Red Hat Enterprise Linux 8

스마트 카드 인증 관리

스마트 카드 인증 구성 및 사용

Last Updated: 2023-09-19

Red Hat Enterprise Linux 8 스마트 카드 인증 관리

스마트 카드 인증 구성 및 사용

법적 공지

초록

Red Hat IdM(Identity Management)을 사용하면 인증 정보를 개인 키와 스마트 카드의 인증서 형태로 저장할 수 있습니다. 그런 다음 암호 대신 이 스마트 카드를 사용하여 서비스에 인증할 수 있습니다. 관리자는 관리 오버헤드를 줄이기 위해 매핑 규칙을 구성할 수 있습니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서 및 웹 속성에서 문제가 있는 언어를 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.

Identity Management에서 계획된 용어 교체는 다음과 같습니다.

- ***차단 목록*** 대체 *블랙리스트*
- ***목록*** 교체 *허용 화이트리스트*
- ***2차*** 대체 *슬레이브*
- *master* 라는 단어는 컨텍스트에 따라 더 정확한 언어로 교체됩니다.
  - ***IdM 서버가*** *IdM 마스터*교체
  - ***CA 갱신 서버가*** *CA 갱신 마스터*교체
  - ***CRL 게시자 서버가*** *CRL 마스터*교체
  - ***멀티 공급자*** 대체 *멀티 마스터*

RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

특정 문구에 대한 의견 제출

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

Jira를 통해 피드백 제출 (등록 필요)

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

1장. 스마트 카드 인증 이해

스마트 카드를 기반으로 하는 인증은 암호에 대한 대안입니다. 개인 키와 인증서의 형태로 스마트 카드에 사용자 자격 증명을 저장할 수 있으며 특수 소프트웨어 및 하드웨어를 사용하여 액세스할 수 있습니다. 스마트 카드를 리더 또는 USB 포트에 저장하고 암호를 제공하는 대신 스마트 카드의 PIN 코드를 제공합니다.

이 섹션에서는 스마트 카드가 무엇인지와 스마트 카드 인증이 작동하는 방식을 설명합니다. 스마트 카드 콘텐츠를 읽고 조작하는 데 사용할 수 있는 도구를 설명합니다. 또한 샘플 사용 사례를 제공하고 스마트 카드 인증을 위한 IdM 서버와 IdM 클라이언트 둘 다의 설정에 대해 설명합니다.

참고

스마트 카드 인증 사용을 시작하려면 하드웨어 요구 사항을 참조하십시오. RHEL8의 스마트 카드 지원.

1.1. 스마트 카드란 무엇인가?

스마트 카드는 물리적 장치이며 일반적으로 카드에 저장된 인증서를 사용하여 개인 인증을 제공할 수 있는 마이크로프로세서가 있는 비결제 카드입니다. 개인 인증은 사용자 암호와 동일한 방식으로 스마트 카드를 사용할 수 있음을 의미합니다.

개인 키와 인증서의 형태로 스마트 카드에 사용자 자격 증명을 저장할 수 있으며 특수 소프트웨어 및 하드웨어를 사용하여 액세스 할 수 있습니다. 스마트 카드를 리더나 USB 소켓에 배치하고 암호를 제공하는 대신 스마트 카드의 PIN 코드를 제공합니다.

1.2. 스마트 카드 인증이란?

공개 키 기반 인증 및 인증서 기반 인증은 암호 기반 인증에 널리 사용되는 두 가지 대안입니다. ID는 암호 대신 공개 및 개인 키를 사용하여 확인됩니다. 인증서는 개인, 서버, 회사 또는 기타 개체를 식별하고 해당 ID를 공개 키와 연결하는 데 사용되는 전자 문서입니다. 드라이버의 라이센스 또는 패스와 마찬가지로, 인증서는 일반적으로 사람의 신원에 대한 증명으로 인식됩니다. 공개 키 암호화는 인증서를 사용하여 가장 문제를 해결합니다.

스마트 카드 인증의 경우 사용자 자격 증명(공개 및 개인 키 및 인증서)은 스마트 카드에 저장되며 스마트 카드가 reader에 삽입되고 Pin이 제공된 후에만 사용할 수 있습니다. 물리적 장치, 스마트 카드 및 해당 Pin을 알아야 하는 경우 스마트 카드 인증은 두 가지 인증 유형으로 간주됩니다.

1.2.1. IdM의 스마트 카드 인증 예

다음 예제에서는 IdM에서 스마트 카드를 사용하는 방법에 대한 두 가지 간단한 시나리오를 설명합니다.

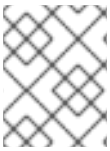
1.2.1.1. 스마트 카드를 사용하여 시스템에 로그인

스마트 카드를 사용하여 로컬 사용자로 RHEL 시스템에 인증할 수 있습니다. 시스템이 스마트 카드 로그인을 적용하도록 구성된 경우 스마트 카드를 삽입하고 해당 DVD를 입력하라는 메시지가 표시되면 시스템에 로그인할 수 없습니다. 또는 스마트 카드 인증 또는 사용자 이름과 암호를 사용하여 인증하도록 시스템을 구성할 수 있습니다. 이 경우 스마트 카드가 삽입되지 않은 경우 사용자 이름과 암호를 입력하라는 메시지가 표시됩니다.

1.2.1.2. 제거 잠금을 사용하여 GDM에 로그인

RHEL 시스템에서 스마트 카드 인증을 구성한 경우 잠금 제거 기능을 활성화할 수 있습니다. GDM(GNOME Display Manager)에 로그인한 후 스마트 카드를 제거하면 화면 잠금이 활성화되고 스마트

카드를 다시 삽입하여 화면 잠금을 해제하려면 StorageClass로 인증해야 합니다. 사용자 이름과 암호를 사용하여 인증할 수 없습니다.



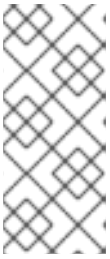
참고

GDM에 로그인한 후 스마트 카드를 제거하면 화면 잠금이 활성화되므로 스마트 카드를 다시 삽입하여 화면 잠금을 해제하려면 PIN으로 인증해야 합니다.

1.3. RHEL의 스마트 카드 인증 옵션

authselect 명령인 **authselect enable-feature <smartcardoption** >을 사용하여 스마트 카드 인증이 특정 IdM(Identity Management) 클라이언트에서 작동하도록 하는 방법을 구성할 수 있습니다. 다음과 같은 스마트 카드 옵션을 사용할 수 있습니다.

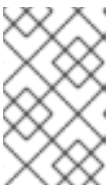
- **with-smartcard**: 사용자는 사용자 이름 및 암호로 또는 스마트 카드로 인증할 수 있습니다.
- **with-smartcard-required**: 사용자는 스마트 카드로 인증할 수 있으며 암호 인증은 비활성화되어 있습니다. 스마트 카드 없이는 시스템에 액세스 할 수 없습니다. 스마트 카드로 인증되면 스마트 카드가 리더로부터 제거된 경우에도 로그인 상태를 유지할 수 있습니다.



참고

with-smartcard-required 옵션은 로그인 ,**gdm**,**xdm**,**xscreensaver**, **gnome-screensaver** 와 같은 **로그인** 서비스에 대해서만 독점 스마트 카드 인증을 적용합니다. 사용자를 전환하는 **su** 또는 **sudo** 와 같은 다른 서비스의 경우 스마트 카드 인증이 적용되지 않으며 스마트 카드가 삽입되지 않은 경우 암호를 입력하라는 메시지가 표시됩니다.

- **with-smartcard-lock-on-removal**: 사용자는 스마트 카드로 인증할 수 있습니다. 그러나 독자에서 스마트 카드를 제거하면 시스템이 자동으로 잠깁니다. 암호 인증은 사용할 수 없습니다.



참고

with-smartcard-lock-on-removal 옵션은 GNOME 데스크탑 환경의 시스템에서만 작동합니다. **tty** 또는 콘솔 기반 시스템을 사용하고 있으며 리더에서 스마트 카드를 제거하는 경우 시스템에서 자동으로 잠기지 않습니다.

자세한 내용은 authselect를 사용하여 스마트 카드 구성을 참조하십시오.

1.4. 스마트 카드 및 콘텐츠를 관리하는 툴

다양한 도구를 사용하여 스마트 카드에 저장된 키와 인증서를 관리할 수 있습니다. 다음 도구를 사용하여 다음을 수행할 수 있습니다.

- 시스템에 연결된 사용 가능한 스마트 카드 리더 나열.
- 사용 가능한 스마트 카드를 나열하고 콘텐츠를 봅니다.
- 스마트 카드 콘텐츠를 조작합니다. 키 및 인증서입니다.

유사한 기능을 제공하지만 시스템의 다른 계층에서 작동하는 많은 도구가 있습니다. 스마트 카드는 여러 구성 요소에 의해 여러 계층에서 관리됩니다. 하한 수준에서 운영 체제는 PC/SC 프로토콜을 사용하여 스마트 카드 리더와 통신하고 이 통신은 pcsc-lite 데몬에 의해 처리됩니다. 데몬은 일반적으로 USB를 통해

수신된 명령을 스마트 카드 리더에 전달하며, 이는 낮은 수준의 CCID 드라이버에서 처리합니다. PC/SC 하위 수준 통신이 애플리케이션 수준에 거의 표시되지 않습니다. RHEL의 애플리케이션이 스마트 카드에 액세스하기 위한 기본 방법은 더 높은 수준의 API(애플리케이션 프로그래밍 인터페이스), OASIS PKCS#11 API를 통해 카드 통신을 암호화 개체(예: 개인 키)에서 작동하는 특정 명령에 추상화하는 것입니다. 스마트 카드 공급업체는 PKCS#11 API를 따르고 스마트 카드의 드라이버 역할을 하는 **.so** 파일과 같은 공유 모듈을 제공합니다.

다음 도구를 사용하여 스마트 카드와 콘텐츠를 관리할 수 있습니다.

- OpenSC 툴: **opensc** 에서 구현된 드라이버와 함께 작동합니다.
  - OpenSC-tool: 스마트 카드 작업을 수행합니다.
  - pkcs15-tool: 토큰에 저장된 pins, 키 및 인증서를 나열 및 읽는 것과 같은 스마트 카드에서 PKCS#15 데이터 구조를 관리합니다.
  - pkcs11-tool: 토큰에 저장된 pins, 키 및 인증서를 나열 및 읽는 것과 같은 스마트 카드에서 PKCS#11 데이터 개체를 관리합니다.
- gnutls utils: 애플리케이션이 네트워크 전송 계층을 통해 보안 통신을 가능하게 하고 X.509, PKCS#12, OpenPGP 및 기타 구조에 액세스할 수 있는 인터페이스를 제공합니다.
  - p11tool: PKCS#11 스마트 카드 및 보안 모듈에서 작업을 수행합니다.
  - certtool: X.509 인증서, 요청 및 개인 키를 구문 분석하고 생성합니다.
- NSS(Network Security Services) 도구: 보안 지원 클라이언트 및 서버 애플리케이션의 플랫폼 간 개발을 지원하도록 설계된 라이브러리 세트입니다. NSS로 빌드된 애플리케이션은 SSL v3, TLS, PKCS #5, PKCS #7, PKCS #11, PKCS #12, S/MIME, X.509 v3 인증서 및 기타 보안 표준을 지원할 수 있습니다.
  - modutil: 보안 모듈 데이터베이스를 사용하여 PKCS#11 모듈 정보를 관리합니다.
  - certutil: NSS 데이터베이스 및 기타 NSS 토큰 모두에서 키와 인증서를 관리합니다.

이러한 툴을 사용하여 스마트 카드 사용 인증 문제를 해결하는 방법에 대한 자세한 내용은 스마트 카드로 인증 문제 해결을 참조하십시오.

추가 리소스

- **OpenSC-tool** man 페이지
- **pkcs15-tool** 매뉴얼 페이지
- **pkcs11-tool** man 페이지
- **p11tool** 도움말 페이지
- **certtool** 도움말 페이지
- **modutil** 도움말 페이지
- **certutil** 도움말 페이지

1.5. 인증서 및 스마트 카드 인증

IdM(Identity Management) 또는 AD(Active Directory)를 사용하여 도메인의 ID 저장소, 인증, 정책 및 권

한 부여 정책을 관리하는 경우 각각 IdM 또는 AD에서 인증에 사용되는 인증서가 생성됩니다. 외부 인증 기관에서 제공하는 인증서를 사용할 수도 있으며 이 경우 외부 공급자의 인증서를 수락하도록 Active Directory 또는 IdM을 구성해야 합니다. 사용자가 도메인에 속하지 않은 경우 로컬 인증 기관에서 생성한 인증서를 사용할 수 있습니다. 자세한 내용은 다음 섹션을 참조하십시오.

- 스마트 카드 인증을 위한 Identity Management 구성
- IdM에서 스마트 카드 인증을 위해 ADCS에서 발급한 인증서 구성
- IdM 사용자, 호스트 및 서비스용 외부 서명된 인증서 관리
- 스마트 카드로 로컬 인증서 구성 및 가져오기

스마트 카드 인증을 받을 수 있는 전체 인증서 목록은 스마트 카드를 사용할 수 있는 인증서를 참조하십시오.

1.6. IDM에서 스마트 카드 인증에 필요한 단계

IdM(Identity Management)에서 스마트 카드로 인증하려면 다음 단계를 따라야 합니다.

- 스마트 카드 인증을 위해 IdM 서버를 구성합니다. 스마트 카드 인증을 위한 IdM 서버 구성을 참조하십시오.
- 스마트 카드 인증을 위해 IdM 클라이언트를 구성합니다. 스마트 카드 인증을 위한 IdM 클라이언트 구성을 참조하십시오.
- IdM의 사용자 항목에 인증서를 추가합니다. IdM 웹 UI의 사용자 항목에 인증서 추가를 참조하십시오.
- 키와 인증서를 스마트 카드에 저장하십시오. 스마트 카드의 인증서 저장을 참조하십시오.

1.7. ACTIVE DIRECTORY에서 발급한 인증서가 있는 스마트 카드 인증에 필요한 단계

AD(Active Directory)에서 발급한 인증서로 스마트 카드를 인증하려면 먼저 다음 단계를 수행해야 합니다.

- Active Directory의 CA 및 사용자 인증서를 IdM 서버 및 클라이언트로 복사합니다 .
- ADCS 인증서를 사용하여 스마트 카드 인증을 위해 IdM 서버 및 클라이언트를 구성합니다 .
- Pcabundle (PKCS#12) 파일을 변환하여 스마트 카드에 인증서 및 개인 키를 저장할 수 있습니다 .
- sssd.conf 파일에서 시간 제한을 구성합니다 .
- 스마트 카드 인증에 대한 인증서 매핑 규칙을 생성합니다 .

2장. 스마트 카드 인증을 위한 ID 관리 구성

IdM(Identity Management)은 다음을 사용하여 스마트 카드 인증을 지원합니다.

- IdM 인증 기관에서 발급한 사용자 인증서
- 외부 인증 기관에서 발급한 사용자 인증서

두 가지 유형의 인증서에 대해 IdM에서 스마트 카드 인증을 구성할 수 있습니다. 이 시나리오에서 **rootca.pem** CA 인증서는 신뢰할 수 있는 외부 인증 기관의 인증서를 포함하는 파일입니다.

IdM의 스마트 카드 인증에 대한 자세한 내용은 스마트 카드 인증 이해 를 참조하십시오.

스마트 카드 인증 구성에 대한 자세한 내용은 다음을 수행합니다.

- 스마트 카드 인증을 위한 IdM 서버 구성
- 스마트 카드 인증을 위한 IdM 클라이언트 구성
- IdM 웹 UI의 사용자 항목에 인증서 추가
- IdM CLI의 사용자 항목에 인증서 추가
- 스마트 카드를 관리하고 사용하기 위한 도구 설치
- 스마트 카드에 인증서 저장
- 스마트 카드를 사용하여 IdM에 로그인
- 스마트 카드 인증을 사용하여 GDM 액세스 구성
- 스마트 카드 인증을 사용하여 su 액세스 구성

2.1. 스마트 카드 인증을 위해 IDM 서버 구성

IdM(Identity Management) CA에서 신뢰하는 <EXAMPLE.ORG> 도메인에서 인증서를 발급한 사용자의 스마트 카드 인증을 활성화하려면 IdM 서버를 구성하는 **ipa-advise** 스크립트를 실행할 때 다음 인증서를 가져와야 합니다.

- <EXAMPLE.ORG> CA 인증서를 직접 발급했거나 하위 CA 중 하나 이상을 통해 발급한 루트 CA의 인증서입니다. 기관에서 인증서를 발급한 웹 페이지에서 인증서 체인을 다운로드할 수 있습니다. 자세한 내용은 인증서 인증을 사용하도록 브라우저 구성에서 1-4a 단계를 참조하십시오.
- IdM CA 인증서입니다. IdM CA 인스턴스가 실행 중인 IdM 서버의 **/etc/ipa/ca.crt** 파일에서 CA 인증서를 가져올 수 있습니다.
- 모든 중간 CA의 인증서입니다. 즉 <EXAMPLE.ORG> CA와 IdM CA 사이입니다.

스마트 카드 인증을 위해 IdM 서버를 구성하려면 다음을 수행합니다.

1. PEM 형식으로 CA 인증서를 사용하여 파일을 가져옵니다.
2. 내장된 **ipa-advise** 스크립트를 실행합니다.
3. 시스템 구성을 다시 로드합니다.

사전 요구 사항

- IdM 서버에 대한 루트 액세스 권한이 있습니다.
- 루트 CA 인증서와 모든 중간 CA 인증서가 있습니다.

절차

1. 설정을 수행할 디렉터리를 생성합니다.

```
[root@server]# mkdir ~/SmartCard/
```

2. 디렉터리로 이동합니다.

```
[root@server]# cd ~/SmartCard/
```

3. PEM 형식의 파일에 저장된 관련 CA 인증서를 가져옵니다. CA 인증서가 DER와 같은 다른 형식의 파일에 저장된 경우 이를 PEM 형식으로 변환합니다. IdM 인증 기관 인증서는 PEM 형식이며 **/etc/ipa/ca.crt** 파일에 있습니다.
DER 파일을 PEM 파일로 변환합니다.

```
# openssl x509 -in <filename>.der -inform DER -out <filename>.pem -outform PEM
```

4. 편의를 위해 구성을 수행하려는 디렉터리에 인증서를 복사합니다.

```
[root@server SmartCard]# cp /tmp/rootca.pem ~/SmartCard/
[root@server SmartCard]# cp /tmp/subca.pem ~/SmartCard/
[root@server SmartCard]# cp /tmp/issuingca.pem ~/SmartCard/
```

5. 선택적으로 외부 인증 기관의 인증서를 사용하는 경우 **openssl x509** 유틸리티를 사용하여 **PEM** 형식으로 파일의 내용을 확인하여 **발급자** 및 **주체** 값이 올바른지 확인합니다.

```
[root@server SmartCard]# openssl x509 -noout -text -in rootca.pem | more
```

6. 관리자 권한을 사용하여 기본 제공 **ipa-advise** 유틸리티로 구성 스크립트를 생성합니다.

```
[root@server SmartCard]# kinit admin
[root@server SmartCard]# ipa-advise config-server-for-smart-card-auth > config-server-for-smart-card-auth.sh
```

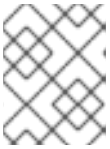
config-server-for-smart-card-auth.sh 스크립트는 다음 작업을 수행합니다.

- IdM Apache HTTP 서버를 구성합니다.
- KDC(키 배포 센터)에서 Kerberos(PKINIT)의 초기 인증에 대한 공개 키 암호화를 활성화합니다.
- 스마트 카드 권한 부여 요청을 수락하도록 IdM 웹 UI를 구성합니다.

7. 스크립트를 실행하여 루트 CA 및 하위 CA 인증서를 포함하는 PEM 파일을 인수로 추가합니다.

```
[root@server SmartCard]# chmod +x config-server-for-smart-card-auth.sh
[root@server SmartCard]# ./config-server-for-smart-card-auth.sh rootca.pem subca.pem issuingca.pem
```

```
Ticket cache:KEYRING:persistent:0:0
Default principal: admin@IDM.EXAMPLE.COM
[...]
Systemwide CA database updated.
The ipa-certupdate command was successful
```



참고

루트 CA 인증서를 하위 CA 인증서 전에 인수로 추가하고 CA 또는 하위 CA 인증서가 만료되지 않았는지 확인합니다.

8. 선택적으로 사용자 인증서를 발행한 인증 기관에서 OCSP(Online Certificate Status Protocol) 응답자를 제공하지 않는 경우 IdM 웹 UI에 대한 인증을 OCSP 검사를 비활성화해야 할 수도 있습니다.

   a. **/etc/httpd/conf.d/ssl.conf 파일에서 SSLOCSPEnable 매개변수를 off 로 설정합니다.**

   ```
   SSLOCSPEnable off
   ```

   b. 변경 사항을 즉시 적용하려면 Apache 데몬(httpd)을 다시 시작하십시오.

   ```
   [root@server SmartCard]# systemctl restart httpd
   ```

주의

IdM CA에서 발급한 사용자 인증서만 사용하는 경우 OCSP 검사를 비활성화하지 마십시오. OCSP 응답자는 IdM의 일부입니다.

OCSP 검사를 계속 활성화하는 방법에 대한 지침이 있지만 사용자 인증서가 OCSP 서비스 요청을 수신 대기하는 CA에 대한 정보가 포함되지 않은 경우 IdM 서버가 사용자 인증서를 거부하지 않는 경우 Apache mod_ssl 구성 옵션의 **SSLOCSPDefaultResponder** 지시문을 참조하십시오.

이제 서버는 스마트 카드 인증을 위해 구성됩니다.

참고

전체 토폴로지에서 스마트 카드 인증을 사용하려면 각 IdM 서버에서 절차를 실행합니다.

2.2. ANSIBLE을 사용하여 스마트 카드 인증을 위해 IDM 서버 구성

Ansible을 사용하면 IdM(Identity Management) CA에서 신뢰하는 <EXAMPLE.ORG> 도메인의 인증서가 발급한 사용자의 스마트 카드 인증을 활성화할 수 있습니다. 이렇게 하려면 **ipasmartcard_server ansible-freeipa** 역할 스크립트로 Ansible 플레이북을 실행할 때 사용할 수 있도록 다음 인증서를 가져와야 합니다.

- <EXAMPLE.ORG> CA 인증서를 직접 발급했거나 하위 CA 중 하나 이상을 통해 발급한 루트 CA의 인증서입니다. 기관에서 인증서를 발급한 웹 페이지에서 인증서 체인을 다운로드할 수 있습니다. 자세한 내용은 Configuring a browser to enable certificate authentication 을 참조하십시오.

- IdM CA 인증서입니다. IdM CA 서버의 **/etc/ipa/ca.crt** 파일에서 CA 인증서를 가져올 수 있습니다.
- <EXAMPLE.ORG> CA와 IdM CA 사이에 있는 모든 CA의 인증서입니다.

사전 요구 사항

- IdM 서버에 대한 **root** 액세스 권한이 있어야 합니다.
- IdM **관리자** 암호를 알고 있습니다.
- 루트 CA 인증서, IdM CA 인증서 및 모든 중간 CA 인증서가 있습니다.
- 다음 요구 사항을 충족하도록 Ansible 제어 노드를 구성했습니다.
  - Ansible 버전 2.14 이상을 사용하고 있습니다.
  - Ansible 컨트롤러에 **ansible-freeipa** 패키지가 설치되어 있습니다.
  - 이 예제에서는 ***~/MyPlaybook/*** 디렉터리에서 IdM 서버의 FQDN(정규화된 도메인 이름)을 사용하여 Ansible 인벤토리 파일을 생성했다고 가정합니다.
  - 이 예제에서는 **secret.yml** Ansible 자격 증명 모음이 **ipaadmin_password** 를 저장하는 것으로 가정합니다.

절차

1. CA 인증서가 **DER** 와 같은 다른 형식의 파일에 저장된 경우 **PEM** 형식으로 변환합니다.

```
# openssl x509 -in <filename>.der -inform DER -out <filename>.pem -outform PEM
```

   IdM 인증 기관 인증서는 **PEM** 형식이며 **/etc/ipa/ca.crt** 파일에 있습니다.

2. 필요한 경우 **openssl x509** 유틸리티를 사용하여 **PEM** 형식의 파일 내용을 보고 **Issuer** 및 **Subject** 값이 올바른지 확인합니다.

```
# openssl x509 -noout -text -in root-ca.pem | more
```

3. ***~/MyPlaybook/*** 디렉토리로 이동합니다.

```
$ cd ~/MyPlaybooks/
```

4. CA 인증서 전용 하위 디렉터리를 생성합니다.

```
$ mkdir SmartCard/
```

5. 편의를 위해 필요한 모든 인증서를 **~/MyPlaybook/SmartCard/** 디렉터리에 복사합니다.

```
# cp /tmp/root-ca.pem ~/MyPlaybooks/SmartCard/
# cp /tmp/intermediate-ca.pem ~/MyPlaybooks/SmartCard/
# cp /etc/ipa/ca.crt ~/MyPlaybooks/SmartCard/ipa-ca.crt
```

6. Ansible 인벤토리 파일에서 다음을 지정합니다.
   - 스마트 카드 인증을 위해 구성할 IdM 서버입니다.

- IdM 관리자 암호입니다.
- 다음 순서로 CA 인증서의 경로입니다.
  - 루트 CA 인증서 파일
  - 중간 CA 인증서 파일
  - IdM CA 인증서 파일

파일은 다음과 같습니다.

```
[ipaserver]
ipaserver.idm.example.com

[ipareplicas]
ipareplica1.idm.example.com
ipareplica2.idm.example.com

[ipacluster:children]
ipaserver
ipareplicas

[ipacluster:vars]
ipaadmin_password=SomeADMINpassword
ipasmartcard_server_ca_certs=/home/<user_name>/MyPlaybooks/SmartCard/root-ca.pem,/home/<user_name>/MyPlaybooks/SmartCard/intermediate-ca.pem,/home/<user_name>/MyPlaybooks/SmartCard/ipa-ca.crt
```

7. 다음 콘텐츠를 사용하여 **install-smartcard-server.yml** 플레이북을 생성합니다.

```
---
- name: Playbook to set up smart card authentication for an IdM server
  hosts: ipaserver
  become: true

  roles:
  - role: ipasmartcard_server
    state: present
```

8. 파일을 저장합니다.
9. Ansible 플레이북을 실행합니다. Playbook 파일, **secret.yml** 파일을 보호하는 암호를 저장하는 파일, 인벤토리 파일을 지정합니다.

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory install-smartcard-server.yml
```

ipasmartcard_server Ansible 역할은 다음 작업을 수행합니다.

- IdM Apache HTTP 서버를 구성합니다.
- KDC(키 배포 센터)에서 Kerberos(PKINIT)의 초기 인증에 대한 공개 키 암호화를 활성화합니다.
- 스마트 카드 권한 부여 요청을 수락하도록 IdM 웹 UI를 구성합니다.

10. 선택적으로 사용자 인증서를 발행한 인증 기관에서 OCSP(Online Certificate Status Protocol) 응답자를 제공하지 않는 경우 IdM 웹 UI에 대한 인증을 OCSP 검사를 비활성화해야 할 수도 있습니다.

    a. **root** 로 IdM 서버에 연결합니다.

    ```
    ssh root@ipaserver.idm.example.com
    ```

    b. **/etc/httpd/conf.d/ssl.conf 파일에서 SSLOCSPEnable 매개변수를 off 로 설정합니다.**

    ```
    SSLOCSPEnable off
    ```

    c. 변경 사항을 즉시 적용하려면 Apache 데몬(httpd)을 다시 시작하십시오.

    ```
    # systemctl restart httpd
    ```

주의

IdM CA에서 발급한 사용자 인증서만 사용하는 경우 OCSP 검사를 비활성화하지 마십시오. OCSP 응답자는 IdM의 일부입니다.

OCSP 검사를 계속 활성화하는 방법에 대한 지침이 있지만 사용자 인증서가 OCSP 서비스 요청을 수신 대기하는 CA에 대한 정보가 포함되지 않은 경우 IdM 서버가 사용자 인증서를 거부하지 않는 경우 Apache mod_ssl 구성 옵션의 **SSLOCSPDefaultResponder** 지시문을 참조하십시오.

인벤토리 파일에 나열된 서버가 스마트 카드 인증을 위해 구성되어 있습니다.

참고

전체 토폴로지에서 스마트 카드 인증을 활성화하려면 Ansible 플레이북의 **hosts** 변수를 **ipacluster** 로 설정합니다.

```
---
- name: Playbook to setup smartcard for IPA server and replicas
  hosts: ipacluster
[...]
```

추가 리소스

- **/usr/share/doc/ansible-freeipa/playbooks/** 디렉터리에서 **ipasmartcard_server** 역할을 사용하는 샘플 플레이북

2.3. 스마트 카드 인증을 위해 IDM 클라이언트 구성

스마트 카드 인증을 위해 IdM 클라이언트를 구성하려면 다음 절차를 따르십시오. 인증을 위해 스마트 카드를 사용하는 동안 연결하려는 각 IdM 시스템, 클라이언트 또는 서버에서 절차를 실행해야 합니다. 예를 들어 호스트 A에서 호스트 B로의 **ssh** 연결을 활성화하려면 스크립트를 호스트 B에서 실행해야 합니다.

관리자는 을 사용하여 스마트 카드 인증을 활성화하려면 다음 절차를 실행합니다.

- **ssh** 프로토콜
  자세한 내용은 스마트 카드 인증을 사용하여 SSH 액세스 구성 을 참조하십시오.
- 콘솔 로그인
- GNOME 디스플레이 관리자(GDM)
- **su** 명령

이 절차는 IdM 웹 UI를 인증하는 데 필요하지 않습니다. IdM 웹 UI에 인증에는 두 개의 호스트가 있으며, 둘 다 IdM 클라이언트여야 합니다.

- 브라우저가 실행 중인 시스템입니다. 시스템이 IdM 도메인 외부에 있을 수 있습니다.
- **httpd** 가 실행 중인 IdM 서버입니다.

다음 절차에서는 IdM 서버가 아닌 IdM 클라이언트에서 스마트 카드 인증을 구성한다고 가정합니다. 따라서 구성 스크립트를 생성하는 IdM 서버와 스크립트를 실행할 IdM 클라이언트라는 두 대의 컴퓨터가 필요합니다.

사전 요구 사항

- 스마트 카드 인증을 위해 IdM 서버 구성에 설명된 대로 IdM 서버가 스마트 카드 인증을 위해 구성되어 있습니다.
- IdM 서버와 IdM 클라이언트에 루트 액세스 권한이 있습니다.
- 루트 CA 인증서와 모든 중간 CA 인증서가 있습니다.
- 원격 사용자가 성공적으로 로그인할 수 **있도록 mkhomedir** 옵션을 사용하여 IdM 클라이언트를 설치했습니다. 홈 디렉터리를 생성하지 않으면 기본 로그인 위치는 디렉터리 구조의 루트입니다. /.

절차

1. IdM 서버에서 관리자 권한을 사용하여 **ipa-advise** 로 구성 스크립트를 생성합니다.

```
[root@server SmartCard]# kinit admin
[root@server SmartCard]# ipa-advise config-client-for-smart-card-auth > config-client-for-smart-card-auth.sh
```

   config-client-for-smart-card-auth.sh 스크립트는 다음 작업을 수행합니다.

   - 스마트 카드 데몬을 구성합니다.
   - 시스템 전체 신뢰 저장소를 설정합니다.
   - 사용자가 사용자 이름 및 암호 또는 스마트 카드로 인증할 수 있도록 SSSD(System Security Services Daemon)를 구성합니다. 스마트 카드 인증에 대한 SSSD 프로필 옵션에 대한 자세한 내용은 RHEL의 스마트 카드 인증 옵션을 참조하십시오.

2. IdM 서버에서 스크립트를 IdM 클라이언트 시스템에서 선택한 디렉터리에 복사합니다.

```
[root@server SmartCard]# scp config-client-for-smart-card-auth.sh
```

```
root@client.idm.example.com:/root/SmartCard/
Password:
config-client-for-smart-card-auth.sh        100%  2419      3.5MB/s   00:00
```

3. IdM 서버에서 이전 단계에서 사용된 것과 동일한 디렉터리에 편의를 위해 PEM 형식으로 CA 인증서 파일을 복사합니다.

```
[root@server SmartCard]# scp {rootca.pem,subca.pem,issuingca.pem}
root@client.idm.example.com:/root/SmartCard/
Password:
rootca.pem                    100%  1237     9.6KB/s   00:00
subca.pem                     100%  2514    19.6KB/s   00:00
issuingca.pem                 100%  2514    19.6KB/s   00:00
```

4. 클라이언트 시스템에서 스크립트를 실행하여 CA 인증서를 인수로 포함하는 PEM 파일을 추가합니다.

```
[root@client SmartCard]# kinit admin
[root@client SmartCard]# chmod +x config-client-for-smart-card-auth.sh
[root@client SmartCard]# ./config-client-for-smart-card-auth.sh rootca.pem subca.pem
issuingca.pem
Ticket cache:KEYRING:persistent:0:0
Default principal: admin@IDM.EXAMPLE.COM
[...]
Systemwide CA database updated.
The ipa-certupdate command was successful
```

참고

루트 CA 인증서를 하위 CA 인증서 전에 인수로 추가하고 CA 또는 하위 CA 인증서가 만료되지 않았는지 확인합니다.

이제 클라이언트는 스마트 카드 인증을 위해 구성됩니다.

2.4. ANSIBLE을 사용하여 스마트 카드 인증을 위해 IDM 클라이언트 구성

IdM 사용자가 스마트 카드로 인증할 수 있도록 **ansible-freeipa ipasmartcard_client** 모듈을 사용하여 특정 IdM(Identity Management) 클라이언트를 구성하려면 다음 절차를 따르십시오. 다음 절차를 실행하여 IdM에 액세스하는 데 다음을 사용하는 IdM 사용자의 스마트 카드 인증을 활성화합니다.

- **ssh** 프로토콜
  자세한 내용은 스마트 카드 인증을 사용하여 SSH 액세스 구성 을 참조하십시오.
- 콘솔 로그인
- GNOME 디스플레이 관리자(GDM)
- **su** 명령

참고

이 절차는 IdM 웹 UI를 인증하는 데 필요하지 않습니다. IdM 웹 UI에 인증에는 두 개의 호스트가 있으며, 둘 다 IdM 클라이언트여야 합니다.

- 브라우저가 실행 중인 시스템입니다. 시스템이 IdM 도메인 외부에 있을 수 있습니다.
- **httpd** 가 실행 중인 IdM 서버입니다.

사전 요구 사항

- Ansible을 사용하여 스마트 카드 인증을 위해 IdM 서버를 구성하는 데 설명된 대로 IdM 서버는 스마트 카드 인증을 위해 구성되어 있습니다.
- IdM 서버와 IdM 클라이언트에 루트 액세스 권한이 있습니다.
- 루트 CA 인증서, IdM CA 인증서 및 모든 중간 CA 인증서가 있습니다.
- 다음 요구 사항을 충족하도록 Ansible 제어 노드를 구성했습니다.
  - Ansible 버전 2.14 이상을 사용하고 있습니다.
  - Ansible 컨트롤러에 **ansible-freeipa** 패키지가 설치되어 있습니다.
  - 이 예제에서는 ***~/MyPlaybook/*** 디렉터리에서 IdM 서버의 FQDN(정규화된 도메인 이름)을 사용하여 Ansible 인벤토리 파일을 생성했다고 가정합니다.
  - 이 예제에서는 **secret.yml** Ansible 자격 증명 모음이 **ipaadmin_password** 를 저장하는 것으로 가정합니다.

절차

1. CA 인증서가 **DER** 와 같은 다른 형식의 파일에 저장된 경우 **PEM** 형식으로 변환합니다.

```
# openssl x509 -in <filename>.der -inform DER -out <filename>.pem -outform PEM
```

   IdM CA 인증서는 **PEM** 형식이며 **/etc/ipa/ca.crt** 파일에 있습니다.

2. 필요한 경우 **openssl x509** 유틸리티를 사용하여 **PEM** 형식의 파일 내용을 보고 **Issuer** 및 **Subject** 값이 올바른지 확인합니다.

```
# openssl x509 -noout -text -in root-ca.pem | more
```

3. Ansible 제어 노드에서 ***~/MyPlaybook/*** 디렉터리로 이동합니다.

```
$ cd ~/MyPlaybooks/
```

4. CA 인증서 전용 하위 디렉터리를 생성합니다.

```
$ mkdir SmartCard/
```

5. 편의를 위해 필요한 모든 인증서를 **~/MyPlaybook/SmartCard/** 디렉터리에 복사합니다. 예를 들면 다음과 같습니다.

```
# cp /tmp/root-ca.pem ~/MyPlaybooks/SmartCard/
# cp /tmp/intermediate-ca.pem ~/MyPlaybooks/SmartCard/
# cp /etc/ipa/ca.crt ~/MyPlaybooks/SmartCard/ipa-ca.crt
```

6. Ansible 인벤토리 파일에서 다음을 지정합니다.
   - 스마트 카드 인증을 위해 구성할 IdM 클라이언트입니다.
   - IdM 관리자 암호입니다.
   - 다음 순서로 CA 인증서의 경로입니다.
     - 루트 CA 인증서 파일
     - 중간 CA 인증서 파일
     - IdM CA 인증서 파일

   파일은 다음과 같습니다.

```
[ipaclients]
ipaclient1.example.com
ipaclient2.example.com

[ipaclients:vars]
ipaadmin_password=SomeADMINpassword
ipasmartcard_client_ca_certs=/home/<user_name>/MyPlaybooks/SmartCard/root-ca.pem,/home/<user_name>/MyPlaybooks/SmartCard/intermediate-ca.pem,/home/<user_name>/MyPlaybooks/SmartCard/ipa-ca.crt
```

7. 다음 콘텐츠를 사용하여 **install-smartcard-clients.yml** 플레이북을 생성합니다.

```
---
- name: Playbook to set up smart card authentication for an IdM client
  hosts: ipaclients
  become: true

  roles:
  - role: ipasmartcard_client
    state: present
```

8. 파일을 저장합니다.
9. Ansible 플레이북을 실행합니다. Playbook 및 인벤토리 파일을 지정합니다.

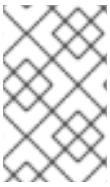
```
$ ansible-playbook --vault-password-file=password_file -v -i inventory install-smartcard-clients.yml
```

ipasmartcard_client Ansible 역할은 다음 작업을 수행합니다.

- 스마트 카드 데몬을 구성합니다.
- 시스템 전체 신뢰 저장소를 설정합니다.

- 사용자가 사용자 이름 및 암호 또는 스마트 카드로 인증할 수 있도록 SSSD(System Security Services Daemon)를 구성합니다. 스마트 카드 인증에 대한 SSSD 프로필 옵션에 대한 자세한 내용은 RHEL의 스마트 카드 인증 옵션을 참조하십시오.

인벤토리 파일의 **ipaclients** 섹션에 나열된 클라이언트가 스마트 카드 인증을 위해 구성됩니다.



참고

--mkhomedir 옵션을 사용하여 IdM 클라이언트를 설치한 경우 원격 사용자가 홈 디렉터리에 로그인할 수 있습니다. 그렇지 않으면 기본 로그인 위치는 디렉터리 구조의 루트입니다. /.

추가 리소스

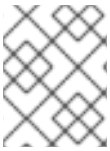
- **/usr/share/doc/ansible-freeipa/playbooks/** 디렉터리에서 **ipasmartcard_server** 역할을 사용하는 샘플 플레이북

2.5. IDM 웹 UI의 사용자 항목에 인증서 추가

IdM 웹 UI의 사용자 항목에 외부 인증서를 추가하려면 다음 절차를 따르십시오.

참고

전체 인증서를 업로드하는 대신, 인증서 매핑 데이터를 IdM의 사용자 항목에 업로드할 수도 있습니다. 전체 인증서 또는 인증서 매핑 데이터를 포함하는 사용자 항목을 해당 인증서 매핑 규칙과 함께 사용하여 시스템 관리자를 위한 스마트 카드 인증 구성을 용이하게 할 수 있습니다. 자세한 내용은 인증 구성을 위한 인증서 매핑 규칙을 참조하십시오.



참고

IdM 인증 기관에서 사용자 인증서를 발급한 경우 인증서가 이미 사용자 항목에 저장되어 있으며 이 절차를 따를 필요가 없습니다.

사전 요구 사항

- 사용자 항목에 추가할 인증서가 있습니다.

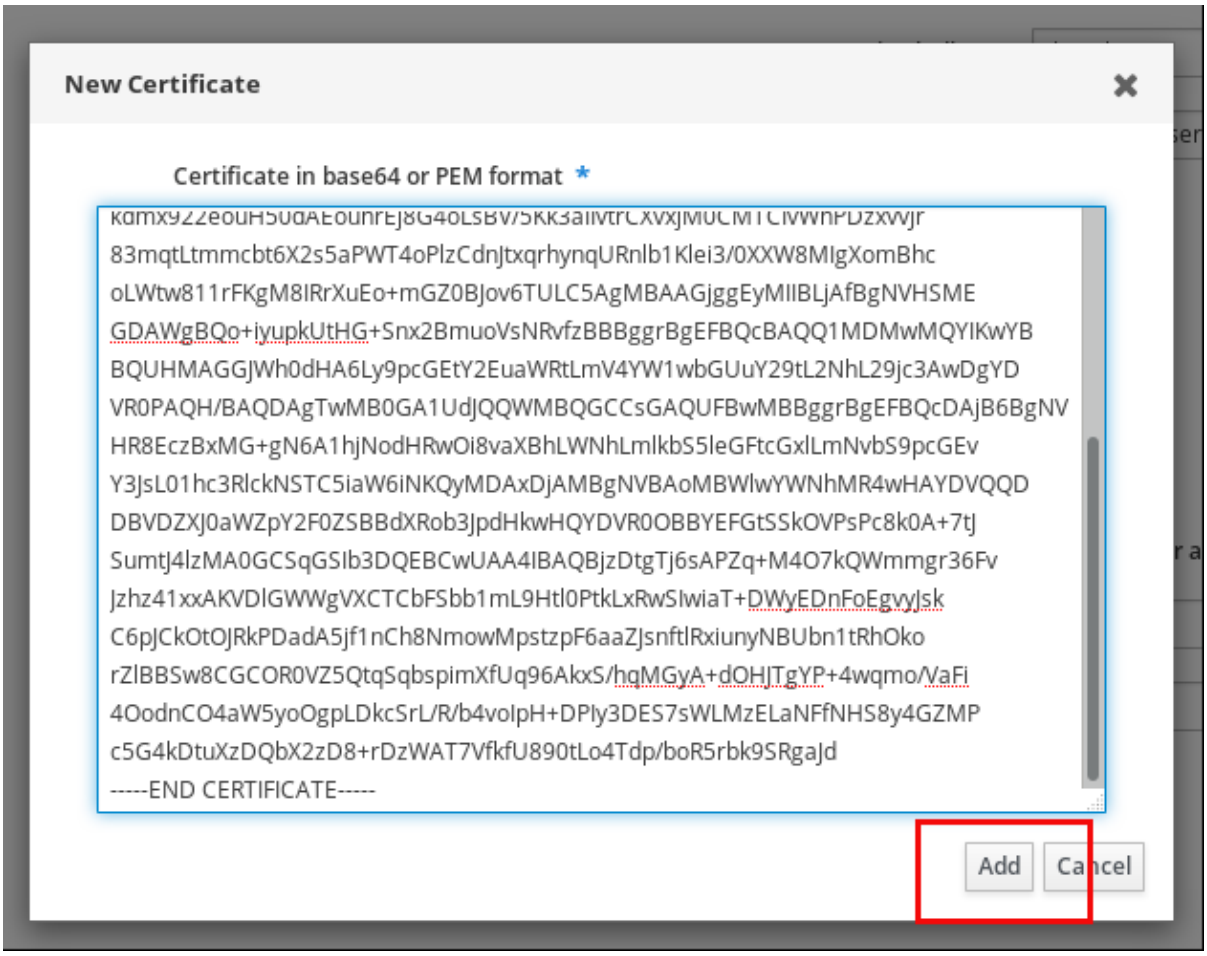
절차

1. 다른 사용자에게 인증서를 추가하려면 IdM 웹 UI에 관리자로 로그인합니다. 자체 프로필에 인증서를 추가하는 데 관리자의 자격 증명이 필요하지 않습니다.
2. **사용자 → 활성 사용자 → sc_user** 로 이동합니다.
3. **Certificate** (인증서) 옵션을 찾아 **Add(추가)**를 클릭합니다.
4. 명령줄 인터페이스에서 **cat** 유틸리티 또는 텍스트 편집기를 사용하여 **PEM** 형식으로 인증서를 표시합니다.

```
[user@client SmartCard]$ cat testuser.crt
```

5. CLI에서 인증서를 복사하여 웹 UI에서 연 창에 붙여넣습니다.
6. **추가**를 클릭합니다.

그림 2.1. IdM 웹 UI에서 새 인증서 추가



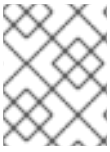
이제 **sc_user** 항목에 외부 인증서가 포함됩니다.

2.6. IDM CLI의 사용자 항목에 인증서 추가

IdM CLI의 사용자 항목에 외부 인증서를 추가하려면 다음 절차를 따르십시오.

참고

전체 인증서를 업로드하는 대신, 인증서 매핑 데이터를 IdM의 사용자 항목에 업로드할 수도 있습니다. 전체 인증서 또는 인증서 매핑 데이터를 포함하는 사용자 항목을 해당 인증서 매핑 규칙과 함께 사용하여 시스템 관리자를 위한 스마트 카드 인증 구성을 용이하게 할 수 있습니다. 자세한 내용은 인증 구성을 위한 인증서 매핑 규칙을 참조하십시오.



참고

IdM 인증 기관에서 사용자 인증서를 발급한 경우 인증서가 이미 사용자 항목에 저장되어 있으며 이 절차를 따를 필요가 없습니다.

사전 요구 사항

- 사용자 항목에 추가할 인증서가 있습니다.

절차

1. 다른 사용자에게 인증서를 추가하려면 IdM CLI에 관리자로 로그인합니다.
   -

```
[user@client SmartCard]$ kinit admin
```

자체 프로필에 인증서를 추가하는 데 관리자의 인증 정보가 필요하지 않습니다.

```
[user@client SmartCard]$ kinit sc_user
```

2. **ipa user-add-cert** 명령에서 필요한 형식인 헤더와 바닥글이 제거되고 한 줄로 연결된 인증서가 포함된 환경 변수를 생성합니다.

```
[user@client SmartCard]$ export CERT=`openssl x509 -outform der -in testuser.crt | base64 -w0 -`
```

testuser.crt 파일의 인증서는 **PEM** 형식이어야 합니다.

3. **ipa user-add-cert** 명령을 사용하여 sc_user의 프로필에 인증서를 추가합니다.

```
[user@client SmartCard]$ ipa user-add-cert sc_user --certificate=$CERT
```

이제 **sc_user** 항목에 외부 인증서가 포함됩니다.

2.7. 스마트 카드 관리 및 사용을 위한 도구 설치

스마트 카드를 구성하려면 인증서를 생성하고 스마트 카드에 저장할 수 있는 도구가 필요합니다.

다음을 수행해야 합니다.

- 인증서 관리에 도움이 되는 **gnutls-utils** 패키지를 설치합니다.
- 스마트 카드로 작업할 라이브러리 및 유틸리티 세트를 제공하는 **opensc** 패키지를 설치합니다.
- 스마트 카드 리더와 통신하는 **pcscd** 서비스를 시작합니다.

절차

1. **opensc** 및 **gnutls-utils** 패키지를 설치합니다.

```
# dnf -y install opensc gnutls-utils
```

2. **pcscd** 서비스를 시작합니다.

```
# systemctl start pcscd
```

pcscd 서비스가 실행 중인지 확인합니다.

2.8. 스마트 카드 준비 및 스마트 카드에 인증서와 키 업로드

다음 절차에 따라 구성하는 데 도움이 되는 **pkcs15-init** 도구를 사용하여 스마트 카드를 구성합니다.

- 스마트 카드 삭제
- 새로운 PIN 및 선택적 PIN 잠금 해제 키 (PUK) 설정
- 스마트 카드에서 새 슬롯 생성

- 인증서, 개인 키 및 공개 키 저장
- 필요한 경우 특정 스마트 카드에 따라 스마트 카드 설정을 잠금하려면 이러한 유형의 최종화가 필요합니다.

참고

pkcs15-init 툴은 모든 스마트 카드에서 작동하지 않을 수 있습니다. 사용 중인 스마트 카드로 작업하는 도구를 사용해야 합니다.

사전 요구 사항

- **pkcs15-init** 툴이 포함된 **opensc** 패키지가 설치됩니다.
  자세한 내용은 스마트 카드 관리 및 사용을 위한 툴 설치를 참조하십시오.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- 스마트 카드에 저장할 개인 키, 공개 키 및 인증서가 있습니다. 이 절차에서 **testuser.key**,**testuserpublic.key** 및 **testuser.crt** 는 개인 키, 공개 키 및 인증서에 사용되는 이름입니다.
- 현재 스마트 카드 사용자 Pin and Security Officer Pin (SO-PIN)이 있습니다.

절차

1. 스마트 카드를 지우고 PIN으로 인증하십시오:

```
$ pkcs15-init --erase-card --use-default-transport-keys
Using reader with a card: Reader name
PIN [Security Officer PIN] required.
Please enter PIN [Security Officer PIN]:
```

   카드가 지워졌습니다.

2. 스마트 카드를 초기화하고 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK를 설정합니다.

```
$ pkcs15-init --create-pkcs15 --use-default-transport-keys \
   --pin 963214 --puk 321478 --so-pin 65498714 --so-puk 784123
Using reader with a card: Reader name
```

   pcks15-init 툴은 스마트 카드에 새 슬롯을 생성합니다.

3. 슬롯의 레이블 및 인증 ID를 설정합니다.

```
$ pkcs15-init --store-pin --label testuser \
   --auth-id 01 --so-pin 65498714 --pin 963214 --puk 321478
Using reader with a card: Reader name
```

   레이블은 사람이 읽을 수 있는 값(이 경우 **testuser**)으로 설정됩니다. **auth-id** 는 16진수 2개 값이어야 합니다(이 경우 **01** 로 설정됨).

4. 스마트 카드의 새 슬롯에 개인 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-private-key testuser.key --label testuser_key \
   --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```

참고

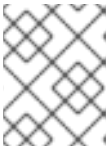
--id 에 지정하는 값은 개인 키를 저장하고 다음 단계에 인증서를 저장할 때 동일해야 합니다. 도구에서 더 복잡한 값을 계산하므로 **--id** 에 대해 자체 값을 지정하는 것이 좋습니다.

5. 스마트 카드의 새 슬롯에 인증서를 저장하고 레이블을 지정합니다.

```
$ pkcs15-init --store-certificate testuser.crt --label testuser_crt \
   --auth-id 01 --id 01 --format pem --pin 963214
Using reader with a card: Reader name
```

6. (선택 사항) 스마트 카드의 새 슬롯에 공개 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-public-key testuserpublic.key
   --label testuserpublic_key --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```



참고

공개 키가 개인 키 또는 인증서에 해당하는 경우 개인 키 또는 인증서의 ID와 동일한 ID를 지정합니다.

7. (선택 사항) 특정 스마트 카드를 사용하려면 설정을 잠금하여 카드를 종료해야 합니다.

```
$ pkcs15-init -F
```

이 단계에서는 새로 생성된 슬롯에 인증서, 개인 키 및 공개 키가 포함되어 있습니다. 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK도 생성했습니다.

2.9. 스마트 카드로 IDM에 로그인

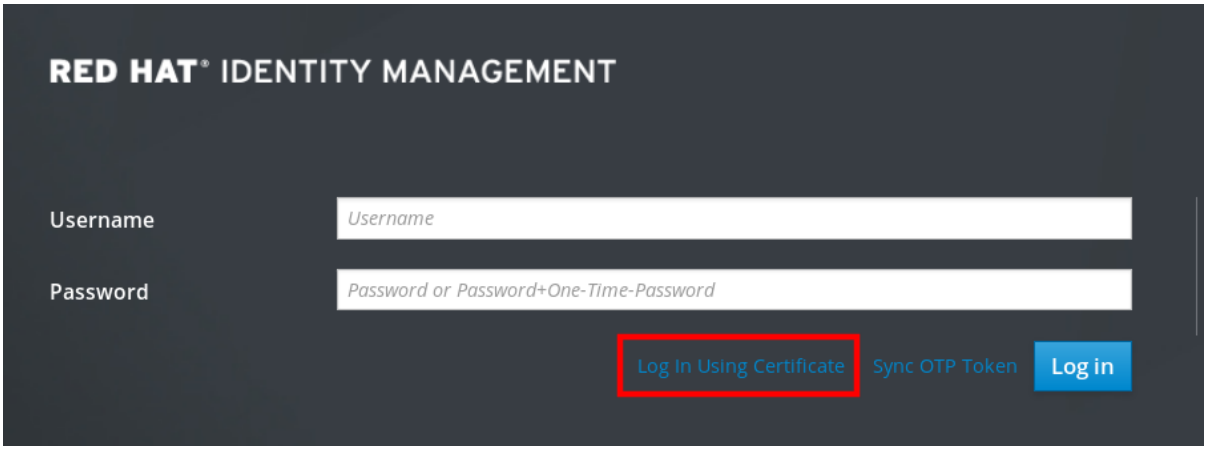
IdM 웹 UI에 로그인하는 데 스마트 카드를 사용하려면 다음 절차를 따르십시오.

사전 요구 사항

- 웹 브라우저는 스마트 카드 인증을 사용하도록 구성됩니다.
- IdM 서버는 스마트 카드 인증을 위해 구성됩니다.
- 스마트 카드에 설치된 인증서는 IdM 서버에서 발행하거나 IdM의 사용자 항목에 추가되었습니다.
- 스마트 카드 잠금을 해제하는 데 필요한 pin을 알고 있습니다.
- 스마트 카드가 리더에 삽입되었습니다.

절차

1. 브라우저에서 IdM 웹 UI를 엽니다.
2. **인증서를 사용하여 로그인을** 클릭합니다.

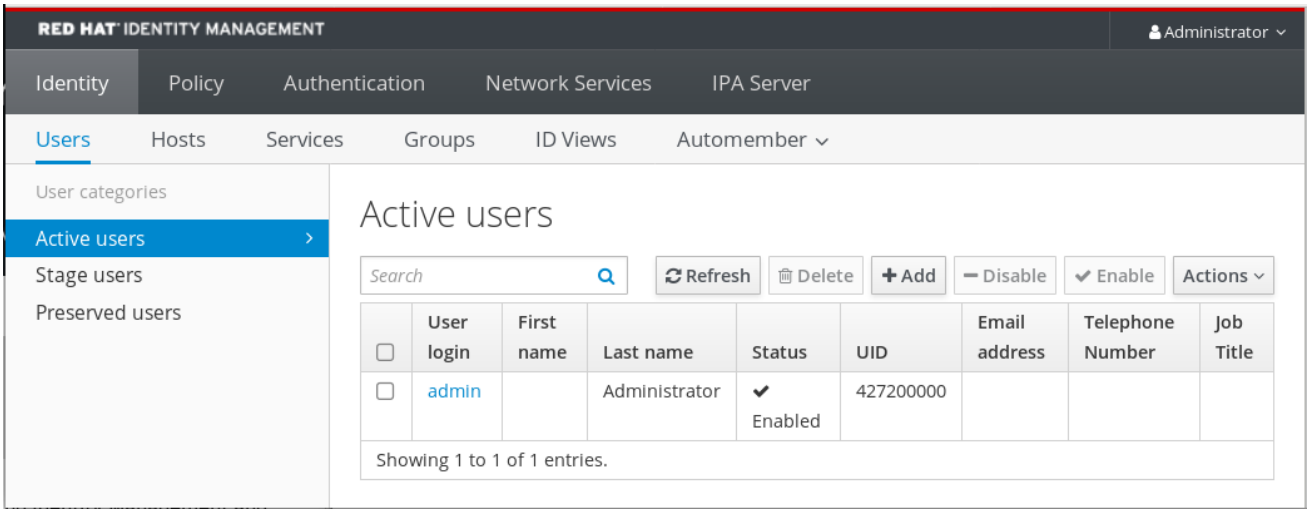


3. **Password Required(암호 필요)** 대화 상자가 열리면 PIN을 추가하여 스마트 카드 잠금을 해제하고 **OK** (확인) 버튼을 클릭합니다.
사용자 식별 요청 대화 상자가 열립니다.

   스마트 카드에 인증서가 두 개 이상 포함된 경우 아래의 드롭다운 목록에서 인증에 사용할 인증서를 **선택하십시오. 식별으로 표시할 인증서를** 선택하십시오.

4. **OK(확인)** 버튼을 클릭합니다.

이제 IdM 웹 UI에 성공적으로 로그인되었습니다.



2.10. IDM 클라이언트에서 스마트 카드 인증을 사용하여 GDM에 로그인

GNOME 데스크탑 관리자(GDM)에는 인증이 필요합니다. 암호를 사용할 수 있지만 인증에 스마트 카드를 사용할 수도 있습니다.

smart 카드 인증을 사용하여 GDM에 액세스하려면 다음 절차를 따르십시오.

사전 요구 사항

- 이 시스템은 스마트 카드 인증을 위해 구성되었습니다. 자세한 내용은 스마트 카드 인증을 위한 IdM 클라이언트 구성을 참조하십시오.
- 스마트 카드에는 인증서와 개인 키가 포함되어 있습니다.

- 사용자 계정은 IdM 도메인의 멤버입니다.
- 스마트 카드의 인증서는 다음을 통해 사용자 항목에 매핑됩니다.
  - 특정 사용자 항목에 인증서 할당. 자세한 내용은 IdM 웹 UI의 사용자 항목에 인증서 추가 또는 IdM CLI의 사용자 항목에 인증서 추가를 참조하십시오.
  - 계정에 적용되는 인증서 매핑 데이터입니다. 자세한 내용은 스마트 카드에 대한 인증을 구성하기 위한 인증서 매핑 규칙을 참조하십시오.

절차

1. 리더에 스마트 카드를 삽입합니다.
2. 스마트 카드 PIN을 입력합니다.
3. **Sign In** (로그인)을 클릭합니다.

RHEL 시스템에 성공적으로 로그인했으며 IdM 서버에서 TGT를 제공합니다.

검증 단계

- **터미널** 창에서 **klist** 를 입력하고 결과를 확인합니다.

```
$ klist
Ticket cache: KEYRING:persistent:1358900015:krb_cache_TObtNMd
Default principal: example.user@REDHAT.COM

Valid starting       Expires              Service principal
04/20/2020 13:58:24  04/20/2020 23:58:24  krbtgt/EXAMPLE.COM@EXAMPLE.COM
 renew until 04/27/2020 08:58:15
```

2.11. SU 명령으로 스마트 카드 인증 사용

다른 사용자로 변경하려면 인증이 필요합니다. 암호 또는 인증서를 사용할 수 있습니다. **su** 명령과 함께 스마트 카드를 사용하려면 다음 절차를 따르십시오. **su** 명령을 입력한 후 스마트 카드 PIN을 묻는 메시지가 표시됩니다.

사전 요구 사항

- 스마트 카드 인증을 위해 IdM 서버 및 클라이언트가 구성되어 있습니다.
  - 스마트 카드 인증을 위한 IdM 서버 구성을 참조하십시오.
  - 스마트 카드 인증을 위한 IdM 클라이언트 구성을 참조하십시오.
- 스마트 카드에는 인증서와 개인 키가 포함되어 있습니다. 스마트 카드의 인증서 저장을 참조하십시오.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.

절차

- 터미널 창에서 **su** 명령을 사용하여 다른 사용자로 변경합니다.

```
$ su - example.user
PIN for smart_card
```

구성이 올바르면 스마트 카드 Pin을 입력하라는 메시지가 표시됩니다.

3장. IDM의 스마트 카드 인증을 위해 ADCS에서 발급한 인증서 구성

AD(Active Directory) 인증서 서비스에서 인증서를 발급한 사용자에 대해 IdM에서 스마트 카드 인증을 구성하려면 다음을 수행합니다.

- 배포는 IdM(Identity Management)과 AD(Active Directory) 간의 교차 포리스트 신뢰성을 기반으로 합니다.
- 계정이 AD에 저장된 사용자에 대해 스마트 카드 인증을 허용하려고 합니다.
- 인증서는 ADCS(Active Directory Certificate Services)에 생성 및 저장됩니다.

스마트 카드 인증 개요는 스마트 카드 인증 이해를 참조하십시오.

구성은 다음 단계에서 수행됩니다.

- Active Directory에서 IdM 서버 및 클라이언트로 CA 및 사용자 인증서 복사
- ADCS 인증서를 사용하여 스마트 카드 인증을 위해 IdM 서버 및 클라이언트 구성
- 인증서 및 개인 키를 스마트 카드에 저장할 수 있도록 Pdatabind (PKCS#12) 파일을 변환
- sssd.conf 파일에서 타임아웃 구성
- 스마트 카드 인증을 위한 인증서 매핑 규칙 생성

사전 요구 사항

- IdM(Identity Management) 및 AD(Active Directory) 신뢰가 설치되어 있습니다.
  자세한 내용은 IdM과 AD 간의 신뢰 설치를 참조하십시오.
- ADCS(Active Directory Certificate Services)가 설치되어 사용자를 위한 인증서가 생성됩니다.

3.1. 신뢰 구성 및 인증서 사용에 필요한 WINDOWS SERVER 설정

Windows Server에서 다음을 구성해야 합니다.

- ADCS(Active Directory Certificate Services)가 설치되어 있습니다.
- 인증 기관이 생성됨
- [선택 사항] 인증 기관 웹 등록을 사용하는 경우 IIS(Internet Information Services)를 구성해야 합니다.

인증서를 내보냅니다.

- 키에는 **2048** 비트 이상이 있어야 합니다
- 개인 키 포함
- 다음 형식의 인증서가 필요합니다. 개인 정보 **교환기 - PKCS #12(.PFX)**
  - 인증서 개인 정보 보호 활성화

3.2. SFTP를 사용하여 ACTIVE DIRECTORY에서 인증서 복사

스마트 카드 인증 기능을 사용하려면 다음 인증서 파일을 복사해야 합니다.

- **CER** 형식의 루트 CA 인증서: IdM 서버의 **adcs-winserver-ca.cer** 입니다.
- **PFX** 형식의 개인 키가 있는 사용자 인증서: IdM 클라이언트의 **aduser1.pfx**.

참고

이 절차에서는 SSH 액세스가 허용되어야 합니다. SSH를 사용할 수 없는 경우 사용자는 AD 서버에서 IdM 서버 및 클라이언트에 파일을 복사해야 합니다.

절차

1. **IdM 서버에서** 연결하고 **adcs-winserver-ca.cer** 루트 인증서를 IdM 서버에 복사합니다.

```
root@idmserver ~]# sftp Administrator@winserver.ad.example.com
Administrator@winserver.ad.example.com's password:
Connected to Administrator@winserver.ad.example.com.
sftp> cd <Path to certificates>
sftp> ls
adcs-winserver-ca.cer    aduser1.pfx
sftp>
sftp> get adcs-winserver-ca.cer
Fetching <Path to certificates>/adcs-winserver-ca.cer to adcs-winserver-ca.cer
<Path to certificates>/adcs-winserver-ca.cer                100%  1254    15KB/s 00:00
sftp quit
```

2. **IdM 클라이언트에서** 연결하고 **aduser1.pfx** 사용자 인증서를 클라이언트에 복사합니다.

```
[root@client1 ~]# sftp Administrator@winserver.ad.example.com
Administrator@winserver.ad.example.com's password:
Connected to Administrator@winserver.ad.example.com.
sftp> cd /<Path to certificates>
sftp> get aduser1.pfx
Fetching <Path to certificates>/aduser1.pfx to aduser1.pfx
<Path to certificates>/aduser1.pfx                 100%  1254    15KB/s 00:00
sftp quit
```

이제 CA 인증서가 IdM 서버에 저장되고 사용자 인증서가 클라이언트 시스템에 저장됩니다.

3.3. ADCS 인증서를 사용하여 스마트 카드 인증을 위해 IDM 서버 및 클라이언트 구성

IdM 환경에서 스마트 카드 인증을 사용할 수 있도록 IdM(ID 관리) 서버 및 클라이언트를 구성해야 합니다. IdM에는 필요한 모든 변경을 수행하는 **ipa-advise** 스크립트가 포함되어 있습니다.

- 필요한 패키지 설치
- IdM 서버 및 클라이언트 구성
- CA 인증서를 예상 위치에 복사

IdM 서버에서 **ipa-advise** 를 실행할 수 있습니다.

스마트 카드 인증을 위해 서버 및 클라이언트를 구성하려면 다음 절차를 따르십시오.

- IdM 서버에서 다음을 수행합니다. 스마트 카드 인증을 위해 IdM 서버를 구성하기 위해 **ipa-advise** 스크립트 준비.
- IdM 서버에서 다음을 수행합니다. 스마트 카드 인증을 위해 IdM 클라이언트를 구성하기 위해 **ipa-advise** 스크립트를 준비합니다.
- IdM 서버에서 다음을 수행합니다. AD 인증서를 사용하여 IdM 서버에 **ipa-advise** 서버 스크립트를 적용합니다.
- 클라이언트 스크립트를 IdM 클라이언트 시스템으로 이동.
- IdM 클라이언트에서 다음을 수행합니다. AD 인증서를 사용하여 IdM 클라이언트에 **ipa-advise** 클라이언트 스크립트를 적용합니다.

사전 요구 사항

- 인증서가 IdM 서버에 복사되었습니다.
- Kerberos 티켓을 받습니다.
- 관리 권한이 있는 사용자로 로그인합니다.

절차

1. IdM 서버에서 **ipa-advise** 스크립트를 사용하여 클라이언트를 구성합니다.

```
[root@idmserver ~]# ipa-advise config-client-for-smart-card-auth > sc_client.sh
```

2. IdM 서버에서 **ipa-advise** 스크립트를 사용하여 서버를 구성합니다.

```
[root@idmserver ~]# ipa-advise config-server-for-smart-card-auth > sc_server.sh
```

3. IdM 서버에서 스크립트를 실행합니다.

```
[root@idmserver ~]# sh -x sc_server.sh adcs-winserver-ca.cer
```

   - IdM Apache HTTP 서버를 구성합니다.
   - KDC(키 배포 센터)에서 Kerberos(PKINIT)의 초기 인증에 대한 공개 키 암호화를 활성화합니다.
   - 스마트 카드 권한 부여 요청을 수락하도록 IdM 웹 UI를 구성합니다.

4. **sc_client.sh** 스크립트를 클라이언트 시스템에 복사합니다.

```
[root@idmserver ~]# scp sc_client.sh root@client1.idm.example.com:/root
Password:
sc_client.sh                  100%  2857   1.6MB/s   00:00
```

5. Windows 인증서를 클라이언트 시스템에 복사합니다.

```
[root@idmserver ~]# scp adcs-winserver-ca.cer root@client1.idm.example.com:/root
Password:
adcs-winserver-ca.cer                 100%  1254   952.0KB/s   00:00
```

6. 클라이언트 시스템에서 클라이언트 스크립트를 실행합니다.

```
[root@idmclient1 ~]# sh -x sc_client.sh adcs-winserver-ca.cer
```

CA 인증서는 IdM 서버 및 클라이언트 시스템에서 올바른 형식으로 설치되고 다음 단계는 사용자 인증서를 스마트 카드 자체에 복사하는 것입니다.

3.4. PFX 파일 변환

PFX (PKCS#12) 파일을 스마트 카드에 저장하기 전에 다음을 수행해야 합니다.

- 파일을 PEM 형식으로 변환합니다.
- 개인 키와 인증서를 두 개의 다른 파일로 추출합니다

사전 요구 사항

- PFX 파일은 IdM 클라이언트 시스템에 복사됩니다.

절차

1. IdM 클라이언트에서 PEM 형식으로 변경합니다.

```
[root@idmclient1 ~]# openssl pkcs12 -in aduser1.pfx -out aduser1_cert_only.pem -clcerts -
nodes
Enter Import Password:
```

2. 키를 별도의 파일에 추출합니다.

```
[root@idmclient1 ~]# openssl pkcs12 -in adduser1.pfx -nocerts -out adduser1.pem >
aduser1.key
```

3. 공용 인증서를 별도의 파일에 추출합니다.

```
[root@idmclient1 ~]# openssl pkcs12 -in adduser1.pfx -clcerts -nokeys -out
aduser1_cert_only.pem > aduser1.crt
```

이제 **aduser1.key 및 aduser1. crt** 를 스마트 카드로 저장할 수 있습니다.

3.5. 스마트 카드 관리 및 사용을 위한 도구 설치

스마트 카드를 구성하려면 인증서를 생성하고 스마트 카드에 저장할 수 있는 도구가 필요합니다.

다음을 수행해야 합니다.

- 인증서 관리에 도움이 되는 **gnutls-utils** 패키지를 설치합니다.
- 스마트 카드로 작업할 라이브러리 및 유틸리티 세트를 제공하는 **opensc** 패키지를 설치합니다.

- 스마트 카드 리더와 통신하는 **pcscd** 서비스를 시작합니다.

절차

1. **opensc** 및 **gnutls-utils** 패키지를 설치합니다.

```
# dnf -y install opensc gnutls-utils
```

2. **pcscd** 서비스를 시작합니다.

```
# systemctl start pcscd
```

pcscd 서비스가 실행 중인지 확인합니다.

3.6. 스마트 카드 준비 및 스마트 카드에 인증서와 키 업로드

다음 절차에 따라 구성하는 데 도움이 되는 **pkcs15-init** 도구를 사용하여 스마트 카드를 구성합니다.

- 스마트 카드 삭제
- 새로운 PIN 및 선택적 PIN 잠금 해제 키 (PUK) 설정
- 스마트 카드에서 새 슬롯 생성
- 인증서, 개인 키 및 공개 키 저장
- 필요한 경우 특정 스마트 카드에 따라 스마트 카드 설정을 잠금하려면 이러한 유형의 최종화가 필요합니다.

참고

pkcs15-init 툴은 모든 스마트 카드에서 작동하지 않을 수 있습니다. 사용 중인 스마트 카드로 작업하는 도구를 사용해야 합니다.

사전 요구 사항

- **pkcs15-init** 툴이 포함된 **opensc** 패키지가 설치됩니다.
  자세한 내용은 스마트 카드 관리 및 사용을 위한 툴 설치를 참조하십시오.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- 스마트 카드에 저장할 개인 키, 공개 키 및 인증서가 있습니다. 이 절차에서 **testuser.key**,**testuserpublic.key** 및 **testuser.crt** 는 개인 키, 공개 키 및 인증서에 사용되는 이름입니다.
- 현재 스마트 카드 사용자 Pin and Security Officer Pin (SO-PIN)이 있습니다.

절차

1. 스마트 카드를 지우고 PIN으로 인증하십시오:

```
$ pkcs15-init --erase-card --use-default-transport-keys
Using reader with a card: Reader name
PIN [Security Officer PIN] required.
```

```
Please enter PIN [Security Officer PIN]:
```

카드가 지워졌습니다.

2. 스마트 카드를 초기화하고 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK를 설정합니다.

```
$ pkcs15-init --create-pkcs15 --use-default-transport-keys \
   --pin 963214 --puk 321478 --so-pin 65498714 --so-puk 784123
Using reader with a card: Reader name
```

pcks15-init 툴은 스마트 카드에 새 슬롯을 생성합니다.

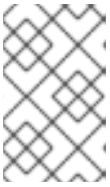
3. 슬롯의 레이블 및 인증 ID를 설정합니다.

```
$ pkcs15-init --store-pin --label testuser \
   --auth-id 01 --so-pin 65498714 --pin 963214 --puk 321478
Using reader with a card: Reader name
```

레이블은 사람이 읽을 수 있는 값(이 경우 **testuser**)으로 설정됩니다. **auth-id** 는 16진수 2개 값이어야 합니다(이 경우 **01** 로 설정됨).

4. 스마트 카드의 새 슬롯에 개인 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-private-key testuser.key --label testuser_key \
   --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```



참고

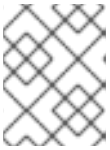
--id 에 지정하는 값은 개인 키를 저장하고 다음 단계에 인증서를 저장할 때 동일해야 합니다. 도구에서 더 복잡한 값을 계산하므로 **--id** 에 대해 자체 값을 지정하는 것이 좋습니다.

5. 스마트 카드의 새 슬롯에 인증서를 저장하고 레이블을 지정합니다.

```
$ pkcs15-init --store-certificate testuser.crt --label testuser_crt \
   --auth-id 01 --id 01 --format pem --pin 963214
Using reader with a card: Reader name
```

6. (선택 사항) 스마트 카드의 새 슬롯에 공개 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-public-key testuserpublic.key
   --label testuserpublic_key --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```



참고

공개 키가 개인 키 또는 인증서에 해당하는 경우 개인 키 또는 인증서의 ID와 동일한 ID를 지정합니다.

7. (선택 사항) 특정 스마트 카드를 사용하려면 설정을 잠금하여 카드를 종료해야 합니다.

```
$ pkcs15-init -F
```

-

  이 단계에서는 새로 생성된 슬롯에 인증서, 개인 키 및 공개 키가 포함되어 있습니다. 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK도 생성했습니다.

3.7. SSSD.CONF에서 시간 제한 설정

스마트 카드 인증서를 사용한 인증은 SSSD에서 사용하는 기본 시간 초과보다 오래 걸릴 수 있습니다. 시간 초과 만료는 다음과 같이 발생할 수 있습니다.

- 느린 리더
- 전달은 물리적 장치를 가상 환경으로 구성합니다.
- 스마트 카드에 너무 많은 인증서가 저장되어 있습니다
- OCSP를 사용하여 인증서를 확인하는 경우 OCSP (Online Certificate Status Protocol) 응답기에서 느린 응답

이 경우 **sssd.conf** 파일에서 다음 시간 제한을 초과할 수 있습니다(예: 60초).

- **p11_child_timeout**
- **krb5_auth_timeout**

사전 요구 사항

- root로 로그인해야 합니다.

절차

1. **sssd.conf** 파일을 엽니다.

```
[root@idmclient1 ~]# vim /etc/sssd/sssd.conf
```

2. **p11_child_timeout** 값을 변경합니다.

```
[pam]
p11_child_timeout = 60
```

3. **krb5_auth_timeout** 값을 변경합니다.

```
[domain/IDM.EXAMPLE.COM]
krb5_auth_timeout = 60
```

4. 설정을 저장합니다.

이제 시간 초과로 인증에 실패하기 전에 스마트 카드와의 상호 작용을 1분(60초) 동안 실행할 수 있습니다.

3.8. 스마트 카드 인증을 위한 인증서 매핑 규칙 생성

AD(Active Directory) 및 IdM(Identity Management)에 계정이 있는 사용자에 대해 하나의 인증서를 사용하려는 경우 IdM 서버에서 인증서 매핑 규칙을 생성할 수 있습니다.

이러한 규칙을 만든 후 사용자는 두 도메인에서 스마트 카드로 인증할 수 있습니다.

인증서 매핑 규칙에 대한 자세한 내용은 인증 구성을 위한 인증서 매핑 규칙을 참조하십시오.

4장. 인증을 구성하기 위한 인증서 매핑 규칙

다음 시나리오에서는 인증서 매핑 규칙을 구성해야 할 수 있습니다.

- 인증서는 IdM 도메인이 신뢰 관계에 있는 AD(Active Directory)의 인증서 시스템에서 발급되었습니다.
- 인증서는 외부 인증 기관에서 발급했습니다.
- IdM 환경은 스마트 카드를 사용하는 많은 사용자에게 큽니다. 이 경우 전체 인증서를 추가하는 것은 복잡할 수 있습니다. 제목과 발행자는 대부분의 시나리오에서 예측할 수 있으므로 전체 인증서보다 미리 추가하기가 더 쉽습니다.

시스템 관리자는 인증서 매핑 규칙을 생성하고 인증서를 특정 사용자에게 발급하기 전에도 사용자 항목에 인증서 매핑 데이터를 추가할 수 있습니다. 인증서가 발급되면 전체 인증서가 사용자 항목에 아직 업로드되지 않았더라도 사용자는 인증서를 사용하여 로그인할 수 있습니다.

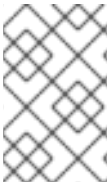
또한 인증서가 정기적으로 갱신되므로 인증서 매핑 규칙이 관리 오버헤드를 줄일 수 있습니다. 사용자의 인증서가 갱신되면 관리자는 사용자 항목을 업데이트할 필요가 없습니다. 예를 들어 매핑이 **주체** 및 **발급자** 값을 기반으로 하고 새 인증서에 이전 인증서와 동일한 제목과 발급자가 있는 경우 매핑이 계속 적용됩니다. 반대로 전체 인증서가 사용된 경우 관리자는 새 인증서를 사용자 항목에 업로드하여 이전 인증서를 교체해야 합니다.

인증서 매핑을 설정하려면 다음을 수행합니다.

1. 관리자는 인증서 매핑 데이터 또는 전체 인증서를 사용자 계정으로 로드해야 합니다.
2. 관리자는 인증서의 정보와 일치하는 인증서 매핑 데이터 항목이 포함된 사용자에 대해 IdM에 성공적으로 로그인할 수 있도록 인증서 매핑 규칙을 생성해야 합니다.

인증서 매핑 규칙이 생성되면 최종 사용자가 인증서를 제공할 때 파일 시스템 또는 스마트 카드에 저장된 인증이 성공적으로 수행됩니다.

매핑 규칙을 구성하는 개별 구성 요소와 이를 가져와서 사용하는 방법에 대한 자세한 내용은 IdM의 ID 매핑 규칙 구성 및 일치하는 규칙에서 사용할 수 있도록 인증서의 발행자 가져오기를 참조하십시오.



참고

인증서 매핑 규칙은 인증서를 사용하는 사용 사례에 따라 달라질 수 있습니다. 예를 들어 인증서가 포함된 SSH를 사용하는 경우 인증서에서 공개 키를 추출하려면 전체 인증서가 있어야 합니다.

5장. 중앙 관리 사용자를 위해 웹 콘솔을 사용하여 스마트 카드 인증 설정

다음을 통해 중앙에서 관리하는 사용자를 위해 RHEL 웹 콘솔에서 스마트 카드 인증을 구성합니다.

- IdM (Identity Management)
- Identity Management를 사용하여 가장 간 신뢰에 연결된 Active Directory

중요

> Smart card authentication does not elevate administrative privileges yet and the web console opens in the web browser in the read-only mode.

> You can run administrative commands in the built-in terminal with `sudo`.

사전 요구 사항

- 스마트 카드 인증을 사용하려는 시스템은 Active Directory 또는 Identity Management 도메인의 멤버여야 합니다.
  웹 콘솔을 사용하여 RHEL 8 시스템을 도메인에 추가하는 방법에 대한 자세한 내용은 웹 콘솔을 사용하여 IdM 도메인에 RHEL 8 시스템 참여를 참조하십시오.
- 스마트 카드 인증에 사용되는 인증서는 ID 관리 또는 Active Directory의 특정 사용자와 연결되어 있어야 합니다.
  ID 관리에서 사용자와 인증서를 연결하는 방법에 대한 자세한 내용은 IdM 웹 UI의 사용자 항목에 인증서 추가를 참조하거나 IdM CLI의 사용자 항목에 인증서 추가를 참조하십시오.

5.1. 중앙 관리 사용자를 위한 스마트 카드 인증

스마트 카드는 물리적 장치이며 카드에 저장된 인증서를 사용하여 개인 인증을 제공할 수 있습니다. 개인 인증은 사용자 암호와 동일한 방식으로 스마트 카드를 사용할 수 있음을 의미합니다.

개인 키 및 인증서 형식으로 스마트 카드에 사용자 자격 증명을 저장할 수 있습니다. 특수 소프트웨어 및 하드웨어가 액세스하는 데 사용됩니다. 스마트 카드를 reader 또는 USB 소켓에 삽입하고 암호를 제공하는 대신 스마트 카드의 Pin 코드를 제공합니다.

IdM(Identity Management)은 다음을 사용하여 스마트 카드 인증을 지원합니다.

- IdM 인증 기관에서 발급한 사용자 인증서. 자세한 내용은 스마트 카드 인증을 위한 ID 관리 구성을 참조하십시오.
- ADCS(Active Directory 인증서 서비스) 인증 기관에서 발급한 사용자 인증서입니다. 자세한 내용은 IdM에서 스마트 카드 인증을 위해 ADCS에서 발급한 인증서 구성을 참조하십시오.

참고

스마트 카드 인증을 사용하려면 하드웨어 요구 사항을 참조하십시오. RHEL8+에서 스마트 카드 지원

5.2. 스마트 카드 관리 및 사용을 위한 도구 설치

스마트 카드를 구성하려면 인증서를 생성하고 스마트 카드에 저장할 수 있는 도구가 필요합니다.

다음을 수행해야 합니다.

- 인증서 관리에 도움이 되는 **gnutls-utils** 패키지를 설치합니다.
- 스마트 카드로 작업할 라이브러리 및 유틸리티 세트를 제공하는 **opensc** 패키지를 설치합니다.
- 스마트 카드 리더와 통신하는 **pcscd** 서비스를 시작합니다.

절차

1. **opensc** 및 **gnutls-utils** 패키지를 설치합니다.

```
# dnf -y install opensc gnutls-utils
```

2. **pcscd** 서비스를 시작합니다.

```
# systemctl start pcscd
```

pcscd 서비스가 실행 중인지 확인합니다.

5.3. 스마트 카드 준비 및 스마트 카드에 인증서와 키 업로드

다음 절차에 따라 구성하는 데 도움이 되는 **pkcs15-init** 도구를 사용하여 스마트 카드를 구성합니다.

- 스마트 카드 삭제
- 새로운 PIN 및 선택적 PIN 잠금 해제 키 (PUK) 설정
- 스마트 카드에서 새 슬롯 생성
- 인증서, 개인 키 및 공개 키 저장
- 필요한 경우 특정 스마트 카드에 따라 스마트 카드 설정을 잠금하려면 이러한 유형의 최종화가 필요합니다.

참고

pkcs15-init 툴은 모든 스마트 카드에서 작동하지 않을 수 있습니다. 사용 중인 스마트 카드로 작업하는 도구를 사용해야 합니다.

사전 요구 사항

- **pkcs15-init** 툴이 포함된 **opensc** 패키지가 설치됩니다.
  자세한 내용은 스마트 카드 관리 및 사용을 위한 툴 설치를 참조하십시오.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- 스마트 카드에 저장할 개인 키, 공개 키 및 인증서가 있습니다. 이 절차에서 **testuser.key**,**testuserpublic.key** 및 **testuser.crt** 는 개인 키, 공개 키 및 인증서에 사용되는 이름입니다.
- 현재 스마트 카드 사용자 Pin and Security Officer Pin (SO-PIN)이 있습니다.

절차

1. 스마트 카드를 지우고 PIN으로 인증하십시오:

```
$ pkcs15-init --erase-card --use-default-transport-keys
Using reader with a card: Reader name
PIN [Security Officer PIN] required.
Please enter PIN [Security Officer PIN]:
```

   카드가 지워졌습니다.

2. 스마트 카드를 초기화하고 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK를 설정합니다.

```
$ pkcs15-init --create-pkcs15 --use-default-transport-keys \
   --pin 963214 --puk 321478 --so-pin 65498714 --so-puk 784123
Using reader with a card: Reader name
```

   pcks15-init 툴은 스마트 카드에 새 슬롯을 생성합니다.

3. 슬롯의 레이블 및 인증 ID를 설정합니다.

```
$ pkcs15-init --store-pin --label testuser \
   --auth-id 01 --so-pin 65498714 --pin 963214 --puk 321478
Using reader with a card: Reader name
```

   레이블은 사람이 읽을 수 있는 값(이 경우 **testuser**)으로 설정됩니다. **auth-id** 는 16진수 2개 값이어야 합니다(이 경우 **01** 로 설정됨).

4. 스마트 카드의 새 슬롯에 개인 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-private-key testuser.key --label testuser_key \
   --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```

참고

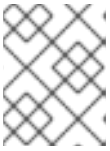
--id 에 지정하는 값은 개인 키를 저장하고 다음 단계에 인증서를 저장할 때 동일해야 합니다. 도구에서 더 복잡한 값을 계산하므로 **--id** 에 대해 자체 값을 지정하는 것이 좋습니다.

5. 스마트 카드의 새 슬롯에 인증서를 저장하고 레이블을 지정합니다.

```
$ pkcs15-init --store-certificate testuser.crt --label testuser_crt \
   --auth-id 01 --id 01 --format pem --pin 963214
Using reader with a card: Reader name
```

6. (선택 사항) 스마트 카드의 새 슬롯에 공개 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-public-key testuserpublic.key
   --label testuserpublic_key --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```



참고

공개 키가 개인 키 또는 인증서에 해당하는 경우 개인 키 또는 인증서의 ID와 동일한 ID를 지정합니다.

7. (선택 사항) 특정 스마트 카드를 사용하려면 설정을 잠금하여 카드를 종료해야 합니다.

```
$ pkcs15-init -F
```

이 단계에서는 새로 생성된 슬롯에 인증서, 개인 키 및 공개 키가 포함되어 있습니다. 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK도 생성했습니다.

5.4. 웹 콘솔에 대한 스마트 카드 인증 활성화

웹 콘솔에서 스마트 카드 인증을 사용하려면 **cockpit.conf** 파일에서 스마트 카드 인증을 활성화합니다.

또한 동일한 파일에서 암호 인증을 비활성화할 수 있습니다.

사전 요구 사항

- RHEL 웹 콘솔이 설치되어 있습니다.
  자세한 내용은 웹 콘솔 설치를 참조하십시오.

절차

1. 관리자 권한으로 RHEL 웹 콘솔에 로그인합니다.
   자세한 내용은 웹 콘솔에 로그인 을 참조하십시오.
2. **터미널** 을 클릭합니다.
3. **/etc/cockpit/cockpit.conf** 에서 **ClientCertAuthentication** 을 **yes** 로 설정합니다.

```
[WebService]
ClientCertAuthentication = yes
```

4. 필요한 경우 **cockpit.conf** 에서 다음을 사용하여 암호 기반 인증을 비활성화합니다.

```
[Basic]
action = none
```

이 설정은 암호 인증을 비활성화하고 항상 스마트 카드를 사용해야 합니다.

5. 웹 콘솔을 다시 시작하여 **cockpit.service** 에서 변경 사항을 수락하는지 확인합니다.

```
# systemctl restart cockpit
```

5.5. 스마트 카드를 사용하여 웹 콘솔에 로그인

스마트 카드를 사용하여 웹 콘솔에 로그인할 수 있습니다.

사전 요구 사항

- Active Directory 또는 Identity Management 도메인에서 생성된 사용자 계정에 연결된 스마트 카드에 저장된 유효한 인증서입니다.
- 스마트 카드의 잠금을 해제하는 unique입니다.
- 스마트 카드가 리더에 배치되었습니다.

절차

1. 웹 브라우저를 열고 주소 표시줄에 웹 콘솔의 주소를 추가합니다.
   브라우저는 스마트 카드에 저장된 인증서를 보호하는 pin을 추가하도록 요청합니다.
2. **Password Required** 대화 상자에서 Pin을 입력하고 **확인을** 클릭합니다.
3. **사용자 식별 요청** 대화 상자에서 스마트 카드에 저장된 인증서를 선택합니다.
4. **Remember this decision** 을 선택합니다.
   다음 번에 이 창을 열지 않습니다.

참고

이 단계는 Google Chrome 사용자에게 적용되지 않습니다.

5. **OK**를 클릭합니다.

이제 연결되고 웹 콘솔에 해당 콘텐츠가 표시됩니다.

5.6. DOS 공격을 방지하기 위해 사용자 세션 및 메모리 제한

인증서 인증은 다른 사용자를 가장하려는 공격자에 대해 **cockpit-ws** 웹 서버의 인스턴스를 분리하고 격리하여 보호됩니다. 그러나 이를 통해 잠재적인 DoS(DoS) 공격이 발생할 수 있습니다. 원격 공격자는 다수의 인증서를 생성하고 서로 다른 인증서를 사용하여 각각 **cockpit-ws** 에 다수의 HTTPS 요청을 보낼 수 있습니다.

이 DoS를 방지하기 위해 이러한 웹 서버 인스턴스의 집합적 리소스는 제한됩니다. 기본적으로 연결 수 및 메모리 사용량에 대한 제한은 200개의 스레드와 75%(소프트) / 90%(하드) 메모리 제한으로 설정됩니다.

다음 절차에서는 연결 및 메모리 수를 제한하여 리소스 보호를 설명합니다.

절차

1. 터미널에서 **system-cockpithttps.slice** 구성 파일을 엽니다.

```
# systemctl edit system-cockpithttps.slice
```

2. **TasksMax** 를 *100* 으로 제한하고 **CPUQuota** 를 *30%* 로 제한합니다.

```
[Slice]
# change existing value
TasksMax=100
# add new restriction
CPUQuota=30%
```

3. 변경 사항을 적용하려면 시스템을 다시 시작하십시오.

```
# systemctl daemon-reload
# systemctl stop cockpit
```

이제 새로운 메모리 및 사용자 세션 제한으로 **cockpit-ws** 웹 서버를 DoS 공격으로부터 보호합니다.

5.7. 추가 리소스

- 스마트 카드 인증을 위한 ID 관리 구성 .
- IdM의 스마트 카드 인증을 위해 ADCS에서 발급한 인증서 구성 .
- 스마트 카드로 로컬 인증서 구성 및 가져오기

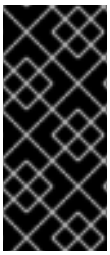
6장. 로컬 인증서를 사용하여 스마트 카드 인증 구성

로컬 인증서를 사용하여 스마트 카드 인증을 구성하려면 다음을 수행합니다.

- 호스트가 도메인에 연결되어 있지 않습니다.
- 이 호스트의 스마트 카드로 인증하려고 합니다.
- 스마트 카드 인증을 사용하여 SSH 액세스를 구성하려고 합니다.
- **authselect** 를 사용하여 스마트 카드를 구성하려고 합니다.

이 시나리오를 수행하려면 다음 구성을 사용합니다.

- 스마트 카드로 인증하려는 사용자의 사용자 인증서를 가져옵니다. 인증서는 도메인에서 사용되는 신뢰할 수 있는 인증 기관에서 생성해야 합니다.
  인증서를 가져올 수 없는 경우 테스트를 위해 로컬 인증 기관에서 서명한 사용자 인증서를 생성할 수 있습니다.
- 인증서 및 개인 키를 스마트 카드에 저장합니다.
- SSH 액세스를 위한 스마트 카드 인증을 구성합니다.



중요

호스트가 도메인의 일부일 수 있는 경우, 호스트를 도메인에 추가하고 Active Directory 또는 Identity Management Certification Authority에서 생성한 인증서를 사용합니다.

스마트 카드의 IdM 인증서를 생성하는 방법에 대한 자세한 내용은 스마트 카드 인증에 대한 ID 관리 구성을 참조하십시오.

사전 요구 사항

- Authselect가 설치됨
  authselect 도구는 Linux 호스트에서 사용자 인증을 구성하며 이를 사용하여 스마트 카드 인증 매개 변수를 구성할 수 있습니다. authselect에 대한 자세한 내용은 authselect 설명에서 참조하십시오.
- RHEL 8에서 지원하는 스마트 카드 또는 USB 장치
  자세한 내용은 RHEL8의 스마트 카드 지원을 참조하십시오.

6.1. 로컬 인증서 생성

다음 작업을 수행하려면 다음 절차를 따르십시오.

- OpenSSL 인증 기관 생성
- 인증서 서명 요청 만들기

주의

다음 단계는 테스트 목적으로만 사용됩니다. 자체 서명된 로컬 인증 기관에서 생성한 인증서는 AD, IdM 또는 RHCS 인증 기관을 사용하는 것만큼 안전하지 않습니다. 호스트가 도메인에 속하지 않더라도 엔터프라이즈 인증 기관에서 생성한 인증서를 사용해야 합니다.

절차

1. 인증서를 생성할 수 있는 디렉터리를 생성합니다. 예를 들면 다음과 같습니다.

```
# mkdir /tmp/ca
# cd /tmp/ca
```

2. 인증서를 설정합니다(이 텍스트를 **ca** 디렉토리의 명령줄에 복사).

```
cat > ca.cnf <<EOF
[ ca ]
default_ca = CA_default

[ CA_default ]
dir             = .
database        = \$dir/index.txt
new_certs_dir   = \$dir/newcerts

certificate     = \$dir/rootCA.crt
serial          = \$dir/serial
private_key     = \$dir/rootCA.key
RANDFILE        = \$dir/rand

default_days    = 365
default_crl_days = 30
default_md      = sha256

policy          = policy_any
email_in_dn     = no

name_opt        = ca_default
cert_opt        = ca_default
copy_extensions  = copy

[ usr_cert ]
authorityKeyIdentifier = keyid, issuer

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints       = CA:true
keyUsage               = critical, digitalSignature, cRLSign, keyCertSign

[ policy_any ]
```

```
organizationName       = supplied
organizationalUnitName = supplied
commonName             = supplied
emailAddress           = optional

[ req ]
distinguished_name = req_distinguished_name
prompt             = no

[ req_distinguished_name ]
O  = Example
OU = Example Test
CN = Example Test CA
EOF
```

3. 다음 디렉터리를 생성합니다.

```
# mkdir certs crl newcerts
```

4. 다음 파일을 생성합니다.

```
# touch index.txt crlnumber index.txt.attr
```

5. 일련 파일에 숫자 01을 작성합니다.

```
# echo 01 > serial
```

이 명령은 직렬 파일에 숫자 01을 작성합니다. 인증서의 일련 번호입니다. 이 CA에서 새로 릴리스하는 각각의 새 인증서에서는 수가 1씩 증가합니다.

6. OpenSSL 루트 CA 키를 생성합니다.

```
# openssl genrsa -out rootCA.key 2048
```

7. 자체 서명된 루트 인증 기관 인증서를 생성합니다.

```
# openssl req -batch -config ca.cnf \
    -x509 -new -nodes -key rootCA.key -sha256 -days 10000 \
    -set_serial 0 -extensions v3_ca -out rootCA.crt
```

8. 사용자 이름에 대한 키를 생성합니다.

```
# openssl genrsa -out example.user.key 2048
```

이 키는 안전하지 않은 로컬 시스템에서 생성되므로 키가 카드에 저장될 때 시스템에서 키를 제거합니다.

스마트 카드에서도 직접 키를 생성할 수 있습니다. 이를 위해 스마트 카드 제조업체가 생성한 지침을 따르십시오.

9. 인증서 서명 요청 구성 파일을 만듭니다(이 텍스트를 ca 디렉토리의 명령줄에 복사).

```
cat > req.cnf <<EOF
[ req ]
```

```
distinguished_name = req_distinguished_name
prompt = no

[ req_distinguished_name ]
O = Example
OU = Example Test
CN = testuser

[ req_exts ]
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "testuser"
subjectKeyIdentifier = hash
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection, msSmartcardLogin
subjectAltName = otherName:msUPN;UTF8:testuser@EXAMPLE.COM,
email:testuser@example.com
EOF
```

10. example.user 인증서에 대한 인증서 서명 요청을 생성합니다.

```
# openssl req -new -nodes -key example.user.key \
    -reqexts req_exts -config req.cnf -out example.user.csr
```

11. 새 인증서를 구성합니다. 만료 기간은 1년으로 설정됩니다.

```
# openssl ca -config ca.cnf -batch -notext \
    -keyfile rootCA.key -in example.user.csr -days 365 \
    -extensions usr_cert -out example.user.crt
```

이때 인증 기관과 인증서가 성공적으로 생성되어 스마트 카드로 가져올 준비가 되었습니다.

6.2. SSSD 디렉터리에 인증서 복사

GNOME 데스크탑 관리자(GDM)에는 SSSD가 필요합니다. GDM을 사용하는 경우 PEM 인증서를 **/etc/sssd/pki** 디렉터리에 복사해야 합니다.

사전 요구 사항

- 로컬 CA 기관 및 인증서가 생성되었습니다.

절차

1. 시스템에 SSSD가 설치되어 있는지 확인합니다.

```
# rpm -q sssd
sssd-2.0.0.43.el8_0.3.x86_64
```

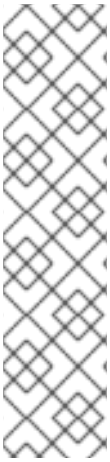
2. **/etc/sssd/pki** 디렉토리를 만듭니다.

```
# file /etc/sssd/pki
/etc/sssd/pki/: directory
```

3. **/etc/sssd/pki/** 디렉터리에서 **rootCA.crt** 를 PEM 파일로 복사합니다.

```
# cp /tmp/ca/rootCA.crt /etc/sssd/pki/sssd_auth_ca_db.pem
```

이제 인증 기관 및 인증서를 성공적으로 생성했으며 **/etc/sssd/pki** 디렉터리에 저장했습니다.



참고

인증 기관 인증서를 다른 애플리케이션과 공유하려면 sssd.conf의 위치를 변경할 수 있습니다.

- SSSD PAM 응답자: **[pam]** 섹션의 **pam_cert_db_path**
- SSSD ssh 응답자: **[ssh]** 섹션의 **ca_db**

자세한 내용은 **sssd.conf** 의 도움말 페이지를 참조하십시오.

Red Hat은 기본 경로를 유지하고 SSSD에 전용 인증 기관 인증서 파일을 사용하여 인증 기관만 여기에 나열되어 있는지 확인하는 것이 좋습니다.

6.3. 스마트 카드 관리 및 사용을 위한 도구 설치

스마트 카드를 구성하려면 인증서를 생성하고 스마트 카드에 저장할 수 있는 도구가 필요합니다.

다음을 수행해야 합니다.

- 인증서 관리에 도움이 되는 **gnutls-utils** 패키지를 설치합니다.
- 스마트 카드로 작업할 라이브러리 및 유틸리티 세트를 제공하는 **opensc** 패키지를 설치합니다.
- 스마트 카드 리더와 통신하는 **pcscd** 서비스를 시작합니다.

절차

1. **opensc** 및 **gnutls-utils** 패키지를 설치합니다.

```
# dnf -y install opensc gnutls-utils
```

2. **pcscd** 서비스를 시작합니다.

```
# systemctl start pcscd
```

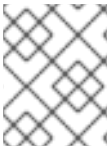
pcscd 서비스가 실행 중인지 확인합니다.

6.4. 스마트 카드 준비 및 스마트 카드에 인증서와 키 업로드

다음 절차에 따라 구성하는 데 도움이 되는 **pkcs15-init** 도구를 사용하여 스마트 카드를 구성합니다.

- 스마트 카드 삭제
- 새로운 PIN 및 선택적 PIN 잠금 해제 키 (PUK) 설정
- 스마트 카드에서 새 슬롯 생성

- 인증서, 개인 키 및 공개 키 저장
- 필요한 경우 특정 스마트 카드에 따라 스마트 카드 설정을 잠금하려면 이러한 유형의 최종화가 필요합니다.



참고

pkcs15-init 툴은 모든 스마트 카드에서 작동하지 않을 수 있습니다. 사용 중인 스마트 카드로 작업하는 도구를 사용해야 합니다.

사전 요구 사항

- **pkcs15-init** 툴이 포함된 **opensc** 패키지가 설치됩니다.
  자세한 내용은 스마트 카드 관리 및 사용을 위한 툴 설치를 참조하십시오.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- 스마트 카드에 저장할 개인 키, 공개 키 및 인증서가 있습니다. 이 절차에서 **testuser.key**,**testuserpublic.key** 및 **testuser.crt** 는 개인 키, 공개 키 및 인증서에 사용되는 이름입니다.
- 현재 스마트 카드 사용자 Pin and Security Officer Pin (SO-PIN)이 있습니다.

절차

1. 스마트 카드를 지우고 PIN으로 인증하십시오:

```
$ pkcs15-init --erase-card --use-default-transport-keys
Using reader with a card: Reader name
PIN [Security Officer PIN] required.
Please enter PIN [Security Officer PIN]:
```

   카드가 지워졌습니다.

2. 스마트 카드를 초기화하고 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK를 설정합니다.

```
$ pkcs15-init --create-pkcs15 --use-default-transport-keys \
   --pin 963214 --puk 321478 --so-pin 65498714 --so-puk 784123
Using reader with a card: Reader name
```

   pcks15-init 툴은 스마트 카드에 새 슬롯을 생성합니다.

3. 슬롯의 레이블 및 인증 ID를 설정합니다.

```
$ pkcs15-init --store-pin --label testuser \
   --auth-id 01 --so-pin 65498714 --pin 963214 --puk 321478
Using reader with a card: Reader name
```

   레이블은 사람이 읽을 수 있는 값(이 경우 **testuser**)으로 설정됩니다. **auth-id** 는 16진수 2개 값이어야 합니다(이 경우 **01** 로 설정됨).

4. 스마트 카드의 새 슬롯에 개인 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-private-key testuser.key --label testuser_key \
   --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```

참고

--id 에 지정하는 값은 개인 키를 저장하고 다음 단계에 인증서를 저장할 때 동일해야 합니다. 도구에서 더 복잡한 값을 계산하므로 **--id** 에 대해 자체 값을 지정하는 것이 좋습니다.

5. 스마트 카드의 새 슬롯에 인증서를 저장하고 레이블을 지정합니다.

```
$ pkcs15-init --store-certificate testuser.crt --label testuser_crt \
   --auth-id 01 --id 01 --format pem --pin 963214
Using reader with a card: Reader name
```

6. (선택 사항) 스마트 카드의 새 슬롯에 공개 키 저장 및 레이블을 지정합니다.

```
$ pkcs15-init --store-public-key testuserpublic.key
   --label testuserpublic_key --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```

참고

공개 키가 개인 키 또는 인증서에 해당하는 경우 개인 키 또는 인증서의 ID와 동일한 ID를 지정합니다.

7. (선택 사항) 특정 스마트 카드를 사용하려면 설정을 잠금하여 카드를 종료해야 합니다.

```
$ pkcs15-init -F
```

이 단계에서는 새로 생성된 슬롯에 인증서, 개인 키 및 공개 키가 포함되어 있습니다. 사용자 PIN 및 PUK, 보안 책임자 PIN 및 PUK도 생성했습니다.

6.5. 스마트 카드 인증을 사용하여 SSH 액세스 구성

SSH 연결에는 인증이 필요합니다. 암호 또는 인증서를 사용할 수 있습니다. 스마트 카드에 저장된 인증서를 사용하여 인증을 활성화하려면 다음 절차를 따르십시오.

authselect를 사용하여 스마트 카드를 구성하는 방법에 대한 자세한 내용은 **authselect** 를 사용하여 스마트 카드 구성을 참조하십시오.

사전 요구 사항

- 스마트 카드에는 인증서와 개인 키가 포함되어 있습니다.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- SSSD가 설치 및 구성되어 있습니다.
- 사용자 이름은 인증서 SUBJECT의 CN(Common Name) 또는 UID(사용자 ID)와 일치합니다.

- **pcscd** 서비스가 로컬 시스템에서 실행되고 있습니다.
  자세한 내용은 스마트 카드를 관리하고 사용하기 위한 툴 설치를 참조하십시오.

절차

1. 스마트 카드 인증을 사용하는 사용자의 홈 디렉터리에 SSH 키에 대한 새 디렉터리를 생성합니다.

```
# mkdir /home/example.user/.ssh
```

2. **opensc** 라이브러리와 함께 **ssh-keygen -D** 명령을 실행하여 스마트 카드의 개인 키와 쌍된 기존 공개 키를 검색하고 SSH 키 디렉터리의 **authorized_keys** 목록에 스마트 카드 인증을 사용하여 SSH 액세스를 활성화합니다.

```
# ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so >>
~example.user/.ssh/authorized_keys
```

3. SSH를 사용하려면 **/.ssh** 디렉토리 및 **authorized_keys** 파일에 대한 올바른 구성이 필요합니다. 액세스 권한을 설정하거나 변경하려면 다음을 입력합니다.

```
# chown -R example.user:example.user ~example.user/.ssh/
# chmod 700 ~example.user/.ssh/
# chmod 600 ~example.user/.ssh/authorized_keys
```

4. 선택적으로 키를 표시합니다.

```
# cat ~example.user/.ssh/authorized_keys
```

   터미널에 키가 표시됩니다.

5. **/etc/sssd/sssd.conf 파일에서 스마트 카드 인증이 활성화되었는지 확인합니다.**
   [pam] 섹션에서 pam 인증서 인증 모듈을 활성화합니다. **pam_cert_auth = True**

   sssd.conf 파일이 아직 생성되지 않은 경우 다음 스크립트를 명령줄에 복사하여 최소 기능 구성을 생성할 수 있습니다.

```
# cat > /etc/sssd/sssd.conf <<EOF
[sssd]
services = nss, pam
domains = shadowutils

[nss]

[pam]
pam_cert_auth = True

[domain/shadowutils]
id_provider = files
EOF
```

6. SSH 키를 사용하려면 **authselect** 명령을 사용하여 인증을 구성합니다.

```
# authselect select sssd with-smartcard --force
```

이제 다음 명령을 사용하여 SSH 액세스를 확인할 수 있습니다.

```
# ssh -I /usr/lib64/opensc-pkcs11.so -l example.user localhost hostname
```

구성이 성공적으로 수행되면 스마트 카드 PIN을 입력하라는 메시지가 표시됩니다.

이제 구성이 로컬로 작동합니다. 이제 공개 키를 복사하여 SSH를 사용하려는 모든 서버에 있는 **authorized_keys** 파일에 배포할 수 있습니다.

7장. AUTHSELECT를 사용하여 스마트 카드 인증 구성

이 섹션에서는 다음 중 하나를 수행하도록 스마트 카드를 구성하는 방법에 대해 설명합니다.

- 암호와 스마트 카드 인증 모두 활성화
- 암호를 비활성화하고 스마트 카드 인증을 활성화합니다.
- 제거 시 잠금 활성화

사전 요구 사항

- Authselect가 설치됨
  authselect 도구는 Linux 호스트에서 사용자 인증을 구성하며 이를 사용하여 스마트 카드 인증 매개 변수를 구성할 수 있습니다. authselect에 대한 자세한 내용은 authselect 를 사용하여 사용자 인증 구성을 참조하십시오.
- RHEL 8에서 지원하는 스마트 카드 또는 USB 장치
  자세한 내용은 RHEL8의 스마트 카드 지원을 참조하십시오.

7.1. 스마트 카드 이용 자격 인증서

authselect 를 사용하여 스마트 카드를 구성하려면 인증서를 카드로 가져와야 합니다. 다음 도구를 사용하여 인증서를 생성할 수 있습니다.

- Active Directory (AD)
- IdM(Identity Management)
  IdM 인증서를 생성하는 방법에 대한 자세한 내용은 새 사용자 인증서 요청 및 클라이언트로 내보내기를 참조하십시오.
- RHCS(Red Hat Certificate System)
  자세한 내용은 Enterprise Security Client를 사용하여 스마트 카드 관리를 참조하십시오.
- 타사 CA(인증 기관)
- 로컬 인증 기관. 사용자가 도메인에 속하지 않거나 테스트 목적으로 로컬 인증 기관에서 생성한 인증서를 사용할 수 있습니다.
  로컬 인증서를 스마트 카드로 생성 및 가져오는 방법에 대한 자세한 내용은 로컬 인증서를 스마트 카드로 구성 및 가져오는 것입니다.

7.2. 스마트 카드 및 암호 인증을 둘 다 사용하도록 시스템 구성

시스템에서 스마트 카드 및 암호 인증을 모두 활성화하려면 다음 절차를 따르십시오.

사전 요구 사항

- 스마트 카드에는 인증서와 개인 키가 포함되어 있습니다.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- **authselect** 도구가 시스템에 설치되어 있습니다.

절차

- 스마트 카드 및 암호 인증을 허용하려면 다음 명령을 입력합니다.

```
# authselect select sssd with-smartcard --force
```

이 시점에서 스마트 카드 인증이 활성화되어 있지만, 가정에서 스마트 카드를 잊어버리면 암호 인증이 작동합니다.

7.3. 스마트 카드 인증을 적용하도록 시스템 구성

authselect 도구를 사용하면 시스템에서 스마트 카드 인증을 구성하고 기본 암호 인증을 비활성화할 수 있습니다. **authselect** 명령에는 다음 옵션이 포함됩니다.

- **with-smartcard** hiera-octetsenables 스마트 카드 인증 및 암호 인증 사용
- **with-smartcard-required** hiera-octets-enables 스마트 카드 인증을 사용하고 암호 인증을 비활성화합니다.

참고

with-smartcard-required 옵션은 로그인 서비스(예: 로그인 , **gdm**,**xdm**,**kdm**, xdm , xscreensdockerfile ,**gnome- screens diff , 및 kscreensaver**)와 같은 **로그인** 서비스에 대해 전용 스마트 카드 인증을 시행합니다. 사용자 전환에 **su** 또는 **sudo** 와 같은 기타 서비스는 기본적으로 스마트 카드 인증을 사용하지 말고 계속 암호를 입력하라는 메시지를 표시합니다.

사전 요구 사항

- 스마트 카드에 인증서 및 개인 키가 포함되어 있습니다.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- **authselect** 툴이 로컬 시스템에 설치되어 있습니다.

절차

- 다음 명령을 입력하여 스마트 카드 인증을 적용합니다.

```
# authselect select sssd with-smartcard  with-smartcard-required --force
```

참고

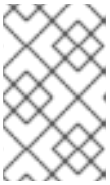
이 명령을 실행하면 암호 인증이 더 이상 작동하지 않으며 스마트 카드로만 로그인할 수 있습니다. 이 명령을 실행하기 전에 스마트 카드 인증이 작동하는지 확인하거나 시스템에서 잠길 수 있습니다.

7.4. 잠금 제거를 통한 스마트 카드 인증 구성

authselect 서비스를 사용하면 리더에서 스마트 카드를 제거한 후 즉시 화면을 잠그도록 스마트 카드 인증을 구성할 수 있습니다. **authselect** 명령에는 다음 변수가 포함되어야 합니다.

- **with-smartcard** 메세지- 스마트 카드 인증 지원

- **스마트카드를 사용하는 경우** 전용 스마트 카드 인증이 필요합니다(암호를 사용한 인증은 비활성화됨)
- **with-smartcard-lock-on-removal** apache-kubeenforcing 스마트 카드 제거 후 로그아웃



참고

with-smartcard-lock-on-removal 옵션은 GNOME 데스크탑 환경의 시스템에서만 작동합니다. **tty** 또는 콘솔 기반 시스템을 사용하고 있으며 리더에서 스마트 카드를 제거하는 경우 시스템에서 자동으로 잠기지 않습니다.

사전 요구 사항

- 스마트 카드에 인증서 및 개인 키가 포함되어 있습니다.
- 카드가 리더에 삽입되고 컴퓨터에 연결됩니다.
- **authselect** 툴이 로컬 시스템에 설치되어 있습니다.

절차

- 다음 명령을 입력하여 스마트 카드 인증을 활성화하고, 암호 인증을 비활성화하고, 제거 시 잠금을 적용합니다.

```
# authselect select sssd  with-smartcard  with-smartcard-required with-smartcard-lock-on-removal --force
```

이제 카드를 제거하면 화면이 잠깁니다. 잠금을 해제하려면 스마트 카드를 다시 삽입해야 합니다.

8장. 스마트 카드를 사용하여 SUDO에 인증

이 섹션에서는 스마트 카드를 사용하여 sudo를 원격으로 인증하는 방법을 설명합니다. **ssh-agent** 서비스가 로컬로 실행되고 **ssh-agent** 소켓을 원격 시스템으로 전달할 수 있는 경우 sudo PAM 모듈에서 SSH 인증 프로토콜을 사용하여 사용자를 원격으로 인증할 수 있습니다.

스마트 카드를 사용하여 로컬로 로그인한 후 SSH를 통해 원격 시스템에 로그인하고 스마트 카드 인증의 SSH 전달을 사용하여 암호를 입력하라는 메시지 없이 **sudo** 명령을 실행할 수 있습니다.

이 예제의 목적을 위해 클라이언트는 SSH를 통해 IPA 서버에 연결하고 스마트 카드에 저장된 자격 증명을 사용하여 IPA 서버에서 sudo 명령을 실행합니다.

- IdM에서 sudo 규칙 생성
- sudo의 PAM 모듈 설정
- 스마트 카드를 사용하여 sudo에 원격으로 연결

8.1. IDM에서 SUDO 규칙 생성

IdM에 sudo 규칙을 생성하여 **ipauser1** 에게 원격 호스트에서 sudo를 실행할 수 있는 권한을 부여합니다.

이 예제의 목적을 위해 **less** 및 **whoami** 명령이 sudo 명령으로 추가되어 절차를 테스트합니다.

사전 요구 사항

- IdM 사용자가 생성되었습니다. 이 예제의 목적을 위해 사용자는 **ipauser1** 입니다.
- sudo를 원격으로 실행 중인 시스템의 호스트 이름이 있습니다. 이 예제에서 호스트는 **server.ipa.test** 입니다.

절차

1. 사용자가 명령을 실행할 수 있도록 **adminrule** 이라는 **sudo** 규칙을 만듭니다.

```
ipa sudorule-add adminrule
```

2. **sudo** 명령으로 **더 적고 whoami** 를 추가합니다.

```
ipa sudocmd-add /usr/bin/less
ipa sudocmd-add /usr/bin/whoami
```

3. **less** 및 **whoami** 명령을 **관리자에게 추가합니다**.

```
ipa sudorule-add-allow-command adminrule --sudocmds /usr/bin/less
ipa sudorule-add-allow-command adminrule --sudocmds /usr/bin/whoami
```

4. **ipauser1** 사용자를 **adminrule** 에 추가합니다.

```
ipa sudorule-add-user adminrule --users ipauser1
```

5. **sudo** 를 실행 중인 호스트를 **adminrule** 에 추가합니다.

```
ipa sudorule-add-host adminrule --hosts server.ipa.test
```

추가 리소스

- **ipa sudorule-add --help** 를 참조하십시오.
- **ipa sudocmd-add --help** 를 참조하십시오.

8.2. SUDO에 대한 PAM 모듈 설정

sudo를 실행하는 모든 호스트에서 스마트 카드로 sudo 인증을 위해 **pam_ssh_agent_auth.so** PAM 모듈을 설치하고 설정하려면 다음 절차를 따르십시오.

절차

1. PAM SSH 에이전트를 설치합니다.

```
dnf -y install pam_ssh_agent_auth
```

2. **pam_ssh _agent_auth.so에 대해 authorized_ keys_command** 를 다른 **auth** 항목 전에 **/etc/pam.d/sudo** 파일에 추가합니다.

```
#%PAM-1.0
auth sufficient pam_ssh_agent_auth.so
authorized_keys_command=/usr/bin/sss_ssh_authorizedkeys
auth       include      system-auth
account    include      system-auth
password   include      system-auth
session    include      system-auth
```

3. sudo 명령을 실행할 때 SSH 에이전트 전달이 작동하도록 하려면 **/etc/sudoers** 파일에 다음을 추가합니다.

```
Defaults env_keep += "SSH_AUTH_SOCK"
```

이를 통해 IPA/SSSD에 저장된 스마트 카드의 공개 키가 있는 사용자는 암호를 입력하지 않고 sudo에 인증할 수 있습니다.

4. **sssd** 서비스를 다시 시작하십시오.

```
systemctl restart sssd
```

추가 리소스

- **pam** 도움말 페이지를 참조하십시오.

8.3. 스마트 카드를 사용하여 SUDO에 원격으로 연결

스마트 카드를 사용하여 **sudo** 에 원격으로 연결하도록 SSH 에이전트 및 클라이언트를 구성하려면 다음 절차를 따르십시오.

사전 요구 사항

- IdM에 **sudo** 규칙이 생성되어 있습니다.
- **sudo** 를 실행할 원격 시스템에서 **sudo** 인증을 위해 **pam_ssh_agent_auth** PAM 모듈을 설치하고 설정했습니다.

절차

1. SSH 에이전트를 시작합니다(아직 실행되지 않은 경우).

```
eval `ssh-agent`
```

2. SSH 에이전트에 스마트 카드를 추가합니다. 메시지가 표시되면 PIN을 입력합니다.

```
ssh-add -s /usr/lib64/opensc-pkcs11.so
```

3. ssh-agent 전달이 활성화된 SSH를 사용하여 원격으로 **sudo** 를 실행해야 하는 시스템에 연결합니다. **-A** 옵션을 사용합니다.

```
ssh -A ipauser1@server.ipa.test
```

검증 단계

- **sudo** 를 사용하여 **whoami** 명령을 실행합니다.

```
sudo /usr/bin/whoami
```

스마트 카드가 삽입될 때 generated 또는 password를 입력하라는 메시지가 표시되지 않습니다.

참고

SSH 에이전트가 GNOME 키링과 같은 다른 소스를 사용하도록 구성되어 있고 스마트 카드를 제거한 후 **sudo** 명령을 실행하는 경우, 다른 소스 중 하나가 유효한 개인 키에 대한 액세스를 제공할 수 있으므로, smart 카드를 제거한 후 sudo 명령을 실행하지 못할 수 있습니다. SSH 에이전트에서 알려진 모든 ID의 공개 키를 확인하려면 **ssh-add -L** 명령을 실행합니다.

추가 리소스

- OpenSSH로 두 시스템 간의 보안 통신 사용
- ssh-agent를 사용하여 SSH 키가 있는 원격 시스템에 연결

9장. 스마트 카드로 인증 문제 해결

다음 섹션에서는 스마트 카드 인증을 설정할 때 발생할 수 있는 몇 가지 문제를 해결하는 방법을 설명합니다.

- 스마트 카드 인증 테스트
- SSSD를 사용하여 스마트 카드 인증 문제 해결
- IdM Kerberos 6443이 PKINIT를 사용하고 CA 인증서가 올바르게 있는지 확인
- SSSD 시간 초과 늘리기
- 인증서 매핑 및 일치 규칙 문제 해결

9.1. 시스템에서 스마트 카드 액세스 테스트

스마트 카드에 액세스할 수 있는지 확인하려면 다음 절차를 따르십시오.

사전 요구 사항

- 스마트 카드에서 사용할 IdM 서버 및 클라이언트를 설치하고 구성했습니다.
- **nss-tools** 패키지에서 **certutil** 툴을 설치했습니다.
- 스마트 카드의 Pin 또는 암호가 있습니다.

절차

1. **lsusb** 명령을 사용하여 스마트 카드 리더가 운영 체제에 표시되는지 확인합니다.

```
$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 072f:b100 Advanced Card Systems, Ltd ACR39U
Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

   RHEL에서 테스트 및 지원되는 스마트 카드 및 리더에 대한 자세한 내용은 RHEL 8의 스마트 카드 지원을 참조하십시오.

2. **pcscd** 서비스 및 소켓이 활성화되어 실행되고 있는지 확인합니다.

```
$ systemctl status pcscd.service pcscd.socket

● pcscd.service - PC/SC Smart Card Daemon
     Loaded: loaded (/usr/lib/systemd/system/pcscd.service; indirect;
vendor preset: disabled)
     Active: active (running) since Fri 2021-09-24 11:05:04 CEST; 2
weeks 6 days ago
TriggeredBy: ● pcscd.socket
       Docs: man:pcscd(8)
   Main PID: 3772184 (pcscd)
      Tasks: 12 (limit: 38201)
     Memory: 8.2M
        CPU: 1min 8.067s
```

```
      CGroup: /system.slice/pcscd.service
              └─3772184 /usr/sbin/pcscd --foreground --auto-exit

● pcscd.socket - PC/SC Smart Card Daemon Activation Socket
     Loaded: loaded (/usr/lib/systemd/system/pcscd.socket; enabled;
vendor preset: enabled)
     Active: active (running) since Fri 2021-09-24 11:05:04 CEST; 2
weeks 6 days ago
   Triggers: ● pcscd.service
     Listen: /run/pcscd/pcscd.comm (Stream)
     CGroup: /system.slice/pcscd.socket
```

3. **p11-kit list-modules** 명령을 사용하여 스마트 카드에 구성된 스마트 카드 및 토큰에 대한 정보를 표시합니다.

```
$ p11-kit list-modules
p11-kit-trust: p11-kit-trust.so
[...]
opensc: opensc-pkcs11.so
   library-description: OpenSC smartcard framework
   library-manufacturer: OpenSC Project
   library-version: 0.20
   token: MyEID (sctest)
      manufacturer: Aventra Ltd.
      model: PKCS#15
      serial-number: 8185043840990797
      firmware-version: 40.1
      flags:
            rng
            login-required
            user-pin-initialized
            token-initialized
```

4. 스마트 카드의 콘텐츠에 액세스할 수 있는지 확인합니다.

```
$ pkcs11-tool --list-objects --login
Using slot 0 with a present token (0x0)
Logging in to "MyEID (sctest)".
Please enter User PIN:
Private Key Object; RSA
  label:      Certificate
  ID:         01
  Usage:      sign
  Access:     sensitive
Public Key Object; RSA 2048 bits
  label:      Public Key
  ID:         01
  Usage:      verify
  Access:     none
Certificate Object; type = X.509 cert
  label:      Certificate
  subject:    DN: O=IDM.EXAMPLE.COM, CN=idmuser1
  ID:         01
```

5. **certutil** 명령을 사용하여 스마트 카드에 인증서 내용을 표시합니다.

a. 다음 명령을 실행하여 인증서의 올바른 이름을 확인합니다.

```
$ certutil -d /etc/pki/nssdb -L -h all

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

Enter Password or Pin for "MyEID (sctest)":
Smart Card CA 0f5019a8-7e65-46a1-afe5-8e17c256ae00           CT,C,C
MyEID (sctest):Certificate                                   u,u,u
```

b. 스마트 카드의 인증서 내용을 표시합니다.

참고

인증서 이름이 이전 단계에 표시된 출력에 대한 정확한 일치 항목인지 확인합니다(이 예에서는 **MyEID(sctest):Certificate**).

```
$ certutil -d /etc/pki/nssdb -L -n "MyEID (sctest):Certificate"

Enter Password or Pin for "MyEID (sctest)":
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 15 (0xf)
        Signature Algorithm: PKCS #1 SHA-256 With RSA Encryption
        Issuer: "CN=Certificate Authority,O=IDM.EXAMPLE.COM"
        Validity:
            Not Before: Thu Sep 30 14:01:41 2021
            Not After : Sun Oct 01 14:01:41 2023
        Subject: "CN=idmuser1,O=IDM.EXAMPLE.COM"
        Subject Public Key Info:
            Public Key Algorithm: PKCS #1 RSA Encryption
            RSA Public Key:
                Modulus:
                    [...]
                Exponent: 65537 (0x10001)
        Signed Extensions:
            Name: Certificate Authority Key Identifier
            Key ID:
                e2:27:56:0d:2f:f5:f2:72:ce:de:37:20:44:8f:18:7f:
                2f:56:f9:1a

            Name: Authority Information Access
            Method: PKIX Online Certificate Status Protocol
            Location:
                URI: "http://ipa-ca.idm.example.com/ca/ocsp"

            Name: Certificate Key Usage
            Critical: True
            Usages: Digital Signature
                    Non-Repudiation
                    Key Encipherment
                    Data Encipherment
```

```
            Name: Extended Key Usage
                TLS Web Server Authentication Certificate
                TLS Web Client Authentication Certificate

            Name: CRL Distribution Points
            Distribution point:
                URI: "http://ipa-ca.idm.example.com/ipa/crl/MasterCRL.bin"
                CRL issuer:
                    Directory Name: "CN=Certificate Authority,O=ipaca"

            Name: Certificate Subject Key ID
            Data:
                43:23:9f:c1:cf:b1:9f:51:18:be:05:b5:44:dc:e6:ab:
                be:07:1f:36

    Signature Algorithm: PKCS #1 SHA-256 With RSA Encryption
    Signature:
        [...]
    Fingerprint (SHA-256):

6A:F9:64:F7:F2:A2:B5:04:88:27:6E:B8:53:3E:44:3E:F5:75:85:91:34:ED:48:A8:0D:F0:31:5
D:7B:C9:E0:EC
    Fingerprint (SHA1):
        B4:9A:59:9F:1C:A8:5D:0E:C1:A2:41:EC:FD:43:E0:80:5F:63:DF:29

    Mozilla-CA-Policy: false (attribute missing)
    Certificate Trust Flags:
        SSL Flags:
            User
        Email Flags:
            User
        Object Signing Flags:
            User
```

추가 리소스

- **certutil(1)** 도움말 페이지를 참조하십시오.

9.2. SSSD를 사용하여 스마트 카드 인증 문제 해결

스마트 카드를 사용하여 SSSD로 인증 문제를 해결하려면 다음 절차를 따르십시오.

사전 요구 사항

- 스마트 카드에서 사용할 IdM 서버 및 클라이언트를 설치하고 구성했습니다.
- **sssd-tools** 패키지가 설치되어 있습니다.
- 스마트 카드 리더를 감지하고 스마트 카드의 내용을 표시할 수 있습니다. 시스템의 스마트 카드 액세스 테스트를 참조하십시오.

절차

1. **su** 를 사용하여 스마트 카드로 인증할 수 있는지 확인하십시오.

```
$ su - idmuser1 -c 'su - idmuser1 -c whoami'
PIN for MyEID (sctest):
idmuser1
```

스마트 카드 PIN을 입력하라는 메시지가 표시되지 않고 암호 프롬프트 또는 권한 부여 오류가 반환되는 경우 SSSD 로그를 확인합니다. SSSD 로그인 에 대한 자세한 내용은 IdM의 SSSD로 인증 문제 해결을 참조하십시오. 다음은 인증 오류의 예입니다.

```
$ su - idmuser1 -c 'su - idmuser1 -c whoami'
PIN for MyEID (sctest):
su: Authentication failure
```

SSSD 로그가 다음과 유사한 KnativeServing **5_h** iera 에서 문제를 나타내는 경우 CA 인증서에 문제가 있을 수 있습니다. 인증서 문제를 해결하려면 IdM Kerberosmtls에서 Pkinit를 사용할 수 있고 CA 인증서가 올바르게 위치하는지 확인합니다.

```
[Pre-authentication failed: Failed to verify own certificate (depth 0): unable to get local issuer certificate: could not load the shared library]
```

SSSD 로그에서 **p11_octets 또는octets 5_** octets 로부터 시간 초과를 나타내는 경우 SSSD 시간 제한을 늘리고 스마트 카드로 다시 인증해야 할 수 있습니다. 시간 제한을 늘리는 방법에 대한 자세한 내용은 SSSD 시간 초과를 참조하십시오.

2. GDM 스마트 카드 인증 구성이 올바른지 확인합니다. 다음과 같이 PAM 인증에 대한 성공 메시지를 반환해야 합니다.

```
# sssctl user-checks -s gdm-smartcard "idmuser1" -a auth
user: idmuser1
action: auth
service: gdm-smartcard

SSSD nss user lookup result:
 - user name: idmuser1
 - user id: 603200210
 - group id: 603200210
 - gecos: idm user1
 - home directory: /home/idmuser1
 - shell: /bin/sh

SSSD InfoPipe user lookup result:
 - name: idmuser1
 - uidNumber: 603200210
 - gidNumber: 603200210
 - gecos: idm user1
 - homeDirectory: /home/idmuser1
 - loginShell: /bin/sh

testing pam_authenticate

PIN for MyEID (sctest)
pam_authenticate for user [idmuser1]: Success

PAM Environment:
 - PKCS11_LOGIN_TOKEN_NAME=MyEID (sctest)
 - KRB5CCNAME=KCM:
```

다음과 유사한 인증 오류가 반환되면 SSSD 로그를 확인하여 문제의 원인이 되는 항목을 확인합니다. SSSD 로그인 에 대한 자세한 내용은 IdM의 SSSD로 인증 문제 해결을 참조하십시오.

```
pam_authenticate for user [idmuser1]: Authentication failure

PAM Environment:
 - no env -
```

PAM 인증이 계속 실패하면 캐시를 지우고 명령을 다시 실행합니다.

```
# sssctl cache-remove
SSSD must not be running. Stop SSSD now? (yes/no) [yes] yes
Creating backup of local data…
Removing cache files…
SSSD needs to be running. Start SSSD now? (yes/no) [yes] yes
```

9.3. IDM KERBEROS 6443이 PKINIT를 사용하고 CA 인증서가 올바르게 있는지 확인

다음 절차에 따라 IdM Kerberos KDC가 PKINIT를 사용할 수 있는지 확인하고 CA 인증서가 올바르게 배치되었는지 확인하는 방법도 설명합니다.

사전 요구 사항

- 스마트 카드에서 사용할 IdM 서버 및 클라이언트를 설치하고 구성했습니다.
- 스마트 카드 리더를 감지하고 스마트 카드의 내용을 표시할 수 있습니다. 시스템의 스마트 카드 액세스 테스트를 참조하십시오.

절차

1. **kinit** 유틸리티를 실행하여 스마트 카드에 저장된 인증서를 사용하여 **idmuser1** 로 인증합니다.

```
$ kinit -X X509_user_identity=PKCS11: idmuser1
MyEID (sctest)                  PIN:
```

2. 스마트 카드 Pin을 입력합니다. Pin을 묻지 않은 경우 스마트 카드 리더를 감지하고 스마트 카드의 내용을 표시할 수 있는지 확인하십시오. 스마트 카드 인증 테스트를 참조하십시오.
3. PIN이 수락되고 암호를 입력하라는 메시지가 표시되면 CA 서명 인증서가 누락되었을 수 있습니다.
   a. **openssl** 명령을 사용하여 CA 체인이 기본 인증서 번들 파일에 나열되어 있는지 확인합니다.

```
$ openssl crl2pkcs7 -nocrl -certfile /var/lib/ipa-client/pki/ca-bundle.pem | openssl pkcs7 -print_certs -noout
subject=O = IDM.EXAMPLE.COM, CN = Certificate Authority

issuer=O = IDM.EXAMPLE.COM, CN = Certificate Authority
```

   b. 인증서의 유효성을 확인합니다.
      i. **idmuser1** 의 사용자 인증 인증서 ID를 찾습니다.

```
$ pkcs11-tool --list-objects --login
[...]
Certificate Object; type = X.509 cert
  label:      Certificate
  subject:    DN: O=IDM.EXAMPLE.COM, CN=idmuser1
 ID: 01
```

ii. DER 형식의 스마트 카드에서 사용자 인증서 정보를 읽습니다.

```
$ pkcs11-tool --read-object --id 01 --type cert --output-file cert.der
Using slot 0 with a present token (0x0)
```

iii. DER 인증서를 PEM 형식으로 변환합니다.

```
$ openssl x509 -in cert.der -inform DER -out cert.pem -outform PEM
```

iv. 인증서에 CA까지 유효한 발행자 서명이 있는지 확인합니다.

```
$ openssl verify -CAfile /var/lib/ipa-client/pki/ca-bundle.pem <path>/cert.pem
cert.pem: OK
```

4. 스마트 카드에 여러 인증서가 포함된 경우 **kinit** 에서 인증을 위해 올바른 인증서를 선택하지 못할 수 있습니다. 이 경우 **certid=<ID>** 옵션을 사용하여 **kinit** 명령에 대한 인수로 인증서 ID를 지정해야 합니다.

a. 스마트 카드에 저장된 인증서 수를 확인하고 사용 중인 인증서의 인증서 ID를 가져옵니다.

```
$ pkcs11-tool --list-objects --type cert --login
Using slot 0 with a present token (0x0)
Logging in to "MyEID (sctest)".
Please enter User PIN:
Certificate Object; type = X.509 cert
  label:      Certificate
  subject:    DN: O=IDM.EXAMPLE.COM, CN=idmuser1
  ID:         01
Certificate Object; type = X.509 cert
  label:      Second certificate
  subject:    DN: O=IDM.EXAMPLE.COM, CN=ipauser1
  ID:         02
```

b. 인증서 ID 01을 사용하여 **kinit** 를 실행합니다.

```
$ kinit -X kinit -X X509_user_identity=PKCS11:certid=01 idmuser1
MyEID (sctest)                  PIN:
```

5. **klist** 를 실행하여 Kerberos 인증 정보 캐시의 내용을 확인합니다.

```
$ klist
Ticket cache: KCM:0:11485
Default principal: idmuser1@EXAMPLE.COM

Valid starting       Expires              Service principal
10/04/2021 10:50:04  10/05/2021 10:49:55  krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

6. 완료되면 활성 Kerberos 티켓을 삭제합니다.

```
$ kdestroy -A
```

추가 리소스

- **kinit** 도움말 페이지를 참조하십시오.
- **kdestroy** man 페이지를 참조하십시오.

9.4. SSSD 시간 초과 늘리기

스마트 카드로 인증하는 데 문제가 있는 경우, 다음과 유사한 시간 초과 항목이 있는지 KnativeServing **5 _hiera.log 및 p11_octets.log** 파일을 확인하십시오.

krb5_child: 하위 시간 [9607] reached…..consider increasing value of krb5_auth_timeout.

로그 파일에 시간 초과 항목이 있는 경우 이 프로세스에 설명된 대로 SSSD 시간 제한을 늘립니다.

사전 요구 사항

- 스마트 카드 인증을 위해 IdM 서버 및 클라이언트를 구성했습니다.

절차

1. IdM 클라이언트에서 **sssd.conf** 파일을 엽니다.

```
# vim /etc/sssd/sssd.conf
```

2. 도메인 섹션에서 **[domain/idm.example.com]** 과 같이 다음 옵션을 추가합니다.

```
krb5_auth_timeout = 60
```

3. **[pam]** 섹션에서 다음을 추가합니다.

```
p11_child_timeout = 60
```

4. SSSD 캐시를 지웁니다.

```
# sssctl cache-remove
SSSD must not be running. Stop SSSD now? (yes/no) [yes] yes
Creating backup of local data…
Removing cache files…
SSSD needs to be running. Start SSSD now? (yes/no) [yes] yes
```

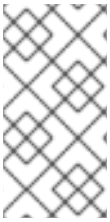
시간 초과를 늘리면 스마트 카드를 사용하여 다시 인증하십시오. 자세한 내용은 스마트 카드 인증 테스트를 참조하십시오.

9.5. 인증서 매핑 및 일치 규칙 문제 해결

스마트 카드로 인증하는 데 문제가 있는 경우 스마트 카드 인증서를 사용자에게 올바르게 연결했는지 확인합니다. 기본적으로 사용자 항목에 **usercertificate** 속성의 일부로 전체 인증서가 포함된 경우 인증서가

사용자와 연결됩니다. 그러나 인증서 매핑 규칙이 정의된 경우 인증서와 사용자 연결 방식이 변경될 수 있습니다. 인증서 매핑 및 일치 규칙의 문제를 해결하려면 다음 섹션을 참조하십시오.

- 인증서가 사용자에게 매핑되는 방법 확인
- 스마트 카드 인증서와 관련된 사용자 확인



참고

스마트 카드를 사용하여 SSH를 사용하여 인증하는 경우 전체 인증서를 IdM(Identity Management)의 사용자 항목에 추가해야 합니다. SSH를 사용하여 인증하기 위해 스마트 카드를 사용하지 않는 경우 **ipa user-add-certmapdata** 명령을 사용하여 인증서 매핑 데이터를 추가할 수 있습니다.

9.5.1. 인증서가 사용자에게 매핑되는 방법 확인

기본적으로 사용자 항목에 **usercertificate** 속성의 일부로 전체 인증서가 포함된 경우 인증서가 사용자와 연결됩니다. 그러나 인증서 매핑 규칙이 정의된 경우 인증서와 사용자 연결 방식이 변경될 수 있습니다. 인증서 매핑 규칙을 확인하려면 다음 절차를 따르십시오.

사전 요구 사항

- 스마트 카드에서 사용할 IdM(Identity Management) 서버 및 클라이언트를 설치하고 구성했습니다.
- 스마트 카드 리더를 감지하고 스마트 카드의 내용을 표시할 수 있습니다. 시스템의 스마트 카드 액세스 테스트를 참조하십시오.
- 스마트 카드 인증서를 IdM 사용자에게 매핑했습니다. 스마트 카드에서 인증을 구성하기 위한 인증서 매핑 규칙을 참조하십시오.

절차

1. 현재 IdM에 대해 구성된 인증서 매핑 규칙을 확인합니다.

```
# ipa certmaprule-find
----------------------------------------
1 Certificate Identity Mapping Rule matched
----------------------------------------
  Rule name: smartcardrule
  Mapping rule: (ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})
  Matching rule: <ISSUER>CN=Certificate Authority,O=IDM.EXAMPLE.COM
  Enabled: TRUE
----------------------------
Number of entries returned 1
----------------------------
```

다음과 같은 매핑 규칙 중 하나가 정의된 것을 확인할 수 있습니다.

- **ipacertmapdata** 는 IdM 사용자 항목 **certmapdata** 특성이 사용되었음을 나타냅니다.
- **altSecurityIdentities** 는 Active Directory의 사용자 항목 이름 매핑 특성이 사용되도록 지정합니다.
- **userCertificate;binary=** 는 IdM 또는 AD의 전체 인증서가 사용됨을 나타냅니다.

다양한 일치 옵션을 정의할 수 있지만 일반적으로 구성된 옵션 중 일부는 다음과 같습니다.

- **<ISSUER>CN=[…]** 는 사용 중인 인증서의 issuer 속성을 확인하여 이 값과 일치하는지 확인합니다.
- **<SUBJECT>.*,DC=MY,DC=DOMAIN** 은 인증서의 제목을 확인합니다.

2. IdM 서버의 **/etc/sssd/sssd.conf** 파일에 **debug_level = 9** 를 추가하여 SSSD(System Security Services Daemon) 로깅을 활성화합니다.

```
[domain/idm.example.com]
...
debug_level = 9
```

3. SSSD를 다시 시작하십시오.

```
# systemctl restart sssd
```

4. 매핑이 올바르게 표시되면 **/var/log/sssd/sssd_idm.example.com.log** 파일에 다음 항목이 표시됩니다.

```
[be[idm.example.com]] [sdap_setup_certmap] (0x4000): Trying to add rule [smartcardrule][-1]
[<ISSUER>CN=Certificate Authority,O=IDM.EXAMPLE.COM][(|(userCertificate;binary=
{cert!bin})(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500}))].
```

5. 매핑 규칙에 잘못된 구문이 포함된 경우 다음과 유사한 항목을 로그 파일에서 확인할 수 있습니다.

```
[be[idm.example.com]]] [sss_certmap_init] (0x0040): sss_certmap initialized.
[be[idm.example.com]]] [ipa_certmap_parse_results] (0x4000): Trying to add rule
[smartcardrule][-1][<ISSUER>CN=Certificate Authority,O=IDM.EXAMPLE.COM]
[(ipacertmapdata=X509:<I>{issuer_dn!x509}<S>{subject_dn})].
[be[idm.example.com]]] [parse_template] (0x0040): Parse template invalid.
[be[idm.example.com]]] [parse_ldap_mapping_rule] (0x0040): Failed to add template.
[be[idm.example.com]]] [parse_mapping_rule] (0x0040): Failed to parse LDAP mapping rule.
[be[idm.example.com]]] [ipa_certmap_parse_results] (0x0020): sss_certmap_add_rule failed
for rule [smartcardrule], skipping. Please check for typos and if rule syntax is supported.
[be[idm.example.com]]] [ipa_subdomains_certmap_done] (0x0040): Unable to parse certmap
results [22]: Invalid argument
[be[idm.example.com]]] [ipa_subdomains_refresh_certmap_done] (0x0020): Failed to read
certificate mapping rules [22]: Invalid argument
```

6. 매핑 규칙 구문을 확인합니다.

```
# ipa certmaprule-show smartcardrule
  Rule name: smartcardrule
  Mapping rule: (|(userCertificate;binary={cert!bin})(ipacertmapdata=X509:<I>
{issuer_dn!nss_x500}<S>{subject_dn!nss_x500}))
  Matching rule: <ISSUER>CN=Certificate Authority,O=IDM.EXAMPLE.COM
  Domain name: ipa.test
  Enabled: TRUE
```

7. 필요한 경우 인증서 매핑 규칙을 수정합니다.

```
# ipa certmaprule-mod smartcardrule --maprule '(ipacertmapdata=X509:<I>
{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})'
```

추가 리소스

- **sss-certmap** 매뉴얼 페이지를 참조하십시오.

9.5.2. 스마트 카드 인증서와 관련된 사용자 확인

스마트 카드로 인증하는 데 문제가 있는 경우 올바른 사용자가 스마트 카드 인증서와 연결되어 있는지 확인합니다.

사전 요구 사항

- 스마트 카드에서 사용할 IdM(Identity Management) 서버 및 클라이언트를 설치하고 구성했습니다.
- 스마트 카드 리더를 감지하고 스마트 카드의 내용을 표시할 수 있습니다. 시스템의 스마트 카드 액세스 테스트를 참조하십시오.
- 스마트 카드 인증서를 IdM 사용자에게 매핑했습니다. 스마트 카드에서 인증을 구성하기 위한 인증서 매핑 규칙을 참조하십시오.
- 스마트 카드의 인증서 사본 (예: **cert.pem**)이 있습니다.

절차

1. 사용자가 스마트 카드 인증서와 연결되어 있는지 확인합니다.

```
# ipa certmap-match cert.pem
--------------
1 user matched
--------------
  Domain: IDM.EXAMPLE.COM
  User logins: idmuser1
----------------------------
Number of entries returned 1
----------------------------
```

사용자 또는 도메인이 올바르지 않으면 인증서가 사용자에게 매핑되는 방법을 확인합니다. 인증서가 사용자에 매핑되는 방법 확인을 참조하십시오.

2. 사용자 항목에 인증서가 포함되어 있는지 확인합니다.

```
# ipa user-show idmuser1
  User login: idmuser1
[...]
Certificate:MIIEejCCAuKgAwIBAgIBCzANBgkqhkiG9w0BAQsFADAzMREwDwYDVQQKDAhJ
UEEuVEVTVDEeMBwGA1UEAwwVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XD
```

3. 사용자 항목에 인증서가 없으면 사용자 항목에 base-64로 인코딩된 인증서를 추가합니다.

   a. **ipa user-add-cert** 명령에서 필요한 형식인 헤더와 바닥글이 제거되고 한 줄로 연결된 인증서가 포함된 환경 변수를 생성합니다.

```
$ export CERT=`openssl x509 -outform der -in idmuser1.crt | base64 -w0 -`
```

idmuser1.crt 파일의 인증서가 PEM 형식이어야 합니다.

b. **ipa user-add-cert** 명령을 사용하여 **idmuser1** 의 프로필에 인증서를 추가합니다.

```
$ ipa user-add-cert idmuser1 --certificate=$CERT
```

c. SSSD(System Security Services Daemon) 캐시를 지웁니다.

```
# sssctl cache-remove
SSSD must not be running. Stop SSSD now? (yes/no) [yes] yes
Creating backup of local data…
Removing cache files…
SSSD needs to be running. Start SSSD now? (yes/no) [yes] yes
```

4. **ipa certmap-match** 를 다시 실행하여 사용자가 스마트 카드 인증서와 연결되어 있는지 확인합니다.