256비트인 대칭 암호	제공됨	제공됨	제공됨	제공되지 않음

[a] SSH에 대해 CBC 암호가 비활성화되어 있습니다

[b] Kerberos를 제외한 모든 프로토콜에 대해 CBC 암호가 비활성화되어 있습니다.

[c] Kerberos를 제외한 모든 프로토콜에 대해 CBC 암호가 비활성화되어 있습니다.

추가 리소스

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

3.2. 시스템 전체 암호화 정책을 이전 릴리스와 호환되는 모드로 전환

Red Hat Enterprise Linux 9의 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.

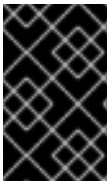
```
# update-crypto-policies --set LEGACY
Setting system policy to LEGACY
```

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.

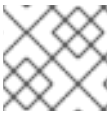
3.3. 시스템을 FIPS 모드로 전환

시스템 차원의 암호화 정책에는 암호화 모듈이 연방 정보 처리 표준(FIPS) Publication 140-3의 요구사항에 따라 자체 점검할 수 있는 정책 수준이 포함되어 있습니다. FIPS 모드를 활성화하거나 비활성화하는 **fips-mode-setup** 툴은 **FIPS** 시스템 전체 암호화 정책 수준을 내부적으로 사용합니다.



중요

나중에 FIPS 모드를 활성화하는 것과 달리 FIPS 모드가 활성화된 Red Hat Enterprise Linux 9를 설치하는 것이 좋습니다. 설치 중에 FIPS 모드를 활성화하면 시스템이 FIPS 승인 알고리즘 및 지속적인 모니터링 테스트로 모든 키를 생성합니다.



참고

RHEL 9의 암호화 모듈은 FIPS 140-3 요구 사항에 대해 아직 인증되지 않았습니다.

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

1. 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
- FIPS 모드에서 시스템 설치
- NIST(National Institute of Standards and Technology) 웹 사이트의 암호화 모듈에 대한 보안 요구 사항.

3.4. 컨테이너에서 FIPS 모드 활성화

FIPS 모드가 활성화된 시스템에서 **podman** 유틸리티는 컨테이너를 FIPS 모드로 자동으로 구성합니다. FIPS 모드가 아닌 시스템에서는 나중에 단일 명령을 사용하여 컨테이너를 FIPS 모드로 전환할 수 있습니다.

참고

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

참고

RHEL 9의 암호화 모듈은 FIPS 140-3 요구 사항에 대해 아직 인증되지 않았습니다.

사전 요구 사항

- 호스트 시스템은 FIPS 모드여야 합니다.

절차

- FIPS 모드로 전환하려는 컨테이너에서 다음 명령을 사용합니다.

```
# mount --bind /usr/share/crypto-policies/back-ends/FIPS /etc/crypto-policies/back-ends
```

추가 리소스

- 시스템을 FIPS 모드로 전환합니다.

- FIPS 모드에서 시스템 설치

3.5. FIPS 140-3와 호환되지 않는 암호화를 사용하는 RHEL 애플리케이션 목록

Red Hat은 FIPS 140-3와 같은 모든 관련 암호화 인증을 통과하고 RHEL 시스템 전체 암호화 정책을 따르기 때문에 핵심 암호화 구성 요소 세트의 라이브러리를 활용하는 것이 좋습니다.

핵심 암호화 구성 요소 개요, 선택 방법, 운영 체제에 통합 방법, 하드웨어 보안 모듈 및 스마트 카드를 지원하는 방법, 암호화 인증서가 적용되는 방법에 대한 정보는 RHEL 핵심 암호화 구성 요소 문서를 참조하십시오.

표 3.1. FIPS 140-3와 호환되지 않는 암호화를 사용하는 RHEL 8 애플리케이션 목록

애플리케이션	세부 정보
Bacula	CRAM-MD5 인증 프로토콜을 구현합니다.
Cyrus SASL	SCRAM-SHA-1 인증 방법을 사용합니다.
Dovecot	SCRAM-SHA-1을 사용합니다.
Emacs	SCRAM-SHA-1을 사용합니다.
FreeRADIUS	인증 프로토콜을 위해 MD5 및 SHA-1을 사용합니다.
Ghostscript	사용자 정의 암호화 구현(MD5, RC4, SHA-2, AES)은 문서를 암호화하고 해독합니다.
GRUB2	SHA-1이 필요한 레거시 펌웨어 프로토콜을 지원하며 **libgcrypt** 라이브러리를 포함합니다.
ipxe	TLS 스택을 구현합니다.
Kerberos	SHA-1(Windows와의 상호 운용성)에 대한 지원을 유지합니다.
lasso	**lasso_wsse_username_token_derive_key()** 키 파생 기능(KDF)은 SHA-1을 사용합니다.
MariaDB, MariaDB 커넥터	**mysql_native_password** 인증 플러그인은 SHA-1을 사용합니다.
MySQL	**mysql_native_password**는 SHA-1을 사용합니다.
OpenIPMI	RAKP-HMAC-MD5 인증 방법은 FIPS 사용에 대해 승인되지 않으며 FIPS 모드에서 작동하지 않습니다.
OVMF (UEFI 펌웨어), Edk2, shim	전체 암호화 스택(OpenSSL 라이브러리의 임베디드 사본).

애플리케이션	세부 정보
perl-CPAN	MD5 인증을 요약합니다.
perl-Digest-HMAC, perl-Digest-SHA	HMAC, HMAC-SHA1, SHA-1, SHA-224 등을 사용합니다.
perl-Mail-DKIM	서명자 클래스는 기본적으로 RSA-SHA1 알고리즘을 사용합니다.
PKCS #12 파일 처리 (OpenSSL, GnuTLS, NSS, Firefox, Java)	PKCS #12의 모든 용도는 FIPS와 호환되지 않습니다. 전체 파일 HMAC를 계산하는 데 사용되는 KDF(Key Derivation Function)는 FIPS 승인되지 않기 때문입니다. 따라서 PKCS #12 파일은 FIPS 준수를 위해 일반 텍스트로 간주됩니다. 키 전송 용도의 경우 FIPS 승인 암호화 체계를 사용하여 PKCS #12(.p12) 파일을 래핑합니다.
poppler	원본 PDF(예: MD5, RC4 및 SHA-1)에 있는 경우 허용되지 않은 알고리즘(예: MD5, RC4 및 SHA-1)을 기반으로 서명, 암호 및 암호화를 사용하여 PDF를 저장할 수 있습니다.
PostgreSQL	KDF는 SHA-1을 사용합니다.
QAT Engine	암호화 프리미티브의 혼합 하드웨어 및 소프트웨어 구현(RSA, EC, DH, AES, ...)
Ruby	비보안 MD5 및 SHA-1 라이브러리 기능을 제공합니다.
Samba	RC4 및 DES(Windows와의 상호 운용성)에 대한 지원을 유지합니다.
Syslinux	BIOS 암호는 SHA-1을 사용합니다.
unbound	DNS 사양을 사용하려면 DNSSEC 확인자가 DNSKEY 레코드에서 SHA-1 기반 알고리즘을 사용하여 유효성 검사를 수행해야 합니다.
valgrind	AES, SHA 해시.[a]

[a] ARM의 AES-NI 또는 SHA-1 및 SHA-2와 같은 소프트웨어 하드웨어 오프로드 작업에서 다시 구현합니다.

3.6. 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.

3.6.1. 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 클라이언트에 대한 시스템 전체 암호화 정책을 비활성화하려면 다음 작업 중 하나를 수행하십시오.

- 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 crypto 정책을 지정하며, 두 자리 숫자 접두사가 **50-redhat.conf** 파일보다 우선하며 **.conf** 접미사(예: **49-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 구성 을 참조하십시오.

추가 리소스

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

3.7. 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.

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 방법 에 대한 Red Hat 블로그 기사

3.8. SHA-1 다시 활성화

서명을 생성하고 확인하는 데 SHA-1 알고리즘을 사용하는 것은 **DEFAULT** 암호화 정책에서 제한됩니다. 시나리오에 기존 또는 타사 암호화 서명을 확인하는 데 SHA-1을 사용해야 하는 경우 RHEL 9에서 기본적으로 제공하는 **SHA1** 하위 정책을 적용하여 활성화할 수 있습니다. 시스템의 보안이 약화된다는 점에 유의하십시오.

사전 요구 사항

- 시스템은 **DEFAULT** 시스템 전체 암호화 정책을 사용합니다.

절차

1. **SHA1** 하위 정책을 **DEFAULT** 암호화 정책에 적용합니다.

```
# update-crypto-policies --set DEFAULT:SHA1
Setting system policy to DEFAULT:SHA1
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.
```

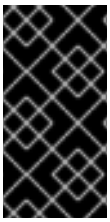
2. 시스템을 다시 시작하십시오.

```
# reboot
```

검증

- 현재 암호화 정책을 표시합니다.

```
# update-crypto-policies --show
DEFAULT:SHA1
```



중요

update-crypto-policies --set LEGACY 명령을 사용하여 LEGACY 암호화 정책으로 전환하면 서명에 SHA-1이 활성화됩니다. 그러나 **LEGACY** 암호화 정책은 다른 약한 암호화 알고리즘도 가능하게 함으로써 시스템이 훨씬 더 취약합니다. SHA-1 서명보다 다른 기존 암호화 알고리즘을 사용해야 하는 시나리오에만 이 해결 방법을 사용하십시오.

추가 리소스

- RHEL 9에서 RHEL 6 시스템으로 SSH가 KCS 문서 작동하지 않음
- SHA-1로 서명된 패키지는 KCS를 설치하거나 업그레이드할 수 없습니다 .

3.9. 사용자 정의 시스템 전체 암호화 정책 생성 및 설정

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

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 방법 에 대한 Red Hat 블로그 기사

4장. 시스템 전체에서 사용자 정의 암호화 정책 설정

관리자는 Cryptographic Policies RHEL 시스템 역할을 사용하여 Ansible Core 패키지를 사용하여 여러 시스템에서 사용자 지정 암호화 정책을 빠르고 일관되게 구성할 수 있습니다.

4.1. 암호화 정책 시스템 역할 변수 및 정보

암호화 정책 시스템 역할 플레이북에서는 환경 설정 및 제한 사항에 따라 암호화 정책 구성 파일의 매개 변수를 정의할 수 있습니다.

변수를 구성하지 않으면 시스템 역할은 시스템을 구성하지 않고 팩트만 보고합니다.

암호화 정책 시스템 역할에 대해 선택한 변수

crypto_policies_policy

시스템 역할이 관리되는 노드에 적용되는 암호화 정책을 결정합니다. 다양한 암호화 정책에 대한 자세한 내용은 시스템 전체 암호화 정책 을 참조하십시오.

crypto_policies_reload

yes로 설정되면 영향을 받는 서비스 (현재 **ipsec**,**bind**, **sshd** 서비스)로 설정된 경우 암호화 정책을 적용한 후 다시 로드합니다. 기본값은 **yes**입니다.

crypto_policies_reboot_ok

yes 로 설정하고 시스템 역할이 crypto 정책을 변경한 후 재부팅해야 하는 경우 **crypto_policies_reboot_required** 를 **yes** 로 설정합니다. 기본값은 **no**입니다.

암호화 정책 시스템 역할에 의해 설정된 팩트

crypto_policies_active

현재 선택한 정책을 나열합니다.

crypto_policies_available_policies

시스템에서 사용 가능한 모든 정책을 나열합니다.

crypto_policies_available_subpolicies

시스템에서 사용 가능한 모든 하위 정책을 나열합니다.

추가 리소스

- 사용자 지정 시스템 전체 암호화 정책 생성 및 설정

4.2. 암호화 정책 시스템 역할을 사용하여 사용자 정의 암호화 정책 설정

암호화 정책 시스템 역할을 사용하여 단일 제어 노드에서 일관되게 많은 수의 관리형 노드를 구성할 수 있습니다.

사전 요구 사항

- Policies 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.
  제어 노드에서 다음을 수행합니다.
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman**과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법 에 대한 지식베이스 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위에 대한 지식베이스 문서를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

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

```
---
- hosts: all
  tasks:
  - name: Configure crypto policies
    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 변수를 사용하면 시스템 역할에서 암호화 정책을 변경한 후 시스템이 재부팅됩니다.

자세한 내용은 Policies Policies System Role variables and facts 를 참조하십시오.

2. 선택 사항: 플레이북 구문을 확인합니다.

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

검증

1. 제어 노드에서 ***verify_playbook.yml***과 같은 다른 플레이북을 생성합니다.

```
- hosts: all
  tasks:
  - name: Verify active crypto policy
```

```
    include_role:
      name: rhel-system-roles.crypto_policies

  - debug:
      var: crypto_policies_active
```

이 플레이북은 시스템의 구성을 변경하지 않고 관리 노드의 활성 정책만 보고합니다.

2. 동일한 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file verify_playbook.yml

TASK [debug] ***************************
ok: [host] => {
    "crypto_policies_active": "FUTURE"
}
```

"crypto_policies_active": 변수는 관리 노드에서 정책을 활성으로 표시합니다.

4.3. 추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md** 파일
- **ansible-playbook(1)** 도움말 페이지.
- RHEL 시스템 역할 설치 .
- 시스템 역할 적용

5장. PKCS #11을 통해 암호화 하드웨어를 사용하도록 애플리케이션 구성

최종 사용자 인증 및 서버 애플리케이션의 HSM(하드웨어 보안 모듈)을 위한 스마트 카드 및 암호화 토큰과 같은 전용 암호화 장치에 시크릿 정보의 일부를 분리하면 추가 보안 계층이 제공됩니다. RHEL에서 PKCS #11 API를 통한 암호화 하드웨어 지원은 서로 다른 애플리케이션에서 일관성이 유지되며 암호화 하드웨어에서 시크릿을 격리하는 작업이 복잡하지 않습니다.

5.1. PKCS #11을 통한 하드웨어 지원

PKCS #11(Public-Key Cryptography Standard)은 암호화 정보를 유지하고 암호화 기능을 수행하는 장치를 암호화하기 위한 API(애플리케이션 프로그래밍 인터페이스)를 정의합니다. 이러한 장치는 토큰이라고 하며 하드웨어 또는 소프트웨어 형태로 구현할 수 있습니다.

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 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

- PKCS #11 URI 스키마
- 스마트 카드에 대한 액세스 제어

5.2. 스마트 카드에 저장된 SSH 키 사용

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

사전 요구 사항

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

절차

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

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

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

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

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

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

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

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

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

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

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

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

추가 리소스

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

5.3. 스마트 카드의 인증서를 사용하여 인증하도록 애플리케이션 구성

애플리케이션의 스마트 카드를 사용한 인증으로 보안이 향상되고 자동화가 간소화될 수 있습니다.

- **wget** 네트워크 다운로드자를 사용하면 로컬에 저장된 개인 키에 대한 경로 대신 PKCS #11 URI를 지정할 수 있으므로 안전하게 저장된 개인 키 및 인증서가 필요한 작업의 스크립트 생성을 간소화할 수 있습니다. 예를 들어 다음과 같습니다.

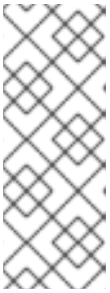
```
$ wget --private-key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --
certificate 'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
```

  자세한 내용은 **wget(1)** 도움말 페이지를 참조하십시오.

- **curl** 툴에서 사용할 PKCS #11 URI를 지정하는 것은 유사합니다.

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

  자세한 내용은 **curl(1)** 도움말 페이지를 참조하십시오.



참고

PIN은 스마트 카드에 저장된 키에 대한 액세스를 제어하는 보안 수단이고 구성 파일에 일반 텍스트 형식의 PIN이 포함되어 있으므로 공격자가 PIN을 읽지 못하도록 추가 보호를 고려하십시오. 예를 들어 **pin-source** 속성을 사용하여 **file:**을 제공할 수 있습니다. 파일에서 PIN을 읽기 위한 URI입니다. RFC 7512: PKCS #11 URI Scheme Query 특성 Semantics에서 자세한 내용을 확인하십시오. 명령 경로를 **pin-source** 속성 값으로 사용하는 것은 지원되지 않습니다.

- **Firefox** 웹 브라우저에서 **p11-kit-proxy** 모듈을 자동으로 로드합니다. 즉, 시스템에서 지원되는 모든 스마트 카드가 자동으로 감지됩니다. TLS 클라이언트 인증을 사용하려면 추가 설정이 필요하지 않으며 서버가 요청할 때 스마트 카드의 키가 자동으로 사용됩니다.

사용자 정의 애플리케이션에서 PKCS #11 URI 사용

애플리케이션에서 **GnuTLS** 또는 **NSS** 라이브러리를 사용하는 경우 PKCS #11 URI에 대한 지원은 PKCS #11에 대한 기본 지원으로 보장됩니다. 또한 **OpenSSL** 라이브러리에 의존하는 애플리케이션은 **openssl-pkcs11** 엔진으로 인해 암호화 하드웨어 모듈에 액세스할 수 있습니다.

스마트 카드에서 개인 키를 사용해야 하고 **NSS**,**GnuTLS** 및 **OpenSSL**을 사용하지 않는 애플리케이션에서는 **p11-kit**을 사용하여 PKCS #11 모듈 등록을 구현합니다.

추가 리소스

- **p11-kit(8)** 도움말 페이지.

5.4. 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 활용하려면 엔진 인터페이스를 통해 PKCS #11 모듈에 액세스할 수 있는 **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.

5.5. 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 지원은 **openssl-pkcs11** 엔진을 통과해야 합니다. **Nginx**는 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 URI에는 **engine:pkcs11:** 접두사가 필요합니다. 이는 다른 **pkcs11** 접두사가 엔진 이름을 참조하기 때문입니다.

5.6. 추가 리소스

- **pkcs11.conf(5)** 도움말 페이지.

6장. 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.

6.1. 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.

추가 리소스

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

6.2. 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.

6.3. 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.

6.4. 추가 리소스

- 스마트 카드에 대한 액세스 제어 Red Hat 블로그 문서.

7장. 공유 시스템 인증서 사용

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

7.1. 시스템 전체 신뢰 저장소

Red Hat Enterprise Linux에서 통합 시스템 전체 신뢰 저장소는 **/etc/pki/ca-trust/** 및 **/usr/share/pki/ca-trust-source/** 디렉토리에 있습니다. **/usr/share/pki/ca-trust-source/**의 신뢰 설정은 **/etc/pki/ca-trust/**의 설정보다 우선 순위가 낮습니다.

인증서 파일은 다음 디렉터리에 설치된 하위 디렉터리에 따라 처리됩니다.

- 신뢰할 수 있는 앵커의 경우
  - **/usr/share/pki/ca-trust-source/anchors/** 또는
  - **/etc/pki/ca-trust/source/anchors/**
- 신뢰할 수 없는 인증서의 경우
  - **/usr/share/pki/ca-trust-source/blacklist/** 또는
  - **/etc/pki/ca-trust/source/blacklist/**
- 확장된 BEGIN TRUSTED 파일 형식의 인증서의 경우
  - **/usr/share/pki/ca-trust-source/** 또는
  - **/etc/pki/ca-trust/source/**

참고

계층 암호화 시스템에서 신뢰 앵커는 다른 당사자가 신뢰할 수 있다고 생각하는 권한있는 엔티티입니다. X.509 아키텍처에서 루트 인증서는 신뢰 체인이 파생되는 신뢰 앵커입니다. 체인 검증을 사용하려면 신뢰할 수 있는 당사자가 신뢰 앵커에 먼저 액세스할 수 있어야 합니다.

7.2. 새 인증서 추가

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

사전 요구 사항

- **ca-certificates** 패키지가 시스템에 있습니다.

절차

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

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

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

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

참고

Firefox 브라우저는 **update-ca-trust** 를 실행하지 않고 추가된 인증서를 사용할 수 있지만 CA 변경 후 **update-ca-trust** 명령을 사용하는 것이 좋습니다. 또한 Firefox, Radphany 또는 Chromium, 캐시 파일 등의 브라우저와 같은 브라우저가 캐시를 지우거나 현재 시스템 인증서 구성을 로드하기 위해 브라우저를 다시 시작해야 할 수도 있습니다.

7.3. 신뢰할 수 있는 시스템 인증서 관리

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

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

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

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

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

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

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

pkcs11:id=%a6%b3%e1%2b%2b%49%b6%d7%73%a1%aa%94%f5%01%e7%73%65%4c%ac%50;type=cert
    type: certificate
    label: ACEDICOM Root
    trust: anchor
    category: authority
...
```

- 신뢰 앵커를 시스템 전체 신뢰 저장소에 저장하려면 **trust anchor** 하위 명령을 사용하고 인증서 경로를 지정합니다. *path.to/certificate.crt* 를 인증서 경로 및 해당 파일 이름으로 교체합니다.

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

- 인증서를 제거하려면 인증서의 경로 또는 인증서의 ID를 사용합니다.

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

추가 리소스

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

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

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

7.4. 추가 리소스

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

8장. 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.

8.1. RHEL의 구성 준수 도구

Red Hat Enterprise Linux는 완전히 자동화된 규정 준수 감사를 수행할 수 있는 툴을 제공합니다. 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 유틸리티를 포함하는 **OpenSCAP** 라이브러리는 로컬 시스템에서 구성 및 취약점 스캔을 수행하고 구성 규정 준수 콘텐츠의 유효성을 검사하고 이러한 스캔 및 평가를 기반으로 보고서 및 가이드를 생성하도록 설계되었습니다.
- SCAP Security Guide(SSG) - **scap-security-guide** 패키지는 Linux 시스템에 대한 최신 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.
- SCE **(Script Check Engine; 스크립트 검사 엔진)** - SCE는 관리자가 Bash, Python, Ruby와 같은 스크립팅 언어를 사용하여 보안 콘텐츠를 작성할 수 있는 SCAP 프로토콜의 확장입니다. 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 .

8.2. 취약점 검사

8.2.1. 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 Security 는 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 Insights 는 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

8.2.2. 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 프런트엔드 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/RHEL9/rhel-9.oval.xml.bz2 | bzip2 --decompress > rhel-9.oval.xml
```

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

```
# oscap oval eval --report vulnerability.html rhel-9.oval.xml
```

검증

- 선택한 브라우저의 결과를 확인합니다. 예를 들면 다음과 같습니다.

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

추가 리소스

- **oscap(8)** 도움말 페이지
- Red Hat OVAL 정의

8.2.3. 원격 시스템에서 취약점 스캔

SSH 프로토콜을 통해 **oscap-ssh** 도구를 사용하여 OpenSCAP 스캐너가 있는 취약점이 원격 시스템에 있는지 확인할 수도 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/RHEL9/rhel-9.oval.xml.bz2 | bzip2 --decompress > rhel-9.oval.xml
```

2. *machine1* 호스트 이름으로 원격 시스템을 스캔하고, 포트 22에서 SSH를 실행하고, *joesec* 사용자 이름을 사용하여 취약점을 검사하고 결과를 *remote-vulnerability.html* 파일에 저장합니다.

```
# oscap-ssh joesec@machine1 22 oval eval --report remote-vulnerability.html rhel-9.oval.xml
```

추가 리소스

- **oscap-ssh(8)**
- Red Hat OVAL 정의

8.3. 구성 규정 준수 검사

8.3.1. 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.

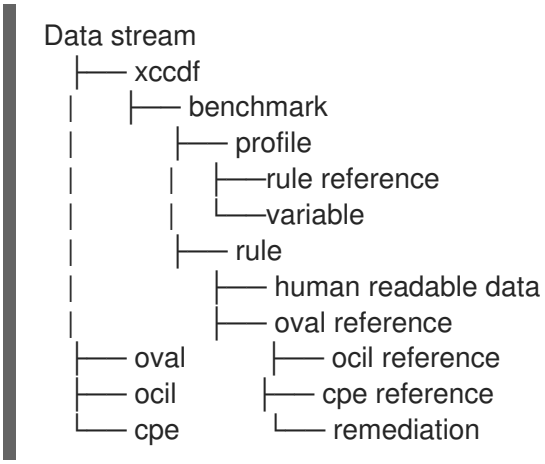
SCAP 보안 가이드 패키지는 SCAP 1.2 및 SCAP 1.3 표준을 준수하는 콘텐츠를 제공합니다. **openscap 스캐너** 유틸리티는 SCAP 보안 가이드 패키지에 제공된 SCAP 1.2 및 SCAP 1.3 콘텐츠와 호환됩니다.

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. RHEL에서는 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



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 맞춤형에 대한 자세한 내용은 SCAP Workbench를 사용하여 보안 프로필 사용자 지정 을 참조하십시오.

8.3.2. OpenSCAP 스캔의 가능한 결과

OpenSCAP 스캔에 적용되는 데이터 스트림 및 프로필과 시스템의 다양한 속성에 따라 각 규칙에서 특정 결과를 생성할 수 있습니다. 이 목록은 가능한 결과의 목록으로 의미에 대한 간략한 설명과 함께 제공됩니다.

표 8.1. OpenSCAP 스캔의 가능한 결과

결과	설명
통과	검사에서 이 규칙과의 충돌을 찾지 못했습니다.
실패	검사에서 이 규칙과 충돌하는 것을 발견했습니다.
확인되지 않음	OpenSCAP에서는 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. OpenSCAP은 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.

8.3.3. 구성 규정 준수 프로필 보기

검사 또는 해결을 위해 프로필을 사용하기 전에 나열하고 **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-rhel9-ds.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-rhel9-ds.xml
Profiles:
...
  Title: Australian Cyber Security Centre (ACSC) Essential Eight
    Id: xccdf_org.ssgproject.content_profile_e8
  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 9
    Id: xccdf_org.ssgproject.content_profile_pci-dss
...
```

3. data-stream 파일에서 프로필을 선택하고 선택한 프로필에 대한 추가 세부 정보를 표시합니다. 이렇게 하려면 **--profile** 옵션 다음에 이전 명령의 출력에 표시된 ID의 마지막 섹션과 함께 **oscap info** 를 사용합니다. 예를 들어 HIPPA 프로필의 ID는 **xccdf_org.ssgproject.content_profile_hipaa** 이고 **--profile** 옵션의 값은 **hipaa** 입니다.

```
$ oscap info --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
...
Profile
 Title: [RHEL9 DRAFT] Health Insurance Portability and Accountability Act (HIPAA)
 Id: xccdf_org.ssgproject.content_profile_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. The Security Rule requires appropriate administrative,
physical and technical safeguards to ensure the confidentiality, integrity, and security of
electronic protected health information.  This profile configures Red Hat Enterprise Linux 9 to
the HIPAA Security Rule identified for securing of electronic protected health information.
Use of this profile in no way guarantees or makes claims against legal compliance against
the HIPAA Security Rule(s).
```

추가 리소스

- **scap-security-guide(8)** 도움말 페이지

8.3.4. 특정 기준의 구성 준수 평가

시스템이 특정 기준을 준수하는지 확인하려면 다음 단계를 따르십시오.

사전 요구 사항

- **openscap-scanner** 및 **scap-security-guide** 패키지가 설치되어 있습니다.
- 시스템이 준수해야 하는 기준 내에서 프로필의 ID를 알고 있습니다. ID를 찾으려면 구성 규정 준수에 대한 프로필 보기를 참조하십시오.

절차

1. 시스템 규정 준수를 선택한 프로필로 평가하고 검사 결과를 report.html HTML 파일에 저장합니다. 예를 들면 다음과 같습니다.

```
$ sudo oscap xccdf eval --report report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

2. 선택 사항: **machine1** 호스트 이름으로 원격 시스템을 스캔하고, 포트 **22** 에서 SSH를 실행하고, **joesec** 사용자 이름을 사용하여 규정 준수를 검색하고 결과를 **remote-report.html** 파일에 저장합니다.

```
$ oscap-ssh joesec@machine1 22 xccdf eval --report remote_report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

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/guidesg-rhel9-guide-index.html** - **scap-security-guide-doc** 패키지를 사용하여 설치한 Red Hat Enterprise Linux 9의 보안 구성에 대한 보안 구성

8.4. 특정 기준선에 맞게 시스템 수정

특정 기준선에 맞게 RHEL 시스템을 수정하려면 다음 절차를 사용하십시오. 이 예에서는 HIPAA(Health Insurance Portability and Accountability Act) 프로필을 사용합니다.

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 RHEL 시스템에 설치되어 있습니다.

절차

1. **--remediate** 옵션과 함께 **oscap** 명령을 사용합니다.

```
$ sudo oscap xccdf eval --profile hipaa --remediate /usr/share/xml/scap/ssg/content/ssg-
rhel9-ds.xml
```

2. 시스템을 다시 시작합니다.

검증

1. HIPAA 프로파일로 시스템 규정 준수를 평가하고 **hipaa_report.html** 파일에 검사 결과를 저장합니다.

```
$ oscap xccdf eval --report hipaa_report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

추가 리소스

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

8.5. SSG ANSIBLE 플레이북을 사용하여 특정 기준과 일치하도록 시스템 수정

SCAP 보안 가이드 프로젝트의 Ansible 플레이북 파일을 사용하여 특정 기준으로 시스템을 해결하려면 다음 절차를 사용하십시오. 이 예에서는 HIPAA(Health Insurance Portability and Accountability Act) 프로필을 사용합니다.

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.
- **ansible-core** 패키지가 설치되어 있습니다. 자세한 내용은 Ansible 설치 가이드를 참조하십시오.

참고

RHEL 8.6 이상 버전에서 Ansible Engine은 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 9의 Red Hat Connector에는 해결 플레이북이 Ansible Core에서 작동하도록 하는 데 필요한 Ansible 모듈이 포함되어 있습니다.

절차

1. Ansible을 사용하여 HIPAA에 맞게 시스템을 교정합니다.

```
# ansible-playbook -i localhost, -c local /usr/share/scap-security-guide/ansible/rhel9-playbook-hipaa.yml
```

2. 시스템을 다시 시작합니다.

검증

1. HIPAA 프로파일로 시스템 규정 준수를 평가하고 **hipaa_report.html** 파일에 검사 결과를 저장합니다.

```
# oscap xccdf eval --profile hipaa --report hipaa_report.html /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

추가 리소스

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

8.6. 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 9에서 Ansible Engine은 기본 제공 모듈만 포함하는 **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 9.0의 Red Hat Connector에는 해결 플레이북이 Ansible Core에서 작동하도록 하는 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-rhel9-ds.xml
```

2. 이전 단계에서 생성한 파일을 기반으로 Ansible 플레이북을 생성합니다.

```
# oscap xccdf generate fix --fix-type ansible --profile hipaa --output hipaa-remediations.yml
hipaa-results.xml
```

3. **hipaa-remediations.yml** 파일에는 1 단계에서 수행한 검사 중에 실패한 규칙에 대한 Ansible 수정이 포함되어 있습니다. 생성된 이 파일을 검토한 후 **ansible-playbook hipaa-remediations.yml** 명령을 사용하여 적용할 수 있습니다.

검증

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

추가 리소스

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

8.7. 이후 애플리케이션에 대한 해결 BASH 스크립트 생성

이 절차를 사용하여 시스템을 HIPAA와 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-rhel9-ds.xml
```

2. 이전 단계에서 생성한 결과 파일을 기반으로 Bash 스크립트를 생성합니다.

```
# oscap xccdf generate fix --profile hipaa --fix-type bash --output hipaa-remediations.sh
hipaa-results.xml
```

3. **hipaa-remediations.sh** 파일에는 1단계에서 수행한 검사 중에 실패한 규칙 수정이 포함되어 있습니다. 생성된 이 파일을 검토한 후 이 파일과 동일한 디렉터리에 있는 경우 **./hipaa-remediations.sh** 명령을 사용하여 적용할 수 있습니다.

검증

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

추가 리소스

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

8.8. 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 형태로 보안 콘텐츠를 처리합니다.

8.8.1. SCAP Workbench를 사용하여 시스템 검사 및 수정

선택한 보안 정책에 대해 시스템을 평가하려면 다음 절차를 사용합니다.

사전 요구 사항

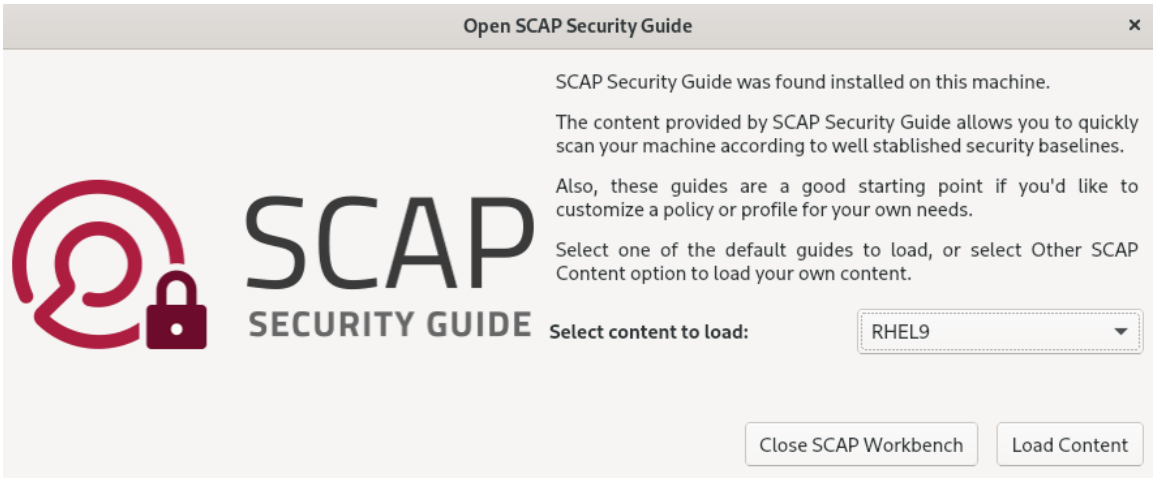
- **scap-workbench** 패키지가 시스템에 설치되어 있습니다.

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 &
```

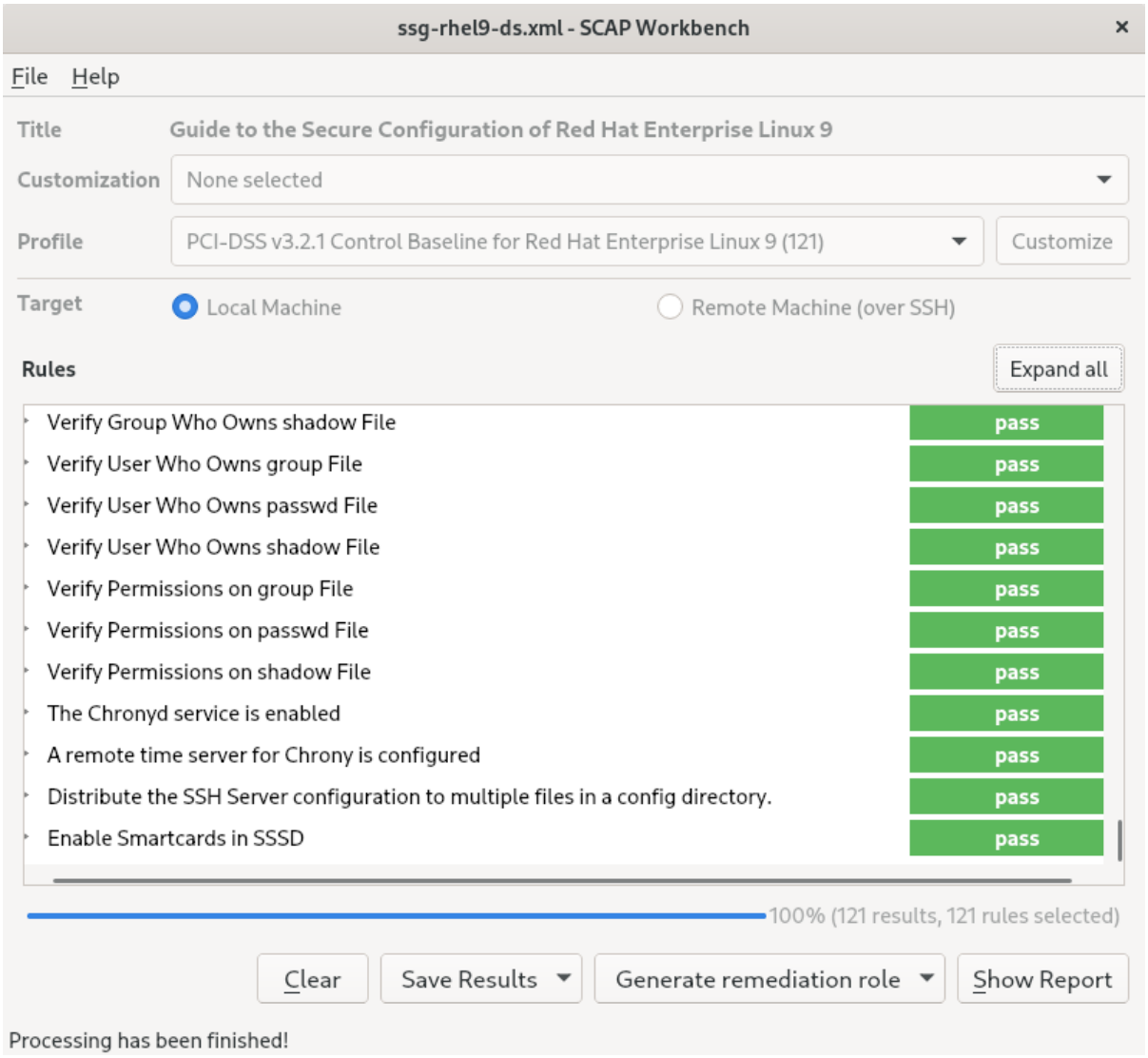
2. 다음 옵션 중 하나를 사용하여 보안 정책을 선택합니다.
   - 시작 창에서 **콘텐츠 로드** 버튼
   - **SCAP 보안 가이드에서 콘텐츠 열기**
   - **File** (파일) 메뉴에서 **기타 콘텐츠를 열고** 해당 XCCDF, SCAP RPM 또는 데이터 스트림 파일을 검색합니다.



3. **Remediate** 확인란을 선택하여 시스템 구성을 자동으로 수정할 수 있습니다. 이 옵션을 활성화하면 **SCAP Workbench**는 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.

8.8.2. 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.

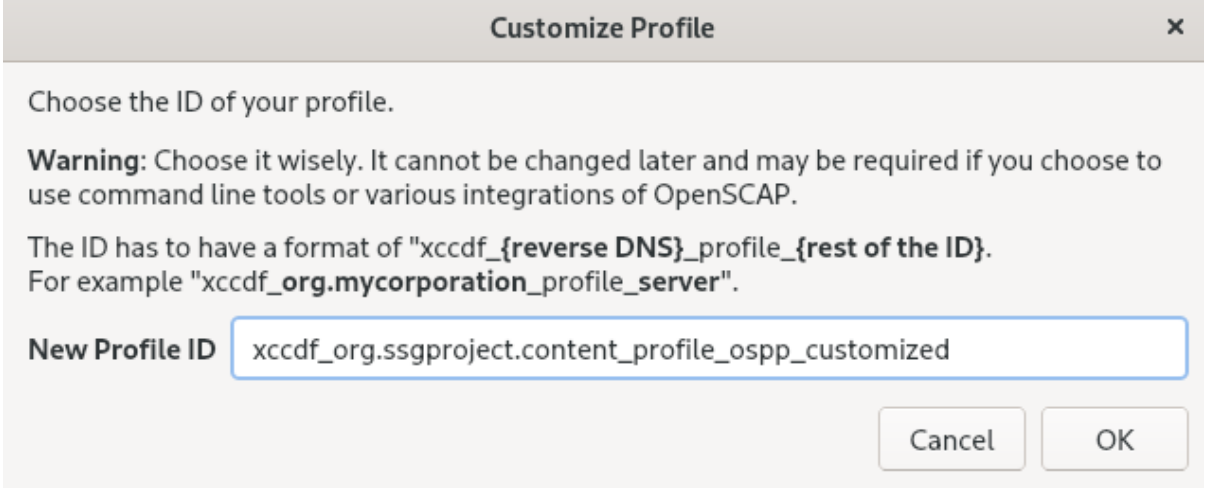
사전 요구 사항

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

절차

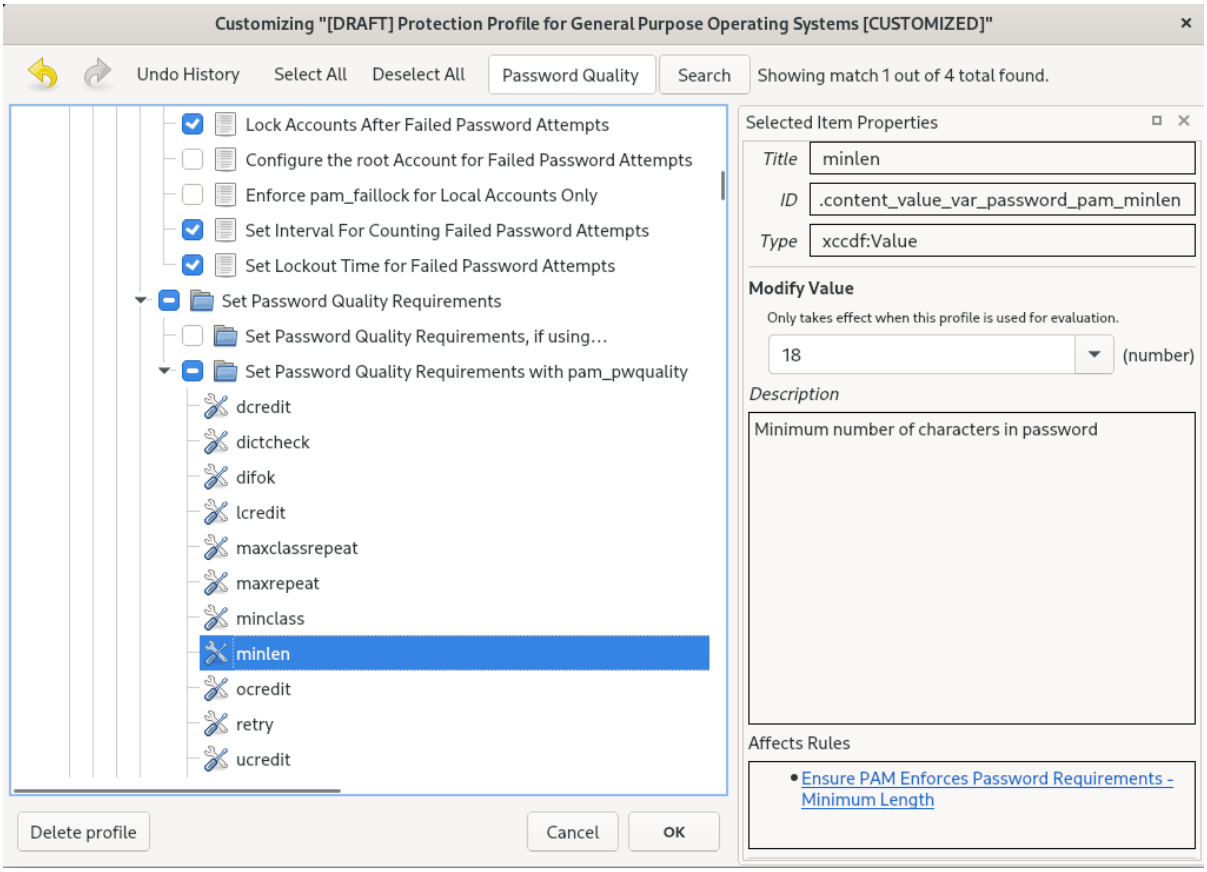
1. **SCAP Workbench**를 실행하고 **File** 메뉴에서 **Open content from SCAP Security Guide** 또는 **Open Other Content**를 열어 사용자 지정할 프로필을 선택합니다.

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 ID를 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.

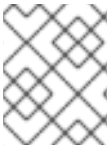
6. 변경 사항을 영구적으로 저장하려면 다음 옵션 중 하나를 사용합니다.

   - **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 Workbench**에 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.

8.8.3. 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

8.9. 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).

8.9.1. GUI에서 서버와 호환되지 않는 프로파일

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

표 8.2. GUI에서 서버와 호환되지 않는 프로파일

프로파일 이름	프로파일 ID	이유	참고
[DRAft] CIS Red Hat Enterprise Linux 9 벤치마크 for Level 2 - Server	**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.	

프로파일 이름	프로파일 ID	이유	참고
[DRAft] CIS Red Hat Enterprise Linux 9 벤치마크 for Level 1 - Server	**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.	
[RHEL9 DRAft] 범용 운영 체제용 보호 프로필	**xccdf_org.ssgproject.content_profile_ospp**	**nfs-utils** 패키지는 **GUI** 패키지가 설정된 서버의 일부이지만 정책을 제거해야 합니다.	BZ#1787156
[DRAFT] DISA STIG for Red Hat Enterprise Linux 9	**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.	DISA DASD **와 일치하는 GUI** 가 있는 서버로 RHEL 시스템을 설치하려면 BZ#1648162의 **GUI 프로파일과 함께 DISA** DASD를 사용할 수 있습니다.

8.9.2. 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 보안프로필은 **GUI** 기본 환경에서 서버에 포함된 확장 패키지 세트와 호환되지 않습니다. 자세한 내용은 GUI 서버와 호환되지 않는 프로파일을 참조하십시오.

사전 요구 사항

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

절차

1. **Installation Summary** 창에서 **Software Selection**을 클릭합니다. **Software Selection** 창이 열립니다.
2. **Base Environment** 창에서 **Server** 환경을 선택합니다. 기본 환경 하나만 선택할 수 있습니다.
3. **Done**을 클릭하여 설정을 적용하고 **Installation Summary** 창으로 돌아갑니다.
4. **Security Policy**를 클릭합니다. **Security Policy** 창이 열립니다.
5. 시스템에서 보안 정책을 활성화하려면 **Apply security policy**을 **ON**으로 전환합니다.
6. 프로필 창에서 General Purpose Operating Systems의 **Protection Profile for General Purpose Operating Systems**를 선택합니다.
7. **Select Profile**을 클릭하여 선택을 확인합니다.
8. **Changes that were done or need to be done**을 확인합니다. 나머지 수동 변경 사항을 완료합니다.
9. OSPP에는 충족해야 하는 엄격한 파티셔닝 요구 사항이 있으므로 **/boot**, **/home**, **/var**, **/var/log**, **/var/tmp**, **/var/log/audit**에 대한 별도의 파티션을 만듭니다.
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-rhel9-ds.xml
```

추가 리소스

- 수동 파티션 설정

8.9.3. Kickstart를 사용하여 기준 준수 RHEL 시스템 배포

특정 기준과 일치하는 RHEL 시스템을 배포하려면 다음 절차를 사용하십시오. 이 예에서는 OSDPP(General Purpose Operating System)에 보안 프로필을 사용합니다.

사전 요구 사항

- **scap-security-guide** 패키지는 RHEL 9 시스템에 설치됩니다.

절차

1. 선택한 편집기에서 **/usr/share/scap-security-guide/kickstart/ssg-rhel9-ospp-ks.cfg** Kickstart 파일을 엽니다.
2. 구성 요구 사항에 맞게 파티션 구성표를 업데이트합니다. OSPP 준수의 경우 **/boot, / home, / var,/var /log,/var/tmp**, **/var/log/audit** 에 대한 개별 파티션을 유지해야 하며 파티션의 크기만 변경할 수 있습니다.
3. Kickstart를 사용하여 자동 설치를 수행하는 방법 에 설명된 대로 Kickstart 설치를 시작합니다.

중요

Kickstart 파일의 암호는 OSPP 요구 사항에 대해 확인되지 않습니다.

검증

1. 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-rhel9-ds.xml
```

추가 리소스

- OSCAP Anaconda 애드온

8.10. 컨테이너 및 컨테이너 이미지에서 취약점 스캔

컨테이너 또는 컨테이너 이미지에서 보안 취약점을 찾으려면 다음 절차를 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/RHEL9/rhel-9.oval.xml.bz2 | bzip2 -
-decompress > rhel-9.oval.xml
```

2. 컨테이너 또는 컨테이너 이미지의 ID를 가져옵니다. 예를 들면 다음과 같습니다.

```
# podman images
REPOSITORY                          TAG     IMAGE ID      CREATED      SIZE
registry.access.redhat.com/ubi9/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-9.oval.xml
```

oscap-podman 명령에는 루트 권한이 필요하며 컨테이너 ID는 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.

8.11. 특정 기준에서 컨테이너 또는 컨테이너 이미지의 보안 준수 평가

다음 단계에 따라 OSPP(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 준수를 평가합니다.

사전 요구 사항

- **openscap-utils** 및 **scap-security-guide** 패키지가 설치되어 있습니다.

절차

1. 컨테이너 또는 컨테이너 이미지의 ID를 가져옵니다. 예를 들면 다음과 같습니다.

```
# podman images
REPOSITORY                              TAG      IMAGE ID       CREATED       SIZE
registry.access.redhat.com/ubi9/ubi     latest   096cae65a207   7 weeks ago   239 MB
```

2. HIPAA 프로필을 사용하여 컨테이너 이미지의 규정 준수를 평가하고 검사 결과를 *report.html* HTML 파일에 저장합니다.

```
# oscap-podman 096cae65a207 xccdf eval --report report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

보안 준수 여부를 OSPP 또는 PCI-DSS 기준으로 평가하면 *096cae65a207* 을 컨테이너 이미지의 ID로, *hipaa* 값을 *ospp* 또는 *pci-dss* 로 바꿉니다. **oscap-podman** 명령에는 root 권한이 필요합니다.

검증

- 선택한 브라우저의 결과를 확인합니다. 예를 들면 다음과 같습니다.

```
$ firefox report.html &
```

참고

*적용되지 않음*으로 표시된 규칙은 컨테이너화된 시스템에 적용되지 않는 규칙입니다. 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.

8.12. RHEL 9에서 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 RHEL 9에서 제공되는 프로필을 프로필과 일치하는 정책 버전과 함께 찾을 수 있습니다.

표 8.3. RHEL 9.0에서 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
[DRAft] CIS Red Hat Enterprise Linux 9 벤치마크 for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	초안[a]
[DRAft] CIS Red Hat Enterprise Linux 9 벤치마크 for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	DRAFT[a]
[DRAft] CIS Red Hat Enterprise Linux 9 벤치마크 for Level 1 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	DRAFT[a]
[DRAft] CIS Red Hat Enterprise Linux 9 벤치마크 for Level 2 - Workstation	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	DRAFT[a]

프로파일 이름	프로파일 ID	정책 버전
[DRAft] 소외 정보 시스템 및 조직(NIST 800-171)에서 분류되지 않은 정보	**xccdf_org.ssgproject.content_profile_cui**	r2
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
[DRAft] 범용 운영 체제용 보호 프로필	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
PCI-DSS v3.2.1 Red Hat Enterprise Linux 9용 Control Baseline	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
[DRAFT] DISA STIG for Red Hat Enterprise Linux 9	**xccdf_org.ssgproject.content_profile_stig**	초안[b]
[DRAft] Red Hat Enterprise Linux 9용 GUI를 사용한 DISA DASD	**xccdf_org.ssgproject.content_profile_stig_gui**	DRAFT[b]

[a] CIS는 아직 RHEL 9에 대한 공식 벤치마크를 게시하지 않았습니다.

[b] DISA는 아직 RHEL 9에 대한 공식적인 벤치마크를 게시하지 않았습니다.

8.13. 추가 리소스

- RHEL에서 지원되는 SCAP 보안 가이드 버전
- OpenSCAP 프로젝트 페이지 - OpenSCAP 프로젝트의 홈 페이지는 SCAP와 관련된 **oscap** 유틸리티 및 기타 구성 요소 및 프로젝트에 대한 자세한 정보를 제공합니다.
- SCAP Workbench 프로젝트 페이지 - SCAP Workbench 프로젝트의 홈 페이지는 **scap-workbench** 애플리케이션에 대한 자세한 정보를 제공합니다.
- SCAP Security Guide(SSG) 프로젝트 페이지 - Red Hat Enterprise Linux에 대한 최신 보안 콘텐츠를 제공하는 SSG 프로젝트의 홈 페이지입니다.
- Red Hat 보안 데모: 보안 규정 준수를 자동화할 수 있는 사용자 지정 보안 정책 콘텐츠 생성 - 업계 표준 보안 정책 및 사용자 지정 보안 정책을 모두 준수하기 위해 Red Hat Enterprise Linux에 포함된 툴을 사용하여 보안 규정 준수 자동화에 초기 환경을 제공하는 실습 랩입니다. 팀을 위해 교

육을 받거나 이러한 실습에 액세스하려면 Red Hat 계정 팀에 자세한 내용을 문의하십시오.

- Red Hat 보안 데모: RHEL Security Technologies로 사용자 보안 강화 - 실습 랩을 통해 OpenSCAP을 포함한 Red Hat Enterprise Linux에서 사용할 수 있는 주요 보안 기술을 사용하여 RHEL 시스템의 모든 수준에서 보안을 구현하는 방법을 배울 수 있습니다. 팀을 위해 교육을 받거나 이러한 실습에 액세스하려면 Red Hat 계정 팀에 자세한 내용을 문의하십시오.
- National Institute of Standards and Technology (NIST) SCAP 페이지 - 이 페이지는 SCAP 게시, 사양 및 SCAP 검증 프로그램을 포함한 SCAP 관련 자료의 광범위한 컬렉션을 나타냅니다.
- NVD(National Vulnerability Database) - 이 페이지는 SCAP 콘텐츠 및 기타 SCAP 표준 기반 취약점 관리 데이터의 가장 큰 리포지토리를 나타냅니다.
- Red Hat OVAL 콘텐츠 리포지토리 - Red Hat Enterprise Linux 시스템의 취약성에 대한 OVAL 정의가 포함된 리포지토리입니다. 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.

9장. AIDE로 무결성 확인

AIDE(Advanced Intrusion Detection Environment)는 시스템에 파일 데이터베이스를 생성한 다음 해당 데이터베이스를 사용하여 파일 무결성을 확인하고 시스템 침입을 탐지하는 유틸리티입니다.

9.1. AIDE 설치

AIDE를 설치하고 데이터베이스를 시작하려면 다음 단계가 필요합니다.

사전 요구 사항

- **AppStream** 리포지토리가 활성화되어 있어야 합니다.

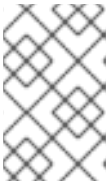
절차

1. *aide* 패키지를 설치하려면 다음을 수행합니다.

```
# dnf install aide
```

2. 초기 데이터베이스를 생성하려면 다음을 수행합니다.

```
# aide --init
```



참고

기본 구성에서 **aide --init** 명령은 **/etc/aide.conf** 파일에 정의된 디렉토리와 파일 집합만 확인합니다. **AIDE** 데이터베이스에 추가 디렉토리나 파일을 포함하고 감시된 매개 변수를 변경하려면 그에 따라 **/etc/aide.conf** 를 편집합니다.

3. 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
```

4. **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.

9.2. AIDE를 사용하여 무결성 검사 수행

사전 요구 사항

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

절차

1. 수동 검사를 시작하려면 다음을 수행합니다.

```
# aide --check
Start timestamp: 2018-07-11 12:41:20 +0200 (AIDE 0.16)
AIDE found differences between database and filesystem!!
```

```
...
[trimmed for clarity]
```

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

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

9.3. AIDE 데이터베이스 업데이트

패키지 업데이트 또는 구성 파일 조정과 같은 시스템 변경 사항을 확인한 후 기본 **AIDE** 데이터베이스를 업데이트할 것을 권장합니다.

사전 요구 사항

- **AIDE** 가 제대로 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.

9.4. 파일 통합 툴: AIDE 및 IMA

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

표 9.1. 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.

질문	AIDE(Advanced Intrusion Detection Environment)	무결성 측정 아키텍처(IMA)
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.

9.5. 추가 리소스

- **AIDE(1)** 도움말 페이지
- 커널 무결성 하위 시스템

10장. LUKS를 사용하여 블록 장치 암호화

디스크 암호화는 이를 암호화하여 블록 장치의 데이터를 보호합니다. 장치의 암호 해독된 콘텐츠에 액세스하려면 사용자가 암호 또는 키를 인증으로 제공해야 합니다. 이는 모바일 컴퓨터와 이동식 미디어와 관련해서는 특히 중요합니다. 이 미디어는 시스템에서 물리적으로 제거된 경우에도 장치의 콘텐츠를 보호하는 데 도움이 됩니다. LUKS 형식은 RHEL에서 블록 장치 암호화의 기본 구현입니다.

10.1. LUKS 디스크 암호화

Linux Unified Key Setup-on-disk-format(LUKS)을 사용하면 블록 장치를 암호화할 수 있으며 암호화된 장치 관리를 간소화하는 도구 세트를 제공합니다. LUKS는 여러 사용자 키를 사용하여 파티션의 대량 암호화에 사용되는 마스터 키를 해독할 수 있습니다.

RHEL은 LUKS를 활용하여 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 해제"합니다. 기본 파티션 테이블을 수정하도록 선택하는 경우 암호화할 파티션을 선택할 수 있습니다. 이는 파티션 테이블 설정에서 설정됩니다.

LUKS의 기능

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

LUKS가 수행하지 *않는* 내용

- LUKS와 같은 디스크 암호화 솔루션은 시스템이 꺼진 경우에만 데이터를 보호합니다. 시스템이 있고 LUKS가 디스크의 암호를 해독하면 해당 디스크의 파일은 일반적으로 액세스할 수 있는 모든 사용자가 사용할 수 있습니다.
- LUKS는 많은 사용자가 동일한 장치에 대해 고유한 액세스 키를 사용해야 하는 시나리오에는 적합하지 않습니다. LUKS1 형식은 8개의 키 슬롯, LUKS2 최대 32개의 키 슬롯을 제공합니다.
- LUKS는 파일 수준 암호화가 필요한 애플리케이션에 적합하지 않습니다.

암호화

LUKS에 사용되는 기본 암호는 **aes-xts-plain64** 입니다. LUKS의 기본 키 크기는 512비트입니다. **Anaconda** (XTS 모드)가 있는 LUKS의 기본 키 크기는 512비트입니다. 사용 가능한 암호는 다음과 같습니다.

- AES - 고급 암호화 표준
- Twofish (128비트 블록 암호)

- Serpent

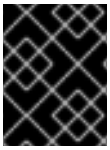
추가 리소스

- LUKS 프로젝트 홈 페이지
- LUKS On-Disk 형식 사양
- FIPS PUB 197

10.2. RHEL의 LUKS 버전

RHEL에서 LUKS 암호화의 기본 형식은 LUKS2입니다. 레거시 LUKS1 형식은 완벽하게 지원되며 이전 RHEL 릴리스와 호환되는 형식으로 제공됩니다.

LUKS2 형식은 바이너리 구조를 수정할 필요 없이 다양한 부분을 나중에 업데이트할 수 있도록 설계되었습니다. LUKS2는 내부적으로 메타데이터에 JSON 텍스트 형식을 사용하고, 메타데이터의 중복성을 제공하고, 메타데이터 손상을 감지하며, 메타데이터 사본의 자동 복구를 허용합니다.



중요

LUKS1만 지원하는 기존 시스템과 호환되어야 하는 시스템에서 LUKS2를 사용하지 마십시오. RHEL 7은 버전 7.6 이후의 LUKS2 형식을 지원합니다.

주의

LUKS2 및 LUKS1은 다른 명령을 사용하여 디스크를 암호화합니다. LUKS 버전에 잘못된 명령을 사용하면 데이터가 손실될 수 있습니다.

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

LUKS2 형식은 LUKS1에서 영향을 받습니다. 특정 상황에서 LUKS1을 LUKS2로 변환할 수 있습니다. 다음 시나리오에서는 변환이 특히 불가능합니다.

- LUKS1 장치는 PBD - Clevis(정책 기반 암호 해독) 솔루션에서 사용하는 것으로 표시됩니다. **cryptsetup** 툴은 일부 **luksmeta** 메타데이터가 감지되면 장치를 변환하는 것을 거부합니다.
- 장치가 활성 상태입니다. 변환이 가능하려면 장치가 비활성 상태여야 합니다.

10.3. LUKS2 재암호화 중에 데이터 보호 옵션

LUKS2는 재암호화 프로세스 중에 성능 또는 데이터 보호 우선 순위를 지정하는 여러 가지 옵션을 제공합니다.

checksum

이는 기본값 모드입니다. 데이터 보호 및 성능 균형을 유지합니다.
이 모드에서는 재암호화 영역에 섹터의 개별 체크섬을 저장하므로 복구 프로세스에서 이미 다시 암호화한 LUKS2 섹터를 감지할 수 있습니다. 이 모드에서는 블록 장치 섹터 쓰기가 atomic이어야 합니다.

journal

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

none

이 모드는 성능에 우선 순위를 지정하며 데이터 보호를 제공하지 않습니다. 이는 **SIGTERM** 신호 또는 **Ctrl+C** 를 누른 사용자와 같은 안전한 프로세스 종료로부터만 데이터를 보호합니다. 예기치 않은 시스템 충돌 또는 애플리케이션 충돌로 인해 데이터가 손상될 수 있습니다.

cryptsetup 의 **--resilience** 옵션을 사용하여 모드를 선택할 수 있습니다.

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

- 다음 LUKS2 장치의 열려 있는 작업 중 자동으로 수행됩니다. 이 작업은 **cryptsetup open** 명령에 의해 트리거되거나 **systemd-cryptsetup** 으로 장치를 연결하여 트리거됩니다.
- 수동으로 LUKS2 장치에서 **cryptsetup repair** 명령을 사용합니다.

10.4. LUKS2를 사용하여 블록 장치의 기존 데이터 암호화

이 절차에서는 LUKS2 형식을 사용하여 아직 암호화되지 않은 장치에서 기존 데이터를 암호화합니다. 새 LUKS 헤더가 장치의 헤드에 저장됩니다.

사전 요구 사항

- 블록 장치에는 파일 시스템이 포함됩니다.
- 데이터를 백업했습니다.

주의

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

절차

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

```
# umount /dev/sdb1
```

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

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

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

   - **parted** 와 같은 파티션 관리 도구를 사용하여 파티션을 확장합니다.
   - 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. 예를 들어 다음과 같습니다.

```
# cryptsetup reencrypt \
        --encrypt \
        --init-only \
        --reduce-device-size 32M \
        /dev/sdb1 sdb1_encrypted
```

명령은 암호를 요청하고 암호화 프로세스를 시작합니다.

4. 장치를 마운트합니다.

```
# mount /dev/mapper/sdb1_encrypted /mnt/sdb1_encrypted
```

5. 온라인 암호화를 시작합니다.

```
# cryptsetup reencrypt --resume-only /dev/sdb1
```

추가 리소스

- **cryptsetup(8)**, **lvextend(8)**, **resize2fs(8)** 및 **parted(8)** 도움말 페이지

10.5. 분리된 헤더로 LUKS2를 사용하여 블록 장치에서 기존 데이터 암호화

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

사전 요구 사항

- 블록 장치에는 파일 시스템이 포함됩니다.
- 데이터를 백업했습니다.

주의

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

절차

1. 장치의 모든 파일 시스템을 마운트 해제합니다. 예를 들어 다음과 같습니다.

```
# umount /dev/sdb1
```

2. 암호화를 초기화합니다.

```
# cryptsetup reencrypt \
        --encrypt \
        --init-only \
        --header /path/to/header \
        /dev/sdb1 sdb1_encrypted
```

*/path/to/header*를 파일 경로로 교체하고 분리된 LUKS 헤더로 바꿉니다. LUKS 분리 헤더는 암호화된 장치를 나중에 잠금 해제할 수 있도록 액세스할 수 있어야 합니다.

명령은 암호를 요청하고 암호화 프로세스를 시작합니다.

3. 장치를 마운트합니다.

```
# mount /dev/mapper/sdb1_encrypted /mnt/sdb1_encrypted
```

4. 온라인 암호화를 시작합니다.

```
# cryptsetup reencrypt --resume-only --header /path/to/header /dev/sdb1
```

추가 리소스

- **cryptsetup(8)** 매뉴얼 페이지

10.6. LUKS2를 사용하여 빈 블록 장치 암호화

이 절차에서는 LUKS2 형식을 사용하여 빈 블록 장치를 암호화하는 방법에 대한 정보를 제공합니다.

사전 요구 사항

- 빈 블록 장치입니다.

절차

1. 파티션을 암호화된 LUKS 파티션으로 설정합니다.

```
# cryptsetup luksFormat /dev/sdb1
```

2. 암호화된 LUKS 파티션을 엽니다.

```
# cryptsetup open /dev/sdb1 sdb1_encrypted
```

이렇게 하면 파티션 잠금 해제하고 장치 매퍼를 사용하여 새 장치에 매핑됩니다. 이 경고는 **device**가 암호화된 장치이므로 암호화된 데이터를 덮어쓰지 않도록 ***/dev/mapper/device_mapped_name*** 을 사용하여 LUKS를 통해 처리해야 합니다.

3. 암호화된 데이터를 파티션에 작성하려면 장치 매핑된 이름을 통해 액세스해야 합니다. 이렇게 하려면 파일 시스템을 생성해야 합니다. 예를 들어 다음과 같습니다.

```
# mkfs -t ext4 /dev/mapper/sdb1_encrypted
```

4. 장치를 마운트합니다.

```
# mount /dev/mapper/sdb1_encrypted mount-point
```

추가 리소스

- **cryptsetup(8)** 매뉴얼 페이지

10.7. 스토리지 시스템 역할을 사용하여 LUKS 암호화된 볼륨 생성

스토리지 역할을 사용하여 Ansible 플레이북을 실행하여 LUKS로 암호화된 볼륨을 생성하고 구성할 수 있습니다.

사전 요구 사항

- Policies 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.
  제어 노드에서 다음을 수행합니다.
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman**과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법 에 대한 지식베이스 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위에 대한 지식베이스 문서를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

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

```
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
         - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: your-password
  roles:
   - rhel-system-roles.storage
```

2. 선택 사항: 플레이북 구문을 확인합니다.

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

추가 리소스

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

11장. 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 *pins*라는 플러그인으로 구성되어 있습니다. 각 핀은 별도의 잠금 해제 기능을 제공합니다. 현재 다음 핀을 사용할 수 있습니다.

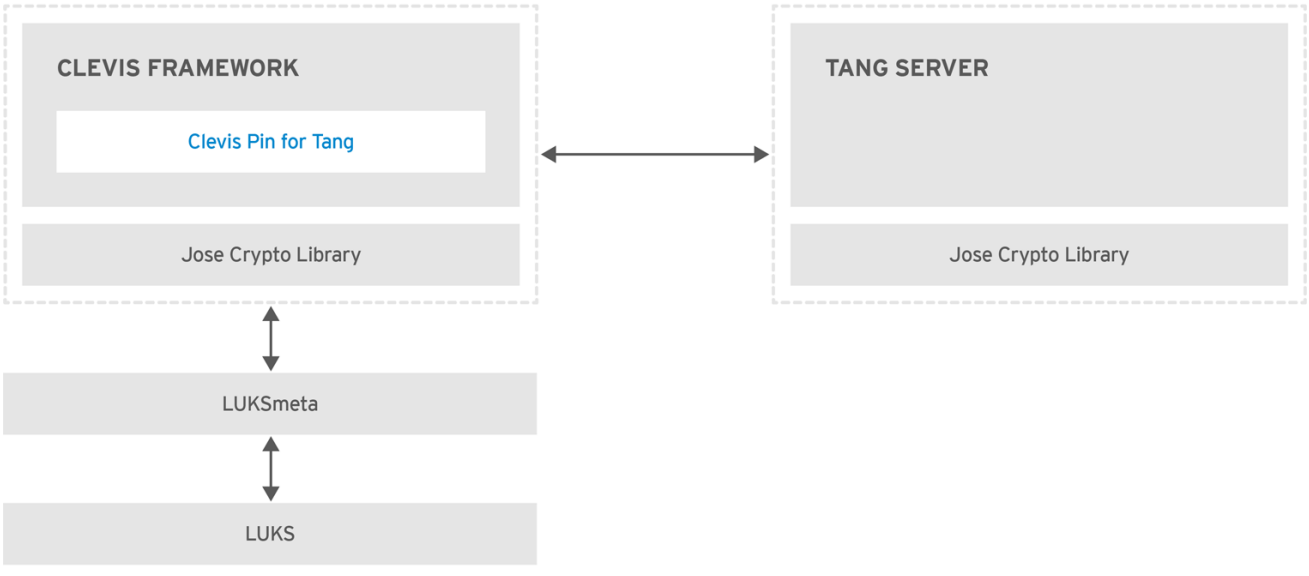
- **Tang** - 네트워크 서버를 사용하여 볼륨 잠금 해제 가능
- **tpm2** - TPM2 정책을 사용하여 볼륨 잠금 해제 가능
- **sss** - Shamir의 Secret Sharing (SSS) 암호화 체계를 사용하여 고가용성 시스템을 배포 가능

NBDE(Network Bound Disc Encryption)는 암호화된 볼륨을 특수 네트워크 서버에 바인딩할 수 있는 PBD의 하위 기능입니다. Clevis의 현재 구현에는 Tang 서버 및 Tang 서버 자체에 대한 Clevis 핀이 포함되어 있습니다.

11.1. 네트워크 바인딩 디스크 암호화

Red Hat Enterprise Linux에서 Clevis는 다음과 같은 구성 요소 및 기술을 통해 구현됩니다.

그림 11.1. LUKS1 암호화 볼륨을 사용할 때의 NBDE 스키마입니다 luksmeta 패키지는 LUKS2 볼륨에 사용되지 않습니다.



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

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

*Clevis PIN*은 Clevis 프레임워크의 플러그인입니다. 이러한 핀 중 하나는 DASD 서버 - Tang과의 상호 작용을 구현하는 플러그인입니다.

Clevis 및 Tang은 네트워크 바인딩 암호화를 제공하는 일반 클라이언트 및 서버 구성 요소입니다. Red Hat Enterprise Linux에서는 LUKS와 함께 네트워크 Bound 디스크 암호화를 수행하기 위해 루트 및 루트가 아닌 스토리지 볼륨을 암호화 및 암호 해독하는 데 사용됩니다.

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 symmetric이므로 Tang의 공개 키 목록을 대역에서 배포할 수 있으므로 클라이언트가 Tang 서버에 액세스하지 않고도 작동할 수 있습니다. 이 모드를 *오프라인 프로비저닝* 이라고 합니다.

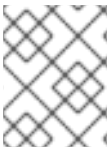
Tang의 Clevis 핀은 공개 키 중 하나를 사용하여 고유하고 암호화 방식으로 암호화 키를 생성합니다. 이 키를 사용하여 데이터를 암호화하면 키가 삭제됩니다. Clevis 클라이언트는 이 프로비저닝 작업에서 생성한 상태를 편리한 위치에 저장해야 합니다. 데이터를 암호화하는 이 프로세스는 *프로비저닝 단계*입니다.

LUKS 버전 2(LUKS2)는 RHEL의 기본 disk-encryption 형식입니다. 따라서 Clevis의 프로비저닝 상태는 LUKS2 헤더에 토큰으로 저장됩니다. **luksmeta** 패키지는 LUKS1로 암호화된 볼륨에만 사용할 수 있습니다.

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 단계*입니다.

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.

11.2. 암호화 클라이언트 설치 - CLEVIS

이 절차를 사용하여 시스템에서 Clevis 플러그형 프레임워크를 배포하고 사용하기 시작합니다.

절차

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

```
# dnf 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
```

11.3. 강제 모드에서 SELINUX를 사용하여 TANG 서버 배포

다음 절차에 따라 사용자 지정 포트에서 실행되는 Tang 서버를 SELinux 강제 모드의 제한된 서비스로 배포합니다.

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.

```
# dnf 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 **ValueError를 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 80에서 이전에 선택한 숫자로 변경합니다.

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

파일을 저장하고 편집기를 종료합니다.

7. 변경된 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

8. 구성이 작동하는지 확인합니다.

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

9. **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

11.4. 클라이언트에서 TANG 서버 키 교체 및 바인딩 업데이트

다음 단계를 사용하여 Tang 서버 키를 교체하고 클라이언트에서 기존 바인딩을 업데이트합니다. 교체해야 하는 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 설정 을 참조하십시오.

사전 요구 사항

- Tang 서버가 실행 중입니다.
- **clevis** 및 **clevis-luks** 패키지가 클라이언트에 설치됩니다.

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

11.5. 웹 콘솔에서 TANG 키를 사용하여 자동 잠금 해제 구성

Tang 서버에서 제공하는 키를 사용하여 LUKS 암호화 스토리지 장치의 자동 잠금 해제를 구성합니다.

사전 요구 사항

- RHEL 9 웹 콘솔이 설치되어 있어야 합니다.
  자세한 내용은 웹 콘솔 설치를 참조하십시오.
- **cockpit-storaged** 패키지가 시스템에 설치됩니다.
- **cockpit.socket** 서비스는 포트 9090에서 실행됩니다.
- **clevis**,**tang**, **clevis-dracut** 패키지가 설치됩니다.
- Tang 서버가 실행 중입니다.

절차

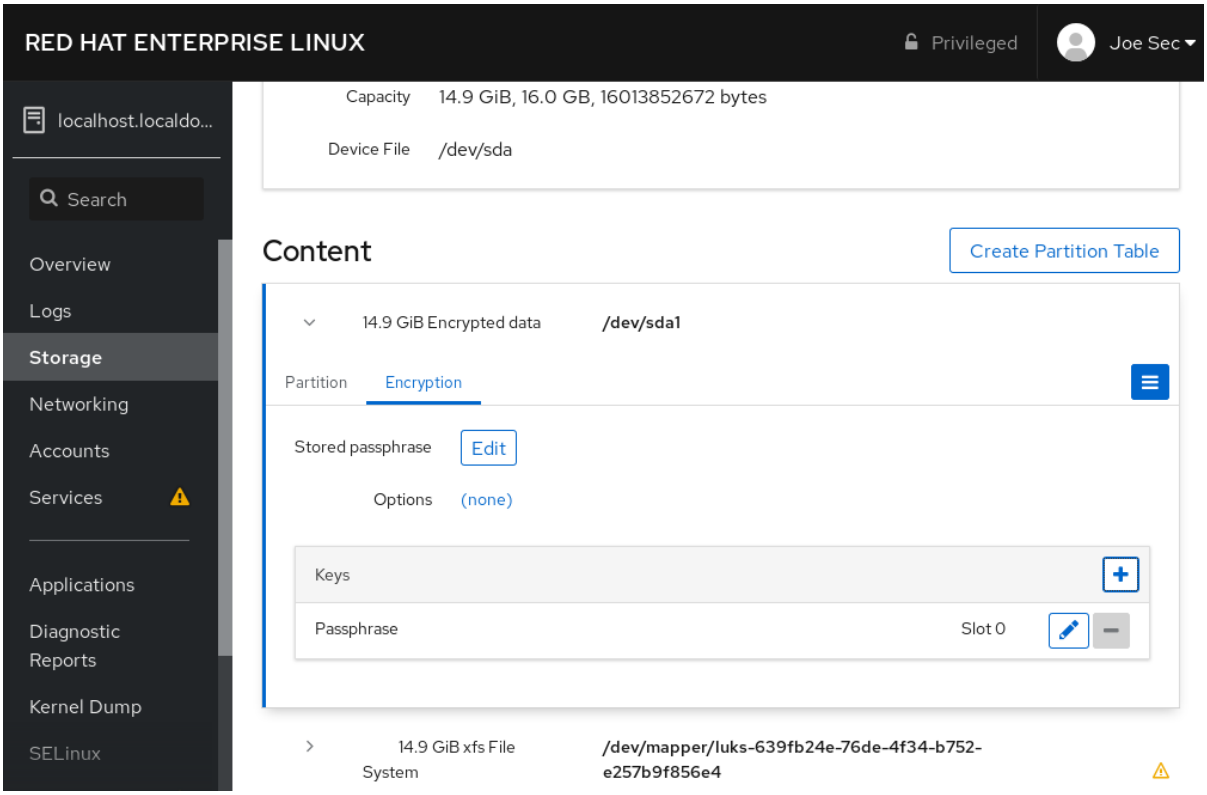
1. 웹 브라우저에 다음 주소를 입력하여 RHEL 웹 콘솔을 엽니다.

   ```
   https://localhost:9090
   ```

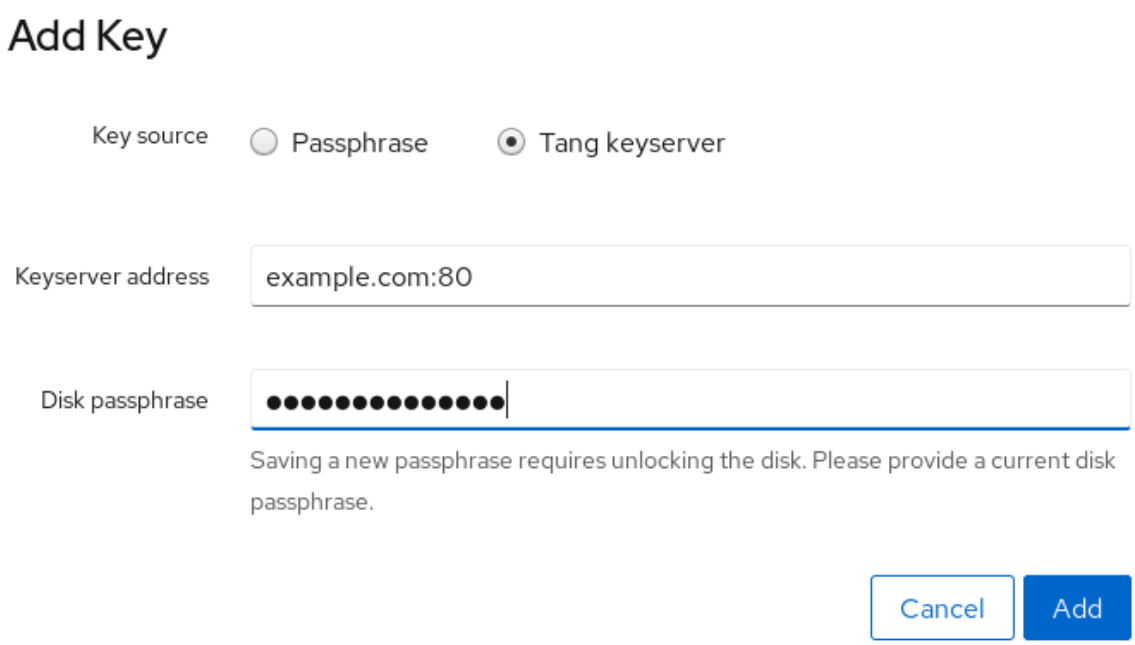
   원격 시스템에 연결할 때 *localhost* 부분을 원격 서버의 호스트 이름 또는 IP 주소로 바꿉니다.

2. 인증 정보를 제공하고 **스토리지**를 클릭합니다. Tang 서버를 사용하여 잠금 해제하려는 암호화된 장치의 세부 정보를 확장하려면 **>**을 클릭하고 **암호화**를 클릭합니다.

3. **키** 섹션에서 **+**를 클릭하여 Tang 키를 추가합니다.



4. Tang 서버의 주소와 LUKS 암호화 장치를 잠금 해제하는 암호를 제공합니다. **추가**를 클릭하여 확인합니다.

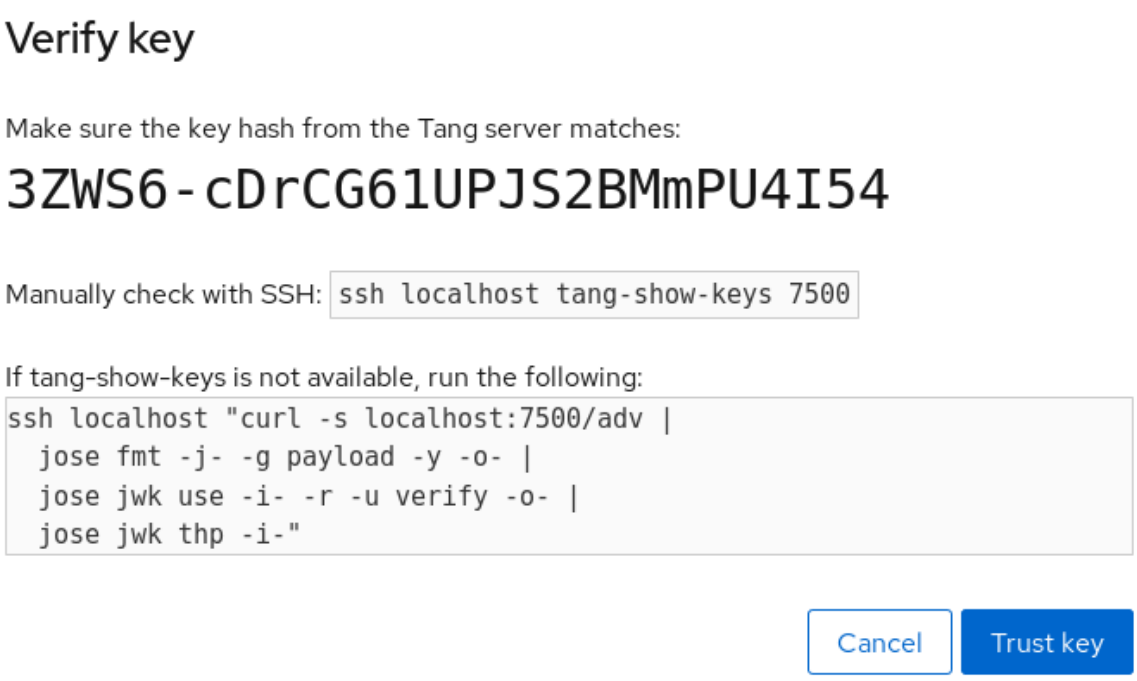


다음 대화 상자 창에서 키 해시가 일치하는지 확인하는 명령을 제공합니다.

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

```
# tang-show-keys 7500
fM-EwYeiTxS66X3s1UAywsGKGnxnpll8ig0KOQmr9CM
```

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



7. 초기 부팅 시스템이 디스크 바인딩을 처리할 수 있도록 하려면 왼쪽 탐색 모음 하단에 있는 **터미널**을 클릭하고 다음 명령을 입력합니다.

```
# dnf install clevis-dracut
# grubby --update-kernel=ALL --args="rd.neednet=1"
# dracut -fv --regenerate-all
```

8. Clevis를 활성화하여 부팅 프로세스 후반에 마운트되는 볼륨도 잠금 해제하려면 시스템을 재시작하기 전에 클라이언트에서 다음 명령을 사용하십시오.

```
# systemctl enable clevis-luks-askpass.path
```

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
clevis
clevis-pin-sss
clevis-pin-tang
clevis-pin-tpm2
-rwxr-xr-x   1 root     root        1600 Feb 11 16:30 usr/bin/clevis
-rwxr-xr-x   1 root     root        1654 Feb 11 16:30 usr/bin/clevis-decrypt
...
-rwxr-xr-x   2 root     root          45 Feb 11 16:30 usr/lib/dracut/hooks/initqueue/settled/60-
clevis-hook.sh
-rwxr-xr-x   1 root     root        2257 Feb 11 16:30 usr/libexec/clevis-luks-askpass
```

추가 리소스

- RHEL 웹 콘솔을 사용하여 시작하기

11.6. 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 Clevis에서 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* URL을 **tang** 이 설치된 서버의 URL과 일치하도록 변경합니다. *secret.jwe* 출력 파일에는 JWE 형식의 암호화된 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* 파일에 있는 광고를 사용하십시오.

```
$ 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 0과 7로 PCR에 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 PCR의 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, KCS를 다시 시작한 후 Clevis TPM2의 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
...
```

11.7. LUKS 암호화 볼륨 수동 등록 구성

다음 단계를 사용하여 Clevis로 LUKS 암호화된 볼륨의 잠금을 구성합니다.

사전 요구 사항

- Tang 서버가 실행 중이고 사용 가능합니다.

절차

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

```
# dnf 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 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:
```

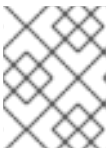
이 명령은 다음 네 가지 단계를 수행합니다.

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. Clevis를 활성화하여 부팅 프로세스 후반에 마운트되는 볼륨도 잠금 해제하려면 시스템을 재시작하기 전에 클라이언트에서 다음 명령을 사용하십시오.

```
# systemctl enable clevis-luks-askpass.path
```

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

```
# dnf install clevis-dracut
```

RHEL에서 Clevis는 호스트별 구성 옵션 없이 일반 **initrd** (초기 램디스크)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"
```

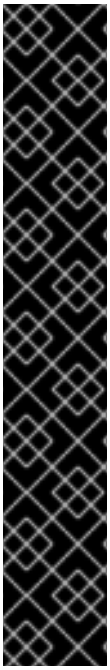
그런 다음 **--hostonly-cmdline** 없이 **dracut** 을 사용할 수 있습니다.

```
# 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 DASD를 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"
```

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"
```

초기 RAM 디스크 이미지를 다시 생성합니다.

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

추가 리소스

- **Clevis-luks-bind(1)** 및 **dracut.cmdline(7)** 도움말 페이지입니다.
- RHEL 네트워크 부팅 옵션

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

다음 단계를 사용하여 신뢰할 수 있는 플랫폼 모듈 2.0(TPM 2.0) 정책을 사용하여 LUKS 암호화 볼륨 잠금을 구성합니다.

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.

```
# dnf 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. 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.

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"}'
```

주의

PCR 해시 값이 밀봉 및 해시를 다시 작성할 때 사용되는 정책과 일치하는 경우에만 데이터가 음소거될 수 있으므로 PCR의 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, KCS를 다시 시작한 후 Clevis TPM2의 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.

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

검증

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

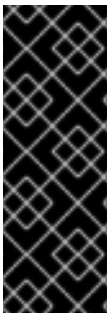
```
# clevis luks list -d /dev/sda2
1: tpm2 '{"hash":"sha256","key":"rsa"}'
```

추가 리소스

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

11.9. LUKS 암호화된 볼륨에서 CLEVIS 핀 제거 수동으로 제거

clevis luks bind 명령으로 생성된 메타데이터를 수동으로 제거하고 Clevis에서 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 초기화합니다.

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

사전 요구 사항

- Clevis 바인딩이 있는 LUKS 암호화 볼륨.

절차

1. 볼륨(예:*/dev/sda* 2)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

11.10. KICKSTART를 사용하여 LUKS 암호화 볼륨 자동 등록 구성

이 절차의 단계에 따라 LUKS 암호화 볼륨 등록에 Clevis를 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 중에 네트워크가 필요한 Tang 핀을 사용하거나 고정 IP 구성과 함께 NetNamespace 클라이언트를 사용하는 경우 LUKS 암호화 볼륨의 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
- Kickstart를 사용하여 Red Hat Enterprise Linux 9 설치

11.11. LUKS 암호화 이동식 스토리지 장치의 자동 잠금 해제 구성

이 절차를 사용하여 LUKS 암호화 USB 스토리지 장치의 자동 잠금 해제 프로세스를 설정합니다.

절차

1. USB 드라이브와 같은 LUKS로 암호화된 이동식 스토리지 장치의 잠금을 자동으로 해제하려면 **clevis-udisks2** 패키지를 설치합니다.

```
# dnf install clevis-udisks2
```

2. 시스템을 재부팅한 다음 LUKS 암호화 볼륨 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

11.12. 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 Hat은 이 방법을 권장하지 않습니다.

11.12.1. Shamir의 Secret Sharing를 사용한 고급 기능

Shamir의 SDS(Secret Sharing)는 시크릿을 여러 가지 고유한 부분으로 나누는 암호화 체계입니다. 시크릿을 복원하려면 여러 부분이 필요합니다. 이 수를 임계값이라고 하며 SSS를 임계값 지정 체계라고도 합니다.

Clevis는 SSS 구현을 제공합니다. 키를 생성하고 여러 조각으로 나눕니다. 각 조각은 SSS를 포함하여 다른 핀을 사용하여 암호화됩니다. 또한 **t** 임계값을 정의합니다. TPM 배포가 최소 **t** 조각을 암호 해독하는 경우 암호화 키를 복구하고 암호 해독 프로세스가 성공합니다. Clevis가 임계값에 지정된 것보다 적은 수의 부분을 감지하면 오류 메시지를 출력합니다.

11.12.1.1. 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.

11.12.1.2. 예 2: Tang 서버 및 TPM 장치의 공유 시크릿

다음 명령은 **tang** 서버와 **tpm2** 장치를 모두 사용할 수 있을 때 LUKS 암호화 장치를 성공적으로 해독합니다.

```
# 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

11.13. NBDE 네트워크에 가상 머신 배포

clevis luks bind 명령은 LUKS 마스터 키를 변경하지 않습니다. 즉, 가상 머신 또는 클라우드 환경에서 사용하기 위해 LUKS로 암호화된 이미지를 생성하면 이 이미지를 실행하는 모든 인스턴스가 마스터 키를 공유합니다. 이는 매우 안전하지 않으며 항상 피해야 합니다.

이는 Clevis의 제한 사항은 아니지만 LUKS의 설계 원칙입니다. 클라우드에 암호화된 루트 볼륨이 필요한 경우 클라우드에서 Red Hat Enterprise Linux의 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(Kickstart를 사용하여 LUKS 암호화 볼륨 자동 등록 구성 참조) 또는 다른 자동화된 프로비저닝 툴을 사용하여각 암호화된 VM에 고유한 마스터 키가 있는지 확인합니다.

추가 리소스

- **clevis-luks-bind(1)** 매뉴얼 페이지

11.14. CLEVIS를 사용하여 클라우드 환경을 위해 자동으로 로그인할 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

이러한 이유로 Red Hat은 데이터가 저장된 위치와 Tang이 실행되는 시스템을 물리적 분리할 것을 강력히 권장합니다. 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.

11.15. 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/rhel9/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/rhel9/tang
```

Tang은 기본적으로 포트 80을 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/rhel9/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 Tang을 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:

x1AIpc6WmnCU-CabD8_4q18vDuw

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

위 예제 명령은 *localhost* URL에서 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

11.16. CLEVIS 및 TANG 시스템 역할 소개

RHEL 시스템 역할은 여러 RHEL 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 Ansible 역할 및 모듈의 컬렉션입니다.

Clevis 및 Tang을 사용하여 PBD(Policy-Based Decryption) 솔루션의 자동화된 배포에 Ansible 역할을 사용할 수 있습니다. **rhel-system-roles** 패키지에는 이러한 시스템 역할, 관련 예제 및 참조 문서가 포함되어 있습니다.

Network Bound Disk Encryption Client System Role을 사용하면 여러 Clevis 클라이언트를 자동화된 방식으로 배포할 수 있습니다. Network Bound Disk Encryption Client 역할은 Tang 바인딩만 지원하며 현재 TPM2 바인딩에는 사용할 수 없습니다.

네트워크 Bound Disk Encryption Client 역할에는 LUKS를 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 Clevis를 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.

Network Bound Disk Encryption Client 역할은 **state** 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.

네트워크 Bound 디스크 암호화 서버 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

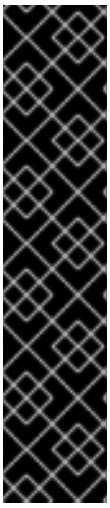
- NFV(Network-Bound Disk Encryption) 역할 변수에 대한 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles/nbde_client/** 및 **/usr/share/doc/rhel-system-roles/nbde_server/** 디렉터리의 **README.md** 및 **README.html** 파일을 참조하십시오.
- 예를 들어 system-roles 플레이북을 **rhel-system-roles** 패키지를 설치하고 **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** 디렉터리를 참조하십시오.
- RHEL 시스템 역할에 대한 자세한 내용은 RHEL 시스템 역할 소개 를 참조하십시오.

11.17. NBDE 서버 시스템 역할을 사용하여 여러 TANG 서버를 설정

단계에 따라 Tang 서버 설정이 포함된 Ansible 플레이북을 준비하고 적용합니다.

사전 요구 사항

- NBDE 서버 시스템 역할을 사용하여 구성할 시스템인 하나 이상의 *관리* 노드에 대한 액세스 및 사용 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.
  제어 노드에서 다음을 수행합니다.
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.



중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman**과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법 에 대한 지식베이스 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위에 대한 지식베이스 문서를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

1. Tang 서버에 대한 설정이 포함된 플레이북을 준비합니다. 처음부터 시작하거나 **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** 디렉터리의 예제 플레이북 중 하나를 사용할 수 있습니다.

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/simple_deploy.yml
./my-tang-playbook.yml
```

2. 선택한 텍스트 편집기에서 플레이북을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi my-tang-playbook.yml
```

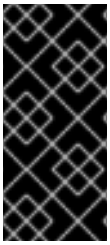
3. 필수 매개 변수를 추가합니다. 다음 예제 플레이북은 Tang 서버와 키 교체의 배포를 확인합니다.

```
---
- hosts: all

  vars:
    nbde_server_rotate_keys: yes

  roles:
    - rhel-system-roles.nbde_server
```

4. 완료된 플레이북을 적용합니다.

```
# ansible-playbook -i inventory-file my-tang-playbook.yml
```

여기서: * ***inventory-file*** 은 인벤토리 파일입니다. * ***logging-playbook.yml*** 은 사용하는 플레이북입니다.



중요

Clevis가 설치된 시스템에서 **grubby** 툴을 사용하여 초기 부팅 시 Tang 핀의 네트워킹을 사용할 수 있도록 하려면 다음을 수행합니다.

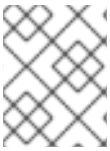
```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

추가 리소스

- 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles/nbde_server/** 및 **usr/share/ansible/rhel-system-roles.nbde_server/** 디렉터리를 참조하십시오.

11.18. 여러 CLEVIS 클라이언트를 설정하는 데 CLEVIS 클라이언트 시스템 역할 사용

단계에 따라 Clevis 클라이언트 설정이 포함된 Ansible 플레이북을 준비하고 적용합니다.



참고

Clevis 클라이언트 시스템 역할은 Tang 바인딩만 지원합니다. 즉, 현재 TPM2 바인딩에 사용할 수 없습니다.

사전 요구 사항

- NBDE 클라이언트 시스템 역할을 사용하여 구성할 시스템인 하나 이상의 *관리* 노드에 대한 액세스 및 사용 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.
- Ansible Core 패키지는 컨트롤 시스템에 설치되어 있어야 합니다.
- **rhel-system-roles** 패키지는 플레이북을 실행하려는 시스템에 설치되어 있어야 합니다.

절차

1. Clevis 클라이언트에 대한 설정을 포함하는 플레이북을 준비합니다. 처음부터 시작하거나 **/usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/** 디렉터리의 예제 플레이북 중 하나를 사용할 수 있습니다.

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/high_availability.yml
./my-clevis-playbook.yml
```

2. 선택한 텍스트 편집기에서 플레이북을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi my-clevis-playbook.yml
```

3. 필수 매개 변수를 추가합니다. 다음 예제 플레이북은 두 개 이상의 Tang 서버 중 하나를 사용할 수 있는 경우 에서 두 개의 LUKS 암호화 볼륨의 잠금 해제를 자동화하도록 Clevis 클라이언트를 구성합니다.

```
---
- hosts: all
```

```
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

roles:
  - rhel-system-roles.nbde_client
```

4. 완료된 플레이북을 적용합니다.

```
# ansible-playbook -i host1,host2,host3 my-clevis-playbook.yml
```

중요

Clevis가 설치된 시스템에서 **grubby** 툴을 사용하여 초기 부팅 시 Tang 핀의 네트워킹을 사용할 수 있도록 하려면 다음을 수행합니다.

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

추가 리소스

- NBDE 클라이언트 시스템 역할에 대한 매개변수 및 추가 정보에 대한 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles/nbde_client/** 및 **/usr/share/ansible/ansible/roles.nbde_client/** 디렉터리를 참조하십시오.

11.19. 추가 리소스

- **Tang(8)**, **clevis(1)**, **jose(1)**, **clevis-luks-unlockers(7)** 매뉴얼 페이지
- 여러 LUKS 장치(Clevis + Tang 잠금 해제) 기술 자료 문서를 사용하여 Network-Bound Disk Encryption을 설정하는 방법

12장. 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.

12.1. LINUX 감사

Linux 감사 시스템은 시스템에서 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.

- 날짜 및 시간, 유형 및 이벤트 결과.
- 주체 및 개체의 민감도 레이블입니다.
- 이벤트를 트리거한 사용자의 ID와 이벤트의 연관
- 감사 구성에 대한 모든 수정 및 감사 로그 파일에 액세스하려고 합니다.
- SSH, Kerberos 등과 같은 인증 메커니즘의 모든 용도.
- 신뢰할 수 있는 데이터베이스(예: **/etc/passwd**)로 변경합니다.
- 시스템 내 또는 시스템에서 정보를 가져오거나 내보내려는 시도.
- 사용자 ID, 주체 및 오브젝트 레이블 및 기타 특성을 기반으로 이벤트를 포함하거나 제외합니다.

감사 시스템을 사용하려면 여러 보안 관련 인증도 필요합니다. 감사는 다음 인증 또는 컴플라이언스 가이드의 요구사항을 충족하거나 초과하도록 설계되었습니다.

- Control Access Protection Profile (CAPP)
- 라벨이 지정된 보안 보호 프로파일 (LSPP)
- 기본 액세스 제어(RSBAC) 규칙 설정
- 국가 산업 보안 프로그램 운영 설명서 (NISPOM)
- 연방 정보 보안 관리법 (FISMA)
- 결제 카드 산업 - PCI-DSS(Data Security Standard)
- 보안 기술 구현 가이드(STIG)

또한 감사는 다음과 같습니다.

- NIAP(National Information Assurance Partnership) 및 BSI(BSI)에 의해 평가되었습니다.
- Red Hat Enterprise Linux 5에서 LSPP/CAPP/RSBAC/EAL4+ 인증.
- Red Hat Enterprise Linux 6에서 운영 체제 보호 프로파일 / evaluation Assurance Level 4+ (OSPP/EAL4+)에 인증되었습니다.

사용 사례

파일 액세스 감시

감사는 파일 또는 디렉터리가 액세스, 수정, 실행 또는 파일의 속성이 변경되었는지 여부를 추적할 수 있습니다. 예를 들어 중요한 파일에 대한 액세스를 감지하고 이러한 파일 중 하나가 손상된 경우 감사 추적을 사용할 수 있도록 하는 것이 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 시간을 추적하는 데 사용할 수 있습니다.

사용자가 실행한 명령 레코딩

감사는 파일이 실행되었는지 여부를 추적할 수 있으므로 특정 명령의 모든 실행을 기록하도록 규칙을 정의할 수 있습니다. 예를 들어 **/bin** 디렉터리의 모든 실행 파일에 대해 규칙을 정의할 수 있습니다. 그런 다음 사용자 ID로 생성된 로그 항목을 검색하여 사용자별로 실행된 명령에 대한 감사 추적을 생성할 수 있습니다.

시스템 경로 이름 실행 기록

규칙 호출 시 경로를 inode로 변환하는 파일 액세스를 감시하는 것 외에도 감사는 이제 규칙 호출 시 존재하지 않거나 파일이 규칙 호출 후 교체되는 경우에도 경로 실행을 모니터링할 수 있습니다. 이를 통해 프로그램 실행 파일을 업그레이드하거나 프로그램을 설치하기 전에 규칙이 계속 작동할 수 있습니다.

보안 이벤트 기록

샌드 **박스 인증** 모듈은 실패한 로그인 시도를 기록할 수 있습니다. 실패한 로그인 시도를 기록하도록 감사를 설정하고 로그인을 시도한 사용자에 대한 추가 정보를 제공할 수 있습니다.

이벤트 검색

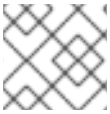
감사에서는 **ausearch** 유틸리티를 제공합니다. 이 유틸리티는 로그 항목을 필터링하고 여러 조건에 따라 완전한 감사 추적을 제공하는 데 사용할 수 있습니다.

요약 보고서 실행

aureport 유틸리티는 기록 된 이벤트에 대한 일일 보고서를 생성하는 데 사용할 수 있습니다. 그런 다음 시스템 관리자는 이러한 보고서를 분석하고 의심 스러운 활동을 더 조사할 수 있습니다.

네트워크 액세스 모니터링

nftables,**iptables**, **ebtables** 유틸리티를 구성하여 감사 이벤트를 트리거하도록 구성하여 시스템 관리자가 네트워크 액세스를 모니터링할 수 있습니다.



참고

감사에서 수집하는 정보의 크기에 따라 시스템 성능에 영향을 줄 수 있습니다.

12.2. 감사 시스템 아키텍처

감사 시스템은 사용자 공간 애플리케이션 및 유틸리티와 커널 측 시스템 호출 처리의 두 가지 주요 부분으로 구성됩니다. 커널 구성 요소는 사용자 공간 애플리케이션에서 시스템 호출을 수신하고 다음 필터 중 하나를 통해 필터링합니다(**사용자**,**작업**,**fstype** 또는 **exit**).

시스템 호출이 **exclude** 필터를 통과하면 감사 규칙 구성을 기반으로 하는 앞서 언급한 필터 중 하나를 통해 전송되어 추가 처리를 위해 감사 데몬으로 보냅니다.

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 여러 설정 및 매개변수를 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 9에서 Audit dispatcher 데몬(**audisp**) 기능이 감사 데몬(**auditd**)에 통합되었습니다. 감사 이벤트와 실시간 분석 프로그램의 상호 작용을 위한 플러그인의 구성 파일은 기본적으로 **/etc/audit/plugins.d/** 디렉토리에 있습니다.

12.3. 보안 환경에 대해 AUDITD 구성

기본 **auditd** 구성은 대부분의 환경에 적합합니다. 그러나 환경에서 엄격한 보안 정책을 충족해야하는 경우 **/etc/audit/auditd.conf** 파일의 감사 데몬 구성에 대해 다음 설정이 제안됩니다.

log_file

감사 로그 파일을 보유한 디렉터리(일반적으로 **/var/log/audit/**)는 별도의 마운트 지점에 있어야 합니다. 이렇게 하면 다른 프로세스가 이 디렉터리의 공간을 소비하지 않도록 하고 감사 데몬의 나머지 공간을 정확하게 탐지할 수 있습니다.

max_log_file

감사 로그 파일이 있는 파티션에서 사용 가능한 공간을 완전히 사용하도록 단일 감사 로그 파일의 최대 크기를 지정합니다.

max_log_file_action

max_log_file 에 설정된 한도에 도달한 후 감사 로그 파일을 덮어쓰지 않도록 **keep_logs** 로 설정해야 하는 작업을 결정합니다.

space_left

space_left_action 매개변수에 설정된 작업이 트리거되는 디스크에 남아 있는 여유 공간의 양을 지정합니다. 관리자에게 응답하는 데 충분한 시간을 제공하고 디스크 공간을 확보할 수 있는 번호로 설정해야 합니다. **space_left** 값은 감사 로그 파일이 생성되는 비율에 따라 달라집니다.

space_left_action

space_left_action 매개변수를 적절한 알림 방법으로 **email** 또는 **exec** 로 설정하는 것이 좋습니다.

admin_space_left

admin_space_left_action 매개변수에 설정된 작업을 트리거하는 데 필요한 최소 최소 공간을 지정하며 관리자가 수행하는 로그 작업에 충분한 공간을 남겨 두는 값으로 설정해야 합니다.

admin_space_left_action

시스템을 **단일** 사용자 모드로 설정하고 관리자가 일부 디스크 공간을 확보하도록 단일으로 설정해야 합니다.

disk_full_action

감사 로그 파일을 보유한 파티션에서 사용 가능한 공간을 사용할 수 없는 경우 트리거되는 작업을 **중지** 하거나 **단일** 으로 설정해야 합니다. 이렇게 하면 감사가 더 이상 이벤트를 로깅할 수 없는 경우 시스템이 단일 사용자 모드에서 종료되거나 작동됩니다.

disk_error_action

하드웨어 중단과 관련된 로컬 보안 정책에 따라 syslog 로그 파일을 보유하는 파티션에서 오류가 감지되는 경우 트리거되는 작업을 **syslog**,**단일** 또는 **정지** 로 설정해야 합니다.

flush

incremental_async 로 설정해야 합니다. **freq** 매개 변수와 함께 작동합니다. 이 매개 변수는 하드 드라이브와 하드 동기화를 강제 적용하기 전에 디스크에 보낼 수 있는 레코드 수를 결정합니다. **freq** 매개 변수는 **100** 으로 설정해야 합니다. 이러한 매개 변수를 사용하면 감사 이벤트 데이터가 디스크의 로그 파일과 동기화되고 버스트 작업에 적합한 성능을 유지할 수 있습니다.

나머지 구성 옵션은 로컬 보안 정책에 따라 설정해야 합니다.

12.4. AUDITD 시작 및 제어

auditd 를 구성한 후 서비스를 시작하여 감사 정보를 수집하고 로그 파일에 저장합니다. 다음 명령을 root 사용자로 사용하여 **auditd** 를 시작합니다.

```
# service auditd start
```

부팅 시 시작되도록 **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 여러 작업을 수행할 수 있습니다. 여기서 *작업* 은 다음 중 하나일 수 있습니다.

중지

auditd 를 중지합니다.

재시작

auditd 를 다시 시작합니다.

reload 또는 force-reload

/etc/audit/auditd.conf 파일에서 **auditd** 구성을 다시 로드합니다.

rotate

/var/log/audit/ 디렉터리에서 로그 파일을 순환합니다.

resume

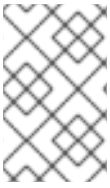
예를 들어 감사 로그 파일을 보유하는 디스크 파티션에 사용 가능한 공간이 충분하지 않은 경우와 같이 이전에 일시 중지된 후 감사 이벤트 로깅을 다시 시작합니다.

condrestart 또는 try-restart

이미 실행 중인 경우에만 **auditd** 를 다시 시작합니다.

status

auditd 의 실행 중인 상태를 표시합니다.



참고

service 명령은 **auditd** 데몬과 올바르게 상호 작용할 수 있는 유일한 방법입니다. **auid** 값이 올바르게 기록되도록 **service** 명령을 사용해야 합니다. **systemctl** 명령은 **활성화** 및 **status** 의 두 가지 작업에만 사용할 수 있습니다.

12.5. 감사 로그 파일 이해

기본적으로 감사 시스템은 로그 항목을 **/var/log/audit/audit.log** 파일에 저장합니다. 로그 순환이 활성화된 경우 순환된 **audit.log** 파일이 동일한 디렉터리에 저장됩니다.

/etc/ssh/sshd_config 파일을 읽거나 수정하려는 모든 시도를 기록하려면 다음 감사 규칙을 추가합니다.

```
# auditctl -w /etc/ssh/sshd_config -p warx -k sshd_config
```

auditd 데몬이 실행 중인 경우 예를 들어 다음 명령을 사용하면 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 공백으로 구분된 여러 개의 ***name=value*** 쌍으로 구성됩니다. 상기 이벤트에 대한 자세한 분석은 다음과 같습니다:

첫 번째 레코드

type=SYSCALL

type 필드에는 레코드 유형이 포함됩니다. 이 예에서 **SYSCALL** 값은 커널에 대한 시스템 호출에 의해 이 레코드가 트리거되었음을 지정합니다.

msg=audit(1364481363.243:24287):

msg 필드는 다음을 기록합니다.

- 타임스탬프와 양식 **audit(*time_stamp*: ID)의 레코드 고유*ID*** 입니다. 동일한 감사 이벤트의 일부로 생성된 경우 여러 레코드가 동일한 타임스탬프와 ID를 공유할 수 있습니다. 타임 스탬프는 1월 1일 00:00:00 UTC 이후의 Unix 시간 형식 - 초를 사용합니다.
- 커널 또는 사용자 공간 애플리케이션에서 제공하는 다양한 이벤트별 ***이름=값*** 쌍입니다.

arch=c000003e

arch 필드에는 시스템의 CPU 아키텍처에 대한 정보가 포함되어 있습니다. **c000003e** 값은 16진수 표기법으로 인코딩됩니다. **ausearch** 명령으로 감사 레코드를 검색할 때 **-i** 또는 **--interpret** 옵션을 사용하여 16진수 값을 사람이 읽을 수 있는 동등한 값으로 자동 변환합니다. **c000003e** 값은 **x86_64** 로 해석됩니다.

syscall=2

syscall 필드는 커널로 전송된 시스템 호출 유형을 기록합니다. 값 **2** 는 **/usr/include/asm/unistd_64.h** 파일에서 사람이 읽을 수 있는 동등한 것과 일치할 수 있습니다. 이 경우 **2** 는 **오픈** 시스템 호출입니다. **ausyscall** 유틸리티를 사용하면 시스템 호출 번호를 사람이 읽을 수 있는 동등한 값으로 변환할 수 있습니다. **ausyscall --dump** 명령을 사용하여 모든 시스템 호출 목록을 번호와 함께 표시합니다. 자세한 내용은 **ausyscall**(8) 매뉴얼 페이지를 참조하십시오.

success=no

success 필드는 해당 특정 이벤트에 기록된 시스템 호출이 성공했는지 또는 실패했는지를 기록합니다. 이 경우 호출이 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 값을 사람이 읽을 수 있는 것으로 해석할 수 있습니다.

```
# ausearch --interpret --exit -13
```

이전 예제에서는 감사 로그에 종료 코드 **-13** 을 사용하여 실패한 이벤트가 포함되어 있다고 가정합니다.

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

a0 ~ **a3** 필드는 이 이벤트에서 시스템 호출의 16진수 표기법으로 인코딩된 처음 4개의 인수를 기록합니다. 이러한 인수는 사용되는 시스템 호출에 따라 달라집니다. **ausearch** 유틸리티에서 해석할 수 있습니다.

items=1

items 필드에는 syscall 레코드를 따르는 PATH 보조 레코드 수가 포함됩니다.

ppid=2686

ppid 필드는 PPID(Parent Process ID)를 기록합니다. 이 경우 **2686** 은 **bash** 와 같은 상위 프로세스의 PPID였습니다.

pid=3538

pid 필드는 Process ID(PID)를 기록합니다. 이 경우 **3538** 은 **cat** 프로세스의 PID입니다.

auid=1000

auid 필드는 loginuid인 감사 사용자 ID를 기록합니다. 이 ID는 로그인 시 사용자에게 할당되며 사용자의 ID가 변경되는 경우에도 모든 프로세스가 상속됩니다(예: **su - john** 명령으로 사용자 계정을 전환함).

uid=1000

uid 필드는 분석된 프로세스를 시작한 사용자의 사용자 ID를 기록합니다. 사용자 ID를 사용자 이름으로 해석할 수 있습니다. **ausearch -i --uid *UID***.

gid=1000

gid 필드는 분석된 프로세스를 시작한 사용자의 그룹 ID를 기록합니다.

euid=1000

euid 필드는 분석 프로세스를 시작한 사용자의 유효한 사용자 ID를 기록합니다.

suid=1000

suid 필드는 분석 프로세스를 시작한 사용자의 set 사용자 ID를 기록합니다.

fsuid=1000

fsuid 필드는 분석 프로세스를 시작한 사용자의 파일 시스템 사용자 ID를 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 분석된 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 실행 시 분석된 프로세스에 레이블이 지정된 SELinux 컨텍스트를 기록합니다.

key="sshd_config"

key 필드는 감사 로그에 이 이벤트를 생성한 규칙과 관련된 관리자 정의 문자열을 기록합니다.

두 번째 레코드

type=CWD

두 번째 레코드에서 **type** 필드 값은 **CWD** - 현재 작업 디렉터리입니다. 이 유형은 첫 번째 레코드에서 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 필드에는 첫 번째 레코드의 값과 동일한 타임스탬프 및 ID 값이 있습니다. 타임 스탬프는 1월 1일 00:00:00 UTC 이후의 Unix 시간 형식 - 초를 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 인수로 시스템 호출에 전달되는 모든 경로에 대한 **PATH**-type 레코드가 포함되어 있습니다. 이 감사 이벤트에서는 하나의 경로(**/etc/ssh/sshd_config**)만 인수로 사용되었습니다.

msg=audit(1364481363.243:24287):

msg 필드에는 첫 번째 및 두 번째 레코드의 값과 동일한 타임스탬프 및 ID 값이 있습니다.

item=0

item 필드는 **SYSCALL** 유형 레코드에서 참조되는 총 항목 수 중 현재 레코드를 나타내는 항목을 나타냅니다. 이 숫자는 0부터 시작합니다. **0** 은 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** 파일입니다.

inode=409248

inode 필드에는 이 이벤트에 기록된 파일 또는 디렉터리와 연결된 inode 번호가 포함되어 있습니다. 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 ID를 지정합니다. 이 경우 값은 **/dev/fd/0** 장치를 나타냅니다.

mode=0100600

mode 필드는 **st_mode** 필드에 있는 **stat** 명령에서 반환된 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

ouid 필드는 오브젝트 소유자의 사용자 ID를 기록합니다.

ogid=0

ogid 필드는 오브젝트 소유자의 그룹 ID를 기록합니다.

rdev=00:00

rdev 필드에는 특수 파일에 대해서만 기록된 장치 식별자가 포함되어 있습니다. 이 경우 기록된 파일이 정규 파일이므로 사용되지 않습니다.

obj=system_u:object_r:etc_t:s0

obj 필드는 실행 시 기록된 파일 또는 디렉터리가 레이블이 지정된 SELinux 컨텍스트를 기록합니다.

nametype=NORMAL

nametype 필드는 지정된 syscall의 컨텍스트에서 각 경로 레코드의 작업의 의도를 기록합니다.

cap_fp=none

cap_fp 필드는 파일 또는 디렉터리 오브젝트의 허용된 파일 시스템 기반 기능과 관련된 데이터를 기록합니다.

cap_fi=none

cap_fi 필드는 파일 또는 디렉토리 오브젝트의 상속된 파일 시스템 기반 기능과 관련된 데이터를 기록합니다.

cap_fe=0

cap_fe 필드는 파일 또는 디렉토리 오브젝트의 유효한 파일 시스템 기반 기능의 설정을 기록합니다.

cap_fver=0

cap_fver 필드는 파일 또는 디렉터리 오브젝트의 파일 시스템 기반 기능 버전을 기록합니다.

네 번째 레코드

type=PROCTITLE

type 필드에는 레코드 유형이 포함됩니다. 이 예에서 **PROCTITLE** 값은 이 레코드가 커널에 대한 시스템 호출에 의해 트리거된 이 감사 이벤트를 트리거한 전체 명령줄을 제공하도록 지정합니다.

proctitle=636174002F6574632F7373682F737368645F636F6E666967

proctitle 필드는 분석 프로세스를 호출하는 데 사용된 명령의 전체 명령줄을 기록합니다. 이 필드는 16진 표기법으로 인코딩되어 사용자가 감사 로그 구문 분석기에 영향을 미칠 수 없습니다. 이 감사 이벤트를 트리거한 명령으로 텍스트를 디코딩합니다. **ausearch** 명령으로 감사 레코드를 검색할 때 **-i** 또는 **--interpret** 옵션을 사용하여 16진수 값을 사람이 읽을 수 있는 동등한 값으로 자동 변환합니다. **636174002F6574632F7373682F63645F636F6E666967** 값은 **cat /etc/ssh/sshd_config** 로 해석됩니다.

12.6. AUDITCTL을 사용하여 감사 규칙 정의 및 실행

감사 시스템은 로그 파일에 캡처된 항목을 정의하는 규칙 세트에서 작동합니다. 감사 규칙은 **auditctl** 유틸리티 또는 **/etc/audit/rules.d/** 디렉터리를 사용하여 명령줄에서 설정할 수 있습니다.

auditctl 명령을 사용하면 감사 시스템의 기본 기능을 제어하고 어떤 감사 이벤트가 기록되는지 결정하는 규칙을 정의할 수 있습니다.

파일 시스템 규칙 예

1. 모든 쓰기 액세스 권한을 기록하는 규칙을 정의하고 모든 속성 변경 사항은 **/etc/passwd** 파일에 있습니다.

```
# auditctl -w /etc/passwd -p wa -k passwd_changes
```

2. 모든 쓰기 권한을 기록하는 규칙을 정의하고 **/etc/selinux/** 디렉터리에 있는 모든 파일의 모든 속성 변경 사항을 기록합니다.

```
# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

시스템 호출 규칙 예

1. 프로그램에서 **adjtimex** 또는 **settimeofday** 시스템 호출이 사용될 때마다 로그 항목을 생성하는 규칙을 정의하기 위해 64비트 아키텍처를 사용합니다.

```
# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
```

2. 파일이 삭제되거나 ID가 1000 이상인 시스템 사용자가 파일 이름을 삭제할 때마다 로그 항목을 생성하는 규칙을 정의하십시오.

```
# 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 UID가 설정되지 않은 사용자를 제외하는 데 사용됩니다.

실행 파일 규칙

/bin/id 프로그램의 모든 실행을 기록하는 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.

12.7. 영구 감사 규칙 정의

재부팅 시 지속되는 감사 규칙을 정의하려면 **/etc/audit/rules.d/audit.rules** 파일에 직접 포함하거나 **/etc/audit/rules.d/** 디렉터리에 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(#) 다음에 있는 빈 줄과 텍스트는 무시됩니다.

또한 **-R** 옵션을 사용하여 **auditctl** 명령을 사용하여 지정된 파일에서 규칙을 읽을 수 있습니다. 예를 들면 다음과 같습니다.

```
# auditctl -R /usr/share/audit/sample-rules/30-stig.rules
```

12.8. 사전 구성된 규칙 파일 사용

/usr/share/audit/sample-rules 디렉터리에서 **audit** 패키지는 다양한 인증 표준에 따라 사전 구성된 규칙 파일 세트를 제공합니다.

30-nispom.rules

국가 산업 보안 프로그램 운영 설명서의 정보 시스템 보안 장에 지정된 요구 사항을 충족하는 감사 규칙 구성입니다.

30-ospp-v42*.rules

OSPP (Protection Profile for General Purpose Operating Systems) 프로파일 버전 4.2에 정의된 요구 사항을 충족하는 감사 규칙 구성.

30-pci-dss-v31.rules

PCI DASD(Payment Card Industry Data Security Standard) v3.1로 설정된 요구 사항을 충족하는 감사 규칙 구성입니다.

30-stig.rules

보안 기술 구현 가이드(STIG)로 설정된 요구 사항을 충족하는 감사 규칙 구성입니다.

이러한 구성 파일을 사용하려면 **/etc/audit/rules.d/** 디렉터리에 파일을 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
```

번호 지정 체계를 사용하여 감사 규칙을 주문할 수 있습니다. 자세한 내용은 **/usr/share/audit/sample-rules/README-rules** 파일을 참조하십시오.

추가 리소스

- **audit.rules-4.8** 매뉴얼 페이지.

12.9. AUGENRULES를 사용하여 영구 규칙 정의

augenrules 스크립트는 **/etc/audit/rules.d/** 디렉터리에 있는 규칙을 읽고 **audit.rules** 파일로 컴파일합니다. 이 스크립트는 자연 정렬 순서에 따라 **.rules** 로 끝나는 모든 파일을 특정 순서로 처리합니다. 이 디렉터리의 파일은 다음과 같은 의미를 사용하여 그룹으로 구성됩니다.

- 10 - kernel 및 auditctl 설정
- 20 - 일반 규칙과 일치할 수 있지만 다른 일치 항목이 필요한 규칙입니다.
- 30 - 주요 규칙
- 40 - 선택적 규칙
- 50 - 서버별 규칙
- 70 - 시스템 로컬 규칙
- 90 - 종료(immutable)

규칙은 한 번에 모두 사용할 수 없습니다. 이러한 정책은 고려해야 할 요소와 **/etc/audit/rules.d/** 에 복사되는 개별 파일입니다. 예를 들어 VDDK 구성에서 시스템을 설정하려면 규칙 **10-base-config**,**30-stig**,**31-privileged**, **99-finalize** 규칙을 복사합니다.

/etc/audit/rules.d/ 디렉터리에 규칙이 있으면 **--load** 지시문을 사용하여 **augenrules** 스크립트를 실행하여 로드하십시오.

```
# 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)** 매뉴얼 페이지.

12.10. AUGENRULES 비활성화

augenrules 유틸리티를 비활성화하려면 다음 단계를 사용합니다. 이 명령은 감사를 전환하여 **/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 주석 처리를 해제합니다.

```
#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 시작합니다.

```
# service auditd restart
```

추가 리소스

- **augenrules(8)** 및 **audit.rules(8)** 매뉴얼 페이지.
- auditd service restart는 /etc/audit/audit.rules의 변경 사항을 덮어씁니다 .

12.11. 소프트웨어 업데이트 모니터링에 감사 설정

사전 구성된 규칙 **44-installers.rules** 를 사용하여 소프트웨어를 설치하는 다음 유틸리티를 모니터링하도록 감사를 구성할 수 있습니다.

- **dnf** [2]
- **yum**
- **pip**
- **npm**
- **CPAN**
- **gem**
- **luarocks**

rpm 유틸리티를 모니터링하려면 **rpm-plugin-audit** 패키지를 설치합니다. 그런 다음 감사는 패키지를 설치하거나 업데이트할 때 **SOFTWARE_UPDATE** 이벤트를 생성합니다. 이러한 이벤트는 명령줄에 **ausearch -m SOWARE_UPDATE**를 입력하여 나열할 수 있습니다.

참고

사전 구성된 규칙 파일은 **ppc64le** 및 **aarch64** 아키텍처가 있는 시스템에서 사용할 수 없습니다.

사전 요구 사항

- **auditd** 는 보안 환경에 대해 auditd 구성을 통해 제공되는 설정에 따라 구성됩니다.

절차

1. 사전 구성된 규칙 파일 **44-installers.rules** 를 **/usr/share/audit/sample-rules/** 디렉터리에서 **/etc/audit/rules.d/** 디렉터리로 복사합니다.

```
# cp /usr/share/audit/sample-rules/44-installers.rules /etc/audit/rules.d/
```

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. 다음과 같이 설치를 수행합니다.

```
# dnf reinstall -y vim-enhanced
```

3. 최근 설치 이벤트에 대한 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"
```

12.12. 감사를 사용하여 사용자 로그인 시간 모니터링

특정 시간에 로그인한 사용자를 모니터링하려면 특별한 방식으로 감사를 구성할 필요가 없습니다. 동일한 정보를 제공하는 다양한 방법을 제공하는 **ausearch** 또는 **aureport** 툴을 사용할 수 있습니다.

사전 요구 사항

- **auditd** 는 보안 환경에 대해 auditd 구성을 통해 제공되는 설정에 따라 구성됩니다.

절차

사용자 로그인을 표시하려면 다음 명령 중 하나를 사용합니다.

- **USER_LOGIN** 메시지 유형을 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 날짜 및 시간을 지정할 수 있습니다. 이 옵션을 사용하지 않으면 **ausearch** 는 오늘의 결과를 제공하고 시간을 생략하면 **ausearch** 는 자정 결과를 제공합니다.
  - **-sv yes** 옵션을 사용하여 성공적인 로그인 시도를 필터링하고 로그인 시도에 대해 **-sv no** 를 사용할 수 있습니다.

- **마지막** 명령의 출력과 유사한 형식으로 출력을 표시하는 **ausearch** 명령의 원시 출력을 **aulast** 유틸리티로 파이프합니다. 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
```

- **--login -i** 옵션과 함께 **aureport** 명령을 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)** 매뉴얼 페이지.
- **aulast(8)** 도움말 페이지.
- **aureport(8)** 도움말 페이지.

12.13. 추가 리소스

- RHEL 감사 시스템 참조 지식 베이스 문서.
- 컨테이너 지식베이스 문서의 감사 실행 옵션 .
- Linux 감사 문서 프로젝트 페이지 .
- **audit** 패키지는 **/usr/share/doc/audit/** 디렉터리의 문서를 제공합니다.

- **auditd(8)**, **auditctl(8)**, **ausearch(8)**, **audit.rules(7)**, **audispd.conf(5)**, **audispd(8)**, **auditd.conf(5)**, **ausearch-expression(5)**, aulast(8) , **aulast(8)**, **aulastlog(8) aureport(8)**, **ausyscall(8)**, **autrace(8)** 및 **auvirt(8)** 도움말 페이지.

---

[2] **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** 로 변경하여 **44-installers.rules** 파일을 수정합니다.

13장. FAPOLICYD를 사용하여 애플리케이션 차단 및 허용

규칙 세트를 기반으로 애플리케이션 실행을 허용하거나 거부하는 정책을 설정하고 적용하면 알 수 없거나 잠재적으로 악성 소프트웨어가 실행되지 않습니다.

13.1. FAPOLICYD 소개

fapolicyd 소프트웨어 프레임워크는 사용자 정의 정책을 기반으로 애플리케이션 실행을 제어합니다. 이는 시스템에서 신뢰할 수 없는 애플리케이션 및 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 감사 가능성을 사용하여 모든 애플리케이션에 대한 **허용** 및 **거부** 실행 규칙을 정의할 수 있습니다.

fapolicyd 프레임워크는 신뢰 개념을 도입합니다. 애플리케이션은 시스템 패키지 관리자에 의해 올바르게 설치될 때 신뢰할 수 있으므로 시스템 RPM 데이터베이스에 등록됩니다. **fapolicyd** 데몬은 RPM 데이터베이스를 신뢰할 수 있는 바이너리 및 스크립트 목록으로 사용합니다. **fapolicyd** RPM 플러그인은 DNF 패키지 관리자 또는 RPM Package Manager에서 처리하는 시스템 업데이트를 등록합니다. 플러그인은 이 데이터베이스의 변경 사항에 대해 **fapolicyd** 데몬에 알립니다. 다른 방법으로 애플리케이션을 추가하려면 사용자 지정 규칙을 생성하고 **fapolicyd** 서비스를 다시 시작해야 합니다.

fapolicyd 서비스 구성은 다음과 같은 구조가 있는 **/etc/fapolicyd/** 디렉터리에 있습니다.

- **/etc/fapolicyd/fapolicyd.trust** 파일에는 신뢰할 수 있는 파일 목록이 포함되어 있습니다. **/etc/fapolicyd/trust.d/** 디렉터리에서 여러 신뢰 파일을 사용할 수도 있습니다.
- **allow** 및 **deny** 실행 규칙이 포함된 파일의 **/etc/fapolicyd/rules.d/** 디렉토리입니다. **fagenrules** 스크립트는 이러한 구성 요소 규칙 파일을 **/etc/fapolicyd/ compiled.rules** 파일에 병합합니다.
- **fapolicyd.conf** 파일에는 데몬의 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 이름 시작 부분에 있는 숫자는 **/etc/fapolicyd/ compiled.rules** 의 순서를 결정합니다.

- 10 - 언어 규칙
- 20 - 배치 관련 규칙
- 21 - 업데이트자 규칙
- 30 - 패턴
- 40 - ELF 규칙

- 41 - 공유 개체 규칙
- 42 - 신뢰할 수 있는 ELF 규칙
- 70 - 신뢰할 수 있는 언어 규칙
- 72 - 쉘 규칙
- 90 - 실행 규칙 거부
- 95% - 오픈 규칙 허용

fapolicyd 무결성 검사 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 따라 무결성 검사 속도가 빠르지만 공격자는 파일의 내용을 교체하고 바이트 크기를 유지할 수 있습니다. SHA-256 체크섬을 계산하고 확인하는 것이 더 안전하지만 시스템의 성능에 영향을 미칩니다. **fapolicyd.conf** 의 **integrity = ima** 옵션에는 실행 파일이 포함된 모든 파일 시스템에서 확장 속성(*xattr*라고도 함)을 지원해야 합니다.

추가 리소스

- **fapolicyd(8)**, **fapolicyd.rules(5)**, **fapolicyd.conf(5)**, **fapolicyd.trust(13)**, **fagenrules(8)** 및 **fapolicyd-cli(1)** 매뉴얼 페이지.
- 커널 관리, 모니터링 및 업데이트 문서의 커널 무결성 하위 시스템으로 보안 활성화 장
- **/usr/share/doc/fapolicyd/** 디렉터리 및 **/usr/share/fapolicyd/sample-rules/README-rules** 파일에 **fapolicyd** 패키지로 설치된 설명서입니다.

13.2. FAPOLICYD 배포

RHEL에서 **fapolicyd** 프레임워크를 배포하려면 다음을 수행합니다.

절차

1. **fapolicyd** 패키지를 설치합니다.

```
# dnf 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 로그인하여 **fapolicyd** 가 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

13.3. 추가 신뢰 소스를 사용하여 파일을 신뢰할 수 있는 것으로 표시

fapolicyd 프레임워크는 RPM 데이터베이스에 포함된 파일을 신뢰합니다. **/etc/fapolicyd/fapolicyd.trust** 일반 텍스트 파일 또는 **/etc/fapolicyd/trust.d/** 디렉터리에 해당 항목을 추가하여 추가 파일을 신뢰할 수 있는 것으로 표시할 수 있으며, 신뢰할 수 있는 파일 목록을 더 많은 파일에 분리할 수 있습니다. 텍스트 편집기를 사용하거나 **fapolicyd -cli** 명령을 통해 직접 **/etc/fapolicyd/trust.d** 의 파일을 수정할 수 있습니다.

참고

사용자 지정 **fapolicyd** 규칙을 작성하는 대신 **fapolicyd.trust** 또는 **trust.d/** 를 사용하여 파일을 신뢰할 수 있는 것으로 표시하는 것을 선호합니다.

사전 요구 사항

- **fapolicyd** 프레임워크가 시스템에 배포됩니다.

절차

1. 사용자 정의 바이너리를 필요한 디렉터리에 복사합니다. 예를 들면 다음과 같습니다.

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

2. 사용자 지정 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 명령에서 해당 행을 **/etc/fapolicyd/fapolicyd.trust** 에 추가합니다.

3. **fapolicyd** 데이터베이스를 업데이트합니다.

```
# fapolicyd-cli --update
```

4. 재시작 **fapolicyd**:

```
# systemctl restart fapolicyd
```

검증

1. 사용자 정의 바이너리가 이제 실행될 수 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ /tmp/ls
ls
```

추가 리소스

- **fapolicyd.trust(13)** 도움말 페이지.

13.4. FAPOLICYD에 대한 사용자 정의 허용 및 거부 규칙 추가

fapolicyd 패키지의 기본 규칙 세트는 시스템 기능에 영향을 미치지 않습니다. 비표준 디렉터리에 바이너리 및 스크립트를 저장하거나 **dnf** 또는 **rpm** 설치 프로그램이 없는 애플리케이션 추가와 같은 사용자 정의 시나리오의 경우 추가 파일을 신뢰할 수 있는 것으로 표시하거나 새 사용자 지정 규칙을 추가해야 합니다.

기본 시나리오 의 경우 추가 신뢰 소스를 사용하여 파일을 신뢰할 수 있는 것으로 표시하는 것을 선호합니다. 특정 사용자 및 그룹 식별자에 대해서만 사용자 지정 바이너리를 실행할 수 있는 등의 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** 프레임워크가 시스템에 배포됩니다.

절차

1. 사용자 정의 바이너리를 필요한 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 자세한 정보이며 **Ctrl**+**C** 눌러만 중지하거나 해당 프로세스를 종료한 후 오류 출력을 파일로 리디렉션할 수 있습니다. 이 경우 **--debug**: 대신 **--debug-deny** 옵션을 사용하여 출력에 액세스 거부만 제한할 수 있습니다.

```
# fapolicyd --debug-deny 2> fapolicy.output &
[1] 51341
```

또는 다른 터미널에서 **fapolicyd** debug 모드를 실행할 수 있습니다.

4. **fapolicyd** 가 거부한 명령을 반복합니다.

```
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

5. 전면에서 디버그 모드를 다시 시작하고 **Ctrl**+**C** 눌러 디버그 모드를 중지하십시오.

```
# fg
fapolicyd --debug 2> fapolicy.output
^C
...
```

또는 **fapolicyd** 디버그 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/** 디렉터리에서 사용자 지정 바이너리 실행을 거부하는 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/ all trust=0
```

9. 사용자 정의 바이너리의 콘텐츠 변경을 방지하려면 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 컴파일된 목록이 **/etc/fapolicyd/rules.d/** 의 규칙 세트와 다른지 확인하고 **/etc/fapolicyd/alias.rules 파일에 저장된 목록을 업데이트합니다.**

```
# fagenrules --check
/usr/sbin/fagenrules: Rules have changed and should be updated
# fagenrules --load
```

11. 실행을 방지하는 규칙보다 먼저 사용자 정의 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 바이너리가 이제 실행될 수 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ /tmp/ls
ls
```

추가 리소스

- **fapolicyd.rules(5)** 및 **fapolicyd-cli(1)** 매뉴얼 페이지.
- **/usr/share/fapolicyd/sample-rules/README-rules** 파일에 **fapolicyd** 패키지로 설치된 문서입니다.

13.5. 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. IMA(Integrity Measurement Architecture) 하위 시스템을 사용하여 무결성 검사를 설정할 수도 있습니다.

사전 요구 사항

- **fapolicyd** 프레임워크가 시스템에 배포됩니다.

절차

1. 선택한 텍스트 편집기에서 **/etc/fapolicyd/fapolicyd.conf** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vi /etc/fapolicyd/fapolicyd.conf
```

2. **무결성** 옵션 값을 **none** 에서 **sha256** 로 변경하고 파일을 저장한 후 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
```

13.6. FAPOLICYD와 관련된 문제 해결

다음 섹션에서는 **rpm** 명령을 사용하여 애플리케이션을 추가하기 위한 **fapolicyd** 애플리케이션 프레임워크 및 지침의 기본 문제 해결에 대한 팁을 제공합니다.

rpm을 사용하여 애플리케이션 설치

- **rpm** 명령을 사용하여 애플리케이션을 설치하는 경우 **fapolicyd** RPM 데이터베이스를 수동으로 새로 고쳐야 합니다.

1. *애플리케이션* 설치:

```
# rpm -i application.rpm
```

2. 데이터베이스를 새로 고침합니다.

```
# fapolicyd-cli --update
```

이 단계를 건너뛰면 시스템이 정지될 수 있으며 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. 예를 들면 다음과 같습니다.

```
# 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 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 모드에서는 일치하는 규칙, 데이터베이스 상태 등에 대한 자세한 정보를 제공합니다. **fapolicyd** 를 디버그 모드로 전환하려면 다음을 수행합니다.

1. **fapolicyd** 서비스를 중지합니다.

```
# systemctl stop fapolicyd
```

2. 디버그 모드를 사용하여 해당 규칙을 확인합니다.

```
# fapolicyd --debug
```

fapolicyd --debug 명령의 출력이 자세한 정보이므로 오류 출력을 파일로 리디렉션할 수 있습니다.

```
# fapolicyd --debug 2> fapolicy.output
```

또는 **fapolicyd** 가 액세스를 거부할 때 항목으로만 출력을 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 포함되어 있습니다. 데이터베이스를 덤프한 후 항목을 확인할 수 있습니다.

```
# fapolicyd-cli --dump-db
```

애플리케이션 파이프

- 드문 경우지만 **fapolicyd** pipe 파일을 제거하면 잠금을 해결할 수 있습니다.

```
# rm -f /var/run/fapolicyd/fapolicyd.fifo
```

추가 리소스

- **fapolicyd-cli(1)** 매뉴얼 페이지.

13.7. 추가 리소스

- **fapolicyd- man -k fapolicyd** 명령을 사용하여 나열된 관련 도움말 페이지.
- FOSDEM 2020 fapolicyd 프레젠테이션입니다.

14장. 침입 USB 장치로부터 시스템 보호

USB 장치는 anti, malware, 트로이 목마 또는 트로이 목마를 사용하여 로드할 수 있습니다.이 장치는 데이터를 도용하거나 시스템을 손상시킬 수 있습니다. Red Hat Enterprise Linux 관리자는 **USBGuard** 를 사용하여 USB 공격을 방지할 수 있습니다.

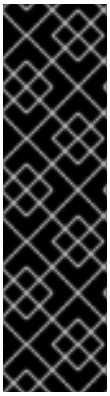
14.1. USBGUARD

USBGuard 소프트웨어 프레임워크를 사용하면 커널의 USB 장치 권한 부여 기능을 기반으로 허용되거나 금지된 장치의 기본 목록을 사용하여 시스템을 침입 USB 장치로부터 보호할 수 있습니다.

USBGuard 프레임워크는 다음 구성 요소를 제공합니다.

- 동적 상호 작용 및 정책 적용을 위한 프로세스 간 통신(IPC) 인터페이스를 사용하는 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 작용을 위한 C++ API

usbguard 시스템 서비스 구성 파일(**/etc/usbguard/usbguard-daemon.conf**)에는 IPC 인터페이스를 사용하도록 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 사용자로 제한됩니다.

IPCAccessControlFiles 옵션 (권장) 또는 **IPCAllowedUsers** 및 **IPCAllowedGroups** 옵션을 설정하여 IPC 인터페이스에 대한 액세스를 제한하는 것이 좋습니다.

IPC 인터페이스가 모든 로컬 사용자에게 노출되고 USB 장치의 권한 부여 상태를 조작하고 USBGuard 정책을 수정할 수 있으므로 ACL(액세스 제어 목록)을 구성 해제하지 않도록 합니다.

14.2. USBGUARD 설치

이 절차를 사용하여 **USBGuard** 프레임워크를 설치하고 시작합니다.

절차

1. **usbguard** 패키지를 설치합니다.

```
# dnf 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)** 매뉴얼 페이지.

14.3. CLI를 사용하여 USB 장치 차단 및 권한 부여

이 절차에서는 **usbguard** 명령을 사용하여 USB 장치를 인증하고 차단하는 방법을 간략하게 설명합니다.

사전 요구 사항

- **usbguard** 서비스가 설치되어 실행 중입니다.

절차

1. **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
```

2. 시스템과 상호 작용하도록 장치 *6* 을 인증합니다.

```
# usbguard allow-device 6
```

3. 장치 *6* 의 인증 해제 및 제거:

```
# usbguard reject-device 6
```

4. 장치 *6* 의 인증을 해제하고 유지합니다.

```
# usbguard block-device 6
```

참고

usbguard 는 *블록을* 사용하고 다음과 같은 의미와 함께 *거부합니다*.

- *block*: 현재 이 장치와 상호 작용하지 마십시오.
- *reject*:이 장치가 없는 것처럼 무시합니다.

추가 리소스

- **usbguard(1)** 도움말 페이지.
- **usbguard --help** 명령을 사용하여 나열된 기본 도움말입니다.

14.4. USB 장치를 영구적으로 차단 및 인증

-p 옵션을 사용하여 USB 장치를 영구적으로 차단 및 인증할 수 있습니다. 그러면 현재 정책에 장치별 규칙이 추가됩니다.

사전 요구 사항

- **usbguard** 서비스가 설치되어 실행 중입니다.

절차

1. **usbguard** 데몬이 규칙을 작성할 수 있도록 SELinux를 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 꺼지면 전원을 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
```

```
"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
```

3. 시스템과 상호 작용하도록 장치 *6* 에 영구적으로 권한을 부여합니다.

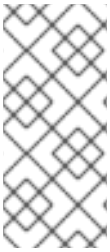
```
# usbguard allow-device 6 -p
```

4. 영구적으로 장치의 인증 해제 및 제거 6:

```
# usbguard reject-device 6 -p
```

5. *영구적으로 인증 해제 및 장치 6 유지:*

```
# usbguard block-device 6 -p
```



참고

usbguard *는 terms block 을 사용하고 다음과 같은 의미를 거부합니다.*

- *block: 현재 이 장치와 상호 작용하지 마십시오.*
- *reject: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.*

14.5. USB 장치용 사용자 정의 정책 생성

다음 절차에서는 시나리오의 요구 사항을 반영하는USB 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 인증하는 정책을 만들고 생성된 규칙을* ***rules.conf*** *파일에 저장합니다.*

```
# usbguard generate-policy --no-hashes > ./rules.conf
```

   no-hashes *옵션은 장치에 대한 해시 속성을 생성하지 않습니다. 구성 설정의 해시 속성이 영구적이 아닐 수 있으므로 피하십시오.*

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:*:* }
```

   규칙 언어 설명 및 자세한 예를 보려면 ***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 다시 시작하여 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.*

14.6. USB 장치에 대한 구조화된 사용자 정의 정책 생성

사용자 지정 USBGuard 정책은 ***/etc/usbguard/rules.d/*** *디렉터리 내의 여러* ***.conf*** *파일로 구성할 수 있습니다. 그런 다음* ***usbguard-daemon*** *은 기본* ***rules.conf*** *파일과 디렉터리 내의* ***.conf*** *파일을 알파벳순으로 결합합니다.*

사전 요구 사항

- ***usbguard*** *서비스가 설치되어 실행 중입니다.*

절차

1. 현재 연결된 USB 장치를 인증하는 정책을 만들고 생성된 규칙을 새 **.conf** 파일에 저장합니다(예: **policy.conf**).

```
# usbguard generate-policy --no-hashes > ./policy.conf
```

no-hashes 옵션은 장치에 대한 해시 속성을 생성하지 않습니다. 구성 설정의 해시 속성이 영구적이 아닐 수 있으므로 피하십시오.

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 이름 시작 부분에 있는 두 자리는 데몬이 구성 파일을 읽는 순서를 지정합니다.

예를 들어 키보드의 규칙을 새 **.conf** 파일에 복사합니다.

```
# grep "USB Keyboard" ./policy.conf > ./10keyboards.conf
```

4. **/etc/usbguard/rules.d/** 디렉터리에 새 정책을 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. 활성 **USBGuard** 규칙을 모두 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*** *파일과 모든* ***.conf*** *파일을 표시합니다.*

```
# cat /etc/usbguard/rules.conf /etc/usbguard/rules.d/*.conf
```

3. *활성 규칙에 파일의 모든 규칙이 포함되고 올바른 순서로 있는지 확인합니다.*

추가 리소스

- ***usbguard-rules.conf(5)*** *도움말 페이지.*

14.7. USBGUARD IPC 인터페이스를 사용하도록 사용자 및 그룹 인증

USBGuard 공용 IPC 인터페이스를 사용하도록 특정 사용자 또는 그룹을 인증하려면 다음 절차를 사용하십시오. 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 편집기로* ***/etc/usbguard/usbguard-daemon.conf*** *파일을 편집합니다.*

```
# vi /etc/usbguard/usbguard-daemon.conf
```

2. *예를 들어,* ***wheel*** *그룹의 모든 사용자가 IPC 인터페이스를 사용하도록 허용하는 규칙이 있는 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 갖습니다. 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 다시 시작하여 변경 사항을 적용합니다.*

```
# systemctl restart usbguard
```

추가 리소스

- ***usbguard(1)*** *및* ***usbguard-rules.conf(5)*** *도움말 페이지.*

14.8. LINUX 감사 로그에 USBGUARD 권한 부여 이벤트 로깅

다음 단계에 따라 USBguard 권한 부여 이벤트의 로깅을 표준 Linux 감사 로그에 통합하십시오. 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 다시 시작하여 구성 변경 사항을 적용합니다.

```
# systemctl restart usbguard
```

검증

1. USB 권한 부여 이벤트에 대한 **감사** 데몬 로그를 쿼리합니다. 예를 들면 다음과 같습니다.

```
# ausearch -ts recent -m USER_DEVICE
```

추가 리소스

- **usbguard-daemon.conf(5)** 도움말 페이지.

14.9. 추가 리소스

- **usbguard(1)**, **usbguard-rules.conf(5)**, **usbguard-daemon(8)** 및 **usbguard-daemon.conf(5)** 도움말 페이지.
- USBGuard 홈 페이지.

15장. 원격 로깅 솔루션 구성

사용자 환경의 다양한 머신의 로그가 로깅 서버에 중앙에 기록되도록 **Rsyslog** 애플리케이션을 구성하여 클라이언트 시스템에서 서버로 특정 기준을 충족하는 로그를 기록할 수 있습니다.

15.1. RSYSLOG 로깅 서비스

systemd-journald 서비스와 함께 Rsyslog 애플리케이션은 Red Hat Enterprise Linux에서 로컬 및 원격 로깅 지원을 제공합니다. ***rsyslogd*** 데몬은 저널에서 ***systemd-journald*** 서비스에서 수신한 ***syslog*** 메시지를 지속적으로 읽습니다. ***rsyslogd*** 는 이러한 ***syslog*** 이벤트를 필터링 및 처리하고 ***rsyslog*** 로그 파일에 기록하거나 구성에 따라 다른 서비스로 전달합니다.

또한 ***rsyslogd*** 데몬은 메시지, 입력 및 출력 모듈의 확장 필터링, 암호화 보호 릴레이, TCP 및 UDP 프로토콜을 사용하여 전송에 대한 지원을 제공합니다.

rsyslog의 기본 설정 파일인 /etc/tekton.conf 에서 ***rsyslog d*** 가 메시지를 처리하는 규칙을 지정할 수 있습니다. 일반적으로 메시지를 소스 및 주제(심각) 및 긴급(우선)별로 분류한 다음 메시지가 이러한 기준에 맞을 때 수행해야 하는 작업을 할당할 수 있습니다.

/etc/tekton.conf에서 rsyslog d 에서 유지 관리하는 로그 파일 목록을 볼 수도 있습니다. 대부분의 로그 파일은 ***/var/log/*** 디렉토리에 있습니다. ***httpd*** 및 ***samba*** 와 같은 일부 애플리케이션은 로그 파일을 ***/var/log/*** 내의 하위 디렉터리에 저장합니다.

추가 리소스

- ***rsyslogd(8)*** 및 ***rsyslog.conf(5)*** 매뉴얼 페이지.
- ***/usr/share/doc/#189/html/index.html 파일의 rsyslog-doc*** 패키지로 설치된 설명서.

15.2. RSYSLOG 문서 설치

Rsyslog 애플리케이션에는 https://www.rsyslog.com/doc/ 에서 사용할 수 있는 광범위한 온라인 문서가 있지만 ***rsyslog-doc*** 설명서 패키지를 로컬로 설치할 수도 있습니다.

사전 요구 사항

- 시스템에서 ***AppStream*** 리포지토리를 활성화했습니다.
- ***sudo*** 를 사용하여 새 패키지를 설치할 수 있습니다.

절차

- ***rsyslog-doc*** 패키지를 설치합니다.

```
# dnf install rsyslog-doc
```

검증

- 선택한 브라우저에서 ***/usr/share/doc/#189/html/index.html*** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
$ firefox /usr/share/doc/rsyslog/html/index.html &
```

15.3. TCP를 통한 원격 로깅을 위한 서버 구성

Rsyslog 애플리케이션을 사용하면 로깅 서버를 실행하고 로그 파일을 로깅 서버로 보내도록 개별 시스템을 구성할 수 있습니다. TCP를 통해 원격 로깅을 사용하려면 서버와 클라이언트를 모두 구성합니다. 서버는 하나 이상의 클라이언트 시스템에서 보낸 로그를 수집 및 분석합니다.

Rsyslog 애플리케이션을 사용하면 로그 메시지가 네트워크를 통해 서버로 전달되는 중앙 집중식 로깅 시스템을 유지 관리할 수 있습니다. 서버를 사용할 수 없는 경우 메시지 손실을 방지하기 위해 전달 작업에 대한 작업 대기열을 구성할 수 있습니다. 이렇게 하면 전송되지 못한 메시지는 서버에 다시 연결할 때까지 로컬로 저장됩니다. 이러한 대기열은 UDP 프로토콜을 사용하는 연결에 대해 구성할 수 없습니다.

omfwd 플러그인은 UDP 또는 TCP를 통해 전달 기능을 제공합니다. 기본 프로토콜은 UDP입니다. 플러그인이 내장되어 있으므로 로드할 필요가 없습니다.

기본적으로 **rsyslog** 는 포트 **514** 에서 TCP를 사용합니다.

사전 요구 사항

- rsyslog가 서버 시스템에 설치되어 있습니다.
- 서버에 **root** 로 로그인되어 있습니다.
- **semanage** 명령을 사용하여 선택적 단계에 대해 **policycoreutils-python-utils** 패키지가 설치됩니다.
- **firewalld** 서비스가 실행 중입니다.

절차

1. 선택 사항: **rsyslog** 트래픽에 다른 포트를 사용하려면 **syslogd_port_t** SELinux 유형을 포트에 추가합니다. 예를 들어 포트 **30514** 를 활성화합니다.

```
# semanage port -a -t syslogd_port_t -p tcp 30514
```

2. 선택 사항: **rsyslog** 트래픽에 다른 포트를 사용하려면 해당 포트에서 들어오는 **rsyslog** 트래픽을 허용하도록 **firewalld** 를 구성합니다. 예를 들어, 포트 **30514** 에서 TCP 트래픽을 허용하십시오.

```
# firewall-cmd --zone=<zone-name> --permanent --add-port=30514/tcp
success
# firewall-cmd --reload
```

3. **/etc/#189.d/** 디렉터리에 새 파일(예: **remotelog.conf**)을 생성하고 다음 내용을 삽입합니다.

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
```

```
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides TCP syslog reception
module(load="imtcp")

# Adding this ruleset to process remote messages
ruleset(name="remote1"){
    authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
     *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="30514" ruleset="remote1")
```

4. ***/etc/qcow.d/remotelog.conf*** *파일에 변경 사항을 저장합니다.*

5. ***/etc/qcow.conf 파일의 구문을 테스트합니다.***

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

6. ***rsyslog*** *서비스가 실행 중이고 로깅 서버에서 활성화되었는지 확인합니다.*

```
# systemctl status rsyslog
```

7. ***rsyslog*** *서비스를 다시 시작합니다.*

```
# systemctl restart rsyslog
```

8. *선택 사항:* ***rsyslog*** *가 활성화되지 않은 경우 재부팅 후* ***rsyslog*** *서비스가 자동으로 시작되는지 확인합니다.*

```
# systemctl enable rsyslog
```

이제 환경의 다른 시스템에서 로그 파일을 수신하고 저장하도록 로그 서버가 구성되어 있습니다.

추가 리소스

- ***rsyslogd(8)***, ***rsyslog.conf(5)***, ***semanage(8)***, and ***firewall-cmd(1)*** man pages.
- ***/usr/share/doc/#189/html/index.html 파일의 rsyslog-doc*** *패키지로 설치된 설명서.*

15.4. TCP를 통해 서버에 대한 원격 로깅 구성

TCP 프로토콜을 통해 로그 메시지를 서버로 전달하도록 시스템을 구성하려면 다음 절차를 따르십시오. ***omfwd*** *플러그인은 UDP 또는 TCP를 통해 전달 기능을 제공합니다. 기본 프로토콜은 UDP입니다. 플러그인이 내장되어 있으므로 이를 로드할 필요가 없습니다.*

사전 요구 사항

- ***rsyslog*** *패키지는 서버에 보고해야 하는 클라이언트 시스템에 설치됩니다.*
- *원격 로깅을 위해 서버를 구성했습니다.*
- *SELinux에서 지정된 포트가 허용되고 방화벽에서 열립니다.*
- *시스템에는 SELinux 구성에 비표준 포트를 추가하기 위해* ***semanage*** *명령을 제공하는* ***policycoreutils-python-utils*** *패키지가 포함되어 있습니다.*

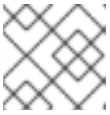
절차

1. ***/etc/#189.d/*** *디렉터리에 (예:* ***10-remotelog.conf*** *)에 새 파일을 생성하고 다음 내용을 삽입합니다.*

```
*.* action(type="omfwd"
    queue.type="linkedlist"
    queue.filename="example_fwd"
    action.resumeRetryCount="-1"
    queue.saveOnShutdown="on"
    target="example.com" port="30514" protocol="tcp"
   )
```

   다음과 같습니다.

   - ***queue.type="linkedlist"*** *는 LinkedList in-memory 큐를 활성화합니다.*
   - ***queue.filename*** *은 디스크 스토리지를 정의합니다. 백업 파일은 이전 글로벌* ***workDirectory*** *지시문으로 지정된 작업 디렉터리에서* ***example_fwd*** *접두사를 사용하여 생성됩니다.*
   - ***action.resumeRetryCount -1*** *설정은 서버가 응답하지 않는 경우 연결을 시도할 때* ***rsyslog*** *가 메시지를 삭제하지 못하도록 합니다.*
   - *활성화된* ***queue.saveOnShutdown="on"*** *은* ***rsyslog*** *가 종료되면 메모리 내 데이터를 저장합니다.*
   - *마지막 줄은 수신된 모든 메시지를 로깅 서버로 전달하며 port 사양은 선택 사항입니다. 이 구성을 사용하면* ***rsyslog*** *가 서버에 메시지를 전송하지만 원격 서버에 연결할 수 없는 경우 메모리에 메시지를 유지합니다. 디스크의 파일은* ***rsyslog*** *가 구성된 메모리 대기열 공간을 벗어나거나 시스템 성능을 종료해야 하는 경우에만 생성됩니다.*



참고

rsyslog는 설정 파일 /etc/#189.d/ *를 어휘 순서로 처리합니다.*

2. ***rsyslog*** *서비스를 다시 시작합니다.*

```
# systemctl restart rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음 단계를 따르십시오.

1. *클라이언트 시스템에서 테스트 메시지를 보냅니다.*

```
# logger test
```

2. 서버 시스템에서 **/var/log/ECDHE 로그를 확인합니다.** 예를 들면 다음과 같습니다.

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

여기서 hostname 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **루트**).

추가 리소스

- **rsyslogd(8)** 및 **rsyslog.conf(5)** 매뉴얼 페이지.
- **/usr/share/doc/#189/html/index.html 파일의 rsyslog-doc** 패키지로 설치된 설명서.

15.5. TLS 암호화 원격 로깅 구성

기본적으로 Rsyslog는 원격-로깅 통신을 일반 텍스트 형식으로 보냅니다. 시나리오가 이 통신 채널을 보안해야 하는 경우 TLS를 사용하여 암호화할 수 있습니다.

TLS를 통해 암호화된 전송을 사용하려면 서버와 클라이언트를 둘 다 구성합니다. 서버는 하나 이상의 클라이언트 시스템에서 보낸 로그를 수집 및 분석합니다.

ossl 네트워크 스트림 드라이버(OpenSSL) 또는 **gtls** 스트림 드라이버(GnuTLS)를 사용할 수 있습니다.

참고

보안이 더 높은 시스템(예: 네트워크에 연결되지 않았거나 엄격한 권한 부여가 있는 시스템)이 있는 경우 별도의 시스템을 인증 기관(CA)으로 사용하십시오.

사전 요구 사항

- 클라이언트 및 서버 시스템에 대한 **루트** 액세스 권한이 있습니다.
- **rsyslog** 및 **rsyslog-openssl** 패키지는 서버 및 클라이언트 시스템에 설치됩니다.
- **gtls** 네트워크 스트림 드라이버를 사용하는 경우 **rsyslog-openssl** 대신 **rsyslog-gnutls** 패키지를 설치합니다.
- **certtool** 명령을 사용하여 인증서를 생성하는 경우 **gnutls-utils** 패키지를 설치합니다.
- 로깅 서버에서 다음 인증서는 **/etc/pki/ca-trust/source/anchors/** 디렉터리에 있으며 **update-ca-trust** 명령을 사용하여 시스템 구성이 업데이트됩니다.
  - **ca-cert.pem** - 로깅 서버 및 클라이언트에서 키와 인증서를 확인할 수 있는 CA 인증서입니다.
  - **server-cert.pem** - 로깅 서버의 공개 키입니다.
  - **server-key.pem** - 로깅 서버의 개인 키입니다.
- 로깅 클라이언트에서 다음 인증서는 **/etc/pki/ca-trust/source/anchors/** 디렉터리에 있으며 **update-ca-trust** 를 사용하여 시스템 구성이 업데이트됩니다.
  - **ca-cert.pem** - 로깅 서버 및 클라이언트에서 키와 인증서를 확인할 수 있는 CA 인증서입니다.

- ***client-cert.pem*** - *클라이언트의 공개 키입니다.*
- ***client-key.pem*** - *클라이언트의 개인 키입니다.*

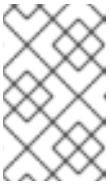
절차

1. *클라이언트 시스템에서 암호화된 로그를 수신하도록 서버를 설정합니다.*

   a. ***/etc/#189.d/*** *디렉터리에 새 파일(예:* ***securelogser.conf*** *)을 만듭니다.*

   b. *통신을 암호화하려면 구성 파일에 서버의 인증서 파일 경로, 선택한 인증 방법 및 TLS 암호화를 지원하는 스트림 드라이버를 포함해야 합니다.* ***/etc/#189.d/securelogser.conf 파일에 다음 행을 추가합니다.***

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/server-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/server-key.pem"
)

# TCP listener
module(
  load="imtcp"
  PermittedPeer=["client1.example.com", "client2.example.com"]
  StreamDriver.AuthMode="x509/name"
  StreamDriver.Mode="1"
  StreamDriver.Name="ossl"
)

# Start up listener at port 514
input(
  type="imtcp"
  port="514"
)
```



참고

GnuTLS 드라이버를 선호하는 경우 ***StreamDriver.Name="gtls"*** *구성 옵션을 사용합니다.* ***x509/name*** *보다 덜 엄격한 인증 모드에 대한 자세한 내용은* ***rsyslog-doc*** *패키지를 사용하여 설치한 설명서를 참조하십시오.*

   c. ***/etc/qcow.d/securelogser.conf*** *파일에 변경 사항을 저장합니다.*

   d. ***/etc/qcow.conf 파일 구문 및 / etc/#189.d/ 디렉터리에 있는 모든 파일을 확인합니다.***

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

   e. ***rsyslog*** *서비스가 실행 중이고 로깅 서버에서 활성화되었는지 확인합니다.*

```
# systemctl status rsyslog
```

f. ***rsyslog*** *서비스를 다시 시작하십시오.*

```
# systemctl restart rsyslog
```

g. *선택 사항: Rsyslog가 활성화되지 않은 경우 재부팅 후* ***rsyslog*** *서비스가 자동으로 시작되는지 확인합니다.*

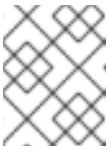
```
# systemctl enable rsyslog
```

2. *암호화된 로그를 서버로 전송하기 위해 클라이언트를 설정합니다.*

a. *클라이언트 시스템에서* ***/etc/#189.d/*** *디렉터리에 새 파일(예:* ***securelogcli.conf****)을 만듭니다.*

b. ***/etc/#189.d/securelogcli.conf 파일에 다음 행을 추가합니다.***

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/client-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/client-key.pem"
)


# Set up the action for all messages
*.* action(
  type="omfwd"
  StreamDriver="ossl"
  StreamDriverMode="1"
  StreamDriverPermittedPeers="server.example.com"
  StreamDriverAuthMode="x509/name"
  target="server.example.com" port="514" protocol="tcp"
)
```



참고

GnuTLS 드라이버를 선호하는 경우 ***StreamDriver.Name="gtls"*** *구성 옵션을 사용합니다.*

c. ***/etc/qcow.d/securelogser.conf*** *파일에 변경 사항을 저장합니다.*

d. ***/etc/#189.conf 파일 및 /etc/#189. d/ 디렉터리에서 기타 파일의 구문을 확인합니다.***

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

e. ***rsyslog*** *서비스가 실행 중이고 로깅 서버에서 활성화되었는지 확인합니다.*

```
# systemctl status rsyslog
```

f. ***rsyslog*** *서비스를 다시 시작하십시오.*

```
# systemctl restart rsyslog
```

g. 선택 사항: Rsyslog가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음 단계를 따르십시오.

1. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

2. 서버 시스템에서 **/var/log/ECDHE 로그를 확인합니다.** 예를 들면 다음과 같습니다.

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

여기서 **hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 logger 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **루트**).

추가 리소스

- **certtool(1)**, **openssl(1)**, **update-ca-trust(8)**, **rsyslogd(8)** 및 **rsyslog.conf(5)** 매뉴얼 페이지.
- **/usr/share/doc/#189/html/index.html에서 rsyslog-doc** 패키지로 설치된 설명서.
- *로깅 시스템 역할 사용* 문서입니다.

15.6. UDP를 통해 원격 로깅 정보를 수신하기 위한 서버 구성

Rsyslog 애플리케이션을 사용하면 원격 시스템에서 로깅 정보를 수신하도록 시스템을 구성할 수 있습니다. UDP를 통해 원격 로깅을 사용하려면 서버와 클라이언트를 둘 다 구성합니다. 수신 서버는 하나 이상의 클라이언트 시스템에서 보낸 로그를 수집 및 분석합니다. 기본적으로 **rsyslog** 는 포트 **514** 에서 UDP를 사용하여 원격 시스템에서 로그 정보를 수신합니다.

UDP 프로토콜을 통해 하나 이상의 클라이언트 시스템에서 보낸 로그를 수집하고 분석하도록 서버를 구성하려면 다음 절차를 따르십시오.

사전 요구 사항

- rsyslog가 서버 시스템에 설치되어 있습니다.
- 서버에 **root** 로 로그인되어 있습니다.
- **semanage** 명령을 사용하여 선택적 단계에 대해 **policycoreutils-python-utils** 패키지가 설치됩니다.
- **firewalld** 서비스가 실행 중입니다.

절차

1. *선택 사항:* ***rsyslog*** *트래픽에 대해 기본 포트* ***514*** *와 다른 포트를 사용하려면 다음을 수행합니다.*

   a. ***syslogd_port_t*** *SELinux 유형을 SELinux 정책 구성에 추가하고* ***portno*** *를* ***rsyslog*** *가 사용하려는 포트 번호로 바꿉니다.*

   ```
   # semanage port -a -t syslogd_port_t -p udp portno
   ```

   b. *들어오는* ***rsyslog*** *트래픽을 허용하고* ***portno*** *를 포트 번호 및* ***영역으로 rsyslog*** *를 사용하려는 영역으로 교체하도록* ***firewalld*** *를 구성합니다.*

   ```
   # firewall-cmd --zone=zone --permanent --add-port=portno/udp
   success
   # firewall-cmd --reload
   ```

   c. *방화벽 규칙을 다시 로드합니다.*

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

2. ***/etc/dpdk.d/ 디렉터리에 새 . conf*** *파일을 만듭니다(예:* ***remotelogserv.conf*** *)는 다음 내용을 삽입합니다.*

   ```
   # Define templates before the rules that use them
   # Per-Host templates for remote systems
   template(name="TmplAuthpriv" type="list") {
      constant(value="/var/log/remote/auth/")
      property(name="hostname")
      constant(value="/")
      property(name="programname" SecurePath="replace")
      constant(value=".log")
      }

   template(name="TmplMsg" type="list") {
      constant(value="/var/log/remote/msg/")
      property(name="hostname")
      constant(value="/")
      property(name="programname" SecurePath="replace")
      constant(value=".log")
      }

   # Provides UDP syslog reception
   module(load="imudp")

   # This ruleset processes remote messages
   ruleset(name="remote1"){
       authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
         *.info;mail.none;authpriv.none;cron.none
   action(type="omfile" DynaFile="TmplMsg")
   }

   input(type="imudp" port="514" ruleset="remote1")
   ```

   여기서 ***514*** *는 기본적으로* ***rsyslog*** *가 사용하는 포트 번호입니다. 대신 다른 포트를 지정할 수 있습니다.*

3. ***/etc/qcow.conf 파일 및 /etc/***dpdk. ***d/ 디렉토리에 있는 모든 . conf*** *파일의 구문을 확인합니다.*

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

4. **rsyslog** 서비스를 다시 시작합니다.

```
# systemctl restart rsyslog
```

5. 선택 사항: **rsyslog** 가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

추가 리소스

- **rsyslogd(8)**, **rsyslog.conf(5)**, **semanage(8)**, and **firewall-cmd(1)** man pages.
- **/usr/share/doc/#189/html/index.html 파일의 rsyslog-doc** 패키지로 설치된 설명서.

15.7. UDP를 통해 서버에 대한 원격 로깅 구성

UDP 프로토콜을 통해 서버로 로그 메시지를 전달하도록 시스템을 구성하려면 다음 절차를 따르십시오. **omfwd** 플러그인은 UDP 또는 TCP를 통해 전달 기능을 제공합니다. 기본 프로토콜은 UDP입니다. 플러그인이 내장되어 있으므로 이를 로드할 필요가 없습니다.

사전 요구 사항

- **rsyslog** 패키지는 서버에 보고해야 하는 클라이언트 시스템에 설치됩니다.
- UDP를 통해 원격 로깅 정보를 받기 위해 서버 구성에 설명된 대로 원격 로깅 서버를 구성했습니다.

절차

1. **/etc/#189.d/ 디렉터리에 새 .conf** 파일을 만듭니다(예: **10-remotelogcli.conf**)는 다음 내용을 삽입합니다.

```
*.* action(type="omfwd"
    queue.type="linkedlist"
    queue.filename="example_fwd"
    action.resumeRetryCount="-1"
    queue.saveOnShutdown="on"
    target="example.com" port="portno" protocol="udp"
   )
```

다음과 같습니다.

- **queue.type="linkedlist"** 는 LinkedList in-memory 큐를 활성화합니다.
- **queue.filename** 은 디스크 스토리지를 정의합니다. 백업 파일은 이전 글로벌 **workDirectory** 지시문으로 지정된 작업 디렉터리에서 **example_fwd** 접두사를 사용하여 생성됩니다.

- ***action.resumeRetryCount -1*** *설정은 서버가 응답하지 않는 경우 연결을 시도할 때* ***rsyslog*** *가 메시지를 삭제하지 못하도록 합니다.*
- ***활성화된 queue.saveOnShutdown="on"*** *은* ***rsyslog*** *가 종료되면 메모리 내 데이터를 저장합니다.*
- ***portno*** *는* ***rsyslog*** *가 사용할 포트 번호입니다. 기본값은* ***514*** *입니다.*
- *마지막 줄은 수신된 모든 메시지를 로깅 서버로 전달하며 port 사양은 선택 사항입니다. 이 구성을 사용하면* ***rsyslog*** *가 서버에 메시지를 전송하지만 원격 서버에 연결할 수 없는 경우 메모리에 메시지를 유지합니다. 디스크의 파일은* ***rsyslog*** *가 구성된 메모리 대기열 공간을 벗어나거나 시스템 성능을 종료해야 하는 경우에만 생성됩니다.*

참고

rsyslog는 설정 파일 /etc/#189.d/ *를 어휘 순서로 처리합니다.*

2. ***rsyslog*** *서비스를 다시 시작합니다.*

```
# systemctl restart rsyslog
```

3. *선택 사항:* ***rsyslog*** *가 활성화되지 않은 경우 재부팅 후* ***rsyslog*** *서비스가 자동으로 시작되는지 확인합니다.*

```
# systemctl enable rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음 단계를 따르십시오.

1. *클라이언트 시스템에서 테스트 메시지를 보냅니다.*

```
# logger test
```

2. *서버 시스템에서* ***/var/log/remote/msg/hostname/root.log*** *로그를 확인합니다. 예를 들면 다음과 같습니다.*

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

여기서 ***hostname*** *은 클라이언트 시스템의 호스트 이름입니다. 로그에는 logger 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우* ***루트****).*

추가 리소스

- ***rsyslogd(8)*** *및* ***rsyslog.conf(5)*** *매뉴얼 페이지.*
- ***/usr/share/doc/#189/html/index.html에서 rsyslog-doc*** *패키지로 설치된 설명서.*

15.8. RSYSLOG의 로드 밸런싱 도우미

RebindInterval *설정은 현재 연결이 끊어지고 재설정되는 간격을 지정합니다. 이 설정은 TCP, UDP 및 RELP 트래픽에 적용됩니다. 로드 밸런서는 이를 새 연결로 인식하고 메시지를 다른 물리적 대상 시스템으로 전달합니다.*

RebindInterval *설정은 대상 시스템이 IP 주소를 변경하는 시나리오에서 도움이 될 수 있음을 증명합니다. Rsyslog 애플리케이션은 연결이 설정될 때 IP 주소를 캐시하므로 동일한 서버로 메시지가 전송됩니다. IP 주소가 변경되면 Rsyslog 서비스가 다시 시작될 때까지 UDP 패킷이 손실됩니다. 연결을 다시 설정하면 DNS에서 IP를 다시 해결할 수 있습니다.*

```
action(type="omfwd" protocol="tcp" RebindInterval="250" target="example.com" port="514" …)

action(type="omfwd" protocol="udp" RebindInterval="250" target="example.com" port="514" …)

action(type="omrelp" RebindInterval="250" target="example.com" port="6514" …)
```

15.9. 신뢰할 수 있는 원격 로깅 구성

RELP(Reliable Event Logging Protocol)를 사용하면 메시지 손실 위험이 크게 단축되어 TCP를 통해 ***syslog*** *메시지를 보내고 받을 수 있습니다. RELP는 이벤트 메시지를 안정적으로 전달하므로 메시지 손실이 허용되지 않는 환경에서 유용합니다. RELP를 사용하려면 서버에서 실행되고 로그를 수신하는* ***imrelp*** *입력 모듈과 클라이언트에서 실행되는* ***omrelp*** *출력 모듈을 구성하고 로깅 서버에 로그를 보냅니다.*

사전 요구 사항

- ***rsyslog***,***librelp***, ***rsyslog-relp*** *패키지를 서버 및 클라이언트 시스템에 설치했습니다.*
- *SELinux에서 지정된 포트가 허용되며 방화벽에서 열립니다.*

절차

1. *신뢰할 수 있는 원격 로깅을 위해 클라이언트 시스템을 구성합니다.*
   a. *클라이언트 시스템에서* ***/etc/#189.d/ 디렉터리에 (예: relpclient . conf)에 새 .conf*** *파일을 생성하고 다음 내용을 삽입합니다.*

   ```
   module(load="omrelp")
   *.* action(type="omrelp" target="_target_IP_" port="_target_port_")
   ```

   다음과 같습니다.

   - ***target_IP*** *는 로깅 서버의 IP 주소입니다.*
   - ***target_port*** *는 로깅 서버의 포트입니다.*

   b. ***/etc/#189.d/relpclient.conf*** *파일에 변경 사항을 저장합니다.*

   c. ***rsyslog*** *서비스를 다시 시작합니다.*

   ```
   # systemctl restart rsyslog
   ```

   d. *선택 사항:* ***rsyslog*** *가 활성화되지 않은 경우 재부팅 후* ***rsyslog*** *서비스가 자동으로 시작되는지 확인합니다.*

   ```
   # systemctl enable rsyslog
   ```

2. *안정적인 원격 로깅을 위해 서버 시스템을 구성합니다.*

a. *서버 시스템에서 **/etc/#189.d/ 디렉터리에 (예: relpserv . conf)에 새 .conf** 파일을 만들고 다음 내용을 삽입합니다.*

```
ruleset(name="relp"){
*.* action(type="omfile" file="_log_path_")
}


module(load="imrelp")
input(type="imrelp" port="_target_port_" ruleset="relp")
```

다음과 같습니다.

- ***log_path** 는 메시지를 저장하는 경로를 지정합니다.*
- ***target_port** 는 로깅 서버의 포트입니다. 클라이언트 구성 파일에서와 동일한 값을 사용합니다.*

b. ***/etc/qcow.d/relpserv.conf** 파일에 변경 사항을 저장합니다.*

c. ***rsyslog** 서비스를 다시 시작합니다.*

```
# systemctl restart rsyslog
```

d. *선택 사항: **rsyslog** 가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.*

```
# systemctl enable rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음 단계를 따르십시오.

1. *클라이언트 시스템에서 테스트 메시지를 보냅니다.*

```
# logger test
```

2. *서버 시스템에서 지정된 **log_path** 의 로그를 확인합니다. 예를 들면 다음과 같습니다.*

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

*여기서 **hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 logger 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **루트**).*

추가 리소스

- ***rsyslogd(8)** 및 **rsyslog.conf(5)** 매뉴얼 페이지.*
- ***/usr/share/doc/#189/html/index.html 파일의 rsyslog-doc** 패키지로 설치된 설명서.*

15.10. 지원되는 RSYSLOG 모듈

Rsyslog 애플리케이션의 기능을 확장하려면 특정 모듈을 사용할 수 있습니다. 모듈은 추가 입력(Input Modules), 출력(Output Modules) 및 기타 기능을 제공합니다. 모듈은 모듈을 로드한 후 사용 가능한 추가 구성 지시문을 제공할 수도 있습니다.

다음 명령을 입력하여 시스템에 설치된 입력 및 출력 모듈을 나열할 수 있습니다.

```
# ls /usr/lib64/rsyslog/{i,o}m*
```

rsyslog -doc 패키지를 설치한 후 **/usr/share/doc/#189/html/configuration/modules/idx_output.html** 파일에서 사용 가능한 모든 rsyslog 모듈 목록을 볼 수 있습니다.

15.11. 추가 리소스

- **file:///usr/share/doc/dpdk/html/index.html에서 rsyslog-doc** 패키지로 설치된 설명서.
- **rsyslog.conf(5)** 및 **rsyslogd(8)** 매뉴얼 페이지.
- journald가 없는 시스템 로깅 구성 또는 journald 사용 Knowledgebase 문서를 참조하십시오.
- 성능 및 완화 방법에 대한 RHEL 기본 로깅 설정이 미치는 영향은 다음과 같습니다 .
- 로깅 시스템 역할 사용 문서입니다.

16장. 로깅 시스템 역할 사용

시스템 관리자는 로깅 시스템 역할을 사용하여 RHEL 호스트를 로깅 서버로 구성하여 여러 클라이언트 시스템에서 로그를 수집할 수 있습니다.

16.1. 로깅 시스템 역할

로깅 시스템 역할을 사용하면 로컬 및 원격 호스트에 로깅 구성을 배포할 수 있습니다.

하나 이상의 시스템에 로깅 시스템 역할을 적용하려면 플레이북에서 로깅 구성을 정의합니다. 플레이북은 하나 이상의 플레이 목록입니다. 플레이북은 사람이 읽을 수 있으며 YAML 형식으로 작성됩니다. 플레이북에 대한 자세한 내용은 Ansible 문서의 플레이북 작업을 참조하십시오.

플레이북에 따라 구성하려는 시스템 세트는 인벤토리 파일에 정의되어 있습니다. 인벤토리 생성 및 사용에 대한 자세한 내용은 Ansible 설명서에서 인벤토리 빌드 방법을 참조하십시오.

로깅 솔루션은 로그를 읽고 여러 로깅 출력을 읽는 다양한 방법을 제공합니다.

예를 들어 로깅 시스템은 다음과 같은 입력을 수신할 수 있습니다.

- 로컬 파일,
- **systemd/journal**,
- 네트워크를 통한 또 다른 로깅 시스템입니다.

또한 로깅 시스템에는 다음과 같은 출력이 포함될 수 있습니다.

- **/var/log** 디렉토리의 로컬 파일에 저장된 로그
- Elasticsearch로 전송된 로그
- 다른 로깅 시스템으로 전달된 로그입니다.

로깅 시스템 역할을 통해 입력 및 출력을 시나리오에 맞게 결합할 수 있습니다. 예를 들어 로컬 파일에 **저널** 의 입력을 저장하는 로깅 솔루션을 구성할 수 있지만 파일에서 읽은 입력은 다른 로깅 시스템으로 전달되고 로컬 로그 파일에 저장됩니다.

16.2. 시스템 역할 매개변수 로깅

로깅 시스템 역할 플레이북에서는 **logging_inputs** 매개변수의 입력, **logging_outputs** 매개변수의 출력 및 **logging_flows** 매개변수의 입력과 출력 간의 관계를 정의합니다. 로깅 시스템 역할은 추가 옵션으로 이러한 변수를 처리하여 로깅 시스템을 구성합니다. 암호화를 활성화할 수도 있습니다.

참고

현재 로깅 시스템 역할에서 사용 가능한 유일한 로깅 시스템은 **Rsyslog** 입니다.

- **logging_inputs**: 로깅 솔루션의 입력 목록입니다.
  - **name**: 입력의 고유 이름입니다. **logging_flows**: 입력 목록 및 생성된 **구성** 파일 이름의 일부입니다.
  - **type**: 입력 요소의 유형입니다. 유형은 **roles/centos/{tasks,vars}/inputs/의** 디렉터리 이름에 해당하는 작업 유형을 지정합니다.

- *기본: **systemd** 저널 또는 **unix** 소켓의 입력을 구성하는 입력입니다.*
  - ***kernel_message**: **true** 로 설정하면 **imklog** 를 로드합니다. 기본값은 **false** 입니다.*
  - ***use_imuxsock**: **imjournal** 대신 **imuxsock** 을 사용하십시오. 기본값은 **false** 입니다.*
  - ***ratelimit_burst**: **ratelimit_interval** 내에서 내보낼 수 있는 최대 메시지 수입니다. **use_imuxsock** 이 false인 경우 default는#189입니다. **use_imuxsock** 이 **true**인 경우 기본값은 **200** 입니다.*
  - ***ratelimit_interval: ratelimit_burst** 를 평가하는 간격입니다. **use_imuxsock** 이 **false**인 경우 기본값은 **600**초입니다. **use_imuxsock** 이 **true**인 경우 기본값은 **0** 입니다. **0**은 속도 제한이 해제되었음을 나타냅니다.*
  - ***persist_state_interval:** 저널 상태는 모든 값 의 메시지로 유지됩니다. 기본값은 **10** 입니다. **use_imuxsock** 이 **false**인 경우에만 유효합니다.*
- ***files:** 로컬 파일의 입력을 구성하는 입력입니다.*
- ***remote:** 네트워크를 통해 다른 로깅 시스템의 입력을 구성하는 입력입니다.*

- ***state:** 구성 파일의 상태. **present** 또는 **absent**. 기본값은 **present** 입니다.*

- ***logging_outputs:** 로깅 솔루션의 출력 목록입니다.*
  - ***files:** 로컬 파일에 출력을 구성하는 출력입니다.*
  - ***Pass e :** 출력을 다른 로깅 시스템에 구성하는 출력입니다.*
  - ***remote_files:** 다른 로깅 시스템의 출력을 로컬 파일로 구성하는 출력 구성입니다.*

- ***logging_flows: logging_inputs** 와 **logging_outputs** 간의 관계를 정의하는 흐름 목록입니다. **logging_flows** 변수에는 다음과 같은 키가 있습니다.*
  - ***name:** 흐름의 고유 이름입니다.*

- 입력: **logging_inputs** 이름 값 목록

- **outputs: logging_outputs** 이름 값 목록입니다.

추가 리소스

- **/usr/share/ansible/roles/ rhel-system-roles.logging.logging/README.html**의 **rhel-system-roles**패키지로 설치된 문서

16.3. 로컬 로깅 시스템 역할 적용

다음 단계에 따라 **Ansible** 플레이북을 준비하고 적용하여 분리된 머신 세트에 로깅 솔루션을 구성합니다. 각 머신은 로그를 로컬로 기록합니다.

사전 요구 사항

- 로깅 시스템 역할로 구성하려는 시스템인 하나 이상의 관리형 노드에 대한 액세스 및 권한.

- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.

  제어 노드에서 다음이 있어야 합니다.

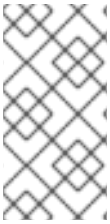
  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 **ansible, ansible-playbook, docker** 및 **podman**과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 확보하고 설치하는 방법에 대한 자세한 내용은 **Red Hat Ansible Engine** 지식베이스를 다운로드하고 설치하는 방법에 대한 지식베이스 문서를 참조하십시오.

RHEL 8.6 및 **9.0**에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모의 기본 제공 **Ansible** 플러그인 세트가 포함된 **Ansible Core(ansible-core** 패키지로 제공)를 도입했습니다. **RHEL**은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 **RHEL 9** 및 **RHEL 8.6** 이상 **AppStream** 리포지토리 지식 베이스에 포함된 **Ansible Core** 패키지에 대한 지원 범위에 대한 지식베이스 문서를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.



참고

시스템 역할이 배포될 때 **rsyslog** 를 설치하기 때문에 **rsyslog** 패키지를 설치할 필요가 없습니다.

절차

1. 필요한 역할을 정의하는 플레이북을 생성합니다.

   a. 새 **YAML** 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

   ```
   # vi logging-playbook.yml
   ```

   b. 다음 콘텐츠를 삽입합니다.

   ```
   ---
   - name: Deploying basics input and implicit files output
     hosts: all
     roles:
       - rhel-system-roles.logging
     vars:
       logging_inputs:
         - name: system_input
           type: basics
   ```

```
logging_outputs:
  - name: files_output
    type: files
logging_flows:
  - name: flow1
    inputs: [system_input]
    outputs: [files_output]
```

2. 특정 인벤토리에서 플레이북을 실행합니다.

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

다음과 같습니다.

- **inventory-file** 은 인벤토리 파일입니다.
- **logging-playbook.yml** 은 사용하는 플레이북입니다.

검증

1. **/etc/qcow.conf** 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 시스템이 로그에 메시지를 전송하는지 확인합니다.

a. 테스트 메시지를 보냅니다.

```
# logger test
```

b. **/var/log/spring** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/messages
Aug  5 13:48:31 hostname root[6778]: test
```

여기서 **'hostname'** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 루트).

16.4. 로컬 로깅 시스템 역할의 로그 필터링

rsyslog 속성 기반 필터를 기반으로 로그를 필터링하는 로깅 솔루션을 배포할 수 있습니다.

사전 요구 사항

- 로깅 시스템 역할로 구성하려는 시스템인 하나 이상의 관리형 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.

  제어 노드에서 다음을 수행합니다.

  - **Red Hat Ansible Core** 설치
  - **rhel-system-roles** 패키지가 설치되어 있습니다.
  - 관리 노드를 나열하는 인벤토리 파일.

참고

배포 시 **System Role**은 **rsyslog** 를 설치하기 때문에 **rsyslog** 패키지를 설치할 필요가 없습니다.

절차

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

```
---
- name: Deploying files input and configured files output
  hosts: all
  roles:
    - linux-system-roles.logging
  vars:
    logging_inputs:
```

```
    - name: files_input
      type: basics
  logging_outputs:
    - name: files_output0
      type: files
      property: msg
      property_op: contains
      property_value: error
      path: /var/log/errors.log
    - name: files_output1
      type: files
      property: msg
      property_op: "!contains"
      property_value: error
      path: /var/log/others.log
  logging_flows:
    - name: flow0
      inputs: [files_input]
      outputs: [files_output0, files_output1]
```

*이 구성을 사용하면 오류 문자열이 포함된 모든 메시지가 **/var/log/errors.log** 에 기록되고 다른 모든 메시지는 **/var/log/others.log** 에 기록됩니다.*

***error** 속성 값을 필터링하려는 문자열로 교체할 수 있습니다.*

환경 설정에 따라 변수를 수정할 수 있습니다.

2. *선택 사항: 플레이북 구문을 확인합니다.*

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

3. *인벤토리 파일에서 플레이북을 실행합니다.*

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

검증

1. ***/etc/qcow.conf** 파일의 구문을 테스트합니다.*

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 시스템에서 오류 문자열이 포함된 메시지를 로그에 전송하는지 확인합니다.

   a. 테스트 메시지를 보냅니다.

   ```
   # logger error
   ```

   b. **/var/log/errors.log** 로그를 확인합니다. 예를 들면 다음과 같습니다.

   ```
   # cat /var/log/errors.log
   Aug  5 13:48:31 hostname root[6778]: error
   ```

   여기서 **hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 루트).

추가 리소스

- **/usr/share/ansible/roles/ rhel-system-roles.logging.logging/README.html**의 **rhel-system-roles**패키지로 설치된 문서

16.5. 로깅 시스템 역할을 사용하여 원격 로깅 솔루션 적용

다음 단계에 따라 **Red Hat Ansible Core** 플레이북을 준비 및 적용하여 원격 로깅 솔루션을 구성합니다. 이 플레이북에서 하나 이상의 클라이언트는 **systemd-journal** 에서 로그를 가져와 원격 서버로 전달합니다. 서버는 **remote_#189** 및 **remote_ files** 에서 원격 입력을 수신하고 원격 호스트 이름에 의해 이름이 지정된 디렉터리의 로컬 파일에 로그를 출력합니다.

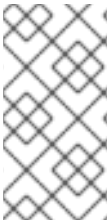
사전 요구 사항

- 로깅 시스템 역할로 구성하려는 시스템인 하나 이상의 관리형 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.

  제어 노드에서 다음이 있어야 합니다.

  - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

- 관리 노드를 나열하는 인벤토리 파일.



참고

배포 시 **System Role**은 **rsyslog** 를 설치하기 때문에 **rsyslog** 패키지를 설치할 필요가 없습니다.

절차

1. 필요한 역할을 정의하는 플레이북을 생성합니다.

a. 새 **YAML** 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

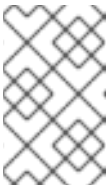
```
# vi logging-playbook.yml
```

b. 다음 내용을 파일에 삽입합니다.

```
---
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: remote_udp_input
        type: remote
        udp_ports: [ 601 ]
      - name: remote_tcp_input
        type: remote
        tcp_ports: [ 601 ]
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: flow_0
        inputs: [remote_udp_input, remote_tcp_input]
        outputs: [remote_files_output]

- name: Deploying basics input and forwards output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
```

```
      type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: _host1.example.com_
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: _host1.example.com_
        tcp_port: 601
    logging_flows:
      - name: flows0
        inputs: [basic_input]
        outputs: [forward_output0, forward_output1]

[basic_input]
[forward_output0, forward_output1]
```

*여기서 **host1.example.com** 은 로깅 서버입니다.*



참고

필요에 맞게 플레이북의 매개변수를 수정할 수 있습니다.

주의

*로깅 솔루션은 서버 또는 클라이언트 시스템의 **SELinux** 정책에 정의된 포트에서만 작동하며 방화벽에서 엽니다. 기본 **SELinux** 정책에는 포트 **601, 514, 6514, 10514** 및 **20514**가 포함됩니다. 다른 포트를 사용하려면 클라이언트 및 서버 시스템에서 **SELinux** 정책을 수정합니다. 시스템 역할을 통한 방화벽 구성은 아직 지원되지 않습니다.*

2. *서버 및 클라이언트를 나열하는 인벤토리 파일을 생성합니다.*

   a. *새 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.*

```
# vi inventory.ini
```

b. *인벤토리 파일에 다음 내용을 삽입합니다.*

```
[servers]
server ansible_host=host1.example.com
[clients]
client ansible_host=host2.example.com
```

다음과 같습니다.

- ***host1.example.com** 은 로깅 서버입니다.*
- ***host2.example.com** 은 로깅 클라이언트입니다.*

3. *인벤토리에서 플레이북을 실행합니다.*

```
# ansible-playbook -i /path/to/file/inventory.ini /path/to/file/_logging-playbook.yml
```

다음과 같습니다.

- ***inventory.ini** 는 인벤토리 파일입니다.*
- ***logging-playbook.yml** 은 생성한 플레이북입니다.*

검증

1. *클라이언트 및 서버 시스템에서 **/etc/#189.conf** 파일의 구문을 테스트합니다.*

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. *클라이언트 시스템이 서버에 메시지를 전송하는지 확인합니다.*

a. *클라이언트 시스템에서 테스트 메시지를 보냅니다.*

```
# logger test
```

b. 서버 시스템에서 **/var/log/ECDHE** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/messages
Aug  5 13:48:31 host2.example.com root[6778]: test
```

여기서 **host2.example.com** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 루트).

추가 리소스

- **RHEL** 시스템 역할 시작하기
- **/usr/share/ansible/roles/ rhel-system-roles.logging.logging/README.html**의 **rhel-system-roles**패키지로 설치된 문서
- **RHEL System Roles KB** 문서

16.6. TLS에서 로깅 시스템 역할 사용

TLS(Transport Layer Security)는 컴퓨터 네트워크를 통해 안전하게 통신하도록 설계된 암호화 프로토콜입니다.

관리자는 **Logging RHEL** 시스템 역할을 사용하여 **Red Hat Ansible Automation Platform**을 사용하여 로그 전송을 설정할 수 있습니다.

16.6.1. TLS를 사용하여 클라이언트 로깅 구성

로깅 시스템 역할을 사용하여 로컬 시스템에 로그인한 **RHEL** 시스템에 로깅을 구성하고 **Ansible** 플레이북을 실행하여 **TLS**를 사용하여 원격 로깅 시스템으로 로그를 전송할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 **clients** 그룹에 있는 모든 호스트에서 **TLS**를 구성합니다. **TLS** 프로토콜은 네트워크를 통해 로그의 보안 전송을 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- ***TLS**를 구성하려는 관리형 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.*
- *관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.*
- ***ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.*

절차

1. *다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.*

```
---
- name: Deploying files input and forwards output with certs
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: files
        input_log_path: /var/log/containers/*.log
    logging_outputs:
      - name: output_name
        type: forwards
        target: your_target_host
        tcp_port: 514
        tls: true
        pki_authmode: x509/name
        permitted_server: 'server.example.com'
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

플레이북은 다음 매개 변수를 사용합니다.

logging_pki_files

*이 매개변수를 사용하면 **TLS**를 구성할 수 있으며 **ca_cert_src,cert_src, private_key_src** 매개 변수를 전달해야 합니다.*

ca_cert

CA 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

cert

인증서 경로를 나타냅니다.**Represents the path to cert.** 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.

private_key

개인 키의 경로를 나타냅니다.**Represents the path to private key.** 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

대상 호스트에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert** 를 지정하면 위치에 복사됩니다.

cert_src

대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. 인증서 가 지정되면 위치에 복사됩니다.

private_key_src

대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 를 지정하면 위치에 복사됩니다.

tls

이 매개변수를 사용하면 네트워크를 통해 로그를 안전하게 전송할 수 있습니다. 보안 래퍼를 원하지 않는 경우 **tls: true** 를 설정할 수 있습니다.

2. 플레이북 구문을 확인합니다.

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

16.6.2. TLS를 사용하여 서버 로깅 구성

*로깅 시스템 역할을 사용하여 **RHEL** 시스템의 로그인을 서버로 구성할 수 있으며 **Ansible Playbook**을 실행하여 **TLS**로 원격 로깅 시스템에서 로그를 수신할 수 있습니다.*

*이 절차에서는 **Ansible** 인벤토리의 서버 그룹에 있는 모든 호스트에서 **TLS**를 구성합니다.*

사전 요구 사항

- *__TLS__를 구성하려는 관리형 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.*
- *관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.*
- ***ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.*

절차

1. *다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.*

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-
replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
```

```
  - name: flow_name
    inputs: [input_name]
    outputs: [output_name]
```

플레이북은 다음 매개 변수를 사용합니다.

logging_pki_files

*이 매개변수를 사용하면 **TLS**를 구성할 수 있으며 **ca_cert_src,cert_src, private_key_src** 매개 변수를 전달해야 합니다.*

ca_cert

***CA** 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.*

cert

*인증서 경로를 나타냅니다.**Represents the path to cert.** 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.*

private_key

*개인 키의 경로를 나타냅니다.**Represents the path to private key.** 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.*

ca_cert_src

*대상 호스트에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert** 를 지정하면 위치에 복사됩니다.*

cert_src

대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. 인증서 가 지정되면 위치에 복사됩니다.

private_key_src

*대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 를 지정하면 위치에 복사됩니다.*

tls

*이 매개변수를 사용하면 네트워크를 통해 로그를 안전하게 전송할 수 있습니다. 보안 래퍼를 원하지 않는 경우 **tls: true** 를 설정할 수 있습니다.*

2. 플레이북 구문을 확인합니다.

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

16.7. RELP에서 로깅 시스템 역할 사용

RELP(Reliability Event Logging Protocol)는 **TCP** 네트워크를 통한 데이터 및 메시지 로깅을 위한 네트워킹 프로토콜입니다. 안정적인 이벤트 메시지를 제공하고 메시지 손실을 허용하지 않는 환경에서 사용할 수 있습니다.

RELP 발신자는 로그 항목을 명령 형태로 전송하고 수신자는 처리 후 이를 승인합니다. 일관성을 보장하기 위해 **RELP**는 모든 종류의 메시지 복구에 대해 전송된 각 명령에 트랜잭션 번호를 저장합니다.

RELP Client와 **RELP Server** 사이에 원격 로깅 시스템을 고려할 수 있습니다. **RELP Client**는 로그를 원격 로깅 시스템으로 전송하고 **RELP** 서버는 원격 로깅 시스템에서 보낸 모든 로그를 수신합니다.

관리자는 로깅 시스템 역할을 사용하여 로그 항목을 안정적으로 전송하고 수신하도록 로깅 시스템을 구성할 수 있습니다.

16.7.1. RELP로 클라이언트 로깅 구성

로깅 시스템 역할을 사용하여 로컬 시스템에 로그인한 **RHEL** 시스템에 로깅을 구성하고 **Ansible Playbook**을 실행하여 **RELP**로 원격 로깅 시스템으로 로그를 전송할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 **clients** 그룹에 있는 모든 호스트에서 **RELP**를 구성합니다. **RELP** 구성은 **TLS(Transport Layer Security)**를 사용하여 네트워크를 통한 로그의 보안 전송을 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- **RELP**를 구성할 관리 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.

- 관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.

절차

1. 다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploying basic input and relp output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: _logging.server.com_
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/client-cert.pem_
        private_key: _/etc/pki/tls/private/client-key.pem_
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: _example_flow_
        inputs: [basic_input]
        outputs: [relp_client]
```

플레이북은 다음 설정을 사용합니다.

- **target:** 이는 원격 로깅 시스템이 실행 중인 호스트 이름을 지정하는 필수 매개 변수입니다.

- 포트: 원격 로깅 시스템이 수신 대기 중인 포트 번호입니다.

- **tls:** 네트워크를 통해 로그를 안전하게 전송할 수 있습니다. 보안 래퍼를 사용하지 않으려면 **tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로 **tls** 매개변수는 **RELP**와 함께 작업

*하는 동안 **true**로 설정되어 있으며 **key/certificates {ca_cert,cert,private_key}** 및/또는 **{ca_cert_src,cert_src,private_key_src}**가 필요합니다.*

- *{ca_cert_src,cert_src,private_key_src} 트립릿이 설정된 경우 기본 위치 /etc/pki/tls/certs 및 /etc/pki/tls/private 이 제어 노드에서 파일을 전송하기 위해 관리 노드의 대상으로 사용됩니다. 이 경우 파일 이름은 트립틀릿의 원래 이름과 동일합니다.*
- *{ca_cert,cert,private_key} 트립릿이 설정된 경우 파일은 로깅 구성 전에 기본 경로에 있어야 합니다.*
- *세로틀릿이 모두 설정된 경우 파일이 제어 노드에서 관리 노드의 특정 경로로 전송됩니다.*

- *ca_cert: CA 인증서의 경로를 나타냅니다. 기본 경로는 /etc/pki/tls/certs/ca.pem 이며 파일 이름은 사용자가 설정합니다.*
- *인증서: 인증서 경로를 나타냅니다.Represents the path to cert. 기본 경로는 /etc/pki/tls/certs/server-cert.pem 이며 파일 이름은 사용자가 설정합니다.*
- *private_key: 개인 키의 경로를 나타냅니다.Represents the path to private key. 기본 경로는 /etc/pki/tls/private/server-key.pem 이며 파일 이름은 사용자가 설정합니다.*
- *ca_cert_src: 대상 호스트에 복사되는 로컬 CA 인증서 파일 경로를 나타냅니다. ca_cert를 지정하면 위치에 복사됩니다.*
- *cert_src: 대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. 인증서가 지정되면 위치에 복사됩니다.*
- *private_key_src: 대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. private_key를 지정하면 위치에 복사됩니다.*
- *pki_authmode: 인증 모드를 이름 또는 지문으로 허용합니다.*
- *permitted_servers: 로깅 클라이언트에서 TLS를 통해 로그를 연결하고 전송하기 위해 허용할 서버 목록입니다.*

- *입력: 로깅 입력 사전 목록입니다.*
- ***outputs:** 로깅 출력 사전 목록입니다.*

2. *선택 사항: 플레이북 구문을 확인합니다.*

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

3. *플레이북을 실행합니다.*

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

16.7.2. RELP를 사용하여 서버 로깅 구성

*로깅 시스템 역할을 사용하여 **RHEL** 시스템의 로그인을 서버로 구성할 수 있으며 **Ansible Playbook**을 실행하여 **RELP**로 원격 로깅 시스템에서 로그를 수신할 수 있습니다.*

*이 절차에서는 **Ansible** 인벤토리의 서버 그룹에 있는 모든 호스트에서 **RELP**를 구성합니다. **RELP** 구성은 **TLS**를 사용하여 네트워크를 통한 로그 보안 전송을 위해 메시지 전송을 암호화합니다.*

사전 요구 사항

- ***RELP**를 구성할 관리 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.*
- *관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.*
- ***ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.*

절차

1. *다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.*

```
---
```

```
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/server-cert.pem_
        private_key: _/etc/pki/tls/private/server-key.pem_
        pki_authmode: name
        permitted_clients:
          - '_*example.client.com_'
    logging_outputs:
      - name: _remote_files_output_
        type: _remote_files_
    logging_flows:
      - name: _example_flow_
        inputs: _relp_server_
        outputs: _remote_files_output_
```

플레이북은 다음 설정을 사용합니다.

- *포트: 원격 로깅 시스템이 수신 대기 중인 포트 번호입니다.*

- ***tls:** 네트워크를 통해 로그를 안전하게 전송할 수 있습니다. 보안 래퍼를 사용하지 않으려면 **tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로 **tls** 매개변수는 **RELP**와 함께 작업하는 동안 **true**로 설정되어 있으며 **key/certificates {ca_cert,cert,private_key}** 및/또는 **{ca_cert_src,cert_src,private_key_src}**가 필요합니다.*

  - ***{ca_cert_src,cert_src,private_key_src}** 트립릿이 설정된 경우 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 이 제어 노드에서 파일을 전송하기 위해 관리 노드의 대상으로 사용됩니다. 이 경우 파일 이름은 트립틀릿의 원래 이름과 동일합니다.*

  - ***{ca_cert,cert,private_key}** 트립릿이 설정된 경우 파일은 로깅 구성 전에 기본 경로에 있어야 합니다.*

  - *세로틀릿이 모두 설정된 경우 파일이 제어 노드에서 관리 노드의 특정 경로로 전송됩니다.*

- ***ca_cert: CA** 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며*

파일 이름은 사용자가 설정합니다.

- *인증서: 인증서 경로를 나타냅니다.**Represents the path to cert.** 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.*
- ***private_key:** 개인 키의 경로를 나타냅니다.**Represents the path to private key.** 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.*
- ***ca_cert_src:** 대상 호스트에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert**를 지정하면 위치에 복사됩니다.*
- ***cert_src:** 대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. 인증서가 지정되면 위치에 복사됩니다.*
- ***private_key_src:** 대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key**를 지정하면 위치에 복사됩니다.*
- ***pki_authmode:** 인증 모드를 이름 또는 지문으로 허용합니다.*
- ***permitted_clients:** 로깅 서버에서 **TLS**를 통해 로그를 연결하고 보내는 데 사용할 클라이언트 목록입니다.*
- *입력: 로깅 입력 사전 목록입니다.*
- ***outputs:** 로깅 출력 사전 목록입니다.*

2. *선택 사항: 플레이북 구문을 확인합니다.*

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

3. *플레이북을 실행합니다.*

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

16.8. 추가 리소스

- RHEL 시스템 역할 시작하기
- /usr/share/ansible/roles/ rhel-system-roles.logging/README.html에 rhel-system-roles 패키지로 설치된 문서입니다.
- RHEL System Roles
- ansible-playbook(1) 도움말 페이지.