Red Hat Enterprise Linux 9

SELinux 사용

SELinux(Security-hardened Linux)의 기본 및 고급 구성

Last Updated: 2022-08-21

Red Hat Enterprise Linux 9 SELinux 사용

SELinux(Security-hardened Linux)의 기본 및 고급 구성

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

법적 공지

초록

이 제목은 사용자와 관리자는 SELinux가 다양한 서비스를 설정하고 구성하는 실제 작업을 설명하는 기본 및 원칙을 익히는 데 도움이 됩니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

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

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. 문서를 개선할 수 있는 방법에 관해 알려주십시오.

- 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. SELINUX 시작하기

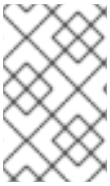
SELinux(Security Enhanced Linux)는 추가 시스템 보안 계층을 제공합니다. SELinux는 근본적으로 질문에 대답합니다. *may <subject> do <action> to <object>? 웹 서버가 사용자의 홈 디렉토리에서 파일에 액세스할 수 있습니까?*

1.1. SELINUX 소개

DDAC(Discretionary Access Control)로 알려진 사용자, 그룹 및 기타 권한에 따른 표준 액세스 정책은 시스템 관리자가 특정 애플리케이션이 로그 파일만 볼 수 있도록 제한하면서 로그 파일에 새 데이터를 추가할 수 있도록 하는 것과 같이 포괄적이고 세분화된 보안 정책을 생성할 수 없습니다.

SELinux(Security Enhanced Linux)는 MAC(Mandatory Access Control)을 구현합니다. 모든 프로세스와 시스템 리소스에는 *SELinux 컨텍스트*라는 특수 보안 레이블이 있습니다. SELinux *레이블 라고도 하는 SELinux* 컨텍스트는 시스템 수준 세부 정보를 추상화하고 엔터티의 보안 속성에 중점을 두는 식별자입니다. 이렇게 하면 SELinux 정책에서 오브젝트를 일관된 방식으로 참조할 수 있을 뿐만 아니라 다른 식별 방법에서 찾을 수 있는 모호성을 제거할 수도 있습니다. 예를 들어, 파일에 바인드 마운트를 사용하는 시스템에 여러 개의 유효한 경로 이름이 있을 수 있습니다.

SELinux 정책은 프로세스가 서로 상호 작용하는 방법과 다양한 시스템 리소스를 정의하는 일련의 규칙에 이러한 컨텍스트를 사용합니다. 기본적으로 정책은 규칙이 액세스를 명시적으로 부여하지 않는 한 모든 상호 작용을 허용하지 않습니다.



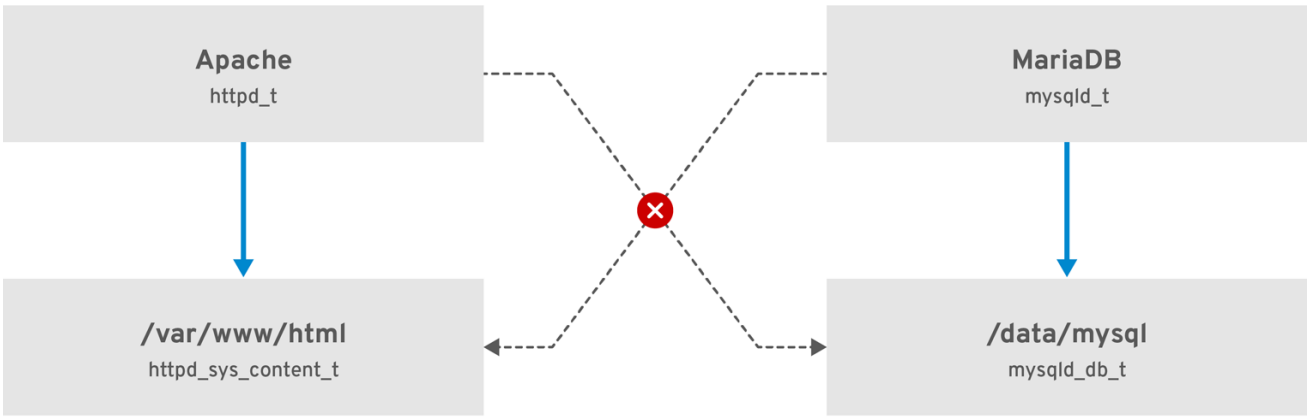
참고

SELinux 정책 규칙은 DAC 규칙 다음에 확인합니다. DAC 규칙에서 액세스를 먼저 거부하는 경우 SELinux 정책 규칙이 사용되지 않으므로 기존 DAC 규칙이 액세스를 방지하는 경우 SELinux 거부가 기록되지 않습니다.

SELinux 컨텍스트에는 user, role, type 및 보안 수준이라는 몇 가지 필드가 있습니다. SELinux 유형 정보는 프로세스와 시스템 리소스 간 허용되는 상호 작용을 정의하는 가장 일반적인 정책 규칙으로 SELinux 유형에서 전체 SELinux 컨텍스트가 아닌 가장 중요한 경우입니다. SELinux 유형은 **_t** 로 끝납니다. 예를 들어 웹 서버의 유형 이름은 **httpd_t** 입니다. **/var/www/html/** 에 있는 파일 및 디렉토리의 유형 컨텍스트는 **httpd_sys_content_t** 입니다. /tmp 및 **/var /tmp /에서 일반적으로 발견되는 파일 및 디렉토리의 유형 컨텍스트는 tmp _t** 입니다. 웹 서버 포트의 유형 컨텍스트는 **http_port_t** 입니다.

Apache(**httpd_t**로 실행되는 웹 서버 프로세스)가 **/var/www/html/** 및 기타 웹 서버 디렉터리 (**httpd_sys_content_t**)에서 일반적으로 발견되는 컨텍스트의 파일 및 디렉터리에 액세스할 수 있도록 허용하는 정책 규칙이 있습니다. **/tmp** 및 **/var/tmp/** 에서 일반적으로 발견되는 파일에 대한 정책에 허용 규칙이 없으므로 액세스가 허용되지 않습니다. SELinux를 사용하면 Apache가 손상되어 악성 스크립트에 액세스하여 **/tmp** 디렉토리에 계속 액세스할 수 없습니다.

그림 1.1. SELinux가 Apache 및 MariaDB를 안전한 방식으로 실행하는 데 도움이 되는 방법의 예는 무엇입니까.



이전 스키마가 표시된 대로 SELinux는 **httpd_t** 로 실행되는 Apache 프로세스가 **/var/www/html/** 디렉터리에 액세스할 수 있도록 허용하고 **httpd_t** 및 **mysqld_db_t** 유형 컨텍스트에 대한 허용 규칙이 없기 때문에 **/data/mysql/** 디렉터리에 액세스하는 동일한 프로세스를 거부합니다. 반면 **mysqld_t** 로 실행되는 MariaDB 프로세스는 **/data/mysql/** 디렉터리에 액세스하고 SELinux도 **mysqld_t** 유형의 프로세스를 올바르게 거부하여 **httpd_sys_content_t** 라는 레이블이 지정된 **/var/www/html/** 디렉터리에 액세스할 수 있습니다.

추가 리소스

- **selinux(8)** 도움말 페이지 및 **apropos selinux** 명령으로 나열된 도움말 페이지 및 도움말 페이지.
- **selinux-policy-doc** 패키지가 설치된 경우 **man -k _selinux** 명령으로 나열된 도움말 페이지.
- SELinux coloring Book은 SELinux 기본 개념을 더 잘 이해하는 데 도움이 됩니다.
- SELinux topics FAQ

1.2. SELINUX 실행의 이점

SELinux는 다음과 같은 이점을 제공합니다.

- 모든 프로세스와 파일에 레이블이 지정됩니다. SELinux 정책 규칙은 프로세스가 파일과 상호 작용하는 방법과 프로세스가 서로 상호 작용하는 방식을 정의합니다. 액세스를 구체적으로 허용하는 SELinux 정책 규칙이 있는 경우에만 액세스가 허용됩니다.
- 세분화된 액세스 제어. 사용자 재량에 따라 제어되는 기존 UNIX 권한을 벗어나 Linux 사용자 및 그룹 ID를 기반으로 하는 SELinux 액세스 결정은 SELinux 사용자, 역할, 유형 및 필요에 따라 보안 수준과 같은 모든 사용 가능한 정보를 기반으로 합니다.
- SELinux 정책은 관리로 정의되며 시스템 전체에서 강제 적용됩니다.
- 권한 에스컬레이션 공격에 대한 완화 조치가 개선되었습니다. 프로세스는 도메인에서 실행되며 서로 분리되어 있습니다. SELinux 정책 규칙은 프로세스가 파일 및 기타 프로세스에 액세스하는 방법을 정의합니다. 프로세스가 손상되면 공격자는 해당 프로세스의 일반 기능에만 액세스하고 프로세스가 액세스할 수 있도록 구성된 파일에만 액세스할 수 있습니다. 예를 들어 Apache HTTP Server가 손상된 경우, 공격자는 이러한 액세스를 허용하도록 특정 SELinux 정책 규칙을 추가하거나 구성하지 않는 한 사용자 홈 디렉터리의 파일을 읽기 위해 해당 프로세스를 사용할 수 없습니다.

- SELinux를 사용하여 데이터 기밀성과 무결성을 적용하고 신뢰할 수 없는 입력에서 프로세스를 보호할 수 있습니다.

그러나 SELinux는 다음을 수행하지 않습니다.

- 안티바이러스 소프트웨어,
- 암호, 방화벽 및 기타 보안 시스템 교체
- All-in-one 보안 솔루션

SELinux는 기존 보안 솔루션을 개선하도록 설계되었으므로 대체하지 않습니다. SELinux를 실행하는 경우에도 하드 추측 암호 및 방화벽을 사용하여 소프트웨어를 최신 상태로 유지하는 등 우수한 보안 관행을 계속 따르는 것이 중요합니다.

1.3. SELINUX 예

다음 예제에서는 SELinux가 보안을 강화하는 방법을 보여줍니다.

- 기본 동작은 거부됩니다. 파일을 여는 프로세스와 같이 액세스를 허용하는 SELinux 정책 규칙이 없으면 액세스가 거부됩니다.
- SELinux는 Linux 사용자를 제한할 수 있습니다. SELinux 정책에 제한된 SELinux 사용자가 많이 있습니다. Linux 사용자는 제한된 SELinux 사용자에게 매핑되어 보안 규칙과 여기에 적용되는 메커니즘을 활용할 수 있습니다. 예를 들어 Linux 사용자를 SELinux **user_u** 사용자에 매핑하면 **sudo** 및 **su** 와 같은 다른 설정의 사용자 ID(setuid) 애플리케이션을 설정하지 않으면 을 실행할 수 없는 Linux 사용자가 생성됩니다.
- 프로세스 및 데이터 분리 증가 SELinux *도메인* 의 개념은 특정 파일 및 디렉터리에 액세스할 수 있는 프로세스를 정의할 수 있습니다. 예를 들어 달리 구성된 경우가 아니면 SELinux를 실행하는 경우 공격자는 Samba 서버를 손상시킬 수 없으며 해당 Samba 서버를 공격 벡터로 사용하여 MariaDB 데이터베이스와 같은 다른 프로세스에서 사용하는 파일을 읽고 쓸 수 있습니다.
- SELinux는 구성 오류로 인한 손상을 완화하는 데 도움이 됩니다. DNS(Domain Name System) 서버는 종종 영역 전송이라고 하는 정보를 서로 복제합니다. 공격자는 영역 전송을 사용하여 잘못된 정보로 DNS 서버를 업데이트할 수 있습니다. Red Hat Enterprise Linux에서 DNS 서버로 Berkeley Internet Name Domain(BIND)을 실행하는 경우, 관리자가 영역 전송을 수행할 수 있는 서버를 제한하는 것을 잊어 버린 경우에도 기본 SELinux 정책은 영역 파일을 방지합니다. [1] 영역 전송을 사용하여 업데이트, 즉 데몬 자체 및 기타 프로세스에 의해 BIND를 통해 업데이트됩니다.

1.4. SELINUX 아키텍처 및 패키지

SELinux는 Linux 커널에 빌드된 Linux Security Module(LSM)입니다. 커널의 SELinux 하위 시스템은 관리자가 제어하고 부팅 시 로드되는 보안 정책에 의해 구동됩니다. 시스템의 모든 보안 수준 액세스 작업은 SELinux에서 가로채고 로드된 보안 정책의 컨텍스트에서 검사합니다. 로드된 정책이 작업을 허용하는 경우 계속 실행됩니다. 그렇지 않으면 작업이 차단되고 프로세스에서 오류가 발생합니다.

액세스 허용 또는 허용하지 않는 등의 SELinux 결정은 캐시됩니다. 이 캐시는 AVC(Access Vector Cache)라고 합니다. 이러한 캐시된 결정을 사용하는 경우 SELinux 정책 규칙을 덜 점검해야 하므로 성능

이 향상됩니다. DAC 규칙에서 먼저 액세스를 거부하는 경우 SELinux 정책 규칙에 영향을 미치지 않습니다. 원시 감사 메시지는 /var/log/audit/audit.log 에 기록되고 type=AVC 문자열로 시작됩니다.

RHEL 9에서 시스템 서비스는 systemd 데몬에서 제어합니다. systemd 는 모든 서비스를 시작하고 중지하며 사용자 및 프로세스는 systemctl 유틸리티를 사용하여 systemd 와 통신합니다. systemd 데몬은 SELinux 정책을 참조하고 호출 프로세스의 레이블과 호출자가 관리하려는 유닛 파일의 레이블을 확인한 다음 SELinux에 호출자가 액세스를 허용하는지 여부를 요청할 수 있습니다. 이 접근 방식은 중요한 시스템 기능에 대한 액세스 제어를 강화하며, 여기에는 시스템 서비스 시작 및 중지가 포함됩니다.

systemd 데몬은 SELinux Access Manager로도 작동합니다. systemctl 을 실행하는 프로세스 레이블 또는 D-Bus 메시지를 systemd 로 보낸 프로세스를 검색합니다. 그런 다음 데몬은 프로세스가 구성하려는 유닛 파일의 레이블을 조회합니다. 마지막으로 SELinux 정책에서 프로세스 레이블과 유닛 파일 레이블 간의 특정 액세스를 허용하는 경우 커널에서 정보를 검색할 수 있습니다. 즉, 특정 서비스에 대해 systemd 와 상호 작용해야 하는 손상된 애플리케이션을 이제 SELinux로 제한할 수 있습니다. 정책 작성자는 관리자가 제한하기 위해 이러한 세분화된 제어를 사용할 수도 있습니다.

프로세스가 D-Bus 메시지를 다른 프로세스로 보내고 SELinux 정책에서 두 프로세스의 D-Bus 통신을 허용하지 않는 경우 시스템은 USER_AVC 거부 메시지를 출력하고 D-Bus 통신 시간이 초과됩니다. 두 프로세스 간의 D-Bus 통신은 양방향으로 작동합니다.

중요

잘못된 SELinux 레이블 지정 및 후속 문제를 방지하려면 systemctl start 명령을 사용하여 서비스를 시작해야 합니다.

RHEL 9는 SELinux를 사용하기 위한 다음 패키지를 제공합니다.

- 정책: selinux-policy-targeted,selinux-policy-mls
- tools: policycoreutils,policycoreutils-gui,libselinux-utils,policycoreutils-python-utils,setools-console,checkpolicy

1.5. SELINUX 상태 및 모드

SELinux는 강제, 허용 또는 비활성화의 세 가지 모드 중 하나로 실행할 수 있습니다.

- 강제 모드는 기본값이며 권장 작업 모드입니다. SELinux 강제 모드에서는 정상적으로 작동하

여 전체 시스템에서 로드된 보안 정책을 적용합니다.

- 허용 모드에서는 **SELinux**가 개체에 레이블을 지정하고 로그에 액세스 거부 항목을 내보내는 등 로드된 보안 정책을 적용하는 것처럼 동작하지만 실제로는 어떤 작업도 거부하지 않습니다. 프로덕션 시스템에는 권장되지 않지만 허용 모드는 **SELinux** 정책 개발 및 디버깅에 유용할 수 있습니다.

- 비활성화 모드는 권장되지 않습니다. 시스템이 **SELinux** 정책을 강제 적용하지 않도록 할 뿐만 아니라 파일과 같은 영구 개체에 레이블을 지정하는 것을 방지하여 나중에 **SELinux**를 활성화하기가 어렵습니다.

setenforce 유틸리티를 사용하여 강제 모드와 허용 모드 간에 변경합니다. **setenforce** 로 변경한 사항은 재부팅 시 유지되지 않습니다. 강제 모드로 변경하려면 **Linux root** 사용자로 **setenforce 1** 명령을 입력합니다. 허용 모드로 변경하려면 **setenforce 0** 명령을 입력합니다. **getenforce** 유틸리티를 사용하여 현재 **SELinux** 모드를 확인합니다.

```
# getenforce
Enforcing
```

```
# setenforce 0
# getenforce
Permissive
```

```
# setenforce 1
# getenforce
Enforcing
```

Red Hat Enterprise Linux에서는 시스템이 강제 모드로 실행되는 동안 개별 도메인을 허용 모드로 설정할 수 있습니다. 예를 들어 ***httpd_t*** 도메인을 허용하도록 설정하려면 다음을 수행합니다.

```
# semanage permissive -a httpd_t
```

허용 도메인은 시스템의 보안을 손상시킬 수 있는 강력한 도구입니다. **Red Hat**은 예를 들어 특정 시나리오를 디버깅할 때 허용 도메인을 주의해서 사용할 것을 권장합니다.

---

[1]
DNS 서버에서 사용하는 **IP** 주소 매핑에 대한 호스트 이름과 같은 정보가 포함된 텍스트 파일입니다.

2장. SELINUX 상태 및 모드 변경

활성화하면 **SELinux**를 두 가지 모드(강제 또는 허용) 중 하나로 실행할 수 있습니다. 다음 섹션에서는 이러한 모드로 영구적으로 변경하는 방법을 보여줍니다.

2.1. SELINUX 상태 및 모드의 영구 변경

SELinux 상태 및 모드에서 설명된 대로 **SELinux**를 활성화 또는 비활성화할 수 있습니다. 활성화된 경우 **SELinux**에는 강제 및 허용의 두 가지 모드가 있습니다.

getenforce 또는 **sestatus** 명령을 사용하여 **SELinux**가 실행 중인 모드를 확인합니다. **getenforce** 명령은 **Enforcing,Permissive** 또는 **Disabled** 를 반환합니다.

sestatus 명령은 **SELinux** 상태 및 사용 중인 **SELinux** 정책을 반환합니다.

```
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      31
```

주의

시스템이 허용 모드에서 **SELinux**를 실행하는 경우 사용자와 프로세스는 다양한 파일 시스템 오브젝트에 잘못 레이블을 지정할 수 있습니다. **SELinux**가 비활성화된 동안 생성된 파일 시스템 개체는 레이블이 지정되지 않습니다. **SELinux**는 파일 시스템 오브젝트의 올바른 레이블에 의존하기 때문에 강제 모드로 전환할 때 문제가 발생합니다.

레이블이 지정되지 않은 파일에 잘못 레이블이 지정되지 않은 파일을 방지하기 위해 **SELinux**는 비활성화 상태에서 허용 또는 강제 모드로 변경될 때 파일 시스템의 레이블을 자동으로 다시 지정합니다. 다음 재부팅 시 파일의 레이블을 다시 지정하려면 **root**로 **fixfiles -F onboot** 명령을 사용하여 **-F** 옵션이 포함된 **/.autorelabel** 파일을 만듭니다.

레이블을 다시 지정하기 위해 시스템을 재부팅하기 전에, 예를 들어 **enforcing=0** 커널 옵션을 사용하여 허용 모드로 부팅되는지 확인합니다. 이렇게 하면 **selinux-autorelabel** 서비스를 시작하기 전에 시스템에 **systemd** 에 필요한 레이블이 지정되지 않은 파일이 포함된 경우 시스템이 부팅되지 않습니다. 자세한 내용은 **RHBZ#2021835** 를 참조하십시오.

2.2. 허용 모드로 변경

다음 절차에 따라 **SELinux** 모드를 허용으로 영구적으로 변경합니다. **SELinux**가 허용 모드에서 실행되면 **SELinux** 정책이 적용되지 않습니다. 시스템이 계속 작동하고 **SELinux**는 모든 작업을 거부하지 않지만 **AVC** 메시지만 기록하면 문제 해결, 디버깅 및 **SELinux** 정책 개선에 사용할 수 있습니다. 이 경우 각 **AVC**는 한 번만 기록됩니다.

사전 요구 사항

- **selinux-policy-targeted,libselinux-utils** 및 **policycoreutils** 패키지가 시스템에 설치됩니다.
- **selinux=0** 또는 **enforcing=0** 커널 매개 변수는 사용되지 않습니다.

절차

1. 선택한 텍스트 편집기에서 **/etc/selinux/config** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vi /etc/selinux/config
```

2. **SELINUX=permissive** 옵션을 구성합니다.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 시스템을 다시 시작하십시오.

```
# reboot
```

검증

1. 시스템이 다시 시작되면 **getenforce** 명령이 **Permissive** 를 반환하는지 확인합니다.

```
$ getenforce
Permissive
```

2.3. 강제 모드로 변경

다음 절차에 따라 **SELinux**를 강제 모드로 전환합니다. **SELinux**가 강제 모드에서 실행되면 **SELinux** 정책을 적용하고 **SELinux** 정책 규칙에 따라 액세스를 거부합니다. **RHEL**에서 강제 모드는 **SELinux**를 사용하여 시스템을 처음 설치할 때 기본적으로 활성화됩니다.

사전 요구 사항

- **selinux-policy-targeted,libselinux-utils** 및 **policycoreutils** 패키지가 시스템에 설치됩니다.
- **selinux=0** 또는 **enforcing=0** 커널 매개 변수는 사용되지 않습니다.

절차

1. 선택한 텍스트 편집기에서 **/etc/selinux/config** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vi /etc/selinux/config
```

2. **SELINUX=enforcing** 옵션을 구성합니다.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 변경 사항을 저장하고 시스템을 다시 시작하십시오.

```
# reboot
```

다음 부팅 시 **SELinux**는 시스템 내의 모든 파일과 디렉터리의 레이블을 다시 지정하고 **SELinux**가 비활성화될 때 생성된 파일 및 디렉터리에 대한 **SELinux** 컨텍스트를 추가합니다.

검증

1. 시스템을 다시 시작한 후 **getenforce** 명령이 **Enforcing** 을 반환하는지 확인합니다.

```
$ getenforce
Enforcing
```

참고

강제 모드로 전환한 후 **SELinux**는 **SELinux** 정책 규칙이 올바르지 않거나 누락되어 일부 작업을 거부할 수 있습니다. **SELinux**가 거부하는 작업을 보려면 **root**로 다음 명령을 입력합니다.

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts today
```

또는 **se rsh-server** 패키지가 설치되어 있으면 다음을 입력합니다.

```
# grep "SELinux is preventing" /var/log/messages
```

SELinux가 활성 상태이고 감사 데몬(**auditd**)이 시스템에서 실행되지 않는 경우 **dmesg** 명령의 출력에서 특정 **SELinux** 메시지를 검색합니다.

```
# dmesg | grep -i -e type=1300 -e type=1400
```

자세한 내용은 **SELinux**와 관련된 문제 해결을 참조하십시오.

2.4. 이전에 비활성화한 시스템에서 SELINUX 활성화

시스템을 부팅하거나 처리할 수 없는 시스템 등의 문제를 방지하려면 이전에 비활성화한 시스템에서 **SELinux**를 활성화할 때 다음 절차를 따르십시오.

주의

시스템이 허용 모드에서 **SELinux**를 실행하는 경우 사용자와 프로세스는 다양한 파일 시스템 오브젝트에 잘못 레이블을 지정할 수 있습니다. **SELinux**가 비활성화된 동안 생성된 파일 시스템 개체는 레이블이 지정되지 않습니다. **SELinux**는 파일 시스템 오브젝트의 올바른 레이블에 의존하기 때문에 강제 모드로 전환할 때 문제가 발생합니다.

레이블이 지정되지 않은 파일에 잘못 레이블이 지정되지 않은 파일을 방지하기 위해 **SELinux**는 비활성화 상태에서 허용 또는 강제 모드로 변경될 때 파일 시스템의 레이블을 자동으로 다시 지정합니다.

레이블을 다시 지정하기 위해 시스템을 재부팅하기 전에, 예를 들어 **enforcing=0** 커널 옵션을 사용하여 허용 모드로 부팅되는지 확인합니다. 이렇게 하면 **selinux-autorelabel** 서비스를 시작하기 전에 시스템에 **systemd** 에 필요한 레이블이 지정되지 않은 파일이 포함된 경우 시스템이 부팅되지 않습니다. 자세한 내용은 **RHBZ#2021835** 를 참조하십시오.

절차

1. 허용 모드에서 **SELinux**를 활성화합니다. 자세한 내용은 허용 모드로 변경을 참조하십시오.

2. 시스템을 다시 시작하십시오.

```
# reboot
```

3. **SELinux** 거부 메시지를 확인합니다. 자세한 내용은 **SELinux** 거부 식별을 참조하십시오.

4. 다음 재부팅 시 파일의 레이블을 다시 지정했는지 확인합니다.

```
# fixfiles -F onboot
```

이렇게 하면 **-F** 옵션이 포함된 **/.autorelabel** 파일이 생성됩니다.

주의

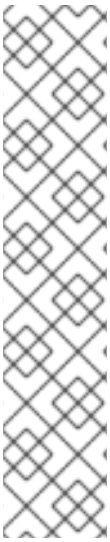
fixfiles -F onboot 명령을 시작하기 전에 항상 허용 모드로 전환합니다. 이렇게 하면 시스템에 레이블이 지정되지 않은 파일이 포함된 경우 시스템이 부팅되지 않습니다. 자세한 내용은 **RHBZ#2021835** 를 참조하십시오.

5. 거부 모드가 없는 경우 강제 모드로 전환합니다. 자세한 내용은 부팅 시 SELinux 모드 변경을 참조하십시오.

검증

1. 시스템을 다시 시작한 후 **getenforce** 명령이 **Enforcing** 을 반환하는지 확인합니다.

```
$ getenforce
Enforcing
```



참고

SELinux를 강제 모드로 사용하여 사용자 정의 애플리케이션을 실행하려면 다음 시나리오 중 하나를 선택합니다.

- **unconfined_service_t** 도메인에서 애플리케이션을 실행합니다.
- 애플리케이션에 대한 새 정책을 작성합니다. 자세한 내용은 사용자 지정 SELinux 정책 만들기 섹션을 참조하십시오.

추가 리소스

- SELinux 상태 및 모드 섹션에서는 모드의 임시 변경 사항을 다룹니다.

2.5. SELINUX 비활성화

SELinux가 비활성화되면 **SELinux** 정책이 전혀 로드되지 않으며 강제 적용되지 않고 **AVC** 메시지가 기록되지 않습니다. 따라서 SELinux 실행의 모든 이점이 손실됩니다.

중요

SELinux를 영구적으로 비활성화하는 대신 허용 모드를 사용하는 것이 좋습니다. 허용 모드에 대한 자세한 내용은 허용 모드 로 변경을 참조하십시오.

사전 요구 사항

- **grubby** 패키지가 설치되어 있습니다.

```
$ rpm -q grubby
grubby-version
```

절차

SELinux를 영구적으로 비활성화하려면 다음을 수행합니다.

1. 커널 명령줄에 **selinux=0** 을 추가하도록 부트로더를 구성합니다.

```
$ sudo grubby --update-kernel ALL --args selinux=0
```

2. 시스템을 다시 시작하십시오.

```
$ reboot
```

검증

- 재부팅 후 **getenforce** 명령이 **Disabled** 를 반환하는지 확인합니다.

```
$ getenforce
Disabled
```

2.6. 부팅 시 SELINUX 모드 변경

부팅 시 여러 커널 매개변수를 설정하여 **SELinux**가 실행되는 방식을 변경할 수 있습니다.

enforcing=0

이 매개변수를 설정하면 시스템이 허용 모드로 시작되어 문제를 해결할 때 유용합니다. 허용 모드를 사용하면 파일 시스템이 너무 손상된 경우 문제를 탐지하는 유일한 옵션이 될 수 있습니다. 또한 허용 모드에서 시스템은 레이블을 올바르게 생성합니다. 이 모드에서 생성된 **AVC** 메시지는 강제 모드와 다를 수 있습니다.

허용 모드에서는 일련의 동일한 거부에서 첫 번째 거부만 보고됩니다. 그러나 강제 모드에서는 디렉터리 읽기와 관련된 거부가 발생할 수 있으며 애플리케이션이 중지됩니다. 허용 모드에서는 동일한 **AVC** 메시지가 표시되지만 애플리케이션은 디렉터리에서 파일을 계속 읽고 각 거부에 대해 **AVC**를 가져옵니다.

selinux=0

이 매개 변수는 커널이 **SELinux** 인프라의 일부를 로드하지 않도록 합니다. **init** 스크립트는 **selinux=0** 매개 변수로 시스템이 부팅되고 **/.autorelabel** 파일을 만지는 것을 알 수 있습니다. 이렇게 하면 시스템이 다음에 **SELinux**를 사용하도록 설정할 때 레이블을 자동으로 다시 지정합니다.

중요

Red Hat은 **selinux=0** 매개 변수를 사용하지 않는 것이 좋습니다. 시스템을 디버깅하려면 허용 모드를 사용하는 것이 좋습니다.

autorelabel=1

이 매개 변수는 시스템에서 다음 명령과 유사하게 레이블을 다시 지정하도록 강제 적용합니다.

```
# touch /.autorelabel
# reboot
```

파일 시스템에 많은 양의 레이블이 있는 경우 허용 모드에서 시스템을 시작하여 자동 레이블 프로세스가 성공적으로 수행되도록 합니다.

추가 리소스

- **checkreqprot** 와 같은 추가 **SELinux** 관련 커널 부팅 매개 변수는 **/usr/share/doc/ kernel-doc -*<KERNEL_VER>*/Documentation/admin-guide/kernel-parameters.txt** 파일을 참조하십시오. ***<KERNEL_VER>*** 문자열을 설치된 커널의 버전 번호로 바꿉니다. 예를 들면 다음과 같습니다.

```
# dnf install kernel-doc
$ less /usr/share/doc/kernel-doc-4.18.0/Documentation/admin-guide/kernel-parameters.txt
```

3장. 제한된 제한 및 제한되지 않은 사용자 관리

다음 섹션에서는 Linux 사용자의 매핑을 SELinux 사용자에게 설명하고, 기본 제한된 사용자 도메인을 설명하고, 새 사용자를 SELinux 사용자에게 매핑하는 방법을 보여줍니다.

3.1. 제한된 제한 및 제한되지 않은 사용자

각 Linux 사용자는 SELinux 정책을 사용하여 SELinux 사용자에게 매핑됩니다. 이를 통해 Linux 사용자는 SELinux 사용자에 대한 제한을 상속할 수 있습니다.

시스템에서 SELinux 사용자 매핑을 보려면 semanage login -l 명령을 root로 사용합니다.

```
# semanage login -l
Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
```

Red Hat Enterprise Linux에서 Linux 사용자는 *기본적으로 SELinux 기본* 로그인에 매핑되며 이는 SELinux unconfined_u 사용자에게 매핑됩니다. 다음 줄은 기본 매핑을 정의합니다.

```
__default__          unconfined_u         s0-s0:c0.c1023       *
```

제한된 사용자는 현재 SELinux 정책에 명시적으로 정의된 SELinux 규칙에 의해 제한됩니다. 제한되지 않은 사용자에게는 SELinux의 최소한의 제한만 적용됩니다.

제한된 Linux 사용자는 실행 가능하고 쓰기 가능한 메모리 검사의 영향을 받으며 MCS 또는 MLS에 의해 제한됩니다.

사용 가능한 SELinux 사용자를 나열하려면 다음 명령을 입력합니다.

```
$ seinfo -u
Users: 8
   guest_u
   root
   staff_u
   sysadm_u
   system_u
```

```
unconfined_u
user_u
xguest_u
```

seinfo 명령은 기본적으로 설치되지 않는 **setools-console** 패키지에서 제공합니다.

제한되지 않은 **Linux** 사용자가 **SELinux** 정책에서 **unconfined_t** 도메인에서 자체 제한된 도메인으로 전환할 수 있는 애플리케이션으로 정의되는 애플리케이션을 실행하는 경우, 제한되지 않은 **Linux** 사용자는 여전히 해당 제한된 도메인의 제한에 따릅니다. 이 보안의 장점은 **Linux** 사용자가 제한되지 않은 상태에서도 애플리케이션이 제한될 수 있다는 것입니다. 따라서 애플리케이션의 결함 악용은 정책에 의해 제한될 수 있습니다.

마찬가지로 이러한 검사를 제한된 사용자에게 적용할 수 있습니다. 제한된 각 사용자는 제한된 사용자 도메인에 의해 제한됩니다. **SELinux** 정책은 제한된 사용자 도메인에서 자체 대상 제한 도메인으로의 전환을 정의할 수도 있습니다. 이러한 경우 제한된 사용자에게는 제한된 대상의 제한 사항이 적용됩니다. 주요 점은 특별 권한이 역할에 따라 제한된 사용자와 연결되어 있다는 것입니다.

3.2. SELINUX 사용자 기능

SELinux 정책은 각 **Linux** 사용자를 **SELinux** 사용자에게 매핑합니다. 이를 통해 **Linux** 사용자는 **SELinux** 사용자의 제한을 상속할 수 있습니다.

정책의 부울을 조정하여 **SELinux** 정책에서 제한된 사용자에 대한 권한을 특정 요구 사항에 따라 사용자 지정할 수 있습니다. **semanage boolean -l** 명령을 사용하여 이러한 부울의 현재 상태를 확인할 수 있습니다.

표 **3.1. SELinux** 사용자 역할

사용자	기본 역할	추가 역할
unconfined_u	**unconfined_r**	**system_r**
guest_u	**guest_r**	
xguest_u	**xguest_r**	
user_u	**user_r**	
staff_u	**staff_r**	**sysadm_r**
		unconfined_r

사용자	기본 역할	추가 역할
		system_r
sysadm_u	**sysadm_r**	
root	**staff_r**	**sysadm_r**
		unconfined_r
		system_r
system_u	**system_r**	

system_u 는 시스템 프로세스 및 오브젝트의 특수 사용자 **ID**이며 **system_r** 은 연결된 역할입니다. 관리자는 이 **system_u** 사용자와 **system_r** 역할을 **Linux** 사용자와 연결하지 않아야 합니다. 또한 **unconfined_u** 및 **root** 는 제한되지 않은 사용자입니다. 이러한 이유로 이러한 **SELinux** 사용자와 관련된 역할은 다음 표의 **SELinux** 역할에 포함되지 않습니다.

각 **SELinux** 역할은 **SELinux** 유형에 해당하며 특정 액세스 권한을 제공합니다.

표 3.2. SELinux 역할 유형 및 액세스

Role	유형	X Window System을 사용하여 로그인	**Su** 및 **sudo**	홈 디렉토리 및 **/tmp** (기본값)에서 실행	네트워킹
unconfined_r	**unconfined_t**	제공됨	제공됨	제공됨	제공됨
guest_r	**guest_t**	제공되지 않음	제공되지 않음	제공됨	제공되지 않음
xguest_r	**xguest_t**	제공됨	제공되지 않음	제공됨	웹 브라우저만 (Fire Firefox, GNOME 웹)
user_r	**user_t**	제공됨	제공되지 않음	제공됨	제공됨
staff_r	**staff_t**	제공됨	**sudo**만	제공됨	제공됨
auditadm_r	**auditadm_t**		제공됨	제공됨	제공됨
secadm_r	**secadm_t**		제공됨	제공됨	제공됨

Role	유형	X Window System을 사용하여 로그인	Su 및 sudo	홈 디렉토리 및 /tmp (기본값)에서 실행	네트워킹
sysadm_r	**sysadm_t**	**xdm_sysadm_login** 부울이 **있는**경우에만	제공됨	제공됨	제공됨

- **SELinux** 정책에서 허용하는 경우 **user_t,guest_t, xguest_t** 도메인의 **Linux** 사용자는 설정된 사용자 **ID(setuid)** 애플리케이션만 실행할 수 있습니다(예: **passwd**). 이러한 사용자는 **su** 및 **sudo setuid** 애플리케이션을 실행할 수 없으므로 이러한 애플리케이션을 사용하여 **root**가 될 수 없습니다.

- **EgressIP_t ,staff _t ,user_t, xguest_t** 도메인의 **Linux** 사용자는 **X Window System** 및 터미널을 사용하여 로그인할 수 있습니다.

- 기본적으로 **staff_t,user_t,guest_t** 도메인의 **Linux** 사용자는 홈 디렉터리 및 **/tmp** 에서 애플리케이션을 실행할 수 있습니다.

  사용자의 권한을 상속하는 애플리케이션을 실행하지 못하도록 하려면 **guest_exec_content** 및 **xguest_exec_content** 부울을 **off** 로 설정합니다. 이는 결함이 있거나 악성 애플리케이션이 사용자의 파일을 수정하지 못하도록 하는 데 도움이 됩니다.

- **xguest_t** 도메인에 있는 유일한 네트워크 액세스 **Linux** 사용자는 **Firefox**가 웹 페이지에 연결하는 것입니다.

- **ca _u** 사용자는 **SSH**를 사용하여 직접 로그인할 수 없습니다. **OSSM _u** 에 대한 **SSH** 로그인을 활성화하려면 **ssh_sysadm_login** 부울을 다음과 같이 설정합니다.

```
# setsebool -P ssh_sysadm_login on
```

이미 언급한 **SELinux** 사용자와 함께 **semanage user** 명령을 사용하여 해당 사용자에게 매핑할 수 있는 특수 역할이 있습니다. 이러한 역할은 **SELinux**에서 사용자가 수행할 수 있는 작업을 결정합니다.

- **webadm_r** 는 **Apache HTTP** 서버와 관련된 **SELinux** 유형만 관리할 수 있습니다.

- **dbadm_r** 는 **MariaDB** 데이터베이스 및 **PostgreSQL** 데이터베이스 관리 시스템과 관련된

SELinux 유형만 관리할 수 있습니다.

- **logadm_r** 는 **syslog** 및 **auditlog** 프로세스와 관련된 **SELinux** 유형만 관리할 수 있습니다.
- **secadm_r** 은 **SELinux**만 관리할 수 있습니다.
- **auditadm_r** 는 감사 하위 시스템과 관련된 프로세스만 관리할 수 있습니다.

사용 가능한 모든 역할을 나열하려면 **seinfo -r** 명령을 입력합니다.

```
$ seinfo -r
Roles: 14
   auditadm_r
   dbadm_r
   guest_r
   logadm_r
   nx_server_r
   object_r
   secadm_r
   staff_r
   sysadm_r
   system_r
   unconfined_r
   user_r
   webadm_r
   xguest_r
```

seinfo 명령은 기본적으로 설치되지 않는 **setools-console** 패키지에서 제공합니다.

추가 리소스

- **seinfo(1), semanage-login(8)** 및 **xguest_selinux(8)** 도움말 페이지

3.3. SELINUX UNCONFINED_U 사용자에게 자동으로 매핑된 새 사용자 추가

다음 절차에서는 시스템에 새 **Linux** 사용자를 추가하는 방법을 보여줍니다. 사용자는 **SELinux unconfined_u** 사용자에게 자동으로 매핑됩니다.

사전 요구 사항

- **root** 사용자는 기본적으로 **Red Hat Enterprise Linux**에서와 마찬가지로 제한되지 않습니다.

절차

1. 다음 명령을 입력하여 ***example.user*** 라는 새 **Linux** 사용자를 생성합니다.

```
# useradd example.user
```

2. **Linux *example.user*** 사용자에게 암호를 할당하려면 다음을 수행합니다.

```
# passwd example.user
Changing password for user example.user.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

3. 현재 세션에서 로그아웃합니다.

4. **Linux *example.user*** 사용자로 로그인합니다. 로그인할 때 **pam_selinux PAM** 모듈은 **Linux** 사용자를 **SELinux** 사용자(이 경우 **unconfined_u**)에 자동으로 매핑하고 결과 **SELinux** 컨텍스트를 설정합니다. 그런 다음 이 컨텍스트로 **Linux** 사용자의 쉘이 시작됩니다.

검증

1. ***example.user*** 사용자로 로그인하면 **Linux** 사용자의 컨텍스트를 확인합니다.

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

추가 리소스

- **pam_selinux(8)** 도움말 페이지.

3.4. SELINUX 제한 사용자로 새 사용자 추가

다음 단계를 사용하여 새 **SELinux** 제한 사용자를 시스템에 추가합니다. 이 예제 절차에서는 사용자 계정을 생성하기 위한 명령을 사용하여 사용자를 **SELinux staff_u** 사용자에게 매핑합니다.

사전 요구 사항

- root 사용자는 기본적으로 Red Hat Enterprise Linux에서와 마찬가지로 제한되지 않습니다.

절차

1. 다음 명령을 입력하여 *example.user* 라는 새 Linux 사용자를 생성하여 SELinux staff_u 사용자에게 매핑합니다.

```
# useradd -Z staff_u example.user
```

2. Linux *example.user* 사용자에게 암호를 할당하려면 다음을 수행합니다.

```
# passwd example.user
Changing password for user example.user.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

3. 현재 세션에서 로그아웃합니다.

4. Linux *example.user* 사용자로 로그인합니다. 사용자의 쉘은 staff_u 컨텍스트로 시작됩니다.

검증

1. *example.user* 사용자로 로그인하면 Linux 사용자의 컨텍스트를 확인합니다.

```
$ id -Z
uid=1000(example.user) gid=1000(example.user) groups=1000(example.user)
context=staff_u:staff_r:staff_t:s0-s0:c0.c1023
```

추가 리소스

- pam_selinux(8) 도움말 페이지.

3.5. 일반 사용자 제한

시스템에 있는 모든 일반 사용자를 user_u SELinux 사용자에게 매핑하여 제한할 수 있습니다.

기본적으로 관리 권한이 있는 사용자를 포함하여 **Red Hat Enterprise Linux**의 모든 **Linux** 사용자는 제한되지 않은 **SELinux** 사용자 **unconfined_u** 에 매핑됩니다. **SELinux** 제한 사용자에게 사용자를 할당하여 시스템의 보안을 개선할 수 있습니다. 이는 **V-71971** 보안 기술 구현 가이드 를 준수하는 데 유용합니다.

절차

1. **SELinux** 로그인 레코드 목록을 표시합니다. 이 목록에는 **Linux** 사용자의 매핑이 **SELinux** 사용자에게 표시됩니다.

```
# semanage login -l

Login Name    SELinux User  MLS/MCS Range   Service

__default__   unconfined_u  s0-s0:c0.c1023      *
root          unconfined_u  s0-s0:c0.c1023      *
```

2. 명시적 매핑이 없는 모든 사용자를 **user_u SELinux** 사용자에게 나타내는 **__default__** 사용자를 매핑합니다.

```
# semanage login -m -s user_u -r s0 __default__
```

검증

1. **__default__** 사용자가 **user_u SELinux** 사용자에게 매핑되었는지 확인합니다.

```
# semanage login -l

Login Name    SELinux User   MLS/MCS Range    Service

__default__   user_u         s0               *
root          unconfined_u   s0-s0:c0.c1023   *
```

2. **user_u:user_r:user_t:s0 SELinux** 컨텍스트에서 새 사용자의 프로세스가 실행되는지 확인합니다.

   a. 새 사용자를 생성합니다.

```
# adduser example.user
```

b.

example.user **에 대한 암호를 정의합니다.**

```
# passwd example.user
```

c.

root **로 로그아웃하고 새 사용자로 로그인합니다.**

d.

사용자 ID의 보안 컨텍스트를 표시합니다.

```
[example.user@localhost ~]$ id -Z
user_u:user_r:user_t:s0
```

e.

사용자 현재 프로세스의 보안 컨텍스트를 표시합니다.

```
[example.user@localhost ~]$ ps axZ
LABEL                             PID TTY      STAT   TIME COMMAND
-                                   1 ?        Ss     0:05 /usr/lib/systemd/systemd --switched-root --
system --deserialize 18
-                                3729 ?        S      0:00 (sd-pam)
user_u:user_r:user_t:s0          3907 ?        Ss     0:00 /usr/lib/systemd/systemd --user
-                                3911 ?        S      0:00 (sd-pam)
user_u:user_r:user_t:s0          3918 ?        S      0:00 sshd: example.user@pts/0
user_u:user_r:user_t:s0          3922 pts/0    Ss     0:00 -bash
user_u:user_r:user_dbusd_t:s0  3969 ?        Ssl    0:00 /usr/bin/dbus-daemon --session --
address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
user_u:user_r:user_t:s0          3971 pts/0    R+     0:00 ps axZ
```

3.6. ADMINISTRATOR_U에 매핑하여 관리자 제한

사용자를tekton _u SELinux 사용자에게 직접 매핑하여 관리 권한으로 사용자를 제한 할 수 있습니다. 사용자가 로그인하면 session이tekton _u:sysadm_r:sysadm_t SELinux 컨텍스트에서 실행됩니다.

기본적으로 관리 권한이 있는 사용자를 포함하여 Red Hat Enterprise Linux의 모든 Linux 사용자는 제한되지 않은 SELinux 사용자 unconfined_u 에 매핑됩니다. SELinux 제한 사용자에게 사용자를 할당하여 시스템의 보안을 개선할 수 있습니다. 이는 V-71971 보안 기술 구현 가이드 를 준수하는 데 유용합니다.

사전 요구 사항

- **root 사용자는 unconfined를 실행합니다. 이는 Red Hat Enterprise Linux 기본값입니다.**

절차

1. 선택 사항: SSH를 사용하여 E gressIP_u 사용자가 시스템에 연결할 수 있도록 하려면 다음을 수행합니다.

```
# setsebool -P ssh_sysadm_login on
```

2. 새 사용자를 만들고 사용자를 wheel 사용자 그룹에 추가한 후 사용자를tekton _u SELinux 사용자에게 매핑 합니다.

```
# adduser -G wheel -Z sysadm_u example.user
```

3. 선택 사항: 기존 사용자를tekton _u SELinux 사용자에게 매핑하고 사용자를 wheel 사용자 그룹에 추가합니다.

```
# usermod -G wheel -Z sysadm_u example.user
```

검증

1. example.user 가tekton _u SELinux 사용자에게 매핑되는지 확인합니다.

```
# semanage login -l | grep example.user
example.user     sysadm_u    s0-s0:c0.c1023   *
```

2. 예를 들어 SSH를 사용하여 example.user 로 로그인하고 사용자의 보안 컨텍스트를 표시합니다.

```
[example.user@localhost ~]$ id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

3. root 사용자로 전환합니다.

```
$ sudo -i
[sudo] password for example.user:
```

4. 보안 컨텍스트가 변경되지 않은 상태로 남아 있는지 확인합니다.

```
# id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

5. 예를 들어 **sshd** 서비스를 다시 시작하십시오.

```
# systemctl restart sshd
```

출력이 없으면 명령이 성공적으로 완료되었습니다.

명령이 성공적으로 완료되지 않으면 다음 메시지가 출력됩니다.

```
Failed to restart sshd.service: Access denied
See system logs and 'systemctl status sshd.service' for details.
```

3.7. SUDO 및TEKTON_R 역할을 사용하여 관리자 제한

관리 권한이 있는 특정 사용자를 **staff_u SELinux** 사용자에게 매핑하고 사용자가 **principal_r SELinux** 관리자 역할을 얻을 수 있도록 **sudo** 를 구성할 수 있습니다. 이 역할을 통해 사용자는 **SELinux** 거부 없이 관리 작업을 수행할 수 있습니다. 사용자가 로그인하면 세션이 **staff_u:staff_r:staff_t SELinux** 컨텍스트에서 실행되지만 **sudo** 를 사용하여 명령을 입력하면 세션이 **staff_u:sysadm_r:sysadm_t** 컨텍스트로 변경됩니다.

기본적으로 관리 권한이 있는 사용자를 포함하여 **Red Hat Enterprise Linux**의 모든 **Linux** 사용자는 제한되지 않은 **SELinux** 사용자 **unconfined_u** 에 매핑됩니다. **SELinux** 제한 사용자에게 사용자를 할당하여 시스템의 보안을 개선할 수 있습니다. 이는 **V-71971** 보안 기술 구현 가이드 를 준수하는 데 유용합니다.

사전 요구 사항

- **root** 사용자는 **unconfined**를 실행합니다. 이는 **Red Hat Enterprise Linux** 기본값입니다.

절차

1. 새 사용자를 만들고 사용자를 **wheel** 사용자 그룹에 추가하고 사용자를 **staff_u SELinux** 사용자에게 매핑합니다.

```
# adduser -G wheel -Z staff_u example.user
```

2. 선택 사항: 기존 사용자를 **staff_u SELinux** 사용자에게 매핑하고 사용자를 **wheel** 사용자 그룹에 추가합니다.

```
# usermod -G wheel -Z staff_u example.user
```

3. ***example.user*** 가 **SELinux** 관리자 역할을 갖도록 하려면 **/etc/sudoers.d/** 디렉터리에 새 파일을 만듭니다. 예를 들면 다음과 같습니다.

```
# visudo -f /etc/sudoers.d/example.user
```

4. 새 파일에 다음 행을 추가합니다.

```
example.user ALL=(ALL) TYPE=sysadm_t ROLE=sysadm_r ALL
```

검증

1. ***example.user*** 가 **staff_u SELinux** 사용자에게 매핑되었는지 확인합니다.

```
# semanage login -l | grep example.user
example.user     staff_u    s0-s0:c0.c1023   *
```

2. 예를 들어 **SSH**를 사용하여 ***example.user*** 로 로그인하고 **root** 사용자로 전환합니다.

```
[example.user@localhost ~]$ sudo -i
[sudo] password for example.user:
```

3. 루트 보안 컨텍스트를 표시합니다.

```
# id -Z
staff_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

4. 예를 들어 **sshd** 서비스를 다시 시작하십시오.

```
# systemctl restart sshd
```

출력이 없으면 명령이 성공적으로 완료되었습니다.

명령이 성공적으로 완료되지 않으면 다음 메시지가 출력됩니다.

> Failed to restart sshd.service: Access denied
> See system logs and 'systemctl status sshd.service' for details.

3.8. 추가 리소스

- **unconfined_selinux(8), user_selinux(8), staff_selinux(8), E gressIP_selinux(8)** 도움말 페이지

- **SELinux 제한 사용자로 시스템 설정 방법**

4장. 비표준 구성을 사용하여 애플리케이션 및 서비스에 대한 SELINUX 구성

SELinux가 강제 모드에 있을 때 기본 정책은 대상 정책입니다. 다음 섹션에서는 프로세스에 대한 구성 기본값(예: 포트, 데이터베이스 위치 또는 파일 시스템 권한)을 변경한 후 다양한 서비스에 대한 SELinux 정책 설정 및 구성에 대한 정보를 제공합니다.

비표준 포트의 SELinux 유형을 변경하고, 기본 디렉터리의 변경 사항에 대해 잘못된 레이블을 식별하고 수정하고, SELinux 부울을 사용하여 정책을 조정하는 방법을 배웁니다.

4.1. 비표준 구성으로 APACHE HTTP 서버에 대한 SELINUX 정책 사용자 정의

다른 포트에서 수신 대기하고 기본이 아닌 디렉터리에 콘텐츠를 제공하도록 Apache HTTP 서버를 구성할 수 있습니다. 연속 SELinux 거부를 방지하려면 이 절차의 단계에 따라 시스템의 SELinux 정책을 조정합니다.

사전 요구 사항

- httpd 패키지가 설치되고 Apache HTTP 서버는 TCP 포트 3131에서 수신 대기하고 기본 /var/www/ 디렉터리 대신 /var/test_www/ 디렉터리를 사용하도록 구성됩니다.
- policycoreutils-python-utils 및 se rsh-server 패키지가 시스템에 설치되어 있습니다.

절차

1. httpd 서비스를 시작하고 상태를 확인합니다.

```
# systemctl start httpd
# systemctl status httpd
...
httpd[14523]: (13)Permission denied: AH00072: make_sock: could not bind to address
[::]:3131
...
systemd[1]: Failed to start The Apache HTTP Server.
...
```

2. SELinux 정책은 httpd 가 포트 80에서 실행되는 것으로 가정합니다.

```
# semanage port -l | grep http
http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t              udp      3130
```

```
http_port_t                tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t        tcp      5988
pegasus_https_port_t       tcp      5989
```

3. 포트 **3131**의 **SELinux** 유형을 포트 **80**과 일치하도록 변경합니다.

```
# semanage port -a -t http_port_t -p tcp 3131
```

4. **httpd** 를 다시 시작합니다.

```
# systemctl start httpd
```

5. 그러나 콘텐츠는 계속 액세스할 수 없습니다.

```
# wget localhost:3131/index.html
...
HTTP request sent, awaiting response... 403 Forbidden
...
```

sealert 툴로 이유를 찾으십시오.

```
# sealert -l "*"
...
SELinux is preventing httpd from getattr access on the file /var/test_www/html/index.html.
...
```

6. **matchpathcon** 도구를 사용하여 표준 및 새 경로의 **SELinux** 유형을 비교합니다.

```
# matchpathcon /var/www/html /var/test_www/html
/var/www/html       system_u:object_r:httpd_sys_content_t:s0
/var/test_www/html  system_u:object_r:var_t:s0
```

7. 새 **/var/test_www/html/** 콘텐츠 디렉터리의 **SELinux** 유형을 기본 **/var/www/html** 디렉터리 유형으로 변경합니다.

```
# semanage fcontext -a -e /var/www /var/test_www
```

8. **/var** 디렉터리의 레이블을 재귀적으로 레이블을 다시 지정합니다.

```
# restorecon -Rv /var/
...
Relabeled /var/test_www/html from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /var/test_www/html/index.html from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

검증

1. **httpd** 서비스가 실행 중인지 확인합니다.

```
# systemctl status httpd
...
Active: active (running)
...
systemd[1]: Started The Apache HTTP Server.
httpd[14888]: Server configured, listening on: port 3131
...
```

2. **Apache HTTP** 서버에서 제공하는 콘텐츠에 액세스할 수 있는지 확인합니다.

```
# wget localhost:3131/index.html
...
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/html]
Saving to: ‘index.html’
...
```

추가 리소스

- **semanage(8), matchpathcon(8), sealert(8)** 매뉴얼 페이지.

4.2. SELINUX 부울을 사용하여 NFS 및 CIFS 볼륨 공유 정책 조정

SELinux 정책 작성에 대한 지식이 없어도 부울을 사용하여 런타임에 **SELinux** 정책의 일부를 변경할 수 있습니다. 이렇게 하면 **SELinux** 정책을 다시 로드하거나 다시 컴파일하지 않고도 **NFS** 볼륨에 대한 서비스 액세스 허용과 같은 변경이 가능합니다. 다음 절차에서는 **SELinux** 부울을 나열하고 정책을 변경하도록 구성하는 방법을 보여줍니다.

클라이언트 측의 **NFS** 마운트는 **NFS** 볼륨의 정책으로 정의된 기본 컨텍스트로 레이블이 지정됩니다. **RHEL**에서 이 기본 컨텍스트는 **nfs_t** 유형을 사용합니다. 또한 클라이언트 측에 마운트된 **Samba** 공유는 정책에 의해 정의된 기본 컨텍스트로 레이블이 지정됩니다. 이 기본 컨텍스트는 **cifs_t** 유형을 사용합니

다. 부울을 활성화하거나 비활성화하여 **nfs_t** 및 **cifs_t** 유형에 액세스할 수 있는 서비스를 제어할 수 있습니다.

Apache HTTP 서버 서비스(**httpd**)가 **NFS** 및 **CIFS** 볼륨에 액세스하고 공유할 수 있도록 하려면 다음 단계를 수행합니다.

사전 요구 사항

- 선택적으로 **selinux-policy-devel** 패키지를 설치하여 **semanage boolean -l** 명령의 출력에서 **SELinux** 부울에 대한 자세한 설명을 얻습니다.

절차

1. **NFS, CIFS** 및 **Apache**와 관련된 **SELinux** 부울을 식별합니다.

```
# semanage boolean -l | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs                 (off  ,  off)  Allow httpd to access cifs file systems
httpd_use_nfs                  (off  ,  off)  Allow httpd to access nfs file systems
```

2. 부울의 현재 상태를 나열합니다.

```
$ getsebool -a | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs --> off
httpd_use_nfs --> off
```

3. 식별된 부울을 활성화합니다.

```
# setsebool httpd_use_nfs on
# setsebool httpd_use_cifs on
```

참고

-P 옵션과 함께 **setsebool** 을 사용하여 다시 시작해도 변경 사항을 지속합니다. **setsebool -P** 명령은 전체 정책을 다시 빌드해야 하며 구성에 따라 다소 시간이 걸릴 수 있습니다.

검증

1. 부울이 있는지 확인합니다.

```
$ getsebool -a | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs --> on
httpd_use_nfs --> on
```

추가 리소스

- **semanage-boolean(8), sepolicy-booleans(8), getsebool(8), setsebool(8), booleans(5),** 및 **booleans(8)** 매뉴얼 페이지

4.3. 추가 리소스

- SELinux 관련 문제 해결

5장. SELINUX 관련 문제 해결

이전에 비활성화되었거나 비표준 구성에서 서비스를 실행하는 시스템에서 **SELinux**를 활성화하려는 경우 **SELinux**에서 잠재적으로 차단하는 상황을 해결해야 할 수 있습니다. 대부분의 경우 **SELinux** 거부는 잘못된 구성의 신호입니다.

5.1. SELINUX 거부 확인

이 절차의 필요한 단계만 따릅니다. 대부분의 경우 **1**단계를 수행해야 합니다.

절차

1. **SELinux**에서 시나리오를 차단하면 **/var/log/audit/audit.log** 파일이 거부에 대한 자세한 정보를 확인하는 첫 번째 위치입니다. 감사 로그를 쿼리하려면 **ausearch** 툴을 사용합니다. 액세스 허용 또는 허용과 같은 **SELinux** 의사 결정이 캐시되고 이 캐시를 **AVC(Access Vector Cache)**로 알려져 있으므로 메시지 유형 매개변수에 **AVC** 및 **USER_AVC** 값을 사용합니다.

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
```

일치하는 항목이 없으면 감사 데몬이 실행 중인지 확인합니다. 그렇지 않으면 **auditd** 를 시작한 후 거부된 시나리오를 반복하고 감사 로그를 다시 확인합니다.

2. **auditd** 가 실행 중인 경우 **ausearch** 의 출력에 일치하는 항목이 없는 경우 **systemd journal**에서 제공하는 메시지를 확인합니다.

```
# journalctl -t setroubleshoot
```

3. **SELinux**가 활성 상태이고 시스템에서 감사 데몬이 실행되지 않는 경우 **dmesg** 명령의 출력에서 특정 **SELinux** 메시지를 검색합니다.

```
# dmesg | grep -i -e type=1300 -e type=1400
```

4. 이전 세 번의 점검 후에도 여전히 아무것도 찾을 수 없습니다. 이 경우 **dontaudit** 규칙으로 인해 **AVC** 거부를 음소거할 수 있습니다.

dontaudit 규칙을 일시적으로 비활성화하려면 모든 거부를 로깅할 수 있습니다.

```
# semodule -DB
```

거부된 시나리오를 다시 실행하고 이전 단계를 사용하여 거부 메시지를 확인한 후 다음 명령을 실행하면 정책에서 **dontaudit** 규칙을 다시 활성화합니다.

```
# semodule -B
```

5. 이전 **4** 단계를 모두 적용하고 문제가 여전히 확인되지 않은 상태로 남아 있는 경우 **SELinux**가 시나리오를 실제로 차단하는지 고려하십시오.

- 허용 모드로 전환합니다.

```
# setenforce 0
$ getenforce
Permissive
```

- 시나리오를 반복합니다.

문제가 계속 발생하면 **SELinux**가 시나리오를 차단하는 것과 다른 문제가 있습니다.

5.2. SELINUX 거부 메시지 분석

SELinux가 시나리오를 차단하고 있음을 확인한 후 수정을 선택하기 전에 근본 원인을 분석해야 할 수 있습니다. **???**

사전 요구 사항

- **policycoreutils-python-utils** 및 **se rsh-server** 패키지가 시스템에 설치되어 있습니다.

절차

1. 다음과 같이 **sealert** 명령을 사용하여 기록된 거부에 대한 세부 정보를 나열합니다.

```
$ sealert -l "*"
SELinux is preventing /usr/bin/passwd from write access on the file
/root/test.

*****  Plugin leaks (86.2 confidence) suggests ******************************

If you want to ignore passwd trying to write access the test file,
```

```
because you believe it should not need this access.
Then you should report this as a bug.
You can generate a local policy module to dontaudit this access.
Do
# ausearch -x /usr/bin/passwd --raw | audit2allow -D -M my-passwd
# semodule -X 300 -i my-passwd.pp

*****  Plugin catchall (14.7 confidence) suggests **************************

...

Raw Audit Messages
type=AVC msg=audit(1553609555.619:127): avc:  denied  { write } for
pid=4097 comm="passwd" path="/root/test" dev="dm-0" ino=17142697
scontext=unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0

...

Hash: passwd,passwd_t,admin_home_t,file,write
```

2. **이전 단계에서 얻은 출력에 명확한 제안이 포함되어 있지 않은 경우:**

   - **전체 경로 감사를 활성화하여 액세스된 오브젝트에 대한 전체 경로를 확인하고 추가 Linux 감사 이벤트 필드를 볼 수 있습니다.**

     ```
     # auditctl -w /etc/shadow -p w -k shadow-write
     ```

   - **se rsh 캐시를 지웁니다.**

     ```
     # rm -f /var/lib/setroubleshoot/setroubleshoot.xml
     ```

   - **문제를 재현합니다.**

   - **1단계를 반복합니다.**

     프로세스를 완료한 후 전체 경로 감사를 비활성화합니다.

     ```
     # auditctl -W /etc/shadow -p w -k shadow-write
     ```

3. **sealert 가 catchall 제안만 반환하거나 audit2allow 툴을 사용하여 새 규칙 추가를 권장하는 경우, 나열된 예제와 일치하는 문제를 감사 로그의 SELinux 거부에 설명되어 있습니다.**

추가 리소스

- **sealert(8)** 도움말 페이지

5.3. 분석된 SELINUX 거부 수정

대부분의 경우 **sealert** 툴에서 제공하는 제안 사항은 **SELinux** 정책과 관련된 문제를 해결하는 방법에 대한 올바른 지침을 제공합니다. **sealert** 를 사용하여 SELinux 거부를 분석하는 방법은 **Analyzing SELinux denial** 메시지를 참조하십시오.

툴에서 구성 변경에 **audit2allow** 툴 사용을 제안하는 경우 주의하십시오. **SELinux** 거부가 표시될 때 **audit2allow** 를 사용하여 로컬 정책 모듈을 첫 번째 옵션으로 생성해서는 안 됩니다. 문제 해결은 레이블 문제가 있는지 확인하여 시작해야 합니다. 두 번째 경우는 프로세스 구성을 변경했으며 **SELinux**에 대해 알리는 것을 잊어 버렸습니다.

라벨 문제

레이블 지정 문제의 일반적인 원인은 비표준 디렉터리가 서비스에 사용되는 경우입니다. 예를 들어, 웹 사이트에 **/var/www/html/** 를 사용하는 대신 관리자는 **/srv/myweb/.**를 사용할 수 있습니다. **Red Hat Enterprise Linux**에서 **/srv** 디렉터리는 **var_t** 유형으로 레이블이 지정됩니다. **/srv** 에서 생성된 파일 및 디렉터리는 이 유형을 상속합니다. 또한 최상위 디렉토리에서 새로 생성된 오브젝트(예: **/myserver**)는 **default_t** 유형으로 레이블을 지정할 수 있습니다. **SELinux**는 **Apache HTTP Server(httpd)**가 이러한 유형에 모두 액세스하지 못하도록 합니다. 액세스를 허용하려면 **SELinux**에서 **httpd:**를 통해 **/srv/myweb/** 의 파일에 액세스할 수 있는지 알아야 합니다.

```
# semanage fcontext -a -t httpd_sys_content_t "/srv/myweb(/.*)?"
```

이 **semanage** 명령은 **/srv/myweb/** 디렉터리 및 그 아래의 모든 파일 및 디렉터리에 대한 컨텍스트를 **SELinux file-context** 구성에 추가합니다. **semanage** 유틸리티는 컨텍스트를 변경하지 않습니다. **root**로 **restorecon** 유틸리티를 사용하여 변경 사항을 적용합니다.

```
# restorecon -R -v /srv/myweb
```

잘못된 컨텍스트

matchpathcon 유틸리티는 파일 경로의 컨텍스트를 확인하고 해당 경로의 기본 레이블과 비교합니다. 다음 예제는 잘못 레이블이 지정된 파일이 포함된 디렉터리에서 **matchpathcon** 을 사용하는 방법을 보여줍니다.

```
$ matchpathcon -V /var/www/html/*
/var/www/html/index.html has context unconfined_u:object_r:user_home_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
```

```
/var/www/html/page1.html has context unconfined_u:object_r:user_home_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
```

이 예에서 **index.html** 및 **page1.html** 파일은 **user_home_t** 유형으로 레이블이 지정됩니다. 이 유형은 사용자 홈 디렉터리의 파일에 사용됩니다. **mv** 명령을 사용하여 홈 디렉토리에서 파일을 이동하면 파일이 **user_home_t** 유형으로 레이블이 지정될 수 있습니다. 이 유형은 홈 디렉터리 외부에 없어야 합니다. **restorecon** 유틸리티를 사용하여 이러한 파일을 올바른 형식으로 복원합니다.

```
# restorecon -v /var/www/html/index.html
restorecon reset /var/www/html/index.html context unconfined_u:object_r:user_home_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

디렉터리에 있는 모든 파일의 컨텍스트를 복원하려면 **-R** 옵션을 사용합니다.

```
# restorecon -R -v /var/www/html/
restorecon reset /var/www/html/page1.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /var/www/html/index.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

비표준 방식으로 구성된 제한된 애플리케이션

서비스는 다양한 방식으로 실행될 수 있습니다. 이를 위해서는 서비스 실행 방법을 지정해야 합니다. **SELinux** 부울을 통해 **SELinux** 정책의 일부를 런타임 시 변경할 수 있습니다. 이렇게 하면 **SELinux** 정책을 다시 로드하거나 다시 컴파일하지 않고도 **NFS** 볼륨에 대한 서비스 액세스 허용과 같은 변경이 가능합니다. 또한 기본이 아닌 포트 번호에서 서비스를 실행하려면 **semanage** 명령을 사용하여 업데이트하는 정책 구성이 필요합니다.

예를 들어 **Apache HTTP** 서버가 **MariaDB**와 통신하도록 허용하려면 **httpd_can_network_connect_db** 부울을 활성화합니다.

```
# setsebool -P httpd_can_network_connect_db on
```

-P 옵션을 사용하면 시스템을 재부팅해도 설정이 지속됩니다.

특정 서비스에 대해 액세스가 거부되면 **getsebool** 및 **grep** 유틸리티를 사용하여 액세스를 허용하는 부울을 사용할 수 있는지 확인합니다. 예를 들어 **getsebool -a | grep ftp** 명령을 사용하여 **FTP** 관련 부울을 검색합니다.

```
$ getsebool -a | grep ftp
ftpd_anon_write --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
```

```
ftpd_use_nfs --> off

ftpd_connect_db --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
```

부울 목록을 가져오고 활성화되어 있는지 확인하려면 **getsebool -a** 명령을 사용합니다. 의미와 같은 부울 목록을 가져오고 해당 의미가 활성화되어 있는지 확인하려면 **selinux-policy-devel** 패키지를 설치하고 **semanage boolean -l** 명령을 **root**로 사용합니다.

포트 번호

정책 구성에 따라 특정 포트 번호에서만 서비스를 실행할 수 있습니다. 정책을 변경하지 않고 서비스가 실행되는 포트를 변경하면 서비스가 시작되지 않을 수 있습니다. 예를 들어 **semanage port -l | grep http** 명령을 **root**로 실행하여 **http** 관련 포트를 나열합니다.

```
# semanage port -l | grep http
http_cache_port_t              tcp      3128, 8080, 8118
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

http_port_t 포트 유형은 포트 **Apache HTTP Server**가 수신할 수 있는 포트를 정의합니다. 이 경우 **TCP** 포트 **80, 443, 488, 8008, 8009** 및 **8443**입니다. 관리자가 **httpd** 가 포트 **9876(List 9876)**에서 수신 대기하도록 **httpd.conf** 를 구성하지만 이를 반영하도록 정책이 업데이트되지 않으면 다음 명령이 실패합니다.

```
# systemctl start httpd.service
Job for httpd.service failed. See 'systemctl status httpd.service' and 'journalctl -xn' for details.

# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: failed (Result: exit-code) since Thu 2013-08-15 09:57:05 CEST; 59s ago
  Process: 16874 ExecStop=/usr/sbin/httpd $OPTIONS -k graceful-stop (code=exited,
status=0/SUCCESS)
  Process: 16870 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
```

다음과 유사한 **SELinux** 거부 메시지는 **/var/log/audit/audit.log** 에 기록됩니다.

```
type=AVC msg=audit(1225948455.061:294): avc:  denied  { name_bind } for  pid=4997
comm="httpd" src=9876 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:port_t:s0 tclass=tcp_socket
```

httpd 가 **http_port_t** 포트 유형에 대해 나열되지 않은 포트에서 수신 대기하도록 허용하려면 **semanage port** 명령을 사용하여 다른 레이블을 포트에 할당합니다.

```
# semanage port -a -t http_port_t -p tcp 9876
```

a 옵션은 새 레코드를 추가합니다. **-t** 옵션은 유형을 정의하고 **-p** 옵션은 프로토콜을 정의합니다. 마지막 인수는 추가할 포트 번호입니다.

코너 케이스, 진화되거나 손상된 애플리케이션, 손상된 시스템

애플리케이션에는 **SELinux**가 액세스를 거부하도록 하는 버그가 포함될 수 있습니다. 또한 **SELinux** 규칙이 진화하고 있습니다. **SELinux**는 애플리케이션이 예상대로 작동하는 경우에도 특정 방식으로 실행되는 애플리케이션이 표시되지 않을 수 있습니다. 예를 들어 **PostgreSQL**의 새 버전이 릴리스되면 현재 정책이 고려하지 않는 작업을 수행할 수 있으므로 액세스가 허용되어야 하는 경우에도 액세스가 거부됩니다.

이러한 경우 액세스가 거부된 후 **audit2allow** 유틸리티를 사용하여 액세스를 허용하는 사용자 지정 정책 모듈을 생성합니다. **Red Hat Bugzilla** 의 **SELinux** 정책에서 누락된 규칙을 보고할 수 있습니다. **Red Hat Enterprise Linux 9**의 경우 **Red Hat Enterprise Linux 9** 제품에 대해 버그를 생성하고 **selinux-policy** 구성 요소를 선택합니다. 이러한 버그 보고서에 **audit2allow -w -a** 및 **audit2allow -a** 명령의 출력을 포함합니다.

애플리케이션에서 주요 보안 권한을 요청하면 애플리케이션이 손상되는 신호일 수 있습니다. 침입 탐지 도구를 사용하여 이러한 의심 스러운 동작을 검사합니다.

Red Hat 고객 포털의 솔루션 엔진은 또한 사용자가 가진 동일한 또는 매우 유사한 문제에 대한 가능한 솔루션이 포함된 문서 형태로 지침을 제공할 수 있습니다. 관련 제품 및 버전을 선택하고 ***selinux*** 또는 ***avc*** 와 같은 **SELinux** 관련 키워드를 사용하여 차단된 서비스 또는 애플리케이션 이름(예: **selinux samba**)을 사용합니다.

5.4. 감사 로그의 SELINUX 거부

Linux 감사 시스템은 기본적으로 **/var/log/audit/audit.log** 파일에 로그 항목을 저장합니다.

SELinux 관련 레코드만 나열하려면 적어도 **AVC** 및 **AVC_USER** 로 설정된 메시지 유형 매개 변수와 함께 **ausearch** 명령을 사용하십시오.

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR
```

감사 로그 파일의 **SELinux** 거부 항목은 다음과 같습니다.

```
type=AVC msg=audit(1395177286.929:1638): avc:  denied  { read } for  pid=6591 comm="httpd"
name="webpages" dev="0:37" ino=2112 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:nfs_t:s0 tclass=dir
```

이 항목의 가장 중요한 부분은 다음과 같습니다.

- **AVC: denied - SELinux**에서 수행하고 **AVC(Access Vector Cache)**에 기록된 동작입니다.
- **{ read } -** 거부된 작업
- **PID=6591 -** 거부된 작업을 수행하려는 주체의 프로세스 식별자
- **Comm="httpd" -** 분석 프로세스를 호출하는 데 사용된 명령의 이름입니다.
- **httpd_t -** 프로세스의 **SELinux** 유형
- **nfs_t -** 프로세스 작업의 영향을 받는 오브젝트의 **SELinux** 유형
- **tclass=dir -** 대상 오브젝트 클래스

이전 로그 항목은 다음과 같이 번역할 수 있습니다.

***SELinux*는 *PID 6591* 및 *httpd_t* 유형의 *httpd* 프로세스를 거부하여 *nfs_t* 유형이 있는 디렉터리에서 읽습니다.**

Apache HTTP Server가 **Samba** 제품군 유형으로 레이블이 지정된 디렉터리에 액세스하려고 할 때 다음 **SELinux** 거부 메시지가 발생합니다.

```
type=AVC msg=audit(1226874073.147:96): avc:  denied  { getattr } for  pid=2465 comm="httpd"
path="/var/www/html/file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file
```

- **{ getattr } - getattr** 항목은 소스 프로세스가 대상 파일의 상태 정보를 읽으려고 한다는 것을 나타냅니다. 이 작업은 파일을 읽기 전에 수행됩니다. 프로세스가 파일에 액세스하고 적절한 레이블이 없기 때문에 **SELinux**는 이 작업을 거부합니다. 일반적으로 표시되는 권한에는 **getattr,read, write** 가 포함됩니다.

- **path="/var/www/html/file1" -** 프로세스가 액세스하려고 시도한 오브젝트(대상)의 경로입니다.

- **scontext="unconfined_u:system_r:httpd_t:s0" -** 거부된 작업을 시도한 프로세스(소스)의 **SELinux** 컨텍스트입니다. 이 경우 **httpd_t** 유형으로 실행 중인 **Apache HTTP** 서버의 **SELinux** 컨텍스트입니다.

- **tcontext="unconfined_u:object_r:hiera_share_t:s0" -** 프로세스가 액세스하려고 시도한 오브젝트(대상)의 **SELinux** 컨텍스트입니다. 이 경우 **file1** 의 **SELinux** 컨텍스트입니다.

이 **SELinux** 거부는 다음과 같이 번역할 수 있습니다.

SELinux는 PID 2465로 httpd 프로세스를 거부하여 samba_share_t 유형을 사용하여 /var/www/html/file1 파일에 액세스했습니다. 이 파일은 달리 구성하지 않는 한 httpd_t 도메인에서 실행되는 프로세스에 액세스할 수 없습니다.

추가 리소스

- **auditd(8)** 및 **ausearch(8)** 도움말 페이지

5.5. 추가 리소스

- CLI의 기본 SELinux 문제 해결

- SELinux가 제공하는 것은 무엇입니까? SELinux 오류 4가지 주요 원인

6장. MULTI-LEVEL SECURITY (MLS) 사용

다단계 보안 (MLS) 정책은 원래 미국 방위에 의해 설계 된 것처럼 명확한 수준을 사용합니다. MLS는 정부 기관과 같이 엄격하게 통제되는 환경에서 정보 관리에 따라 매우 좁은 보안 요구 사항을 충족합니다.

MLS 사용은 복잡하고 일반적인 사용 사례 시나리오에 잘 매핑되지 않습니다.

6.1. 다단계 보안 (MLS)

Multi-Level Security (MLS) 기술은 정보 보안 수준을 사용하여 계층 분류의 데이터를 분류합니다. 예를 들면 다음과 같습니다.

- [lowest] Unclassified
- [low] 신뢰할 수 있습니다.
- [high] 보안
- [highest] Top secret

기본적으로 MLS SELinux 정책은 16개의 민감도 수준을 사용합니다.

- s0 은 최소 민감합니다.
- S 15 가 가장 민감합니다.

MLS는 민감도 수준을 해결하기 위해 특정 용어를 사용합니다.

- 사용자 및 프로세스는 중요도 수준을 명확한이라고 하는 주제 라고 합니다.

- 시스템의 파일, 장치 및 기타 수동 구성 요소는 분류 라고 하는 개체 라고 합니다.

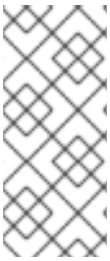
MLS를 구현하기 위해 **SELinux**는 **BLP(P hat-La Padula Model)** 모델을 사용합니다. 이 모델은 각 주체 및 개체에 연결된 라벨에 따라 시스템 내에서 정보가 이동하는 방법을 지정합니다.

BLP의 기본 원칙이 "미읽기 없음, 쓰기없음." 즉, 사용자가 자신의 민감도 수준에서만 파일을 읽을 수 있으며 데이터는 하위 수준에서 더 높은 수준으로만 이동할 수 있으며 그 반대의 경우도 없습니다.

RHEL에서 **MLS**를 구현하는 **MLS SELinux** 정책은 쓰기 같음을 사용하여 **Bell-La Padula** 라는 수정된 원칙을 적용합니다. 즉, 사용자는 자신의 민감도 수준에서 파일을 읽을 수 있지만 정확히 자신의 수준에서만 쓸 수 있습니다. 이렇게 하면, 예를 들어 저해 사용자가 콘텐츠를 **top-secret** 파일에 쓰지 못하게 합니다.

예를 들어 기본적으로 명확한 수준 **s2** 를 가진 사용자:

- 민감도 수준 **s0,s1, s2** 로 파일을 읽을 수 있습니다.
- 민감도 수준 **s3** 이상이 있는 파일을 읽을 수 없습니다.
- 정확히 **s2** 의 민감도 수준으로 파일을 수정할 수 있습니다.
- 민감도 수준이 **s2** 와 다른 파일을 수정할 수 없습니다.



참고

보안 관리자는 시스템의 **SELinux** 정책을 수정하여 이 동작을 조정할 수 있습니다. 예를 들어 사용자가 더 낮은 수준에서 파일을 수정할 수 있도록 허용하여 파일의 민감도 수준을 사용자의 명확한 수준으로 높일 수 있습니다.

실제로 사용자는 일반적으로 광범위한 명확한 레벨(예: **s1-s2**)에 할당됩니다. 사용자는 사용자의 최대 수준보다 민감도 수준이 낮은 파일을 읽고 해당 범위 내의 모든 파일에 쓸 수 있습니다.

예를 들어 기본적으로 범위가 **s1-s2** 인 사용자는 다음을 수행합니다.

- 민감도 수준 **s0** 및 **s1** 을 사용하여 파일을 읽을 수 있습니다.
- 민감도 수준 **s2** 이상의 파일을 읽을 수 없습니다.
- 민감도 수준 **s1** 로 파일을 수정할 수 있습니다.
- 민감도 수준이 **s1** 과 다른 파일을 수정할 수 없습니다.
- 자신의 명료 수준을 **s2** 로 변경할 수 있습니다.

MLS 환경에서 권한이 없는 사용자의 보안 컨텍스트는 다음과 같습니다.

```
user_u:user_r:user_t:s1
```

다음과 같습니다.

user_u

은 **SELinux** 사용자입니다.

user_r

은 **SELinux** 역할입니다.

user_t

SELinux 유형입니다.

s1

MLS 민감도 수준의 범위입니다.

시스템은 항상 **MLS** 액세스 규칙과 기존 파일 액세스 권한을 결합합니다. 예를 들어, **"Secret"**의 보안 수준을 가진 사용자가 **DC(Discretionary Access Control)**를 사용하여 다른 사용자의 파일에 대한 액세

스를 차단하면, "Top Secret" 사용자도 해당 파일에 액세스할 수 없습니다. 보안 수준이 높은 경우 사용자가 전체 파일 시스템을 검색하도록 자동으로 허용하지 않습니다.

최상위 수준의 명확한 권한을 가진 사용자는 다단계 시스템에 대한 관리 권한을 자동으로 취득하지 않습니다. 시스템에 대한 모든 중요한 정보에 액세스할 수 있지만 이는 관리 권한이 있는 것과 다릅니다.

또한 관리자 권한은 중요한 정보에 대한 액세스 권한을 제공하지 않습니다. 예를 들어, 사용자가 **root** 로 로그인한 경우에도 **top-secret** 정보를 읽을 수 없습니다.

범주를 사용하여 **MLS** 시스템 내에서 액세스를 추가로 조정할 수 있습니다. **MCCS(Multi-Category Security)**를 사용하면 프로젝트 또는 부서와 같은 카테고리를 정의할 수 있으며 사용자는 할당된 카테고리의 파일에만 액세스할 수 있습니다. 자세한 내용은 데이터 기밀성을 위해 **MCS(Multi-Category Security)** 사용을 참조하십시오.

6.2. MLS의 SELINUX 역할

SELinux 정책은 각 **Linux** 사용자를 **SELinux** 사용자에게 매핑합니다. 이를 통해 **Linux** 사용자는 **SELinux** 사용자의 제한을 상속할 수 있습니다.

중요

MLS 정책에는 제한되지 않은 사용자, 유형 및 역할을 포함하여 제한되지 않은 모듈이 포함되어 있지 않습니다. 결과적으로 **root** 를 포함하여 제한되지 않은 사용자는 모든 오브젝트에 액세스할 수 없으며 대상 정책에서 수행할 수 있는 모든 조치를 수행할 수 없습니다.

정책의 부울을 조정하여 **SELinux** 정책에서 제한된 사용자에 대한 권한을 특정 요구 사항에 따라 사용자 지정할 수 있습니다. **semanage boolean -l** 명령을 사용하여 이러한 부울의 현재 상태를 확인할 수 있습니다.

표 **6.1. MLS**의 **SELinux** 사용자 역할

사용자	기본 역할	추가 역할
guest_u	**guest_r**	
xguest_u	**xguest_r**	
user_u	**user_r**	

사용자	기본 역할	추가 역할
staff_u	**staff_r**	**auditadm_r**
		secadm_r
		sysadm_r
		staff_r
sysadm_u	**sysadm_r**	
root	**staff_r**	**auditadm_r**
		secadm_r
		sysadm_r
		system_r
system_u	**system_r**	

system_u 는 시스템 프로세스 및 오브젝트의 특수 사용자 **ID**이며 **system_r** 은 연결된 역할입니다. 관리자는 이 **system_u** 사용자와 **system_r** 역할을 **Linux** 사용자와 연결하지 않아야 합니다. 또한 **unconfined_u** 및 **root** 는 제한되지 않은 사용자입니다. 이러한 이유로 이러한 **SELinux** 사용자와 관련된 역할은 다음 표의 **SELinux** 역할에 포함되지 않습니다.

각 **SELinux** 역할은 **SELinux** 유형에 해당하며 특정 액세스 권한을 제공합니다.

표 6.2. MLS의 SELinux 역할 유형 및 액세스

Role	유형	X Window System을 사용하여 로그인	Su 및 sudo	홈 디렉토리 및 /tmp (기본값)에서 실행	네트워킹
guest_r	**guest_t**	제공되지 않음	제공되지 않음	제공됨	제공되지 않음
xguest_r	**xguest_t**	제공됨	제공되지 않음	제공됨	웹 브라우저만 (Fire Firefox, GNOME 웹)
user_r	**user_t**	제공됨	제공되지 않음	제공됨	제공됨
staff_r	**staff_t**	제공됨	**sudo**만	제공됨	제공됨

Role	유형	X Window System을 사용하여 로그인	Su 및 sudo	홈 디렉토리 및 /tmp (기본값)에서 실행	네트워킹
auditadm_r	**auditadm_t**		제공됨	제공됨	제공됨
secadm_r	**secadm_t**		제공됨	제공됨	제공됨
sysadm_r	**sysadm_t**	**xdm_sysadm_login** 부울이 **있는**경우에만	제공됨	제공됨	제공됨

- 기본적으로 **HEALTH _r** 역할에는 **secadm_r** 역할의 권한이 있습니다. 즉, **sensitive _r** 역할의 사용자는 보안 정책을 관리할 수 있습니다. 이 사용 사례와 일치하지 않는 경우 정책에서 **RuntimeClass _secadm** 모듈을 비활성화하여 두 역할을 분리할 수 있습니다. 자세한 내용은 **MLS의 보안 관리에서 시스템 관리 분리**를참조하십시오.

- 비로그인 역할 **dbadm_r,logadm_r, webadm_r** 는 관리 작업의 하위 집합에 사용할 수 있습니다. 기본적으로 이러한 역할은 **SELinux** 사용자와 연결되지 않습니다.

6.3. SELINUX 정책을 MLS로 전환

다음 단계를 사용하여 **SELinux** 정책을 대상에서 다단계 보안(**MLS**)으로 전환합니다.

중요

Red Hat은 **X Window System**을 실행 중인 시스템에서 **MLS** 정책을 사용하지 않는 것이 좋습니다. 또한 **MLS** 레이블을 사용하여 파일 시스템의 레이블을 다시 지정하면 시스템에서 제한된 도메인이 액세스하지 못하도록 시스템이 올바르게 시작되지 않을 수 있습니다. 따라서 파일의 레이블을 다시 지정하기 전에 **SELinux**를 허용 모드로 전환해야 합니다. 대부분의 시스템에서 **MLS**로 전환한 후 많은 **SELinux** 거부가 표시되며, 많은 시스템에서 수정할 수 없는 경우가 많습니다.

절차

1. **selinux-policy-mls** 패키지를 설치합니다.

```
# dnf install selinux-policy-mls
```

2. 선택한 텍스트 편집기에서 **/etc/selinux/config** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vi /etc/selinux/config
```

3. **SELinux** 모드를 **enforcing**에서 허용으로 변경하고 대상 정책에서 **MLS**로 전환합니다.

```
SELINUX=permissive
SELINUXTYPE=mls
```

변경 사항을 저장하고 편집기를 종료합니다.

4. **MLS** 정책을 활성화하려면 파일 시스템의 각 파일의 레이블을 **MLS** 레이블로 다시 지정해야 합니다.

```
# fixfiles -F onboot
System will relabel on next boot
```

5. 시스템을 다시 시작하십시오.

```
# reboot
```

6. **SELinux** 거부를 확인합니다.

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent -i
```

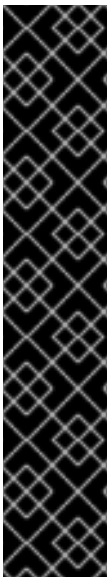
이전 명령은 모든 시나리오를 다루지 않으므로 **SELinux 거부 식별, 분석 및 수정에 대한 지침은 SELinux와 관련된 문제 해결**을 참조하십시오.

7. 시스템에서 **SELinux**와 관련된 문제가 없는지 확인한 후 **/etc/selinux/config** 에서 해당 옵션을 변경하여 **SELinux**를 강제 모드로 다시 전환합니다.

```
SELINUX=enforcing
```

8. 시스템을 다시 시작하십시오.

```
# reboot
```



중요

시스템이 시작되지 않거나 **MLS**로 전환한 후 로그인할 수 없는 경우 커널 명령줄에 **enforcing=0** 매개 변수를 추가합니다. 자세한 내용은 부팅 시 **SELinux** 모드 변경을 참조하십시오.

또한 **MLS**에서 **configures _r SELinux** 역할에 매핑된 **root** 사용자로 **SSH** 로그인은 **staff_r** 에서 **root** 로 로그인하는 것과 다릅니다. **MLS**에서 시스템을 시작하기 전에 **ssh_sysadm _login SELinux** 부울을 **1** 로 설정하여 **etcdctl_r** 로 **SSH** 로그인을 허용하십시오. 나중에 **ssh_sysadm_login** 을 활성화하려면 이미 **MLS**에 있는 **root** 로 로그인하고, **newrole - r tekton _r** 명령을 사용하여 **root** 로 전환한 다음, 부울을 **1** 로 설정해야 합니다.

검증

1. **SELinux**가 강제 모드에서 실행되는지 확인합니다.

```
# getenforce
Enforcing
```

2. **SELinux**의 상태가 **mls** 값을 반환하는지 확인합니다.

```
# sestatus | grep mls
Loaded policy name:             mls
```

추가 리소스

- **fixfiles(8), setsebool(8),** 및 **ssh_selinux(8)** 도움말 페이지.

6.4. MLS에서 사용자 명확한 설정

SELinux 정책을 **MLS**로 전환한 후에는 제한된 **SELinux** 사용자에게 매핑하여 보안 명확한 수준을 사용자에게 할당해야 합니다. 기본적으로 지정된 보안 명확한 사용자가 있습니다.

- 민감도 수준이 높은 개체를 읽을 수 없습니다.
- 다른 민감도 수준에서 오브젝트에 쓸 수 없습니다.

사전 요구 사항

- **SELinux** 정책은 **mls** 로 설정됩니다.
- **SELinux** 모드는 강제 모드로 설정되어 있습니다.
- **policycoreutils-python-utils** 패키지가 설치됩니다.
- **SELinux** 제한 사용자에게 할당된 사용자:
  - **user_u** 에 할당된 권한이 없는 사용자의 경우 다음 절차의*example_user* 입니다.
  - 권한이 있는 사용자의 경우 **staff_u** (다음 절차의 *담당자*)에 할당되었습니다.

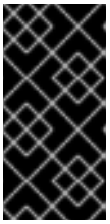
참고

MLS 정책이 활성화되었을 때 사용자가 생성되었는지 확인합니다. 다른 **SELinux** 정책에서 생성된 사용자는 **MLS**에서 사용할 수 없습니다.

절차

1. 선택 사항: **SELinux** 정책에 오류를 추가하지 않도록 하려면 허용 **SELinux** 모드로 전환하여 문제 해결에 도움이 됩니다.

```
# setenforce 0
```



중요

허용 모드에서 **SELinux**는 활성 정책을 적용하지 않지만 문제 해결 및 디버깅에 사용할 수 있는 **AVC(Access Vector Cache)** 메시지만 기록합니다.

2. **staff_u SELinux** 사용자의 명확한 범위를 정의합니다. 예를 들어 이 명령은 **s1**에서 **s1** 로 **clearance** 범위를 설정하고 **s1** 은 기본 명확한 수준입니다.

```
# semanage user -m -L s1 -r s1-s15 _staff_u
```

3. 사용자 홈 디렉터리에 대한 **SELinux** 파일 컨텍스트 구성 항목을 생성합니다.

```
# genhomedircon
```

4. 파일 보안 컨텍스트를 기본값으로 복원합니다.

```
# restorecon -R -F -v /home/
Relabeled /home/staff from staff_u:object_r:user_home_dir_t:s0 to
staff_u:object_r:user_home_dir_t:s1
Relabeled /home/staff/.bash_logout from staff_u:object_r:user_home_t:s0 to
staff_u:object_r:user_home_t:s1
Relabeled /home/staff/.bash_profile from staff_u:object_r:user_home_t:s0 to
staff_u:object_r:user_home_t:s1
Relabeled /home/staff/.bashrc from staff_u:object_r:user_home_t:s0 to
staff_u:object_r:user_home_t:s1
```

5. 사용자에게 명확한 수준을 할당합니다.

```
# semanage login -m -r s1 example_user
```

여기서 ***s1*** 은 사용자에게 할당된 명확한 수준입니다.

6. 사용자의 홈 디렉터리의 레이블을 사용자의 명확한 수준으로 다시 지정합니다.

```
# chcon -R -l s1 /home/example_user
```

7. 선택 사항: 이전에 허용 **SELinux** 모드로 전환한 후 모든 항목이 예상대로 작동하는지 확인한 후 강제 **SELinux** 모드로 다시 전환합니다.

```
# setenforce 1
```

검증 단계

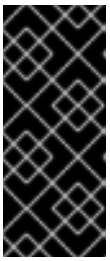
1. 사용자가 올바른 **SELinux** 사용자에게 매핑되고 올바른 명확한 수준이 할당되어 있는지 확인합니다.

```
# semanage login -l
Login Name      SELinux User         MLS/MCS Range        Service
__default__     user_u               s0-s0                *
```

```
example_user    user_u          s1              *
...
```

2. **MLS 내에서 사용자로 로그인합니다.**

3. **사용자의 보안 수준이 올바르게 작동하는지 확인합니다.**



중요

확인에 사용하는 파일에는 구성이 올바르지 않은 경우 중요한 정보가 포함되어 있지 않아야 하며 사용자가 실제로 권한 부여 없이 파일에 액세스할 수 있습니다.

a. **사용자가 더 높은 수준의 민감도를 가진 파일을 읽을 수 없는지 확인합니다.**

b. **사용자가 동일한 민감도로 파일에 쓸 수 있는지 확인합니다.**

c. **사용자가 낮은 수준의 민감도를 사용하여 파일을 읽을 수 있는지 확인합니다.**

추가 리소스

- **6.3절. "SELinux 정책을 MLS로 전환" .**
- **3.4절. "SELinux 제한 사용자로 새 사용자 추가" .**
- ***2장. SELinux 상태 및 모드 변경* .**
- ***5장. SELinux 관련 문제 해결* .**
- **CLI 지식베이스에서 기본 SELinux 문제 해결 문서.**

6.5. MLS에서 정의된 보안 범위 내에서 사용자의 명확한 수준 변경

Multi-Level Security (MLS)의 사용자는 관리자가 할당한 범위 내에서 현재 명확한 수준을 변경할 수 있습니다. 범위의 상한을 초과하거나 범위의 낮은 한도 미만의 수준을 줄일 수 없습니다. 이렇게 하면 예를 들어 민감도 수준을 최고 수준의 명확한 수준으로 늘리지 않고도 낮은 수준의 민감도 파일을 수정할 수 있습니다.

예를 들어, 범위 **s1-s3** 에 할당된 사용자는 다음과 같습니다.

- 레벨 **s1,s2, s3** 으로 전환할 수 있습니다.
- **s1-s2** 및 **s2-s3** 범위로 전환할 수 있습니다.
- **s0-s3** 또는 **s1-s4** 범위로 전환할 수 없습니다.

참고

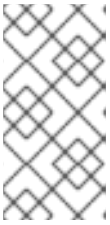
다른 수준으로 전환하면 다른 명확한 설명이 포함된 새 쉘이 열립니다. 즉, 원래의 명확한 수준으로 돌아가는 것과 동일한 방식으로 돌아갈 수 없습니다. 그러나 **exit** 를 입력하여 항상 이전 쉘로 돌아갈 수 있습니다.

사전 요구 사항

- **SELinux** 정책은 **mls** 로 설정됩니다.
- **SELinux** 모드는 **enforcing** 으로 설정되어 있습니다.
- 다양한 **MLS clearance** 수준에 할당된 사용자로 로그인할 수 있습니다.

절차

1. 보안 터미널에서 사용자로 로그인합니다.



참고

보안 터미널은 **/etc/selinux/mls/contexts/securetty_types** 파일에 정의되어 있습니다. 기본적으로 콘솔은 보안 터미널이지만 **SSH**는 그렇지 않습니다.

2. 현재 사용자의 보안 컨텍스트를 확인합니다.

```
$ id -Z
user_u:user_r:user_t:s0-s2
```

이 예에서 사용자는 **user_u SELinux** 사용자, **user_r** 역할, **user_t** 유형 및 **MLS** 보안 범위 **s0-s2** 에 할당됩니다.

3. 현재 사용자의 보안 컨텍스트를 확인합니다.

```
$ id -Z
user_u:user_r:user_t:s1-s2
```

4. 사용자 명확한 범위 내에서 다른 보안 명확한 범위로 전환합니다.

```
$ newrole -l s1
```

최대 범위가 더 낮거나 할당된 범위와 동일한 모든 범위로 전환할 수 있습니다. 단일 수준 범위를 입력하면 할당된 범위의 하한 제한이 변경됩니다. 예를 들어 **s0-s2** 범위가 있는 사용자로 **newrole -l s1** 을 입력하는 것은 **newrole -l s1-s2** 를 입력하는 것과 동일합니다.

검증

1. 현재 사용자의 보안 컨텍스트를 표시합니다.

```
$ id -Z
user_u:user_r:user_t:s1-s2
```

2. 현재 쉘을 종료하여 원래 범위로 이전 쉘로 돌아갑니다.

```
$ exit
```

추가 리소스

- using-multi-level-security-mls_using-selinux.xml
- newrole(1) 매뉴얼 페이지
- securetty_types(5) 도움말 페이지

6.6. MLS에서 파일 민감성 수준 증가

기본적으로 Multi-Level Security (MLS) 사용자는 파일 민감도 수준을 높일 수 없습니다. 그러나 보안 관리자(secadm_r)는 사용자가 시스템의 SELinux 정책에 로컬 모듈 mlsfilewrite 를 추가하여 파일의 민감성을 높일 수 있도록 이 기본 동작을 변경할 수 있습니다. 그런 다음 policy 모듈에 정의된 SELinux 유형에 할당된 사용자는 파일을 수정하여 파일 분류 수준을 늘릴 수 있습니다. 사용자가 파일을 수정할 때마다 파일의 민감도 수준이 사용자의 현재 보안 범위 값으로 늘어납니다.

참고

보안 관리자는 secadm_r 역할에 할당된 사용자로 로그인하면 chcon -l *s0 /path/to/file* 명령을 사용하여 파일의 보안 수준을 변경할 수 있습니다. 자세한 내용은 참조하십시오. 6.7절. “MLS에서 파일 민감성 변경”

사전 요구 사항

- SELinux 정책은 mls 로 설정됩니다.
- SELinux 모드는 강제 모드로 설정되어 있습니다.
- policycoreutils-python-utils 패키지가 설치됩니다.
- mlsfilewrite 로컬 모듈이 SELinux MLS 정책에 설치됩니다.
- MLS에서 다음과 같은 사용자로 로그인했습니다.
  - 정의된 보안 범위에 할당됩니다. 이 예에서는 보안 범위 s0-s2 가 있는 사용자를 보여줍

니다.

- **mlsfilewrite** 모듈에 정의된 동일한 **SELinux** 유형에 할당됩니다. 이 예제에서는 **(attributeset mlsfilewrite(user_t))** 모듈이 필요합니다.

절차

1. 선택 사항: 현재 사용자의 보안 컨텍스트를 표시합니다.

```
$ id -Z
user_u:user_r:user_t:s0-s2
```

2. 사용자의 **MLS** 지우기 범위의 하위 수준을 파일에 할당할 수준으로 변경합니다.

```
$ newrole -l s1-s2
```

3. 선택 사항: 현재 사용자의 보안 컨텍스트를 표시합니다.

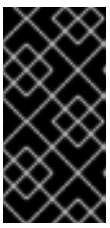
```
$ id -Z
user_u:user_r:user_t:s1-s2
```

4. 선택 사항: 파일의 보안 컨텍스트를 표시합니다.

```
$ ls -Z /path/to/file
user_u:object_r:user_home_t:s0 /path/to/file
```

5. 파일을 수정하여 파일의 민감도 수준을 사용자의 명확한 수준으로 변경합니다.

```
$ touch /path/to/file
```



중요

restorecon 명령이 시스템에서 사용되는 경우 분류 수준은 기본값으로 되돌립니다.

6. 선택 사항: 쉘을 종료하여 사용자의 이전 보안 범위로 돌아갑니다.

```
$ exit
```

검증

- 파일의 보안 컨텍스트를 표시합니다.

```
$ ls -Z /path/to/file
user_u:object_r:user_home_t:s1 /path/to/file
```

추가 리소스

- 6.10절. “MLS 사용자가 더 낮은 수준에서 파일을 편집 가능”.

6.7. MLS에서 파일 민감성 변경

MLS SELinux 정책에서 사용자는 자신의 민감도 수준에서만 파일을 수정할 수 있습니다. 이는 매우 민감한 정보가 더 낮은 수준에서 사용자에게 노출되지 않도록 하고, 낮은 수준의 사용자도 방지하기 위한 것입니다. 그러나 관리자는 파일의 분류를 수동으로 늘릴 수 있습니다(예: 상위 수준에서 처리할 파일 분류).

사전 요구 사항

- SELinux 정책은 mls 로 설정됩니다.
- SELinux 모드는 enforcing으로 설정됩니다.
- 보안 관리 권한이 있습니다. 즉, 다음 중 하나에 할당됨을 의미합니다.
  - head adm_r 역할입니다.
  - tekton _secadm 모듈이 활성화된 경우, to thetekton _r 역할로 설정합니다. RuntimeClass _secadm 모듈은 기본적으로 활성화되어 있습니다.
- policycoreutils-python-utils 패키지가 설치됩니다.
- 모든 명확한 수준에 할당된 사용자입니다. 자세한 내용은 MLS의 사용자 삭제 수준을 참조하

십시오.

이 예에서 **user** *1에는 명확한* 수준 **s1** 이 있습니다.

- 분류 수준이 할당되고 액세스 권한이 있는 파일입니다.

  이 예에서 ***/path/to/file*** 에는 분류 수준 **s1** 이 있습니다.

절차

1. 파일의 분류 수준을 확인합니다.

```
# ls -lZ /path/to/file
-rw-r-----. 1 User1 User1 user_u:object_r:user_home_t:s1 0 12. Feb 10:43 /path/to/file
```

2. 파일의 기본 분류 수준을 변경합니다.

```
# semanage fcontext -a -r s2 /path/to/file
```

3. 파일의 **SELinux** 컨텍스트 레이블을 강제로 다시 지정합니다.

```
# restorecon -F -v /path/to/file
Relabeled /path/to/file from user_u:object_r:user_home_t:s1 to
user_u:object_r:user_home_t:s2
```

검증

1. 파일의 분류 수준을 확인합니다.

```
# ls -lZ /path/to/file
-rw-r-----. 1 User1 User1 user_u:object_r:user_home_t:s2 0 12. Feb 10:53 /path/to/file
```

2. 선택 사항**: lower-clearance** 사용자가 파일을 읽을 수 없는지 확인합니다.

```
$ cat /path/to/file
cat: file: Permission denied
```

추가 리소스

- 6.4절. “MLS에서 사용자 명확한 설정” .

6.8. MLS의 보안 관리와 시스템 관리 분리

기본적으로 **HEALTH _r** 역할에는 **secadm_r** 역할의 권한이 있습니다. 즉, **sensitive _r** 역할의 사용자는 보안 정책을 관리할 수 있습니다. 보안 권한을 보다 잘 제어해야 하는 경우 **Linux** 사용자를 **secadm_r** 역할에 할당하고 **SELinux** 정책에서 **etcdctl _secadm** 모듈을 비활성화하여 보안 관리에서 시스템 관리를 분리할 수 있습니다.

사전 요구 사항

- **SELinux** 정책은 **mls** 로 설정됩니다.
- **SELinux** 모드는 강제 모드로 설정되어 있습니다.
- **policycoreutils-python-utils** 패키지가 설치됩니다.
- **secadm_r** 역할에 할당할 **Linux** 사용자:
  - 사용자가 **staff_u SELinux** 사용자에게 할당됩니다.
  - 이 사용자의 암호가 정의되었습니다.

주의

secadm 역할에 할당할 사용자로 로그인할 수 있는지 확인합니다. 그러지 않으면 시스템의 **SELinux** 정책을 향후 수정하는 것을 방지할 수 있습니다.

절차

1. **/etc/sudoers.d 디렉토리에 사용자의 새 sudoers 파일을 만듭니다.**

```
# visudo -f /etc/sudoers.d/<sec_adm_user>
```

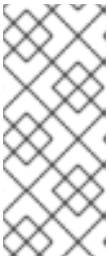
sudoers 파일을 정리하여 유지하려면 < *sec_adm_user* >를 secadm 역할에 할당할 Linux 사용자로 바꿉니다.

2. **/etc/sudoers.d/ <*sec_adm_user*> 파일에 다음 내용을 추가합니다.**

```
<sec_adm_user> ALL=(ALL) TYPE=secadm_t ROLE=secadm_r ALL
```

이 행은 모든 호스트에서 < *secadmuser* >를 인증하여 모든 명령을 수행하고 기본적으로 사용자를 secadm SELinux 유형 및 역할에 매핑합니다.

3. **< *sec_adm_user*> *사용자로* 로그인합니다.**



참고

SELinux 컨텍스트(SELinux 사용자, 역할 및 유형으로 구성됨)가 변경되도록 하려면 ssh, 콘솔 또는 xdm 을 사용하여 로그인합니다. su 및 sudo 와 같은 다른 방법은 전체 SELinux 컨텍스트를 변경할 수 없습니다.

4. **사용자의 보안 컨텍스트를 확인합니다.**

```
$ id
uid=1000(<sec_adm_user>) gid=1000(<sec_adm_user>) groups=1000(<sec_adm_user>)
context=staff_u:staff_r:staff_t:s0-s15:c0.c1023
```

5. **root 사용자에 대해 대화형 쉘을 실행합니다.**

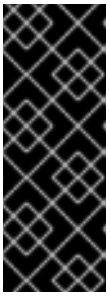
```
$ sudo -i
[sudo] password for <sec_adm_user>:
```

6. **현재 사용자의 보안 컨텍스트를 확인합니다.**

```
# id
uid=0(root) gid=0(root) groups=0(root) context=staff_u:secadm_r:secadm_t:s0-s15:c0.c1023
```

7. 정책에서 **cryptsetup_secadm** 모듈을 비활성화합니다.

```
# semodule -d sysadm_secadm
```



중요

semodule -r 명령을 사용하여 시스템 정책 모듈을 제거하는 대신 **semodule -d** 명령을 사용합니다. **semodule -r** 명령은 시스템 스토리지에서 모듈을 삭제합니다. 즉 **selinux-policy-mls** 패키지를 다시 설치하지 않고도 다시 로드할 수 없습니다.

검증

1. **secadm** 역할에 할당된 사용자로 **root** 사용자의 대화형 셸에서 보안 정책 데이터에 액세스할 수 있는지 확인합니다.

```
# seinfo -xt secadm_t

Types: 1
  type secadm_t, can_relabelto_shadow_passwords, (...) userdomain;
```

2. **root** 셸에서 로그아웃합니다.

```
# logout
```

3. **< *sec_adm_user* >** 사용자로부터 로그아웃합니다.

```
$ logout
Connection to localhost closed.
```

4. 현재 보안 컨텍스트를 표시합니다.

```
# id
uid=0(root) gid=0(root) groups=0(root) context=root:sysadm_r:sysadm_t:s0-s15:c0.c1023
```

5. **authorization _secadm** 모듈을 활성화합니다. 이 명령은 실패합니다.

```
# semodule -e sysadm_secadm
SELinux:  Could not load policy file /etc/selinux/mls/policy/policy.31:  Permission denied
/sbin/load_policy:  Can't load policy:  Permission denied
libsemanage.semanage_reload_policy: load_policy returned error code 2. (No such file or
directory).
SELinux:  Could not load policy file /etc/selinux/mls/policy/policy.31:  Permission denied
/sbin/load_policy:  Can't load policy:  Permission denied
libsemanage.semanage_reload_policy: load_policy returned error code 2. (No such file or
directory).
semodule:  Failed!
```

6.

GDM _t SELinux 유형에 대한 세부 정보를 표시하려고 합니다. 이 명령은 실패합니다.

```
# seinfo -xt sysadm_t
[Errno 13] Permission denied: '/sys/fs/selinux/policy'
```

6.9. MLS에서 보안 터미널 정의

SELinux 정책은 사용자가 연결된 터미널 유형을 확인하고, 예를 들어 newrole 과 같이 보안 터미널에서만 특정 SELinux 애플리케이션을 실행할 수 있습니다. 비보안 터미널에서 이 작업을 시도하면 오류가 발생합니다. 오류: 보안 이외의 터미널에서 수준을 변경할 수 없습니다..

/etc/selinux/mls/contexts/securetty_types 파일은 MLS(Multi-Level Security) 정책에 대한 보안 터미널을 정의합니다.

파일의 기본 콘텐츠:

```
console_device_t
sysadm_tty_device_t
user_tty_device_t
staff_tty_device_t
auditadm_tty_device_t
secureadm_tty_device_t
```

주의

보안 터미널 목록에 터미널 유형을 추가하면 시스템이 보안 위험에 노출될 수 있습니다.

사전 요구 사항

- SELinux 정책은 mls 로 설정됩니다.
- 이미 보안된 터미널에서 연결되어 있거나 SELinux가 허용 모드에 있습니다.
- 보안 관리 권한이 있습니다. 즉, 다음 중 하나에 할당됨을 의미합니다.
  - head adm_r 역할입니다.
  - tekton _secadm 모듈이 활성화된 경우, to thetekton _r 역할로 설정합니다. RuntimeClass _secadm 모듈은 기본적으로 활성화되어 있습니다.
- policycoreutils-python-utils 패키지가 설치됩니다.

절차

1. 현재 터미널 유형을 확인합니다.

```
# ls -Z `tty`
root:object_r:user_devpts_t:s0 /dev/pts/0
```

이 예제 출력에서 user_devpts_t 는 현재 터미널 유형입니다.

2. /etc/selinux/mls/contexts/securetty_types 파일의 새 행에 관련 SELinux 유형을 추가합니다.

3. 선택 사항: SELinux를 강제 모드로 전환합니다.

```
# setenforce 1
```

검증

- 이전에 안전하지 않은 터미널에서 /etc/selinux/mls/contexts/securetty_types 파일에 추가했습니다.

추가 리소스

- **securetty_types(5)** 도움말 페이지

6.10. MLS 사용자가 더 낮은 수준에서 파일을 편집 가능

기본적으로 **MLS** 사용자는 클리어스 범위의 낮은 값보다 민감도 수준이 있는 파일에 쓸 수 없습니다. 사용자가 더 낮은 수준에서 파일을 편집하도록 허용하는 경우 로컬 **SELinux** 모듈을 생성하여 이를 수행할 수 있습니다. 그러나 파일에 쓰는 경우 민감도 수준이 사용자의 현재 범위보다 낮은 값으로 증가합니다.

사전 요구 사항

- **SELinux** 정책은 **mls** 로 설정됩니다.
- **SELinux** 모드는 강제 모드로 설정되어 있습니다.
- **policycoreutils-python-utils** 패키지가 설치됩니다.
- 확인을 위한 **setools-console** 및 감사 패키지.

절차

1. 선택 사항: 문제 해결을 위해 허용 모드로 전환합니다.

```
# setenforce 0
```

2. 텍스트 편집기로 새 **.cil** 파일(예: **~/local_mlsfilewrite.cil**)을 열고 다음 사용자 지정 규칙을 삽입합니다.

```
(typeattributeset mlsfilewrite (_staff_t_))
```

staff_t 를 다른 **SELinux** 유형으로 교체할 수 있습니다. **SELinux** 유형을 지정하면 하위 수준 파일을 편집할 수 있는 **SELinux** 역할을 제어할 수 있습니다.

로컬 모듈을 더 잘 정리하려면 로컬 **SELinux** 정책 모듈의 이름에 **local_** 접두사를 사용합니다.

3. **policy** 모듈을 설치합니다.

```
# semodule -i ~/local_mlsfilewrite.cil
```

참고

로컬 정책 모듈을 제거하려면 **semodule -r ~/*local_mlsfilewrite*** 를 사용합니다. **.cil** 접미사 없이 모듈 이름을 참조해야 합니다.

4. 선택 사항: 이전에 허용 모드로 다시 전환한 경우 강제 모드로 돌아갑니다.

```
# setenforce 1
```

검증

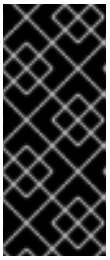
1. 설치된 **SELinux** 모듈 목록에서 로컬 모듈을 찾습니다.

```
# semodule -lfull | grep "local_mls"
400 local_mlsfilewrite  cil
```

로컬 모듈에는 우선 순위가 **400** 이므로 **semodule -lfull | grep -v ^100** 명령을 사용하여 나열할 수도 있습니다.

2. 사용자 지정 규칙에 정의된 유형에 할당된 사용자로 로그인합니다(예: ***staff_t***).

3. 민감도 수준이 낮은 파일에 쓰기를 시도합니다. 이렇게 하면 파일의 분류 수준이 사용자의 명확한 수준으로 증가합니다.



중요

확인에 사용하는 파일에는 구성이 올바르지 않은 경우 중요한 정보가 포함되어 있지 않아야 하며 사용자가 실제로 권한 부여 없이 파일에 액세스할 수 있습니다.

7장. 데이터 기밀성을 위해 MCS(MULTI-CATEGORY SECURITY) 사용

MCS를 사용하여 데이터를 분류한 다음 특정 프로세스 및 사용자에게 특정 카테고리에 대한 액세스 권한을 부여하여 시스템의 데이터 기밀성을 향상시킬 수 있습니다.

7.1. MCS(MULTI-CATEGORY SECURITY)

MCS(Multi-Category Security)는 프로세스 및 파일에 할당된 카테고리를 사용하는 액세스 제어 메커니즘입니다. 그런 다음 동일한 카테고리에 할당된 프로세스에서만 파일에 액세스할 수 있습니다. MCS의 목적은 시스템에서 데이터 기밀성을 유지하는 것입니다.

MCS 카테고리는 c 1023 사이의 값으로 정의되지만, "Personnel", "ProjectX" 또는 "Projectnel"과 같은 카테고리의 각 카테고리 또는 조합에 대한 텍스트 레이블을 정의할 수도 있습니다. 그런 다음 MCS Translation 서비스(mcstrans)는 범주 값을 시스템 입력 및 출력에서 적절한 레이블로 대체하여 사용자가 카테고리 값 대신 이러한 레이블을 사용할 수 있도록 합니다.

사용자가 카테고리에 할당되면 해당 파일에 할당된 범주 중 하나로 레이블을 지정할 수 있습니다.

MCS는 파일에 액세스하려면 파일에 할당된 모든 카테고리에 대해 사용자를 할당해야 합니다. MCS 확인은 일반 Linux DAC(disretionary Access Control) 및 SELinux 유형 강제(TE) 규칙 이후에 적용되므로 기존 보안 구성만 추가로 제한할 수 있습니다.

다단계 보안 내 MCS

MCS를 자체적으로 비hierarchical 시스템으로 사용하거나 MLS(Multi-Level Security)와 함께 계층 구조의 계층으로 사용할 수 있습니다.

MLS 내 MCS의 예로는 파일이 다음과 같이 분류되는 비밀 연구 조직이 될 수 있습니다.

표 7.1. 보안 수준 및 카테고리 조합의 예

보안 수준	카테고리			
	지정되지 않음	프로젝트 X	프로젝트 Y	프로젝트 Z
분류되지 않음	**s0**	**s0:c0**	**s0:c1**	**s0:c2**
confidential	**s1**	**s1:c0**	**s1:c1**	**s1:c2**
Secret	**s2**	**s2:c0**	**s2:c1**	**s2:c2**

상단 시크릿	**s3**	**s3:c0**	**s3:c1**	**s3:c2**

참고

s0:c0.1023 범위가 있는 사용자는 **TPM** 또는 유형 적용 정책 규칙과 같은 다른 보안 메커니즘에서 액세스를 금지하지 않는 한 수준 **s0** 의 모든 카테고리에 할당된 모든 파일에 액세스할 수 있습니다.

결과적으로 파일 또는 프로세스의 보안 컨텍스트는 다음과 같은 조합입니다.

- **SELinux** 사용자
- **SELinux** 역할
- **SELinux** 유형
- **MLS** 민감도 수준
- **MCS** 카테고리

예를 들어 **MLS/MCS** 환경에서 민감도 수준 **1** 및 카테고리 **2**에 액세스할 수 있는 권한이 없는 사용자는 다음과 같은 **SELinux** 컨텍스트를 가질 수 있습니다.

```
user_u:user_r:user_t:s1:c2
```

추가 리소스

- Multi-Level Security (MLS) 사용

7.2. 데이터 기밀성을 위해 다중 범주 보안 구성

기본적으로 **MCS**는 대상 및 **mls SELinux** 정책에서 활성화되지만 사용자에 대해 구성되지 않습니다. 대상 정책에서 **MCS**는 다음에 대해서만 구성됩니다.

- OpenShift
- virt
- sandbox
- 네트워크 레이블링
- 컨테이너 (container-selinux)

유형 적용 외에도 MCS 규칙으로 user_t SELinux 유형을 제한하는 규칙으로 로컬 SELinux 모듈을 생성하여 사용자를 분류하도록 MCS를 구성할 수 있습니다.

주의

특정 파일의 카테고리를 변경하면 일부 서비스가 작동하지 않을 수 있습니다. 전문가가 아닌 경우 Red Hat 영업 담당자에게 연락하여 컨설팅 서비스를 요청하십시오.

사전 요구 사항

- SELinux 모드는 강제 모드로 설정되어 있습니다.
- SELinux 정책은 대상 또는 mls 로 설정됩니다.
- policycoreutils-python-utils 및 setools-console 패키지가 설치됩니다.

절차

1. 이라는 새 파일을 만듭니다(예: **local_mcs_user.cil**):

```
# vim local_mcs_user.cil
```

2. 다음 규칙을 삽입합니다.

```
(typeattributeset mcs_constrained_type (user_t))
```

3. **policy** 모듈을 설치합니다.

```
# semodule -i local_mcs_user.cil
```

검증

- 각 사용자 도메인의 모든 구성 요소에 대한 추가 세부 정보를 표시합니다.

```
# seinfo -xt user_t

Types: 1
type user_t, application_domain_type, nsswitch_domain, corenet_unlabeled_type, domain,
kernel_system_state_reader, mcs_constrained_type, netlabel_peer_type, privfd,
process_user_target, scsi_generic_read, scsi_generic_write, syslog_client_type,
pcmcia_typeattr_1, user_usertype, login_userdomain, userdomain, unpriv_userdomain,
userdom_home_reader_type, userdom_filetrans_type, xdmhomewriter, x_userdomain,
x_domain, dridomain, xdrawable_type, xcolormap_type;
```

추가 리소스

- 로컬 **SELinux** 정책 모듈 생성
- 컨테이너 컨텍스트에서 **MCS**에 대한 자세한 내용은 **SELinux**가 다단계 보안을 사용하여 컨테이너를 분리하는 방법 및 **Linux** 컨테이너용 **Multi-Category Security**를 사용해야 하는 이유를 참조하십시오.

7.3. MCS에서 카테고리 레이블 정의

setrans.conf 파일을 편집하여 **MCS** 범주 또는 **MLS** 수준의 **MCS** 카테고리 조합을 관리하고 유지 관리할 수 있습니다. 이 파일에서 **SELinux**는 내부 민감도와 범주 수준과 사람이 읽을 수 있는 레이블 간의 매핑을 유지 관리합니다.

참고

카테고리 레이블만 있으면 사용자가 카테고리를 더 쉽게 사용할 수 있습니다. **MCS**는 레이블을 정의하든 관계 없이 동일하게 작동합니다.

사전 요구 사항

- **SELinux** 모드는 강제 모드로 설정되어 있습니다.
- **SELinux** 정책은 대상 또는 **mls** 로 설정됩니다.
- **policycoreutils-python-utils** 및 **mcstrans** 패키지가 설치됩니다.

절차

1. 텍스트 편집기에서 **/etc/selinux/*<selinuxpolicy>*/setrans.conf** 파일을 편집하여 기존 카테고리를 수정하거나 새 범주를 만듭니다. 사용하는 **SELinux** 정책에 따라 ***<selinuxpolicy>*** 를 대상 또는 **mls** 로 바꿉니다. 예를 들면 다음과 같습니다.

```
# vi /etc/selinux/targeted/setrans.conf
```

2. 정책에 대한 **setrans.conf** 파일에서 **s_<security level>_:c_<category number>_=*<category.name>*** *구문을 사용하여 시나리오에 필요한 카테고리 조합을 정의합니다.* 예를 들면 다음과 같습니다.

```
s0:c0=Marketing
s0:c1=Finance
s0:c2=Payroll
s0:c3=Personnel
```

- **c0**에서 **c 1023** 까지 범주 번호를 사용할 수 있습니다.
- 타겟 정책에서 **s0** 보안 수준을 사용합니다.
- **mls** 정책에서는 민감도 수준 및 범주의 각 조합에 레이블을 지정할 수 있습니다.

3. 선택 사항: **setrans.conf** 파일에서 **MLS** 민감도에 레이블을 지정할 수도 있습니다.

4. 파일을 저장하고 종료합니다.

5. 변경 사항을 적용하려면 **MCS** 변환 서비스를 다시 시작하십시오.

```
# systemctl restart mcstrans
```

검증

- 현재 카테고리를 표시합니다.

```
# chcat -L
```

위의 예제에서는 다음 출력을 생성합니다.

```
s0:c0                Marketing
s0:c1                Finance
s0:c2                Payroll
s0:c3                Personnel
s0
s0-s0:c0.c1023              SystemLow-SystemHigh
s0:c0.c1023              SystemHigh
```

추가 리소스

- **setrans.conf(5)** 도움말 페이지.

7.4. MCS의 사용자에게 카테고리 할당

Linux 사용자에게 범주를 할당하여 사용자 권한을 정의할 수 있습니다. 범주가 할당된 사용자는 사용자 카테고리의 하위 집합이 있는 파일에 액세스하고 수정할 수 있습니다. 사용자는 할당된 범주에 고유 파일을 할당할 수도 있습니다.

Linux 사용자는 관련 **SELinux** 사용자에 대해 정의된 보안 범위를 벗어나는 범주에 할당할 수 없습니다.

참고

범주 액세스는 로그인 중에 할당됩니다. 따라서 사용자는 다시 로그인할 때까지 새로 할당된 카테고리에 액세스할 수 없습니다. 마찬가지로, 범주에 대한 사용자 액세스를 취소하는 경우 사용자가 다시 로그인한 후에만 적용됩니다.

사전 요구 사항

- **SELinux** 모드는 강제 모드로 설정되어 있습니다.
- **SELinux** 정책은 대상 또는 **mls** 로 설정됩니다.
- **policycoreutils-python-utils** 패키지가 설치됩니다.
- **Linux** 사용자는 **SELinux** 제한 사용자에게 할당됩니다.
  - 권한이 없는 사용자는 **user_u** 에 할당됩니다.
  - 권한이 있는 사용자는 **staff_u** 에 할당됩니다.

절차

1. **SELinux** 사용자의 보안 범위를 정의합니다.

```
# semanage user -m -rs0:c0,c1-s0:c0.c9 <user_u>
```

setrans.conf 파일에 정의된 대로 범주 번호 **c0 ~ c1023** 또는 **category** 레이블을 사용합니다. 자세한 내용은 **MCS의 카테고리 라벨 정의를** 참조하십시오.

2. **Linux** 사용자에게 **MCS** 범주를 할당합니다. 관련 **SELinux** 사용자에게 정의된 범위 내의 범위만 지정할 수 있습니다.

```
# semanage login -m -rs0:c1 <Linux.user1>
```

참고

chcat 명령을 사용하여 **Linux** 사용자의 카테고리를 추가하거나 제거할 수 있습니다. 다음 예제에서는 ***<category1>*** 를 추가하고 ***<Linux.user 1>*** *및* ***<Linux.user2>*** *에서* ***<category 2>****를 제거합니다.*

```
# chcat -l -- +<category1>,-<category2> <Linux.user1>,<Linux.user2>
```

-<category> 구문을 사용하기 전에 명령줄에서 **--** 을 지정해야 합니다. 그렇지 않으면 **chcat** 명령이 카테고리 제거를 명령 옵션으로 잘못 해석합니다.

검증

- **Linux** 사용자에게 할당된 카테고리를 나열합니다.

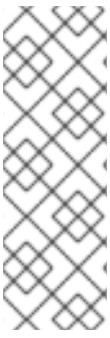
```
# chcat -L -l <Linux.user1>,<Linux.user2>
<Linux.user1>: <category1>,<category2>
<Linux.user2>: <category1>,<category2>
```

추가 리소스

- **chcat(8)** 도움말 페이지.

7.5. MCS의 파일에 카테고리 할당

사용자에게 카테고리를 할당하려면 관리자 권한이 필요합니다. 그런 다음 사용자는 파일에 카테고리를 할당할 수 있습니다. 파일의 카테고리를 수정하려면 사용자에게 해당 파일에 대한 액세스 권한이 있어야 합니다. 사용자는 할당된 범주에만 파일을 할당할 수 있습니다.



참고

시스템은 범주 액세스 규칙을 기존의 파일 액세스 권한과 결합합니다. 예를 들어, 범주가 **bigfoot** 인 사용자가 **DAC**(임의 액세스 제어)를 사용하여 다른 사용자가 파일에 대한 액세스를 차단하는 경우 다른 **bigfoot** 사용자는 해당 파일에 액세스할 수 없습니다. 사용 가능한 모든 카테고리에 할당된 사용자는 여전히 전체 파일 시스템에 액세스하지 못할 수 있습니다.

사전 요구 사항

- SELinux 모드는 강제 모드로 설정되어 있습니다.
- SELinux 정책은 대상 또는 mls 로 설정됩니다.
- policycoreutils-python-utils 패키지가 설치됩니다.
- Linux 사용자는 다음과 같은 액세스 권한 및 권한입니다.
  - SELinux 사용자에게 할당.
  - 파일을 할당하려는 범주에 할당됨. 자세한 내용은 MCS의 사용자에게 카테고리 할당을 참조하십시오.
- 범주에 추가할 파일에 대한 액세스 및 권한.
- 검증 목적으로: 이 범주에 할당되지 않은 Linux 사용자에 대한 액세스 및 권한

절차

- 파일에 카테고리 추가:

```
$ chcat -- +<category1>,+<category2> <path/to/file1>
```

setrans.conf 파일에 정의된 대로 범주 번호 c0 ~ c1023 또는 category 레이블을 사용합니다. 자세한 내용은 MCS의 카테고리 라벨 정의를 참조하십시오.

동일한 구문을 사용하여 파일에서 카테고리를 제거할 수 있습니다.

```
$ chcat -- -<category1>,-<category2> <path/to/file1>
```

참고

범주를 제거하는 경우 *-<category>* 구문을 사용하기 전에 명령줄에서 -- 을 지정해야 합니다. 그렇지 않으면 chcat 명령이 카테고리 제거를 명령 옵션으로 잘못 해석할 수 있습니다.

검증

1. **파일의 보안 컨텍스트를 표시하여 올바른 카테고리가 있는지 확인합니다.**

```
$ ls -lZ <path/to/file>
-rw-r--r--  <LinuxUser1> <Group1> root:object_r:user_home_t:_<sensitivity>_:_<category>_
<path/to/file>
```

파일의 특정 보안 컨텍스트는 다를 수 있습니다.

2. **선택 사항: 파일과 동일한 범주에 할당되지 않은 Linux 사용자로 로그인하면 파일 액세스를 시도합니다.**

```
$ cat <path/to/file>
cat: <path/to/file>: Permission Denied
```

추가 리소스

- **semanage(8) 도움말 페이지.**
- **chcat(8) 도움말 페이지.**

8장. 사용자 지정 SELINUX 정책 작성

이 섹션에서는 SELinux로 제한된 애플리케이션을 실행할 수 있는 사용자 지정 정책을 작성하고 사용하는 방법을 안내합니다.

8.1. 사용자 정의 SELINUX 정책 및 관련 툴

SELinux 보안 정책은 SELinux 규칙 컬렉션입니다. 정책은 SELinux의 핵심 구성 요소이며 SELinux 사용자 공간 도구를 통해 커널에 로드됩니다. 커널은 SELinux 정책을 사용하여 시스템의 액세스 요청을 평가합니다. 기본적으로 SELinux는 로드된 정책에 지정된 규칙에 해당하는 요청을 제외한 모든 요청을 거부합니다.

각 SELinux 정책 규칙은 프로세스와 시스템 리소스 간의 상호 작용을 설명합니다.

```
ALLOW apache_process apache_log:FILE READ;
```

이 예제 규칙은 다음과 같습니다. *Apache 프로세스는 로깅 파일을 읽을 수 있습니다.* 이 규칙에서 apache_process 및 apache_log 는 레이블 입니다. SELinux 보안 정책은 프로세스에 레이블을 할당하고 시스템 리소스에 대한 관계를 정의합니다. 이렇게 하면 정책이 운영 체제 엔터티를 SELinux 계층에 매핑합니다.

SELinux 레이블은 ext2 와 같은 파일 시스템의 확장된 속성으로 저장됩니다. getfattr 유틸리티 또는 ls -Z 명령을 사용하여 나열할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ ls -Z /etc/passwd
system_u:object_r:passwd_file_t:s0 /etc/passwd
```

여기서 system_u 는 SELinux 사용자이고 object_r 은 SELinux 역할의 예이며 passwd_file_t 는 SELinux 도메인입니다.

selinux-policy 패키지에서 제공하는 기본 SELinux 정책에는 Red Hat Enterprise Linux 9의 일부인 애플리케이션 및 데몬에 대한 규칙이 포함되어 있으며 리포지토리의 패키지에서 제공합니다. 이 배포 정책의 규칙에 설명되지 않은 애플리케이션은 SELinux에 의해 제한되지 않습니다. 이를 변경하려면 추가 정의 및 규칙이 포함된 policy 모듈을 사용하여 정책을 수정해야 합니다.

Red Hat Enterprise Linux 9에서는 설치된 SELinux 정책을 쿼리하고 sepolicy 도구를 사용하여 새 정책 모듈을 생성할 수 있습니다. sepolicy 가 정책 모듈과 함께 생성하는 스크립트에는 restorecon 유틸리

티를 사용하는 명령이 항상 포함됩니다. 이 유틸리티는 파일 시스템의 선택한 부분에서 레이블 문제를 해결하는 기본 도구입니다.

추가 리소스

- **sepolicy(8)** 및 **getfattr(1)** 도움말 페이지

8.2. 사용자 지정 애플리케이션에 대한 SELINUX 정책 생성 및 적용

이 예제 절차에서는 **SELinux**에서 간단한 데몬을 제한하는 단계를 제공합니다. 데몬을 사용자 정의 애플리케이션으로 교체하고 해당 애플리케이션 및 보안 정책의 요구 사항에 따라 예제 규칙을 수정합니다.

사전 요구 사항

- **policycoreutils-devel** 패키지 및 해당 종속 항목은 시스템에 설치됩니다.

절차

1. 이 예제 프로시저의 경우 쓰기를 위해 **/var/log/message** 파일을 여는 간단한 데몬을 준비합니다.

   a. 새 파일을 생성하고 선택한 텍스트 편집기에서 엽니다.

   ```
   $ vi mydaemon.c
   ```

   b. 다음 코드를 삽입합니다.

   ```
   #include <unistd.h>
   #include <stdio.h>

   FILE *f;

   int main(void)
   {
   while(1) {
   f = fopen("/var/log/messages","w");
         sleep(5);
         fclose(f);
      }
   }
   ```

c.

파일을 컴파일합니다.

```
$ gcc -o mydaemon mydaemon.c
```

d.

데몬의 systemd 장치 파일을 생성합니다.

```
$ vi mydaemon.service
[Unit]
Description=Simple testing daemon

[Service]
Type=simple
ExecStart=/usr/local/bin/mydaemon

[Install]
WantedBy=multi-user.target
```

e.

데몬을 설치하고 시작합니다.

```
# cp mydaemon /usr/local/bin/
# cp mydaemon.service /usr/lib/systemd/system
# systemctl start mydaemon
# systemctl status mydaemon
● mydaemon.service - Simple testing daemon
   Loaded: loaded (/usr/lib/systemd/system/mydaemon.service; disabled; vendor preset:
disabled)
   Active: active (running) since Sat 2020-05-23 16:56:01 CEST; 19s ago
 Main PID: 4117 (mydaemon)
    Tasks: 1
   Memory: 148.0K
   CGroup: /system.slice/mydaemon.service
           └─4117 /usr/local/bin/mydaemon

May 23 16:56:01 localhost.localdomain systemd[1]: Started Simple testing daemon.
```

f.

새 데몬이 SELinux에 의해 제한되지 않았는지 확인합니다.

```
$ ps -efZ | grep mydaemon
system_u:system_r:unconfined_service_t:s0 root 4117    1  0 16:56 ?        00:00:00
/usr/local/bin/mydaemon
```

2.

데몬에 대한 사용자 정의 정책을 생성합니다.

```
$ sepolicy generate --init /usr/local/bin/mydaemon
```

```
Created the following files:
/home/example.user/mysepol/mydaemon.te # Type Enforcement file
/home/example.user/mysepol/mydaemon.if # Interface file
/home/example.user/mysepol/mydaemon.fc # File Contexts file
/home/example.user/mysepol/mydaemon_selinux.spec # Spec file
/home/example.user/mysepol/mydaemon.sh # Setup Script
```

3. **이전 명령으로 생성한 설정 스크립트를 사용하여 새 정책 모듈로 시스템 정책을 다시 빌드합니다.**

```
# ./mydaemon.sh
Building and Loading Policy
+ make -f /usr/share/selinux/devel/Makefile mydaemon.pp
Compiling targeted mydaemon module
Creating targeted mydaemon.pp policy package
rm tmp/mydaemon.mod.fc tmp/mydaemon.mod
+ /usr/sbin/semodule -i mydaemon.pp
...
```

설정 스크립트는 restorecon 명령을 사용하여 파일 시스템의 해당 부분을 레이블을 다시 지정합니다.

```
restorecon -v /usr/local/bin/mydaemon /usr/lib/systemd/system
```

4. **데몬을 다시 시작하고 이제 SELinux에 의해 제한되는지 확인합니다.**

```
# systemctl restart mydaemon
$ ps -efZ | grep mydaemon
system_u:system_r:mydaemon_t:s0 root        8150       1  0 17:18 ?        00:00:00
/usr/local/bin/mydaemon
```

5. **이제 데몬이 SELinux로 제한되므로 SELinux는 /var/log/message에 액세스하지 못하도록 합니다. 해당 거부 메시지를 표시합니다.**

```
# ausearch -m AVC -ts recent
...
type=AVC msg=audit(1590247112.719:5935): avc:  denied  { open } for  pid=8150
comm="mydaemon" path="/var/log/messages" dev="dm-0" ino=2430831
scontext=system_u:system_r:mydaemon_t:s0 tcontext=unconfined_u:object_r:var_log_t:s0
tclass=file permissive=1
...
```

6. **sealert 도구를 사용하여 추가 정보를 얻을 수 있습니다.**

```
$ sealert -l "*"
SELinux is preventing mydaemon from open access on the file /var/log/messages.

 Plugin catchall (100. confidence) suggests *

If you believe that mydaemon should be allowed open access on the messages file by
default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'mydaemon' --raw | audit2allow -M my-mydaemon
# semodule -X 300 -i my-mydaemon.pp

Additional Information:
Source Context                system_u:system_r:mydaemon_t:s0
Target Context                unconfined_u:object_r:var_log_t:s0
Target Objects                /var/log/messages [ file ]
Source                        mydaemon

...
```

7. **audit2allow 툴을 사용하여 변경 사항을 제안하십시오.**

```
$ ausearch -m AVC -ts recent | audit2allow -R

require {
 type mydaemon_t;
}

#============= mydaemon_t ==============
logging_write_generic_logs(mydaemon_t)
```

8. **audit2allow 에서 제공하는 규칙은 특정 사례에 잘못 될 수 있으므로 출력의 일부만 사용하여 해당 정책 인터페이스를 찾습니다.**

```
$ grep -r "logging_write_generic_logs" /usr/share/selinux/devel/include/ | grep .if
/usr/share/selinux/devel/include/system/logging.if:interface(`logging_write_generic_logs',`
```

9. **인터페이스 정의를 확인합니다.**

```
$ cat /usr/share/selinux/devel/include/system/logging.if
...
interface(`logging_write_generic_logs',`
        gen_require(`
                type var_log_t;
        ')

        files_search_var($1)
```

```
        allow $1 var_log_t:dir list_dir_perms;
        write_files_pattern($1, var_log_t, var_log_t)
')
...
```

10. 이 경우 제안된 인터페이스를 사용할 수 있습니다. 유형 적용 파일에 해당 규칙을 추가합니다.

```
$ echo "logging_write_generic_logs(mydaemon_t)" >> mydaemon.te
```

또는 인터페이스를 사용하는 대신 이 규칙을 추가할 수 있습니다.

```
$ echo "allow mydaemon_t var_log_t:file { open write getattr };" >> mydaemon.te
```

11. 정책을 다시 설치합니다.

```
# ./mydaemon.sh
Building and Loading Policy
+ make -f /usr/share/selinux/devel/Makefile mydaemon.pp
Compiling targeted mydaemon module
Creating targeted mydaemon.pp policy package
rm tmp/mydaemon.mod.fc tmp/mydaemon.mod
+ /usr/sbin/semodule -i mydaemon.pp
...
```

검증

1. 다음과 같이 애플리케이션이 **SELinux**에 의해 제한적으로 실행되는지 확인합니다.

```
$ ps -efZ | grep mydaemon
system_u:system_r:mydaemon_t:s0 root        8150       1  0 17:18 ?        00:00:00
/usr/local/bin/mydaemon
```

2. 사용자 지정 애플리케이션에서 **SELinux** 거부를 유발하지 않는지 확인합니다.

```
# ausearch -m AVC -ts recent
<no matches>
```

추가 리소스

- **sepolgen(8), ausearch(8), audit2allow(1), audit2why(1), sealert(8)** 및 **restorecon(8)** 도움말 페이지

8.3. 로컬 SELINUX 정책 모듈 생성

활성 SELinux 정책에 특정 SELinux 정책 모듈을 추가하면 SELinux 정책의 특정 문제를 해결할 수 있습니다. 이 절차를 사용하여 Red Hat 릴리스 노트에 설명된 특정 알려진 문제를 수정하거나 특정 Red Hat 솔루션을 구현할 수 있습니다.

주의

Red Hat에서 제공하는 규칙만 사용하십시오. Red Hat은 제품 지원 적용 범위를 벗어나므로 사용자 지정 규칙을 사용하여 SELinux 정책 모듈 생성 을 지원하지 않습니다. 전문가가 아닌 경우 Red Hat 영업 담당자에게 연락하여 컨설팅 서비스를 요청하십시오.

사전 요구 사항

- 확인을 위한 setools-console 및 감사 패키지.

절차

1. 텍스트 편집기로 새 .cil 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vim <local_module>.cil
```

로컬 모듈을 더 잘 정리하려면 로컬 SELinux 정책 모듈의 이름에 local_ 접두사를 사용합니다.

2. Known Issue 또는 Red Hat 솔루션에서 사용자 지정 규칙을 삽입합니다.

중요

자신의 규칙을 작성하지 마십시오. 특정 Known Issue 또는 Red Hat 솔루션에서 제공되는 규칙만 사용하십시오.

예를 들어 SELinux에서 RHEL 솔루션의 cups.sock에 대한 cups-lpd 읽기 액세스를 거부하

려면 다음 규칙을 삽입합니다.

참고

RHBA-2021:4420 의 RHEL에서 예제 솔루션이 영구적으로 수정되었습니다. 따라서 이 솔루션과 관련된 절차의 일부는 업데이트된 RHEL 8 및 9 시스템에 영향을 미치지 않으며 구문의 예로만 포함됩니다.

```
(allow cupsd_lpd_t cupsd_var_run_t (sock_file (read)))
```

두 SELinux 규칙 구문 중 하나인 CIL(Common Intermediate Language) 및 m4를 사용할 수 있습니다. 예를 들어, CIL의 cupsd_lpd_t cupsd_var_run_t (sock_file (read)) 는 m4에서 다음과 같습니다.

```
module local_cupslpd-read-cupssock 1.0;

require {
    type cupsd_var_run_t;
    type cupsd_lpd_t;
    class sock_file read;
}

#============= cupsd_lpd_t ==============
allow cupsd_lpd_t cupsd_var_run_t:sock_file read;
```

3. 파일을 저장하고 닫습니다.

4. policy 모듈을 설치합니다.

```
# semodule -i <local_module>.cil
```

참고

semodule -i 를 사용하여 생성한 로컬 정책 모듈을 제거하려면 .cil 접미사가 없는 모듈 이름을 참조하십시오. 로컬 정책 모듈을 제거하려면 semodule -r <*local_module*>*을 사용합니다*.

5. 규칙과 관련된 서비스를 다시 시작합니다.

```
# systemctl restart <service-name>
```

-

검증

1. **SELinux 정책에 설치된 로컬 모듈을 나열합니다.**

```
# semodule -lfull | grep "local_"
400 local_module  cil
```

참고

로컬 모듈에는 우선 순위가 400 이므로 예를 들어 semodule -lfull | grep -v ^100 명령을 사용하여 목록에서도 필터링할 수 있습니다.

2. **관련 allow 규칙에 대한 SELinux 정책을 검색합니다.**

```
# sesearch -A --source=<SOURCENAME> --target=<TARGETNAME> --
class=<CLASSNAME> --perm=<P1>,<P2>
```

여기서 *<SOURCENAME>* 은 소스 SELinux 유형이며 *<TARGETNAME>* 은 대상 SELinux 유형인 *<CLASSNAME>* 은 보안 클래스 또는 오브젝트 클래스 이름이고 *<P1>* 및 *<P2>* 는 규칙의 특정 권한입니다.

예를 들어 SELinux에서 RHEL 솔루션의 cups.sock에 대한 cups-lpd 읽기 액세스를 거부합니다.

```
# sesearch -A --source=cupsd_lpd_t --target=cupsd_var_run_t --class=sock_file --
perm=read
allow cupsd_lpd_t cupsd_var_run_t:sock_file { append getattr open read write };
```

마지막 행에는 읽기 작업이 포함되어야 합니다.

3. **관련 서비스가 SELinux에 의해 제한적으로 실행되는지 확인합니다.**

a. **관련 서비스와 관련된 프로세스를 식별합니다.**

```
$ systemctl status <service-name>
```

b. **이전 명령의 출력에 나열된 프로세스의 SELinux 컨텍스트를 확인합니다.**

```
$ ps -efZ | grep <process-name>
```

4. **서비스에서 SELinux 거부를 유발하지 않는지 확인합니다.**

```
# ausearch -m AVC -ts recent
<no matches>
```

추가 리소스

- *5장. SELinux 관련 문제 해결*

8.4. 추가 리소스

- SELinux 정책 개요

9장. 컨테이너에 대한 SELINUX 정책 생성

Red Hat Enterprise Linux 9는 udica 패키지를 사용하여 컨테이너의 SELinux 정책을 생성하는 툴을 제공합니다. udica 를 사용하면 컨테이너가 호스트 시스템 리소스에 액세스하는 방법(예: 스토리지, 장치, 네트워크)을 보다 효과적으로 제어하기 위한 맞춤형 보안 정책을 생성할 수 있습니다. 이를 통해 보안 위반에 대해 컨테이너 배포를 강화하고 규정 준수를 단순화할 수 있습니다.

9.1. UDICA SELINUX 정책 생성기 소개

사용자 지정 컨테이너에 대한 새 SELinux 정책 생성을 간소화하기 위해 RHEL 9에서는 udica 유틸리티를 제공합니다. 이 도구를 사용하여 Linux-capabilities, 마운트 지점, 포트 정의가 포함된 컨테이너 JavaScript Object Notation(JSON) 파일의 검사를 기반으로 정책을 생성할 수 있습니다. 결과적으로 툴은 지정된 SELinux CIL(Common Intermediate Language) 블록에서 상속된 규칙의 결과를 사용하여 생성된 규칙을 결합합니다.

udica 를 사용하여 컨테이너에 대한 SELinux 정책을 생성하는 프로세스에는 다음 세 가지 주요 부분이 있습니다.

1. JSON 형식의 컨테이너 사양 파일 구문 분석
2. 첫 번째 파트의 결과에 따라 적절한 허용 규칙을 찾습니다.
3. 최종 SELinux 정책 생성

구문 분석 단계에서 udica 는 Linux 기능, 네트워크 포트 및 마운트 지점을 찾습니다.

결과에 따라 udica 는 컨테이너에 필요한 Linux 기능을 감지하고 모든 기능을 허용하는 SELinux 규칙을 만듭니다. 컨테이너가 특정 포트에 바인딩된 경우 udica 는 SELinux 사용자 공간 라이브러리를 사용하여 검사된 컨테이너에서 사용하는 포트의 올바른 SELinux 레이블을 가져옵니다.

그런 다음 udica 는 호스트에서 컨테이너 파일 시스템 이름 공간에 마운트된 디렉토리를 감지합니다.

CIL의 블록 상속 기능을 사용하면 udica 가 특정 작업에 중점을 둔 SELinux *허용 규칙을* 생성할 수 있습니다. 예를 들면 다음과 같습니다.

- *홈 디렉토리에 액세스 허용*
- *로그 파일 액세스 허용*
- ***Xserver**와의 통신을 허용합니다.*

이러한 템플릿을 블록이라고 하며 블록을 병합하여 최종 **SELinux** 정책을 생성합니다.

추가 리소스

- **udica Red Hat** 블로그 문서를 사용하여 컨테이너에 대한 **SELinux** 정책 생성

9.2. 사용자 지정 컨테이너의 SELINUX 정책 생성 및 사용

사용자 지정 컨테이너에 대한 **SELinux** 보안 정책을 생성하려면 다음 절차의 단계를 따르십시오.

사전 요구 사항

- 컨테이너를 관리하기 위한 **podman** 툴이 설치되어 있습니다. 그렇지 않은 경우 **dnf install podman** 명령을 사용합니다.
- 사용자 지정 **Linux** 컨테이너 **-** 이 예에서는 ***ubi8*** 입니다.

절차

1. **udica** 패키지를 설치합니다.

```
# dnf install -y udica
```

또는 **udica** 를 포함한 컨테이너 소프트웨어 패키지 세트를 제공하는 **container-tools** 모듈을 설치합니다.

```
# dnf module install -y container-tools
```

2.

읽기 전용 권한과 /var/spool 디렉터리를 읽기 및 쓰기 권한을 사용하여 /home 디렉터리를 마운트하는 *ubi8* 컨테이너를 시작합니다. 컨테이너는 포트 *21* 을 노출합니다.

```
# podman run --env container=podman -v /home:/home:ro -v /var/spool:/var/spool:rw -p
21:21 -it ubi8 bash
```

이제 컨테이너는 container_t SELinux 유형으로 실행됩니다. 이 유형은 SELinux 정책의 모든 컨테이너에 대한 일반 도메인이며 시나리오에 너무 엄격한 경우가 있거나 너무 느릴 수 있습니다.

3.

새 터미널을 열고 podman ps 명령을 입력하여 컨테이너의 ID를 가져옵니다.

```
# podman ps
CONTAINER ID  IMAGE                                 COMMAND  CREATED         STATUS
PORTS  NAMES
37a3635afb8f  registry.access.redhat.com/ubi8:latest  bash     15 minutes ago  Up 15
minutes ago          heuristic_lewin
```

4.

컨테이너 JSON 파일을 생성하고 JSON 파일의 정보를 기반으로 정책 모듈을 생성하는 데 udica 를 사용합니다.

```
# podman inspect 37a3635afb8f > container.json
# udica -j container.json my_container
Policy my_container with container id 37a3635afb8f created!
[...]
```

또는 다음을 수행합니다.

```
# podman inspect 37a3635afb8f | udica my_container
Policy my_container with container id 37a3635afb8f created!

Please load these modules using:
# semodule -i my_container.cil
/usr/share/udica/templates/{base_container.cil,net_container.cil,home_container.cil}

Restart the container with: "--security-opt label=type:my_container.process" parameter
```

5.

이전 단계에서 udica 의 출력에 표시된 대로 policy 모듈을 로드합니다.

```
# semodule -i my_container.cil
/usr/share/udica/templates/{base_container.cil,net_container.cil,home_container.cil}
```

6. **컨테이너를 중지하고 --security-opt label=type:my_container.process 옵션을 사용하여 다시 시작합니다.**

```
# podman stop 37a3635afb8f
# podman run --security-opt label=type:my_container.process -v /home:/home:ro -v
/var/spool:/var/spool:rw -p 21:21 -it ubi8 bash
```

검증

1. **컨테이너가 my_container.process 유형으로 실행되는지 확인합니다.**

```
# ps -efZ | grep my_container.process
unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023 root 2275 434  1 13:49 pts/1
00:00:00 podman run --security-opt label=type:my_container.process -v /home:/home:ro -v
/var/spool:/var/spool:rw -p 21:21 -it ubi8 bash
system_u:system_r:my_container.process:s0:c270,c963 root 2317 2305  0 13:49 pts/0
00:00:00 bash
```

2. **SELinux가 이제 /home 및 /var/spool 마운트 지점에 액세스할 수 있는지 확인합니다.**

```
[root@37a3635afb8f /]# cd /home
[root@37a3635afb8f home]# ls
username
[root@37a3635afb8f ~]# cd /var/spool/
[root@37a3635afb8f spool]# touch test
[root@37a3635afb8f spool]#
```

3. **SELinux가 포트 21에만 바인딩을 허용하는지 확인합니다.**

```
[root@37a3635afb8f /]# dnf install nmap-ncat
[root@37a3635afb8f /]# nc -lvp 21
...
Ncat: Listening on :::21
Ncat: Listening on 0.0.0.0:21
^C
[root@37a3635afb8f /]# nc -lvp 80
...
Ncat: bind to :::80: Permission denied. QUITTING.
```

추가 리소스

- **udica(8) 및 podman(1) 도움말 페이지**

- 컨테이너 빌드, 실행 및 관리

9.3. 추가 리소스

- udica - 컨테이너에 대한 SELinux 정책 생성

10장. 여러 시스템에 동일한 SELINUX 구성 배포

이 섹션에서는 여러 시스템에 인증된 SELinux 구성을 배포하는 데 권장되는 두 가지 방법을 제공합니다.

- RHEL 시스템 역할 및 Ansible 사용
- 스크립트에서 semanage export 및 import 명령 사용

10.1. SELINUX 시스템 역할 소개

RHEL 시스템 역할은 여러 RHEL 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 Ansible 역할 및 모듈의 컬렉션입니다. SELinux 시스템 역할을 사용하면 다음 작업을 수행할 수 있습니다.

- SELinux 부울, 파일 컨텍스트, 포트 및 로그인과 관련된 로컬 정책 수정을 정리합니다.
- SELinux 정책 부울 설정, 파일 컨텍스트, 포트, 로그인.
- 지정된 파일 또는 디렉터리에서 파일 컨텍스트 복원.
- SELinux 모듈 관리.

다음 표에서는 SELinux 시스템 역할에서 사용할 수 있는 입력 변수에 대한 개요를 제공합니다.

표 10.1. SELinux 시스템 역할 변수

역할 변수	설명	CLI 대체
selinux_policy	대상 프로세스 또는 다중 수준 보안 보호를 보호하는 정책을 선택합니다.	/etc/selinux/config의 SELINUXTYPE
selinux_state	은 SELinux 모드를 대체합니다.	/etc/selinux/config 의 setenforce 및 SELINUX.

역할 변수	설명	CLI 대체
selinux_booleans	SELinux 부울을 활성화 및 비활성화합니다.	**setsebool**
selinux_fcontexts	SELinux 파일 컨텍스트 매핑을 추가하거나 제거합니다.	**semanage fcontext**
selinux_restore_dirs	파일 시스템 트리에서 SELinux 레이블을 복원합니다.	**restorecon -R**
selinux_ports	포트에 SELinux 레이블을 설정합니다.	**semanage port**
selinux_logins	사용자를 SELinux 사용자 매핑으로 설정합니다.	**semanage login**
selinux_modules	SELinux 모듈을 설치, 활성화, 비활성화 또는 제거합니다.	**semodule**

rhel-system-roles 패키지에서 설치한 /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml 예제 플레이북은 강제 모드에서 대상 정책을 설정하는 방법을 보여줍니다. 또한 플레이북은 /tmp/test_dir/ 디렉터리의 여러 로컬 정책 수정 및 복원 파일 컨텍스트를 적용합니다.

SELinux 역할 변수에 대한 자세한 참조는 rhel-system-roles 패키지를 설치하고 /usr/share/doc/rhel-system-roles/selinux/ 디렉터리에서 README.md 또는 README.html 파일을 참조하십시오.

추가 리소스

- RHEL 시스템 역할 소개.

10.2. SELINUX 시스템 역할을 사용하여 여러 시스템에 SELINUX 설정 적용

단계에 따라 검증된 **SELinux** 설정으로 **Ansible** 플레이북을 준비 및 적용합니다.

사전 요구 사항

- **SELinux** 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 파일이 있어야 합니다.

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 방법 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 범위를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일이 있어야 합니다.

절차

1. 플레이북을 준비합니다. 처음부터 시작하거나 **rhel-system-roles** 패키지의 일부로 설치된 예제 플레이북을 수정할 수 있습니다.

```
# cp /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml my-selinux-playbook.yml
# vi my-selinux-playbook.yml
```

2. 시나리오에 맞게 플레이북의 내용을 변경합니다. 예를 들어 다음 부분에서는 시스템이 **selinux-local-1.pp SELinux** 모듈을 설치하고 활성화합니다.

```
selinux_modules:
- { path: "selinux-local-1.pp", priority: "400" }
```

3. 변경 사항을 저장하고 텍스트 편집기를 종료합니다.

4. ***host1*,*host2*** 및 ***host3*** 시스템에서 플레이북을 실행합니다.

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

추가 리소스

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

10.3. SEMANAGE를 사용하여 SELINUX 설정을 다른 시스템으로 전송

RHEL 9 기반 시스템 간에 사용자 지정 및 검증된 **SELinux** 설정을 전송하려면 다음 단계를 사용합니다.

사전 요구 사항

- **policycoreutils-python-utils** 패키지가 시스템에 설치되어 있습니다.

절차

1. 확인된 **SELinux** 설정을 내보냅니다.

```
# semanage export -f ./my-selinux-settings.mod
```

2. 설정을 사용하여 파일을 새 시스템에 복사합니다.

```
# scp ./my-selinux-settings.mod new-system-hostname:
```

3. 새 시스템에서 로그인합니다.

```
$ ssh root@new-system-hostname
```

-

4. **새 시스템에서 설정을 가져옵니다.**

```
new-system-hostname# semanage import -f ./my-selinux-settings.mod
```

추가 리소스

- **semanage-export(8)** 및 **semanage-import(8)** 도움말 페이지