Red Hat Enterprise Linux 8

スマートカード認証の管理

スマートカード認証の設定と使用

Last Updated: 2024-04-10

Red Hat Enterprise Linux 8 スマートカード認証の管理

スマートカード認証の設定と使用

法律上の通知

概要

Red Hat Identity Management (IdM) を使用すると、秘密鍵と証明書の形で認証情報をスマートカードに保存できます。これにより、パスワードの代わりにこのスマートカードを使用してサービスに対する認証を行うことができます。管理者はマッピングルールを設定して、管理オーバーヘッドを削減できます。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施してまいります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

Identity Management では、次のような用語の置き換えが計画されています。

- **ブラックリスト** から **ブロックリスト**
- **ホワイトリスト** から **許可リスト**
- **スレーブ** から **セカンダリー**
- **マスター** という言葉は、文脈に応じて、より正確な言葉に置き換えられています。
  - **IdM マスター** から **IdM サーバー**
  - **CA 更新マスター** から **CA 更新サーバー**
  - **CRL マスター** から **CRL パブリッシャーサーバー**
  - **マルチマスター** から **マルチサプライヤー**

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 スマートカード認証について

スマートカードに基づいた認証は、パスワードの代替手段です。ユーザー認証情報は、秘密鍵と証明書の形式でスマートカードに格納され、特別なソフトウェアやハードウェアを使用してその鍵にアクセスします。スマートカードをリーダーまたは USB ポートに挿入して、パスワードを入力する代わりに、スマートカードの PIN コードを入力します。

このセクションでは、スマートカードとは何か、およびスマートカード認証のしくみについて説明します。スマートカードコンテンツの読み取りと操作に使用できるツールについて説明します。また、サンプルのユースケースを提供し、スマートカード認証のための IdM サーバーと IdM クライアントの両方のセットアップについて説明します。

注記

スマートカード認証の使用を開始する場合は、ハードウェア要件 Smart Card support in RHEL8 を参照してください。

1.1. スマートカードとは

スマートカードは物理デバイスであり、通常はマイクロプロセッサーを搭載したプラスチックカードであり、カードに格納された証明書を使用して個人認証を行うことができます。個人認証とは、ユーザーパスワードと同じ方法でスマートカードを使用できることを意味します。

ユーザー認証情報は、秘密鍵と証明書の形式でスマートカードに格納され、特別なソフトウェアやハードウェアを使用してその鍵にアクセスします。スマートカードをリーダーまたは USB ソケットに挿入して、パスワードを入力する代わりに、スマートカードの PIN コードを入力します。

1.2. スマートカード認証とは

公開鍵ベースの認証と証明書ベースの認証は、パスワードベースの認証に代わるものとして広く使用されています。パスワードの代わりに公開鍵と秘密鍵を使用して、身元が確認されます。証明書とは、個人、サーバー、会社、または他のエンティティーを特定し、そのアイデンティティーを公開鍵に関連付けるために使用される電子ドキュメントです。運転免許または乗車のように、証明書は、一般的にユーザーのアイデンティティーを証明する証明を提供します。公開鍵暗号では、証明書を使用してなりすましの問題に対処します。

スマートカード認証の場合、ユーザー認証情報 (公開鍵と秘密鍵、および証明書) はスマートカードに保存され、スマートカードがリーダーに挿入されて PIN が提供された後にのみ使用できます。物理デバイスであるスマートカードを所有し、その PIN を知る必要があるため、スマートカード認証は 2 要素認証の一種と見なされます。

1.2.1. IdM でのスマートカード認証の例

以下の例では、IdM でスマートカードを使用する方法に関する 2 つの簡単なシナリオについて説明します。

1.2.1.1. スマートカードを使用してシステムにログインする

スマートカードを使用して、ローカルユーザーとして RHEL システムに認証することができます。システムがスマートカードログインを強制するように設定されている場合、スマートカードを挿入してその PIN を入力するように求められます。それが失敗すると、システムにログインできません。または、スマートカード認証またはユーザー名とパスワードのいずれかを使用して認証するようにシステムを設定することもできます。この場合、スマートカードが挿入されていないと、ユーザー名とパスワードの入力を求められます。

1.2.1.2. ロックを解除して GDM にログインする

RHEL システムでスマートカード認証を設定している場合は、ロックオンリムーバル機能を有効にすることができます。GNOME Display Manager (GDM) にログインしているときにスマートカードを取り外すと、画面ロックが有効になり、スマートカードを再度挿入し、PIN で認証して画面のロックを解除する必要があります。ユーザー名とパスワードを使用して認証することはできません。

注記

GDM にログインしているときにスマートカードを取り外すと、画面ロックが有効になり、スマートカードを再度挿入し、PIN で認証して画面のロックを解除する必要があります。

1.3. RHEL のスマートカード認証オプション

authselect コマンド **authselect enable-feature <smartcardoption>** を使用して、特定の Identity Management (IdM) クライアントでスマートカード認証を機能させる方法を設定できます。次のスマートカードオプションを使用できます。

- **with-smartcard**: ユーザーは、ユーザー名とパスワード、またはスマートカードで認証できます。
- **with-smartcard-required**: ユーザーはスマートカードで認証でき、パスワード認証は無効になります。スマートカードがないとシステムにアクセスできません。スマートカードで認証されると、スマートカードがリーダーから取り外されてもログイン状態を維持できます。

  注記

  with-smartcard-required オプションは、**login**、**gdm**、**xdm**、**xscreensaver**、および **gnome-screensaver** などのログインサービスに対してのみ、排他的なスマートカード認証を適用します。ユーザーを切り替えるための **su** や **sudo** などの他のサービスでは、スマートカード認証は適用されず、スマートカードが挿入されていない場合は、パスワードの入力を求められます。

- **with-smartcard-lock-on-removal**: ユーザーはスマートカードで認証できます。ただし、リーダーからスマートカードを取り外すと、システムから自動的にロックアウトされます。パスワード認証は使用できません。

  注記

  with-smartcard-lock-on-removal オプションは、GNOME デスクトップ環境を持つシステムでのみ機能します。**tty** またはコンソールベースのシステムを使用しており、そのリーダーからスマートカードを削除すると、システムから自動的にロックアウトされません。

詳しくは、authselect を使用したスマートカードの設定 を参照してください。

1.4. スマートカードとその内容を管理するためのツール

さまざまなツールを使用して、スマートカードに格納されているキーと証明書を管理できます。これらのツールを使用して、次のことができます。

- システムに接続されている使用可能なスマートカードリーダーをリスト表示します。

- 利用可能なスマートカードをリスト表示し、その内容を表示します。
- キーと証明書であるスマートカードの内容を操作します。

同様の機能を提供するツールは多数ありますが、システムの異なるレイヤーで機能するツールもあります。スマートカードは、複数のコンポーネントによって複数のレイヤーで管理されます。下位レベルでは、オペレーティングシステムは PC/SC プロトコルを使用してスマートカードリーダーと通信し、この通信は pcsc-lite デーモンによって処理されます。デーモンは、受信したコマンドを通常は USB 経由でスマートカードリーダーに転送します。これは、低レベルの CCID ドライバーによって処理されます。PC/SC の低レベル通信は、アプリケーションレベルではほとんど見られません。アプリケーションがスマートカードにアクセスするための RHEL での主な方法は、高レベルのアプリケーションプログラミングインターフェイス (API) である OASIS PKCS#11 API を使用することです。この API は、秘密鍵などの暗号化オブジェクトを操作する特定のコマンドにカード通信を抽象化します。スマートカードベンダーは、PKCS#11 API に従い、スマートカードのドライバーとして機能する **.so** ファイルなどの共有モジュールを提供します。

次のツールを使用して、スマートカードとその内容を管理できます。

- OpenSC ツール: **opensc** に実装されているドライバーと連携します。
  - opensc-tool: スマートカード操作を実行します。
  - pkcs15-tool: スマートカードの PKCS#15 データ構造を管理します (トークンに保存されている PIN、キー、および証明書のリスト表示と読み取りなど)。
  - pkcs11-tool: スマートカード上の PKCS#11 データオブジェクトを管理します。たとえば、トークンに格納されている PIN、キー、および証明書のリスト表示と読み取りを行います。
- GnuTLS ユーティリティー: アプリケーションがネットワークトランスポート層を介した安全な通信を可能にするための API と、X.509、PKCS#12、OpenPGP、およびその他の構造にアクセスするためのインターフェイス。
  - p11tool: PKCS#11 スマートカードとセキュリティーモジュールで操作を実行します。
  - certtool: X.509 証明書、要求、秘密鍵を解析して生成します。
- ネットワークセキュリティーサービス (NSS) ツール: セキュリティー対応のクライアントおよびサーバーアプリケーションのクロスプラットフォーム開発をサポートするように設計された一連のライブラリー。NSS で構築されたアプリケーションは、SSL v3、TLS、PKCS #5、PKCS #7、PKCS #11、PKCS #12、S/MIME、X.509 v3 証明書、およびその他のセキュリティー標準をサポートできます。
  - modutil: PKCS#11 モジュール情報をセキュリティーモジュールデータベースで管理します。
  - certutil: NSS データベースと他の NSS トークンの両方でキーと証明書を管理します。

これらのツールを使用し、スマートカードを使用した認証に関する問題をトラブルシューティングする方法の詳細は、スマートカードによる認証のトラブルシューティング を参照してください。

関連情報

- **opensc-tool** man ページ
- **pkcs15-tool** man ページ

- **pkcs11-tool** man ページ
- **p11tool** man ページ
- **certtool** man ページ
- **modutil** man ページ
- **certutil** man ページ

1.5. 証明書とスマートカード認証

Identity Management (IdM) または Active Directory (AD) を使用してドメイン内の ID ストア、認証、ポリシー、および認可ポリシーを管理する場合、認証に使用される証明書はそれぞれ IdM または AD によって生成されます。外部認証局から提供された証明書を使用することもできます。この場合、外部プロバイダーからの証明書を受け入れるように Active Directory または IdM を設定する必要があります。ユーザーがドメインに属していない場合は、ローカル認証局によって生成された証明書を使用できます。詳細は、次のセクションを参照してください。

- スマートカード認証用の Identity Management の設定
- IdM でスマートカード認証用に ADCS が発行した証明書の設定
- IdM ユーザー、ホスト、およびサービスの外部署名証明書の管理
- ローカル証明書のスマートカードへの設定およびインポート

スマートカード認証に適格な証明書の完全なリストについては、スマートカードに適格な証明書 を参照してください。

1.6. IDM でのスマートカード認証に必要な手順

Identity Management (IdM) でスマートカードを使用して認証する前に、以下の手順に従っていることを確認する必要があります。

- スマートカード認証用に IdM サーバーを設定します。スマートカード認証用の IdM サーバーの設定 を参照してください。
- IdM クライアントをスマートカード認証用に設定します。スマートカード認証用の IdM クライアントの設定 を参照してください。
- IdM のユーザーエントリーに証明書を追加します。IdM Web UI のユーザーエントリーへの証明書の追加 を参照してください。
- キーと証明書をスマートカードに保存します。スマートカードでの証明書の保存 を参照してください。

1.7. ACTIVE DIRECTORY によって発行された証明書を使用したスマートカード認証に必要な手順

Active Directory (AD) によって発行された証明書を使用してスマートカードで認証する前に、次の手順に従っていることを確認する必要があります。

- Active Directory から IdM サーバーおよびクライアントへの CA 証明書およびユーザー証明書のコピー

- ADCS 証明書を使用したスマートカード認証用の IdM サーバーおよびクライアントの設定
- PFX (PKCS#12) ファイルを変換して、証明書と秘密鍵をスマートカードに保存できるようにします。
- sssd.conf ファイルでタイムアウトを設定します。
- スマートカード認証用の証明書マッピングルールの作成

第2章 スマートカード認証用の IDENTITY MANAGEMENT の設定

Identity Management (IdM) では、以下によるスマートカード認証に対応しています。

- IdM 認証局が発行するユーザー証明書
- 外部認証局が発行するユーザー証明書

両方のタイプの証明書に対して、IdM でスマートカード認証を設定できます。このシナリオでは、**rootca.pem** CA 証明書は、信頼された外部の認証局の証明書を含むファイルです。

IdM でのスマートカード認証については、スマートカード認証について を参照してください。

スマートカード認証の設定に関する詳細は、以下を参照してください。

- スマートカード認証用の IdM サーバーの設定
- スマートカード認証用の IdM クライアントの設定
- IdM Web UI のユーザーエントリーへの証明書の追加
- IdM CLI でユーザーエントリーへの証明書の追加
- スマートカードを管理および使用するツールのインストール
- スマートカードでの証明書の保存
- スマートカードを使用して IdM へのログイン
- スマートカード認証で GDM アクセスの設定
- スマートカード認証で su アクセスの設定

2.1. スマートカード認証用の IDM サーバーの設定

Identity Management (IdM) CA が信頼する <EXAMPLE.ORG> ドメインの認証局 (CA) によって証明書が発行されたユーザーに対してスマートカード認証を有効にする場合は、次の証明書を取得し、IdM サーバーを設定する **ipa-advise** スクリプトを実行するときにそれらを追加できるようにする必要があります。

- <EXAMPLE.ORG> CA の証明書を直接、または 1 つ以上のサブ CA を通じて発行した root CA の証明書。証明書チェーンは、その CA が証明書を発行した Web ページからダウンロードできます。詳細は、証明書認証を有効にするためのブラウザーの設定 の手順 1 - 4a を参照してください。
- IdM CA 証明書。IdM CA インスタンスが実行されている IdM サーバーの **/etc/ipa/ca.crt** ファイルから CA 証明書を取得できます。
- すべての中間 CA、つまり <EXAMPLE.ORG> CA と IdM CA の中間 CA の証明書。

スマートカード認証用に IdM サーバーを設定するには、以下を行います。

1. PEM 形式の CA 証明書を含むファイルを取得します。
2. ビルトイン **ipa-advise** スクリプトを実行します。
3. システム設定をリロードします。

前提条件

- IdM サーバーへの root アクセス権限がある。
- ルート CA 証明書とすべての中間 CA 証明書があります。

手順

1. 設定を行うディレクトリーを作成します。

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

2. そのディレクトリーに移動します。

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

3. PEM 形式のファイルに保存されている関連する CA 証明書を取得します。CA 証明書が DER などの異なる形式のファイルに保存されている場合は、これを PEM 形式に変換します。IdM 認証局の証明書は PEM 形式で、**/etc/ipa/ca.crt** ファイルにあります。
DER ファイルを PEM ファイルに変換します。

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

4. 便宜上、設定を行うディレクトリーに証明書をコピーします。

```
[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. 必要に応じて、外部の認証局の証明書を使用する場合は、**openssl x509** ユーティリティーを使用して **PEM** 形式のファイルの内容を表示し、**Issuer** および **Subject** の値が正しいことを確認します。

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

6. 管理者の権限を使用して、組み込みの **ipa-advise** ユーティリティーで設定スクリプトを生成します。

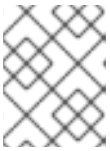
```
[root@server SmartCard]# kinit admin
[root@server SmartCard]# ipa-advise config-server-for-smart-card-auth > config-server-for-smart-card-auth.sh
```

config-server-for-smart-card-auth.sh スクリプトは、以下の操作を実行します。

- IdM Apache HTTP サーバーを設定します。
- キー配布センター (KDC) の Kerberos (PKINIT) で、初回認証用の公開鍵暗号化機能を有効にします。
- スマートカード認可要求を受け入れるように IdM Web UI を設定します。

7. スクリプトを実行し、root CA と サブ CA 証明書が含まれる PEM ファイルを引数として追加します。

```
[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
```



注記

ルート CA 証明書を、サブ CA 証明書の前に引数として追加します。また、CA またはサブ CA 証明書の有効期限が切れていないことを確認します。

8. 必要に応じて、ユーザー証明書を発行した認証局が Online Certificate Status Protocol (OCSP) レスポンダーを提供しない場合は、IdM Web UI への認証に対する OCSP チェックを無効にすることが必要になる場合があります。

   a. **/etc/httpd/conf.d/ssl.conf** ファイルで **SSLOCSPEnable** パラメーターを **off** に設定します。

   ```
   SSLOCSPEnable off
   ```

   b. 変更をすぐに有効にするには、Apache デーモン (httpd) を再起動します。

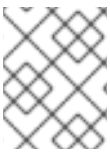
   ```
   [root@server SmartCard]# systemctl restart httpd
   ```

警告

IdM CA が発行したユーザー証明書のみを使用する場合は、OCSP チェックを無効にしないでください。OCSP レスポンダーは IdM に含まれます。

ユーザー証明書を発行した CA が、OCSP サービスリクエストをリッスンする場所に関する情報がユーザー証明書に含まれていない場合に、OCSP チェックを有効にしたまま、ユーザー証明書が IdM サーバーにより拒否されないようにする方法は、Apache mod_ssl 設定オプションの **SSLOCSPDefaultResponder** ディレクティブを参照してください。

これで、スマートカード認証にサーバーが設定されました。



注記

トポロジー全体でスマートカード認証を有効にするには、各 IdM サーバーで手順を実行します。

2.2. ANSIBLE を使用したスマートカード認証用の IDM サーバー設定

Ansible を使用して、Identity Management (IdM) CA が信頼する <EXAMPLE.ORG> ドメインの認証局 (CA) によって証明書が発行されたユーザーのスマートカード認証を有効にできます。そのために

は、**ipasmartcard_server ansible-freeipa** ロールスクリプトを使用して Ansible Playbook を実行するときに使用できるように、次の証明書を取得する必要があります。

- <EXAMPLE.ORG> CA の証明書を直接、または 1 つ以上のサブ CA を通じて発行した root CA の証明書。証明書チェーンは、その CA が証明書を発行した Web ページからダウンロードできます。詳細は、証明書認証を有効にするためのブラウザーの設定 の手順 4 を参照してください。
- IdM CA 証明書。CA 証明書は、任意の IdM CA サーバーの **/etc/ipa/ca.crt** ファイルから取得できます。
- <EXAMPLE.ORG> CA と IdM CA の中間にあるすべての CA の証明書。

前提条件

- IdM サーバーへの **root** アクセス権限がある。
- IdM **admin** のパスワードを把握している。
- ルート CA 証明書、IdM CA 証明書、すべての中間 CA の証明書がある。
- 次の要件を満たすように Ansible コントロールノードを設定している。
  - Ansible バージョン 2.14 以降を使用している。
  - Ansible コントローラーに **ansible-freeipa** パッケージがインストールされている。
  - この例では、**~/MyPlaybooks/** ディレクトリーに、IdM サーバーの完全修飾ドメイン名 (FQDN) を使用して Ansible インベントリーファイル を作成したことを前提としている。
  - この例では、**secret.yml** Ansible ボールトに **ipaadmin_password** が保存されていることを前提としている。
- ターゲットノード (**ansible-freeipa** モジュールが実行されるノード) は、IdM クライアント、サーバー、またはレプリカとして IdM ドメインの一部です。

手順

1. CA 証明書が **DER** などをはじめとする別の形式のファイルに保存されている場合、それらを **PEM** 形式に変換します。

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

   IdM 認証局の証明書は **PEM** 形式で、**/etc/ipa/ca.crt** ファイルにあります。

2. 必要に応じて、**openssl x509** ユーティリティーを使用して **PEM** 形式のファイルの内容を表示し、**Issuer** および **Subject** の値が正しいことを確認します。

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

3. **~/MyPlaybooks/** ディレクトリーに移動します。

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

4. CA 証明書専用のサブディレクトリーを作成します。

```
$ mkdir SmartCard/
```

5. 便宜上、必要なすべての証明書を **~/MyPlaybooks/SmartCard/** ディレクトリーにコピーします。

```
# 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 インベントリーファイルで、以下を指定します。
   - スマートカード認証用に設定する IdM サーバー。
   - IdM 管理者パスワード。
   - CA の証明書へのパス (次の順序に従う)。
     - ルート CA 証明書ファイル
     - 中間 CA 証明書ファイル
     - IdM CA 証明書ファイル

   ファイルは次のようになります。

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

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

[ipacluster:children]
ipaserver
ipareplicas

[ipacluster:vars]
ipaadmin_password= "{{ ipaadmin_password }}"
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. 次の内容で **install-smartcard-server.yml** playbook を作成します。

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

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

8. ファイルを保存します。

9. Ansible Playbook を実行します。Playbook ファイル、**secret.yml** ファイルを保護するパスワードを格納するファイル、およびインベントリーファイルを指定します。

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

ipasmartcard_server Ansible ロールは、次のアクションを実行します。

- IdM Apache HTTP サーバーを設定します。
- キー配布センター (KDC) の Kerberos (PKINIT) で、初回認証用の公開鍵暗号化機能を有効にします。
- スマートカード認可要求を受け入れるように IdM Web UI を設定します。

10. 必要に応じて、ユーザー証明書を発行した認証局が Online Certificate Status Protocol (OCSP) レスポンダーを提供しない場合は、IdM Web UI への認証に対する OCSP チェックを無効にすることが必要になる場合があります。

a. **root** として IdM サーバーに接続します。

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

b. **/etc/httpd/conf.d/ssl.conf** ファイルで **SSLOCSPEnable** パラメーターを **off** に設定します。

```
SSLOCSPEnable off
```

c. 変更をすぐに有効にするには、Apache デーモン (httpd) を再起動します。

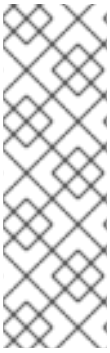
```
# systemctl restart httpd
```

警告

IdM CA が発行したユーザー証明書のみを使用する場合は、OCSP チェックを無効にしないでください。OCSP レスポンダーは IdM に含まれます。

ユーザー証明書を発行した CA が、OCSP サービスリクエストをリッスンする場所に関する情報がユーザー証明書に含まれていない場合に、OCSP チェックを有効にしたまま、ユーザー証明書が IdM サーバーにより拒否されないようにする方法は、Apache mod_ssl 設定オプションの **SSLOCSPDefaultResponder** ディレクティブを参照してください。

これで、インベントリーファイルにリストされているサーバーがスマートカード認証用に設定されました。



注記

トポロジー全体でスマートカード認証を有効にするには、Ansible Playbook の **hosts** 変数を **ipacluster** に設定します。

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

関連情報

- **/usr/share/doc/ansible-freeipa/playbooks/** ディレクトリーで **ipasmartcard_server** ロールを使用するサンプル Playbook

2.3. スマートカード認証用の IDM クライアントの設定

以下の手順に従って、スマートカード認証用に IdM クライアントを設定します。この手順は、認証にスマートカードを使用しているときに接続する各 IdM システム、クライアント、またはサーバーで実行する必要があります。たとえば、ホスト A からホスト B への **ssh** 接続を有効にするには、スクリプトをホスト B で実行する必要があります。

管理者として、以下を使用して、この手順でスマートカード認証を有効にします。

- **ssh** プロトコル
  詳細は、スマートカード認証で SSH アクセスの設定 を参照してください。
- コンソールのログイン
- GNOME Display Manager (GDM)
- **su** コマンド

この手順は、IdM Web UI に対する認証には必要ありません。IdM Web UI の認証には 2 つのホストが関係しますが、どちらも IdM クライアントである必要はありません。

- ブラウザーが実行されているマシン。マシンは IdM ドメインの外にある場合があります。
- **httpd** が実行している IdM サーバー

以下の手順は、IdM サーバーではなく、IdM クライアントでスマートカード認証を設定していることを前提としています。このため、2 台のコンピューターが必要です。設定スクリプトを生成する IdM サーバーと、スクリプトを実行する IdM クライアントが必要になります。

前提条件

- Configuring the IdM server for smart card authentication に従って、IdM サーバーがスマートカード認証用に設定されている。
- IdM サーバーと IdM クライアントに root アクセス権限がある。
- ルート CA 証明書とすべての中間 CA 証明書があります。

- **--mkhomedir** オプションを使用して IdM クライアントをインストールし、リモートユーザーが正常にログインできるようにしている。ホームディレクトリーを作成しない場合、デフォルトのログイン場所はディレクトリー構造のルート / になります。

手順

1. IdM サーバーで、管理者権限を使用して、**ipa-advise** で設定スクリプトを生成します。

```
[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 スクリプトは、以下の操作を実行します。

- スマートカードデーモンを設定する。
- システム全体のトラストストアを設定します。
- System Security Services Daemon (SSSD) を設定して、ユーザーがユーザー名とパスワード、またはスマートカードで認証できるようにします。スマートカード認証用の SSSD プロファイルオプションの詳細は、RHEL のスマートカード認証オプション を参照してください。

2. IdM サーバーから、IdM クライアントマシンの任意のディレクトリーに、スクリプトをコピーします。

```
[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. 便宜上、IdM サーバーから、上の手順で使用した IdM クライアントマシンのディレクトリーに、PEM 形式の CA 証明書ファイルをコピーします。

```
[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. クライアントマシンで、スクリプトを実行し、CA 証明書を含む PEM ファイルを引数として追加します。

```
[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
```

注記

ルート CA 証明書を、サブ CA 証明書の前に引数として追加します。また、CA またはサブ CA 証明書の有効期限が切れていないことを確認します。

これで、クライアントがスマートカード認証に対して設定されました。

2.4. ANSIBLE を使用したスマートカード認証用の IDM クライアント設定

ansible-freeipa ipasmartcard_client モジュールを使用して特定の Identity Management (IdM) クライアントを設定し、IdM ユーザーがスマートカードで認証できるようにするには、次の手順に従います。この手順を実行し、以下のいずれかを使用して IdM にアクセスする IdM ユーザーのスマートカード認証を有効にします。

- **ssh** プロトコル
  詳細は、スマートカード認証で SSH アクセスの設定 を参照してください。
- コンソールのログイン
- GNOME Display Manager (GDM)
- **su** コマンド

注記

この手順は、IdM Web UI に対する認証には必要ありません。IdM Web UI の認証には 2 つのホストが関係しますが、どちらも IdM クライアントである必要はありません。

- ブラウザーが実行されているマシン。マシンは IdM ドメインの外にある場合があります。
- **httpd** が実行している IdM サーバー

前提条件

- Ansible を使用したスマートカード認証用の IdM サーバー設定 に説明されているとおり、IdM サーバーがスマートカード認証用に設定されている。
- IdM サーバーと IdM クライアントに root アクセス権限がある。
- ルート CA 証明書、IdM CA 証明書、すべての中間 CA の証明書がある。
- 次の要件を満たすように Ansible コントロールノードを設定している。
  - Ansible バージョン 2.14 以降を使用している。
  - Ansible コントローラーに **ansible-freeipa** パッケージがインストールされている。
  - この例では、**~/MyPlaybooks/** ディレクトリーに、IdM サーバーの完全修飾ドメイン名 (FQDN) を使用して Ansible インベントリーファイル を作成したことを前提としている。
  - この例では、**secret.yml** Ansible ボールトに **ipaadmin_password** が保存されていることを前提としている。
- ターゲットノード (**ansible-freeipa** モジュールが実行されるノード) は、IdM クライアント、サーバー、またはレプリカとして IdM ドメインの一部です。

手順

1. CA 証明書が **DER** などをはじめとする別の形式のファイルに保存されている場合、それらを **PEM** 形式に変換します。

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

IdM CA 証明書は **PEM** 形式で、**/etc/ipa/ca.crt** ファイルにあります。

2. 必要に応じて、**openssl x509** ユーティリティーを使用して **PEM** 形式のファイルの内容を表示し、**Issuer** および **Subject** の値が正しいことを確認します。

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

3. Ansible コントロールノードで、**~/MyPlaybooks/** ディレクトリーに移動します。

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

4. CA 証明書専用のサブディレクトリーを作成します。

```
$ mkdir SmartCard/
```

5. 便宜上、必要なすべての証明書を **~/MyPlaybooks/SmartCard/** ディレクトリーにコピーします。以下はその例です。

```
# 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 インベントリーファイルで、以下を指定します。

- スマートカード認証用に設定する IdM クライアント。
- IdM 管理者パスワード。
- CA の証明書へのパス (次の順序に従う)。
  - ルート CA 証明書ファイル
  - 中間 CA 証明書ファイル
  - IdM CA 証明書ファイル

ファイルは次のようになります。

```
[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. 次の内容で **install-smartcard-clients.yml** playbook を作成します。

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

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

8. ファイルを保存します。

9. Ansible Playbook を実行します。Playbook とインベントリーファイルを指定します。

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

ipasmartcard_client Ansible ロールは、次のアクションを実行します。

- スマートカードデーモンを設定する。
- システム全体のトラストストアを設定します。
- System Security Services Daemon (SSSD) を設定して、ユーザーがユーザー名とパスワード、またはスマートカードで認証できるようにします。スマートカード認証用の SSSD プロファイルオプションの詳細は、RHEL のスマートカード認証オプション を参照してください。

これで、インベントリーファイルの **ipaclients** セクションにリストされているクライアントがスマートカード認証用に設定されました。

注記

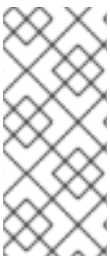
--mkhomedir オプションを使用して IdM クライアントをインストールしている場合、リモートユーザーはホームディレクトリーにログインできます。それ以外の場合、デフォルトのログイン場所はディレクトリー構造のルート / です。

関連情報

- **/usr/share/doc/ansible-freeipa/playbooks/** ディレクトリーで **ipasmartcard_server** ロールを使用するサンプル Playbook

2.5. IDM WEB UI のユーザーエントリーへの証明書の追加

IdM Web UI のユーザーエントリーに外部証明書を追加するには、以下の手順に従います。



注記

証明書全体をアップロードする代わりに、IdM のユーザーエントリーに証明書マッピングデータをアップロードすることもできます。システム管理者向けのスマートカード認証の設定を容易にするために、完全な証明書または証明書マッピングデータのいずれかが含まれるユーザーエントリーを、対応する証明書マッピングルールと併用できます。詳細は、認証を設定するための証明書マッピングルール を参照してください。

注記

ユーザーの証明書が IdM 認証局によって発行された場合、証明書はユーザーエントリーにすでに保存されているため、この手順に従う必要はありません。

前提条件

- ユーザーエントリーに追加できる証明書がある。

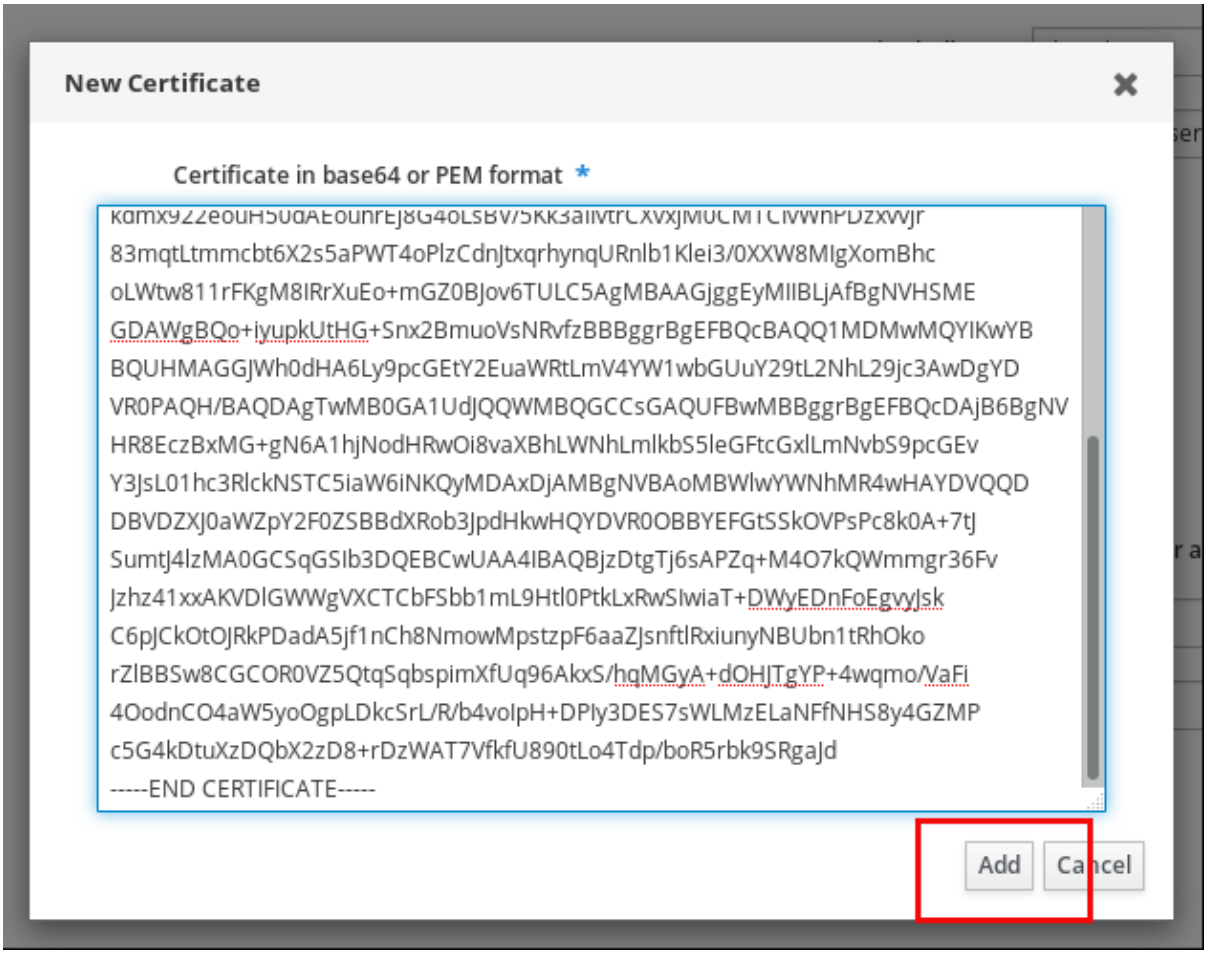
手順

1. 別のユーザーに証明書を追加する場合は、管理者として IdM Web UI にログインします。独自のプロファイルに証明書を追加する場合は、管理者の認証情報が必要ありません。
2. **Users → Active users → sc_user** の順に移動します。
3. **Certificate** オプションを探して、**Add** をクリックします。
4. コマンドラインインターフェイスで、**cat** ユーティリティーまたはテキストエディターを使用して、**PEM** 形式の証明書を表示します。

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

5. CLI で、証明書をコピーし、Web UI で開いたウィンドウにこれを貼り付けます。
6. **Add** をクリックします。

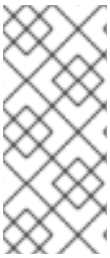
図2.1 IdM Web UI で新しい証明書の追加



sc_user エントリーに外部証明書が含まれるようになりました。

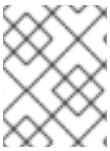
2.6. IDM CLI でユーザーエントリーへの証明書の追加

IdM CLI のユーザーエントリーに外部証明書を追加するには、以下の手順に従います。



注記

証明書全体をアップロードする代わりに、IdM のユーザーエントリーに証明書マッピングデータをアップロードすることもできます。システム管理者向けのスマートカード認証の設定を容易にするために、完全な証明書または証明書マッピングデータのいずれかが含まれるユーザーエントリーを、対応する証明書マッピングルールと併用できます。詳細は、認証を設定するための証明書マッピングルール を参照してください。



注記

ユーザーの証明書が IdM 認証局によって発行された場合、証明書はユーザーエントリーにすでに保存されているため、この手順に従う必要はありません。

前提条件

- ユーザーエントリーに追加できる証明書がある。

手順

1. 別のユーザーに証明書を追加する場合は、管理者として IdM Web CLI にログインします。

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

独自のプロファイルに証明書を追加する場合は、管理者の認証情報は必要ありません。

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

2. ヘッダーとフッターのある証明書を含む環境変数を作成し、1 行に連結します。これは、**ipa user-add-cert** コマンドで必要な形式です。

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

testuser.crt ファイルの証明書は、**PEM** 形式である必要があることに注意してください。

3. **ipa user-add-cert** コマンドを使用して、sc_user のプロファイルに証明書を追加します。

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

sc_user エントリーに外部証明書が含まれるようになりました。

2.7. スマートカードを管理および使用するツールのインストール

前提条件

- **gnutls-utils** パッケージがインストールされている。

- **opensc** パッケージがインストールされている。
- **pcscd** サービスを実行している。

スマートカードを設定する前に、対応するツール (証明書を生成して **pscd** サービスを起動できるもの) をインストールする必要があります。

手順

1. **opensc** パッケージおよび **gnutls-utils** パッケージをインストールします。

```
# {PackageManagerCommand} -y install opensc gnutls-utils
```

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

検証手順

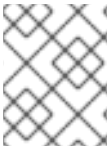
- **pcscd** サービスが稼働していることを確認します。

```
# systemctl status pcscd
```

2.8. スマートカードを準備し、証明書と鍵をスマートカードにアップロードする

pkcs15-init ツールを使用してスマートカードを設定するには、この手順に従います。このツールは、以下を設定するのに役立ちます。

- スマートカードの消去
- 新しい PIN および任意の PIN ブロック解除キー (PUK) の設定
- スマートカードでの新規スロットの作成
- スロットへの証明書、秘密鍵、および公開鍵の保存
- 必要に応じて、特定のスマートカードではこのタイプのファイナライズが必要なため、スマートカードの設定をロックします。



注記

pkcs15-init ツールは、すべてのスマートカードで機能するとは限りません。使用しているスマートカードで動作するツールを使用する必要があります。

前提条件

- **pkcs15-init** ツールを含む **opensc** パッケージがインストールされている。
  詳細は スマートカードを管理および使用するツールのインストール を参照してください。
- カードがリーダーに挿入され、コンピューターに接続されている。

- スマートカードに保存する秘密鍵、公開鍵、および証明書がある。この手順の **testuser.key**、**testuserpublic.key**、および **testuser.crt** は、秘密鍵、公開鍵、および証明書に使用される名前です。
- 現在のスマートカードユーザー PIN およびセキュリティーオフィス PIN (SO-PIN)

手順

1. スマートカードを消去して PIN で自身を認証します。

```
$ 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]:
```

カードが削除されました。

2. スマートカードを初期化し、ユーザーの PIN と PUK を設定します。また、セキュリティー担当者の PIN と PUK を設定します。

```
$ 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 ツールは、スマートカードに新しいスロットを作成します。

3. スロットのラベルと認証 ID を設定します。

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

ラベルは人間が判読できる値に設定されます (この場合は **testuser**)。**auth-id** は 16 進数の値である必要があります。この場合、**01** に設定されます。

4. スマートカードの新しいスロットに秘密鍵を保存し、ラベルを付けます。

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

注記

--id に指定する値は、秘密鍵を保存するときと、次の手順で証明書を保存するときに同じである必要があります。**--id** に独自の値を指定することを推奨します。そうしないと、より複雑な値がツールによって計算されます。

5. スマートカードの新しいスロットに証明書を保存し、ラベル付けします。

```
$ 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. (必要に応じて) スマートカードの新しいスロットに公開鍵を保存し、ラベルを付けます。

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

注記

公開鍵が秘密鍵または証明書に対応する場合は、秘密鍵または証明書の ID と同じ ID を指定します。

7. (オプション) スマートカードの中には、設定をロックしてカードを最終処理する必要があります。

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

この段階では、スマートカードには、新たに作成されたスロットに証明書、秘密鍵、および公開鍵が含まれます。ユーザーの PIN と PUK、およびセキュリティー担当者の PIN と PUK も作成しました。

2.9. スマートカードを使用して IDM へのログイン

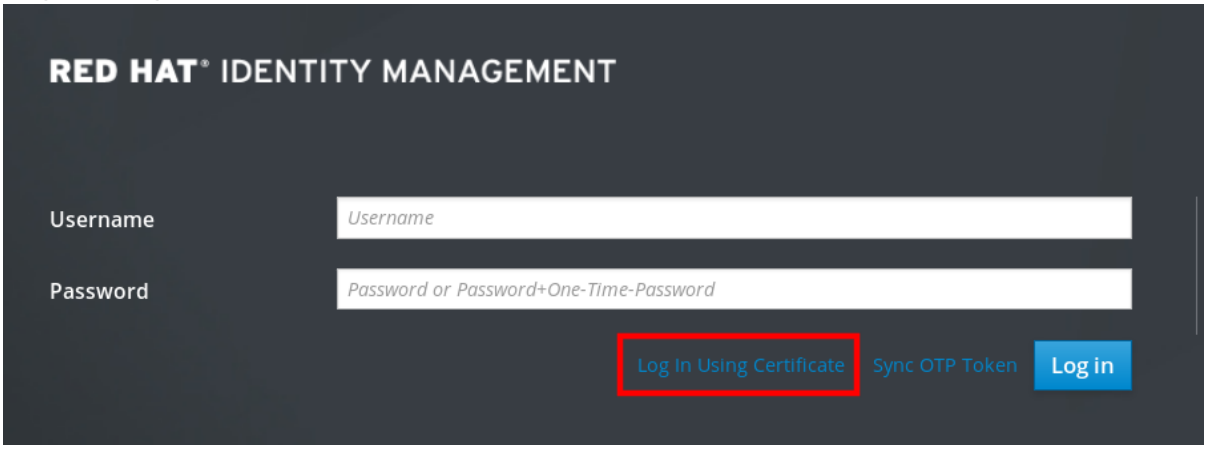
IdM Web UI へのログインにスマートカードを使用するには、以下の手順に従います。

前提条件

- Web ブラウザーが、スマートカード認証を使用できるように設定されている。
- IdM サーバーはスマートカード認証用に設定されています。
- スマートカードにインストールされた証明書は、IdM サーバーによって発行されるか、IdM のユーザーエントリーに追加されています。
- スマートカードのロックを解除するために必要な PIN を知っています。
- スマートカードがリーダーに挿入されました。

手順

1. ブラウザーで IdM Web UI を開きます。
2. **Log In Using Certificate** をクリックします。

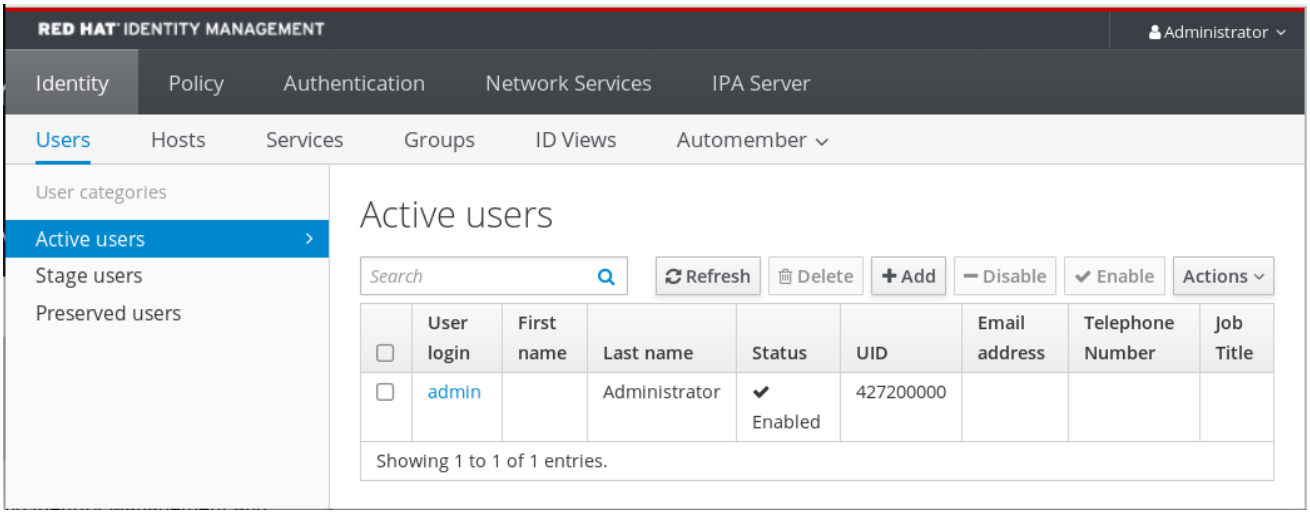


3. **Password Required** ダイアログボックスが開いたら、スマートカードのロックを解除する PIN を追加して、**OK** ボタンをクリックします。
   User Identification Request ダイアログボックスが開きます。

   スマートカードに複数の証明書が含まれている場合は、**Choose a certificate to present as identification** の下にあるドロップダウンリストで、認証に使用する証明書を選択します。

4. **OK** ボタンをクリックします。

これで、IdM Web UI に正常にログインできるようになりました。



2.10. IDM クライアントでスマートカード認証を使用して GDM にログインする

GNOME Desktop Manager (GDM) には認証が必要です。パスワードは使用できますが、認証にスマートカードを使用することもできます。

以下の手順に従って、スマートカード認証を使用して GDM にアクセスします。

前提条件

- システムはスマートカード認証用に設定されています。詳細は、スマートカード認証用の IdM クライアントの設定 を参照してください。
- スマートカードに、証明書と秘密鍵が含まれている。
- ユーザーアカウントは、IdM ドメインのメンバーです。
- スマートカードの証明書は、以下を使用してユーザーエントリーにマッピングします。
  - 特定のユーザーエントリーへの証明書の割り当て。詳細は Adding a certificate to a user entry in the IdM Web UI または Adding a certificate to a user entry in the IdM CLI を参照してください。
  - アカウントに適用される証明書マッピングデータ。詳細は、スマートカードにおける認証を設定するための証明書マッピングルール を参照してください。

手順

1. スマートカードをリーダーに挿入します。

2. スマートカード PIN を入力します。
3. **Sign In** をクリックします。

RHEL システムにログインし、IdM サーバーが提供する TGT がある。

検証手順

- **端末** ウィンドウで **klist** を入力し、結果を確認します。

```
$ 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 コマンドでのスマートカード認証の使用

別のユーザーへの変更には認証が必要です。パスワードまたは証明書を使用できます。以下の手順に従って、**su** コマンドでスマートカードを使用します。これは、**su** コマンドを入力すると、スマートカード PIN の入力が求められます。

前提条件

- IdM サーバーとクライアントがスマートカード認証用に設定されました。
  - スマートカード認証用の IdM サーバーの設定 を参照してください。
  - スマートカード認証用の IdM クライアントの設定 を参照してください。
- スマートカードに、証明書と秘密鍵が含まれている。スマートカードでの証明書の保存 を参照してください。
- カードがリーダーに挿入され、コンピューターに接続されている。

手順

- 端末で、**su** コマンドで別のユーザーに移動します。

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

設定が正しい場合は、スマートカードの PIN を入力するよう求められます。

第3章 IDM でスマートカード認証用に ADCS が発行した証明書の設定

Active Directory (AD)証明書サービスが証明書を発行するユーザーに IdM でスマートカード認証を設定するには、次のコマンドを実行します。

- デプロイメントは、Identity Management (IdM) と Active Directory (AD) と間のフォレスト間の信頼に基づいている場合
- AD にアカウントが保存されているユーザーに対してスマートカード認証を許可したい場合
- 証明書が作成され、Active Directory Certificate Services (ADCS) に保存される場合

スマートカード認証の概要については、スマートカード認証について を参照してください。

設定は、次の手順で行われます。

- Active Directory から IdM サーバーおよびクライアントへの CA 証明書およびユーザー証明書のコピー
- ADCS 証明書を使用したスマートカード認証用の IdM サーバーおよびクライアントの設定
- 証明書および秘密鍵をスマートカードに保存できるように PFX (PKCS#12) ファイルの変換
- sssd.conf ファイルでのタイムアウトの設定
- スマートカード認証用の証明書マッピングルールの作成

前提条件

- Identity Management (IdM) および Active Directory (AD) 信頼がインストールされている。詳細は、IdM と AD との間の信頼のインストール を参照してください。
- Active Directory 証明書サービス (ADCS) がインストールされ、ユーザーの証明書が生成されている。

3.1. 信頼の設定と証明書の使用に必要な WINDOWS SERVER 設定

Windows Server で以下を設定する必要があります。

- Active Directory 証明書サービス (ADCS) がインストールされる
- 認証局が作成される
- 必要に応じて、認証機関の Web 登録を使用している場合は、IIS (Internet Information Services) を設定する必要がある。

証明書をエクスポートします。

- 鍵には **2048** ビット以上が必要
- 秘密鍵を含める
- Personal Information Exchange (**PKCS #12(.PFX)**) の形式の証明書が必要
    - 証明書のプライバシーを有効にする

3.2. SFTP を使用して ACTIVE DIRECTORY から証明書のコピー

スマートカード認証を使用できるようにするには、次の証明書ファイルをコピーする必要があります。

- IdM サーバーにある **CER** 形式のルート CA 証明書 (**adcs-winserver-ca.cer**)
- IdM クライアントの **PFX** 形式の秘密鍵を持つユーザー証明書 (**aduser1.pfx**)

注記

この手順では、SSH アクセスが許可されていることを想定しています。SSH が使用できない場合、ユーザーは AD サーバーから IdM サーバーおよびクライアントにファイルをコピーする必要があります。

手順

1. **IdM サーバー** から接続し、**adcs-winserver-ca.cer** ルート証明書を IdM サーバーにコピーします。

```
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 クライアント** から接続し、**aduser1.pfx** ユーザー証明書をクライアントにコピーします。

```
[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
```

これで、CA 証明書は IdM サーバーに保存され、ユーザー証明書はクライアントマシンに保存されます。

3.3. ADCS 証明書を使用したスマートカード認証用の IDM サーバーおよびクライアントの設定

IdM 環境でスマートカード認証を使用できるように、IdM (Identity Management) サーバーおよびクライアントを設定する必要があります。IdM には、必要なすべての変更を行う **ipa-advise** スクリプトが含まれています。

- 必要なパッケージをインストールする

- IdM サーバーおよびクライアントの設定
- CA 証明書を予想される場所にコピーします。

IdM サーバーで **ipa-advise** を実行できるようになります。

以下の手順に従って、スマートカード認証用にサーバーとクライアントを設定します。

- IdM サーバー - **ipa-advise** スクリプトを準備して、スマートカード認証用に IdM サーバーを設定します。
- IdM サーバー - **ipa-advise** スクリプトを準備して、スマートカード認証用に IdM クライアントを設定します。
- IdM サーバー - AD 証明書を使用して IdM サーバーに **ipa-advise** サーバースクリプトを適用します。
- クライアントスクリプトを IdM クライアントマシンに移動します。
- IdM サーバー - AD 証明書を使用して IdM クライアントに **ipa-advise** クライアントスクリプトを適用します。

前提条件

- 証明書が IdM サーバーにコピーされている。
- Kerberos チケットを取得している。
- 管理者権限を持つユーザーとしてログインしている。

手順

1. IdM サーバーで、クライアントを設定する **ipa-advise** スクリプトを使用します。

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

2. IdM サーバーで、サーバーを設定する **ipa-advise** スクリプトを使用します。

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

3. IdM サーバーで、スクリプトを実行します。

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

   - IdM Apache HTTP サーバーを設定します。
   - キー配布センター (KDC) の Kerberos (PKINIT) で、初回認証用の公開鍵暗号化機能を有効にします。
   - スマートカード認可要求を受け入れるように IdM Web UI を設定します。

4. **sc_client.sh** スクリプトをクライアントシステムにコピーします。

```
[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 証明書をクライアントシステムにコピーします。

```
[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. クライアントシステムで、クライアントスクリプトを実行します。

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

CA 証明書が IdM サーバーとクライアントシステムに正しい形式でインストールされました。次の手順は、ユーザー証明書をスマートカード自体にコピーすることです。

3.4. PFX ファイルの変換

PFX (PKCS#12) ファイルをスマートカードに保存する前に、以下を行う必要があります。

- ファイルを PEM 形式に変換する。
- 秘密鍵と証明書を 2 つの異なるファイルに抽出する。

前提条件

- PFX ファイルが IdM クライアントマシンにコピーされます。

手順

1. IdM クライアントで、PEM 形式に変換します。

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

2. 鍵を別のファイルにデプロイメントします。

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

3. パブリック証明書を別のファイルにデプロイメントします。

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

この時点で、**aduser1.key** および **aduser1.crt** をスマートカードに保存できます。

3.5. スマートカードを管理および使用するツールのインストール

前提条件

- **gnutls-utils** パッケージがインストールされている。
- **opensc** パッケージがインストールされている。
- **pcscd** サービスを実行している。

スマートカードを設定する前に、対応するツール (証明書を生成して **pscd** サービスを起動できるもの) をインストールする必要があります。

手順

1. **opensc** パッケージおよび **gnutls-utils** パッケージをインストールします。

```
# {PackageManagerCommand} -y install opensc gnutls-utils
```

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

検証手順

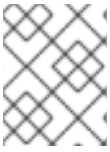
- **pcscd** サービスが稼働していることを確認します。

```
# systemctl status pcscd
```

3.6. スマートカードを準備し、証明書と鍵をスマートカードにアップロードする

pkcs15-init ツールを使用してスマートカードを設定するには、この手順に従います。このツールは、以下を設定するのに役立ちます。

- スマートカードの消去
- 新しい PIN および任意の PIN ブロック解除キー (PUK) の設定
- スマートカードでの新規スロットの作成
- スロットへの証明書、秘密鍵、および公開鍵の保存
- 必要に応じて、特定のスマートカードではこのタイプのファイナライズが必要なため、スマートカードの設定をロックします。



注記

pkcs15-init ツールは、すべてのスマートカードで機能するとは限りません。使用しているスマートカードで動作するツールを使用する必要があります。

前提条件

- **pkcs15-init** ツールを含む **opensc** パッケージがインストールされている。
  詳細は スマートカードを管理および使用するツールのインストール を参照してください。

- カードがリーダーに挿入され、コンピューターに接続されている。
- スマートカードに保存する秘密鍵、公開鍵、および証明書がある。この手順の **testuser.key**、**testuserpublic.key**、および **testuser.crt** は、秘密鍵、公開鍵、および証明書に使用される名前です。
- 現在のスマートカードユーザー PIN およびセキュリティーオフィス PIN (SO-PIN)

手順

1. スマートカードを消去して PIN で自身を認証します。

```
$ 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]:
```

   カードが削除されました。

2. スマートカードを初期化し、ユーザーの PIN と PUK を設定します。また、セキュリティー担当者の PIN と PUK を設定します。

```
$ 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 ツールは、スマートカードに新しいスロットを作成します。

3. スロットのラベルと認証 ID を設定します。

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

   ラベルは人間が判読できる値に設定されます (この場合は **testuser**)。**auth-id** は 16 進数の値である必要があります。この場合、**01** に設定されます。

4. スマートカードの新しいスロットに秘密鍵を保存し、ラベルを付けます。

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

注記

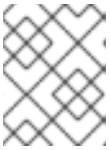
--id に指定する値は、秘密鍵を保存するときと、次の手順で証明書を保存するときに同じである必要があります。**--id** に独自の値を指定することを推奨します。そうしないと、より複雑な値がツールによって計算されます。

5. スマートカードの新しいスロットに証明書を保存し、ラベル付けします。

```
$ 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. (必要に応じて) スマートカードの新しいスロットに公開鍵を保存し、ラベルを付けます。

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



注記

公開鍵が秘密鍵または証明書に対応する場合は、秘密鍵または証明書の ID と同じ ID を指定します。

7. (オプション) スマートカードの中には、設定をロックしてカードを最終処理する必要があります。

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

この段階では、スマートカードには、新たに作成されたスロットに証明書、秘密鍵、および公開鍵が含まれます。ユーザーの PIN と PUK、およびセキュリティー担当者の PIN と PUK も作成しました。

3.7. SSSD.CONF でタイムアウトの設定

スマートカード証明書による認証は、SSSD で使用されるデフォルトのタイムアウトよりも時間がかかる場合があります。タイムアウトの期限切れは次の原因で発生する可能性があります。

- 読み込みが遅い
- 物理デバイスから仮想環境への転送
- スマートカードに保存されている証明書が多すぎる
- OCSP (Online Certificate Status Protocol) を使用して証明書を検証する場合は、OCSP レスポンダーからの応答が遅い

この場合は、**sssd.conf** ファイルにある次のタイムアウトを、たとえば 60 秒まで延長できます。

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

前提条件

- root としてログインしている。

手順

1. **sssd.conf** ファイルを開きます。

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

2. **p11_child_timeout** の値を変更します。

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

3. **krb5_auth_timeout** の値を変更します。

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

4. 設定を保存します。

現在、スマートカードとの相互作用は 1 分間 (60 秒) 実行でき、その後、認証はタイムアウトで失敗します。

3.8. スマートカード認証用の証明書マッピングルールの作成

AD (Active Directory) および IdM (Identity Management) にアカウントを持つユーザーに対して証明書を 1 つ使用する場合は、IdM サーバーで証明書マッピングルールを作成できます。

このようなルールを作成すると、ユーザーは両方のドメインのスマートカードで認証できます。

証明書マッピングルールの詳細は、認証を設定するための証明書マッピングルール を参照してください。

第4章 認証を設定するための証明書マッピングルール

次のシナリオでは、証明書マッピングルールの設定が必要になる場合があります。

- 証明書は、IdM ドメインが信頼関係にある Active Directory (AD) の証明書システムによって発行されています。
- 証明書は外部の認証局によって発行されています。
- IdM 環境は大規模で、多くのユーザーがスマートカードを使用しています。この場合、完全な証明書の追加は複雑になる可能性があります。件名と発行者はほとんどのシナリオで予測可能なため、完全な証明書よりも事前に追加する方が簡単です。

システム管理者は、証明書マッピングルールを作成し、特定のユーザーに証明書を発行する前に、ユーザーエントリーに証明書マッピングデータを追加できます。証明書を発行すると、完全な証明書がまだユーザーエントリーにアップロードされていなくても、ユーザーは証明書を使用してログインできます。

さらに、証明書は定期的に更新されるため、証明書マッピングルールにより管理オーバーヘッドが削減されます。ユーザーの証明書が更新される場合、管理者はユーザーエントリーを更新する必要はありません。たとえば、マッピングが **Subject** と **Issuer** の値に基づいている場合、および新しい証明書の Subject と Issuer が以前と同じ場合は、マッピングは引き続き適用されます。一方で、完全な証明書を使用した場合、管理者は古い証明書に置き換わる新しい証明書をユーザーエントリーにアップロードする必要があります。

証明書マッピングを設定するには、以下を実行します。

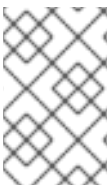
1. 管理者は、証明書マッピングデータまたは完全な証明書をユーザーアカウントにロードする必要があります。
2. 管理者は、証明書の情報と一致する証明書マッピングデータエントリーがアカウントに含まれているユーザーが IdM に正常にログインできるように、証明書マッピングルールを作成する必要があります。

証明書マッピングルールが作成されると、エンドユーザーが ファイルシステム または スマートカード に保存されている証明書を提示すると、認証は成功します。

注記

キー配布センター (KDC) には、証明書マッピングルールのキャッシュがあります。キャッシュは最初の **certauth** 要求の際に入力され、タイムアウトは 300 秒にハードコーディングされています。KDC は、再起動するかキャッシュが期限切れにならない限り、証明書マッピングルールへの変更を認識しません。

マッピングルールを設定する個々のコンポーネントの詳細と、そのコンポーネントの取得方法および使用方法は、IdM における ID マッピングルールのコンポーネント および マッチングルールで使用する証明書からの発行者の取得 を参照してください。



注記

証明書マッピングルールは、証明書を使用するユースケースに応じて異なります。たとえば、証明書を使用して SSH を使用している場合、証明書から公開鍵を抽出するには完全な証明書が必要です。

第5章 集中管理ユーザー向けに WEB コンソールを使用したスマートカード認証の設定

RHEL Web コンソールでスマートカード認証を集中管理しているユーザーに設定します。

- Identity Management
- Identity Management を使用してフォレスト間の信頼に接続する Active Directory

重要

> 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`.

前提条件

- スマートカード認証を使用するシステムは、Active Directory または Identity Management ドメインのメンバーである必要があります。
  Web コンソールを使用して RHEL 8 システムをドメインに参加させる方法は Web コンソールで RHEL 8 システムを IdM ドメインに参加 を参照してください。
- スマートカード認証に使用される証明書は、Identity Management または Active Directory の特定のユーザーに関連付けられている必要があります。
  Identity Management のユーザーと証明書の関連付けの詳細は、Adding a certificate to a user entry in the IdM Web UI または Adding a certificate to a user entry in the IdM CLI を参照してください。

5.1. 集中管理ユーザーのスマートカード認証

スマートカードは、カードに保存されている証明書を使用して個人認証を提供できる物理デバイスです。個人認証とは、ユーザーパスワードと同じ方法でスマートカードを使用できることを意味します。

秘密鍵と証明書の形式で、スマートカードにユーザーの認証情報を保存できます。特別なソフトウェアおよびハードウェアを使用して、そのソフトウェアにアクセスします。スマートカードをリーダーまたは USB ソケットに挿入して、パスワードを入力する代わりに、スマートカードの PIN コードを入力します。

Identity Management (IdM) では、以下によるスマートカード認証に対応しています。

- IdM 認証局が発行するユーザー証明書。詳細は、スマートカード認証用の Identity Management の設定 を参照してください。
- Active Directory Certificate Service (ADCS) 認証局が発行するユーザー証明書。詳細は IdM でスマートカード認証用に ADCS が発行した証明書の設定 を参照してください。

注記

スマートカード認証の使用を開始する場合は、ハードウェア要件 Smart Card support in RHEL8+ を参照してください。

5.2. スマートカードを管理および使用するツールのインストール

前提条件

- **gnutls-utils** パッケージがインストールされている。
- **opensc** パッケージがインストールされている。
- **pcscd** サービスを実行している。

スマートカードを設定する前に、対応するツール (証明書を生成して **pscd** サービスを起動できるもの) をインストールする必要があります。

手順

1. **opensc** パッケージおよび **gnutls-utils** パッケージをインストールします。

```
# {PackageManagerCommand} -y install opensc gnutls-utils
```

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

検証手順

- **pcscd** サービスが稼働していることを確認します。

```
# systemctl status pcscd
```

5.3. スマートカードを準備し、証明書と鍵をスマートカードにアップロードする

pkcs15-init ツールを使用してスマートカードを設定するには、この手順に従います。このツールは、以下を設定するのに役立ちます。

- スマートカードの消去
- 新しい PIN および任意の PIN ブロック解除キー (PUK) の設定
- スマートカードでの新規スロットの作成
- スロットへの証明書、秘密鍵、および公開鍵の保存
- 必要に応じて、特定のスマートカードではこのタイプのファイナライズが必要なため、スマートカードの設定をロックします。

注記

pkcs15-init ツールは、すべてのスマートカードで機能するとは限りません。使用しているスマートカードで動作するツールを使用する必要があります。

前提条件

- **pkcs15-init** ツールを含む **opensc** パッケージがインストールされている。
  詳細は スマートカードを管理および使用するツールのインストール を参照してください。
- カードがリーダーに挿入され、コンピューターに接続されている。
- スマートカードに保存する秘密鍵、公開鍵、および証明書がある。この手順の **testuser.key**、**testuserpublic.key**、および **testuser.crt** は、秘密鍵、公開鍵、および証明書に使用される名前です。
- 現在のスマートカードユーザー PIN およびセキュリティーオフィス PIN (SO-PIN)

手順

1. スマートカードを消去して PIN で自身を認証します。

```
$ 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]:
```

   カードが削除されました。

2. スマートカードを初期化し、ユーザーの PIN と PUK を設定します。また、セキュリティー担当者の PIN と PUK を設定します。

```
$ 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 ツールは、スマートカードに新しいスロットを作成します。

3. スロットのラベルと認証 ID を設定します。

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

   ラベルは人間が判読できる値に設定されます (この場合は **testuser**)。**auth-id** は 16 進数の値である必要があります。この場合、**01** に設定されます。

4. スマートカードの新しいスロットに秘密鍵を保存し、ラベルを付けます。

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

注記

--id に指定する値は、秘密鍵を保存するときと、次の手順で証明書を保存するときに同じである必要があります。**--id** に独自の値を指定することを推奨します。そうしないと、より複雑な値がツールによって計算されます。

5. スマートカードの新しいスロットに証明書を保存し、ラベル付けします。

```
$ 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. (必要に応じて) スマートカードの新しいスロットに公開鍵を保存し、ラベルを付けます。

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

注記

公開鍵が秘密鍵または証明書に対応する場合は、秘密鍵または証明書の ID と同じ ID を指定します。

7. (オプション) スマートカードの中には、設定をロックしてカードを最終処理する必要があります。

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

この段階では、スマートカードには、新たに作成されたスロットに証明書、秘密鍵、および公開鍵が含まれます。ユーザーの PIN と PUK、およびセキュリティー担当者の PIN と PUK も作成しました。

5.4. WEB コンソールのスマートカード認証の有効化

Web コンソールでスマートカード認証を使用できるようにするには、**cockpit.conf** ファイルでスマートカード認証を有効にします。

また、同じファイルでパスワード認証を無効にすることもできます。

前提条件

- RHEL Web コンソールがインストールされている。
  詳細は、Web コンソールのインストール を参照してください。

手順

1. 管理者権限で RHEL Web コンソールにログインしている。
   詳細は、Web コンソールへのログイン を参照してください。
2. **Terminal** をクリックします。
3. **/etc/cockpit/cockpit.conf** で **ClientCertAuthentication** を **yes** に設定します。

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

4. 必要に応じて、以下のようにして **cockpit.conf** でパスワードベースの認証を無効にします。

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

この設定ではパスワード認証が無効になり、常にスマートカードを使用する必要があります。

5. Web コンソールを再起動して、**cockpit.service** が変更を受け入れることを確認します。

```
# systemctl restart cockpit
```

5.5. スマートカードを使用して WEB コンソールへのログイン

スマートカードを使用して、Web コンソールにログインできます。

前提条件

- 有効な証明書が、Active Directory または Identity Management ドメインで作成されたユーザーアカウントに関連付けられているスマートカードに保存されている。
- スマートカードのロックを解除するピン。
- スマートカードがリーダーに追加されている。

手順

1. Web ブラウザーを開き、アドレスバーに Web コンソールのアドレスを追加します。
ブラウザーは、スマートカードに保存されている証明書を PIN で保護するよう要求します。
2. **Password Required** ダイアログボックスで PIN を入力し、**OK** をクリックします。
3. **User Identification Request** ダイアログボックスで、スマートカードに保存されている証明書を選択します。
4. **Remember this decision** を選択します。
次回、このウィンドウが開きません。

注記

この手順は、Google Chrome ユーザーには適用されません。

5. **OK** をクリックします。

これで接続され、Web コンソールがそのコンテンツを表示します。

5.6. DOS 攻撃を防ぐためのユーザーセッションおよびメモリーの制限

証明書認証は、別のユーザーの権限を借用しようとする攻撃者に対して Web サーバー **cockpit-ws** のインスタンスを分離して孤立させることで保護されます。ただし、これによりサービス拒否攻撃 (DoS) 攻撃が発生する可能性があります。リモートの攻撃者は大量の証明書を作成し、異なる証明書を使用してそれぞれ cockpit-ws に多数の HTTPS 要求を送信することができます。

この DoS を防ぐために、これらの Web サーバーインスタンスの共同リソースは制限されます。デフォルトでは、接続数に制限され、メモリー使用量の制限は 200 スレッドと、75% (ソフト) または 90% (ハード) のメモリーに設定されます。

以下の手順では、接続およびメモリーの数を制限することで、リソースの保護を説明します。

手順

1. 端末で **system-cockpithttps.slice** 設定ファイルを開きます。

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

2. **TasksMax** を 100 に、**CPUQuota** を 30% に制限します。

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

3. 変更を適用するには、システムを再起動します。

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

これで、新しいメモリーとユーザーセッションの制限により、Web サーバー **cockpit-ws** が DoS 攻撃から保護されるようになりました。

5.7. 関連情報

- Configuring Identity Management for smart card authentication .
- Configuring certificates issued by ADCS for smart card authentication in IdM .
- ローカル証明書のスマートカードへの設定およびインポート

第6章 ローカル証明書を使用したスマートカード認証の設定

ローカル証明書を使用してスマートカード認証を設定するには、次の手順を実行します。

- ホストがドメインに接続されていない
- このホストで、スマートカードで認証する必要がある
- スマートカード認証を使用して SSH アクセスを設定する必要がある
- **authselect** を使用してスマートカードを設定する必要がある

このシナリオを行うには、以下の設定を使用します。

- スマートカードで認証するユーザーのユーザー証明書を取得する。証明書は、ドメインで使用される信頼できる認証局によって生成される必要があります。
  証明書を取得できない場合は、テスト目的で、ローカルの認証局が署名したユーザー証明書を生成します。
- スマートカードに証明書と秘密鍵を保存する。
- SSH アクセス用のスマートカード認証を設定する。

重要

ホストがドメインの一部である場合は、そのホストをドメインに追加し、Active Directory または Identity Management 認証局が生成した証明書を使用します。

スマートカードに IdM 証明書を作成する方法は、Configuring Identity Management for smart card authentication を参照してください。

前提条件

- authselect がインストールされている。
  authselect ツールは、Linux ホストでユーザー認証を設定し、スマートカード認証パラメーターを設定するのに使用できます。authselect の詳細は、authselect とは を参照してください。
- RHEL 8 で対応しているスマートカードまたは USB デバイス
  詳細は、Smart Card support in RHEL8 を参照してください。

6.1. ローカル証明書の作成

この手順に従って、次のタスクを実行します。

- OpenSSL 認証局の生成
- 証明書署名リクエストの作成

警告

以下の手順は、テスト目的のみを想定しています。ローカルの自己署名証明局により生成される証明書は、AD、IdM、または RHCS 認証局を使用する場合と同じように安全ではありません。ホストがドメインに含まれていない場合でも、企業の認定機関が生成した証明書を使用する必要があります。

手順

1. 証明書を生成するディレクトリーを作成します。以下に例を示します。

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

2. 証明書を設定します (このテキストは、**ca** ディレクトリーのコマンドラインにコピーします)。

```
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. 以下のディレクトリーを作成します。

```
# mkdir certs crl newcerts
```

4. 以下のファイルを作成します。

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

5. シリアルファイルに番号 01 を書き込みます。

```
# echo 01 > serial
```

このコマンドは、シリアル番号内に 01 を書き込みます。これは証明書のシリアル番号です。この CA によってリリースされる新しい証明書ごとに、数が 1 つ増えます。

6. OpenSSL root CA キーを作成します。

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

7. 自己署名 root 認証局証明書を作成します。

```
# 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. ユーザー名のキーを作成します。

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

このキーは、セキュリティーが保護されていないローカルシステムで生成されるため、キーがカードに格納されると、キーがシステムから削除されます。

キーはスマートカードで直接作成できます。これを行う場合は、スマートカードの製造元が作成した手順に従います。

9. 証明書署名リクエスト設定ファイルを作成します (このテキストを ca ディレクトリーでコマンドラインにコピーします)。

```
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 証明書用の証明書署名リクエストを作成します。

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

11. 新しい証明書を設定します。有効期限は 1 年に設定されています。

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

この時点で、認証局と証明書が正常に生成され、スマートカードにインポートできる状態になります。

6.2. SSSD ディレクトリーへの証明書のコピー

GNOME Desktop Manager (GDM) には SSSD が必要です。GDM を使用する場合は、**/etc/sssd/pki** ディレクトリーに PEM 証明書をコピーする必要があります。

前提条件

- ローカル CA の認証局および証明書が生成されている。

手順

1. システムに SSSD がインストールされていることを確認します。

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

2. **/etc/sssd/pki** ディレクトリーを作成します。

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

3. **rootCA.crt** を PEM ファイルとして **/etc/sssd/pki/** ディレクトリーにコピーします。

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

これで、認証局と証明書が生成され、**/etc/sssd/pki** ディレクトリーに保存されました。

注記

別のアプリケーションで認証局の証明書を共有する場合は、sssd.conf の場所を変更してください。

- SSSD PAM レスポンダー: **[pam]** セクションの **pam_cert_db_path**
- SSSD ssh レスポンダー: **[ssh]** セクションの **ca_db**

詳細は、man ページの **sssd.conf** を参照してください。

Red Hat では、デフォルトのパスを維持して、SSSD に専用の認証局証明書ファイルを使用し、認証に信頼されている認証局のみをここに登録するようにすることを推奨します。

6.3. スマートカードを管理および使用するツールのインストール

前提条件

- **gnutls-utils** パッケージがインストールされている。
- **opensc** パッケージがインストールされている。
- **pcscd** サービスを実行している。

スマートカードを設定する前に、対応するツール (証明書を生成して **pscd** サービスを起動できるもの) をインストールする必要があります。

手順

1. **opensc** パッケージおよび **gnutls-utils** パッケージをインストールします。

```
# {PackageManagerCommand} -y install opensc gnutls-utils
```

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

検証手順

- **pcscd** サービスが稼働していることを確認します。

```
# systemctl status pcscd
```

6.4. スマートカードを準備し、証明書と鍵をスマートカードにアップロードする

pkcs15-init ツールを使用してスマートカードを設定するには、この手順に従います。このツールは、以下を設定するのに役立ちます。

- スマートカードの消去
- 新しい PIN および任意の PIN ブロック解除キー (PUK) の設定
- スマートカードでの新規スロットの作成
- スロットへの証明書、秘密鍵、および公開鍵の保存
- 必要に応じて、特定のスマートカードではこのタイプのファイナライズが必要なため、スマートカードの設定をロックします。

注記

pkcs15-init ツールは、すべてのスマートカードで機能するとは限りません。使用しているスマートカードで動作するツールを使用する必要があります。

前提条件

- **pkcs15-init** ツールを含む **opensc** パッケージがインストールされている。
  詳細は スマートカードを管理および使用するツールのインストール を参照してください。
- カードがリーダーに挿入され、コンピューターに接続されている。
- スマートカードに保存する秘密鍵、公開鍵、および証明書がある。この手順の **testuser.key**、**testuserpublic.key**、および **testuser.crt** は、秘密鍵、公開鍵、および証明書に使用される名前です。
- 現在のスマートカードユーザー PIN およびセキュリティーオフィス PIN (SO-PIN)

手順

1. スマートカードを消去して PIN で自身を認証します。

```
$ 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]:
```

   カードが削除されました。

2. スマートカードを初期化し、ユーザーの PIN と PUK を設定します。また、セキュリティー担当者の PIN と PUK を設定します。

```
$ 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 ツールは、スマートカードに新しいスロットを作成します。

3. スロットのラベルと認証 ID を設定します。

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

ラベルは人間が判読できる値に設定されます (この場合は **testuser**)。**auth-id** は 16 進数の値である必要があります。この場合、**01** に設定されます。

4. スマートカードの新しいスロットに秘密鍵を保存し、ラベルを付けます。

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

注記

--id に指定する値は、秘密鍵を保存するときと、次の手順で証明書を保存するときに同じである必要があります。**--id** に独自の値を指定することを推奨します。そうしないと、より複雑な値がツールによって計算されます。

5. スマートカードの新しいスロットに証明書を保存し、ラベル付けします。

```
$ 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. (必要に応じて) スマートカードの新しいスロットに公開鍵を保存し、ラベルを付けます。

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

注記

公開鍵が秘密鍵または証明書に対応する場合は、秘密鍵または証明書の ID と同じ ID を指定します。

7. (オプション) スマートカードの中には、設定をロックしてカードを最終処理する必要があります。

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

この段階では、スマートカードには、新たに作成されたスロットに証明書、秘密鍵、および公開鍵が含まれます。ユーザーの PIN と PUK、およびセキュリティー担当者の PIN と PUK も作成しました。

6.5. スマートカード認証で SSH アクセスの設定

SSH 接続には認証が必要です。パスワードまたは証明書を使用できます。スマートカードに保存されている証明書を使用した認証を有効にするには、次の手順に従います。

authselect を使用したスマートカードの設定の詳細は Configuring smart cards using authselect を参照してください。

前提条件

- スマートカードに、証明書と秘密鍵が含まれている。
- カードがリーダーに挿入され、コンピューターに接続されている。
- SSSD がインストールされ、設定されている。
- ユーザー名が、証明書の SUBJECT の CN (Common Name) または UID (User ID) と一致する。
- **pcscd** サービスがローカルマシンで実行している。
  詳細は スマートカードを管理および使用するツールのインストール を参照してください。

手順

1. スマートカード認証を使用するユーザーのホームディレクトリーで、SSH キー用の新しいディレクトリーを作成します。

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

2. **opensc** ライブラリーで **ssh-keygen -D** コマンドを実行して、スマートカードの秘密鍵とペアの既存の公開鍵を取得し、そのユーザーの SSH キーディレクトリーの **authorized_keys** リストに追加し、スマートカード認証を使用した SSH アクセスを有効にします。

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

3. SSH では、**/.ssh** ディレクトリーおよび **authorized_keys** ファイルのアクセス権が必要です。アクセス権を設定または変更するには、以下を入力します。

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

4. 必要に応じて、キーを表示します。

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

   端末にキーが表示されます。

5. **/etc/sssd/sssd.conf** ファイルでスマートカード認証が有効になっていることを確認します。**[pam]** セクションで、pam 証明書認証モジュールである **pam_cert_auth = True** を有効にします。

   sssd.conf ファイルがまだ作成されていない場合は、以下のスクリプトをコマンドラインにコピーして、最小限の機能設定を作成できます。

```
# 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 キーを使用するには、**authselect** コマンドで認証を設定します。

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

これで、次のコマンドを実行して SSH アクセスを確認できます。

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

設定に成功すると、スマートカードの PIN を入力するように求められます。

設定がローカルで機能するようになりました。これで、公開鍵をコピーして、SSH を使用するすべてのサーバーにある **authorized_keys** ファイルに配布できます。

第7章 AUTHSELECT を使用したスマートカード認証の設定

本セクションでは、以下のいずれかを達成するためのスマートカードの設定方法を説明します。

- パスワードとスマートカード認証の両方を有効化
- パスワードの無効化およびスマートカード認証の有効化
- 削除時にロックの有効化

前提条件

- authselect がインストールされている。
  authselect ツールは、Linux ホストでユーザー認証を設定し、スマートカード認証パラメーターを設定するのに使用できます。authselect の詳細は、authselect でユーザー認証の設定 を参照してください。
- RHEL 8 で対応しているスマートカードまたは USB デバイス
  詳細は、Smart Card support in RHEL8 を参照してください。

7.1. スマートカードの対象となる証明書

authselect を使用してスマートカードを設定する前に、証明書をカードにインポートする必要があります。以下のツールを使用して証明書を生成できます。

- Active Directory (AD)
- Identity Management (IdM)
  IdM 証明書を作成する方法は、新しいユーザー証明書を要求し、クライアントにエクスポート を参照してください。
- Red Hat Certificate System (RHCS)
  詳細は、Managing Smart Cards with the Enterprise Security Client を参照してください。
- サードパーティー認証局 (CA)
- ローカルの認証局ユーザーがドメインの一部ではない場合、またはテスト目的である場合は、ローカル認証局が生成した証明書を使用可能。
  ローカル認証を作成してスマートカードにインポートする方法の詳細は、Configuring and importing local certificates to a smart card を参照してください。

7.2. スマートカード認証とパスワード認証の両方を有効にするようにシステムを設定する

システムでスマートカード認証とパスワード認証の両方を有効にするには、次の手順に従ってください。

前提条件

- スマートカードに、証明書と秘密鍵が含まれる。
- そのカードがリーダーに挿入され、コンピューターに接続されている。
- **authselect** ツールがシステムにインストールされている。

手順

- 以下のコマンドを実行して、スマートカードとパスワード認証を許可します。

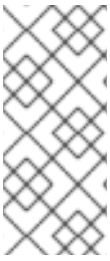
```
# authselect select sssd with-smartcard --force
```

この時点で、スマートカード認証が有効になります。ただし、スマートカードを忘れてしまった場合は、パスワード認証が動作します。

7.3. スマートカード認証を適用するようにシステムを設定する

authselect ツールを使用すると、システムでスマートカード認証を設定でき、デフォルトのパスワード認証を無効にできます。**authselect** コマンドには、以下のオプションが含まれます。

- **with-smartcard** – パスワード認証に加えてスマートカード認証を有効にします。
- **with-smartcard-required** – スマートカード認証を有効にし、パスワード認証を無効にします。



注記

with-smartcard-required オプションは、**login**、**gdm**、**xdm**、**kdm**、**xscreensaver**、**gnome-screensaver**、および **kscreensaver** などのログインサービスに対してのみ、排他的なスマートカード認証を適用します。ユーザーを切り替えるための **su** や **sudo** などの他のサービスは、デフォルトではスマートカード認証を使用せず、引き続きパスワードの入力を求めます。

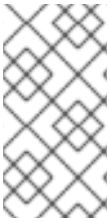
前提条件

- スマートカードに、証明書と秘密鍵が含まれている。
- そのカードがリーダーに挿入され、コンピューターに接続されている。
- **authselect** ツールがローカルシステムにインストールされている。

手順

- 以下のコマンドを実行して、スマートカード認証を強制します。

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



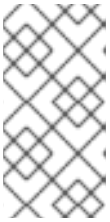
注記

このコマンドを実行すると、パスワード認証は機能しなくなり、スマートカードでのみログインできます。このコマンドを実行する前に、スマートカード認証が機能していることを確認してください。機能していないと、システムからロックアウトされる可能性があります。

7.4. 削除時にロックを使用したスマートカード認証の設定

authselect サービスを使用すると、スマートカードの認証を設定して、リーダーからスマートカードを削除した後も、画面を即時にロックできます。**authselect** コマンドには以下の変数が含まれている必要があります。

- **with-smartcard** - スマートカード認証の有効化
- **with-smartcard-required** - 排他的なスマートカード認証の有効化 (パスワードによる認証は無効になっています)
- **with-smartcard-lock-on-removal** - スマートカードの削除後に強制的にログアウト



注記

with-smartcard-lock-on-removal オプションは、GNOME デスクトップ環境を持つシステムでのみ機能します。**tty** またはコンソールベースのシステムを使用しており、そのリーダーからスマートカードを削除すると、システムから自動的にロックアウトされません。

前提条件

- スマートカードに、証明書と秘密鍵が含まれている。
- そのカードがリーダーに挿入され、コンピューターに接続されている。
- **authselect** ツールがローカルシステムにインストールされている。

手順

- 以下のコマンドを実行して、スマートカード認証の有効化、パスワード認証の無効化、削除時のロックの強制を行います。

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

これで、カードを削除すると、画面がロックされます。ロックを解除するには、スマートカードを再度挿入する必要があります。

第8章 スマートカードを使用した SUDO のリモート認証

本セクションでは、スマートカードを使用して sudo にリモートで認証する方法を説明します。**ssh-agent** サービスがローカルで実行され、**ssh-agent** ソケットをリモートマシンに転送できるようになると、sudo PAM モジュールの SSH 認証プロトコルを使用してユーザーをリモートで認証できます。

スマートカードを使用してローカルにログインした後、SSH 経由でリモートマシンにログインして、スマートカード認証の SSH 転送を使用してパスワードの入力を要求されずに **sudo** コマンドを実行できます。

この例では、クライアントは SSH 経由で IPA サーバーに接続され、スマートカードに保存されている認証情報を使用して IPA サーバーで sudo コマンドを実行します。

- IdM での sudo ルールの作成
- sudo 用の PAM モジュールの設定
- スマートカードを使用した sudo へのリモート接続

8.1. IDM での SUDO ルールの作成

この手順に従って、IdM で sudo ルールを作成し、リモートホストで sudo を実行する権限を **ipauser1** に付与します。

この例では、手順をテストするために、**less** コマンドと **whoami** コマンドが sudo コマンドとして追加されます。

前提条件

- IdM ユーザーが作成されている。この例では、ユーザーは **ipauser1** です。
- sudo をリモートで実行するシステムのホスト名を認識している。この例では、ホストは **server.ipa.test** です。

手順

1. **adminrule** という名前の **sudo** ルールを作成し、ユーザーがコマンドを実行できるようにします。

```
ipa sudorule-add adminrule
```

2. **sudo** コマンドとして **less** および **whoami** を追加します。

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

3. **less** および **whoami** コマンドを **adminrule** に追加します。

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

4. **ipauser1** ユーザーを **adminrule** に追加します。

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

5. **sudo** を実行しているホストを **adminrule** に追加します。

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

関連情報

- **ipa sudorule-add --help** を参照してください。
- **ipa sudocmd-add --help** を参照してください。

8.2. SUDO 用の PAM モジュールの設定

この手順に従って、sudo を実行している任意のホストで、スマートカードを使用した sudo 認証のために **pam_ssh_agent_auth.so** PAM モジュールをインストールして設定します。

手順

1. PAM SSH エージェントをインストールします。

```
dnf -y install pam_ssh_agent_auth
```

2. その他の **auth** エントリーの前に、**pam_ssh_agent_auth.so** の **authorized_keys_command** を **/etc/pam.d/sudo** ファイルに追加します。

```
#%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 コマンドを実行する際に SSH エージェントの転送が機能するようにするには、以下を **/etc/sudoers** ファイルに追加します。

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

これにより、IPA/SSSD に保存されたスマートカードの公開鍵があるユーザーは、パスワードを入力せずに sudo に対して認証できます。

4. **sssd** サービスを再起動します。

```
systemctl restart sssd
```

関連情報

- **pam** の man ページを参照してください。

8.3. スマートカードを使用した SUDO へのリモート接続

スマートカードを使用してリモートで **sudo** に接続するように SSH エージェントとクライアントを設定するには、次の手順に従います。

前提条件

- IdM で **sudo** ルールを作成している。
- **sudo** を実行するリモートシステムで **sudo** 認証用に **pam_ssh_agent_auth** PAM モジュールをインストールして設定している。

手順

1. SSH エージェントを起動します (まだ実行されていない場合には)。

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

2. スマートカードを SSH エージェントに追加します。プロンプトが表示されたら PIN を入力します。

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

3. ssh-agent 転送を有効にした SSH を使用して、**sudo** をリモートで実行する必要があるシステムに接続します。**-A** オプションを使用します。

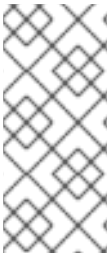
```
ssh -A ipauser1@server.ipa.test
```

検証手順

- **sudo** で **whoami** コマンドを実行します。

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

スマートカードの挿入時に PIN またはパスワードの入力を求められることはありません。



注記

SSH エージェントが GNOME キーリングなどの他のソースを使用するように設定されている場合に、スマートカードを取り外した後に **sudo** コマンドを実行すると、他のソースのいずれかが有効な秘密鍵へのアクセスを提供する可能性があるため、PIN またはパスワードの入力を求められないことがあります。SSH エージェントが認識しているすべての ID の公開鍵を確認するには、**ssh-add -L** コマンドを実行します。

関連情報

- 2 台のシステム間で OpenSSH を使用した安全な通信の使用
- ssh-agent を使用して SSH キーでリモートマシンに接続する手順

第9章 スマートカードによる認証のトラブルシューティング

以下のセクションでは、スマートカード認証の設定時に発生する可能性のある問題を解決する方法を説明します。

- スマートカード認証のテスト
- SSSD を使用したスマートカード認証のトラブルシューティング
- IdM Kerberos KDC が PKINIT を使用でき、CA 証明書が正しく配置されていることの確認
- SSSD タイムアウトの増加
- 証明書マッピングとマッチングルールのトラブルシューティング

9.1. システムでのスマートカードアクセスのテスト

この手順に従って、スマートカードにアクセスできるかどうかをテストします。

前提条件

- スマートカードで使用する IdM サーバーおよびクライアントをインストールして設定している。
- **nss-tools** パッケージから **certutil** ツールをインストールしている。
- スマートカードの PIN またはパスワードがある。

手順

1. **lsusb** コマンドを使用して、スマートカードリーダーがオペレーティングシステムに表示されることを確認します。

```
$ 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
```

   RHEL でテストおよびサポートされているスマートカードとリーダーの詳細は、Smart Card support in RHEL 9 を参照してください。

2. **pcscd** サービスおよびソケットが有効になっており、実行していることを確認します。

```
$ 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** コマンドを使用して、設定されたスマートカードと、スマートカードにあるトークンに関する情報を表示します。

```
$ 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. スマートカードの内容にアクセスできることを確認します。

```
$ 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** コマンドを使用して、スマートカードの証明書の内容を表示します。

   a. 以下のコマンドを実行して、証明書の正しい名前を確認します。

```
$ 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. スマートカードに証明書の内容を表示します。

注記

証明書の名前が、前の手順で表示された出力 (この例では **MyEID (sctest):Certificate**) と完全に一致していることを確認してください。

```
$ 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
```

関連情報

- **certutil(1)** の man ページを参照してください。

9.2. SSSD を使用したスマートカード認証のトラブルシューティング

スマートカードを使用した SSSD での認証のトラブルシューティングを行うには、次の手順に従います。

前提条件

- スマートカードで使用する IdM サーバーおよびクライアントをインストールして設定している。
- **sssd-tools** パッケージがインストールされている。
- スマートカードリーダーを検出し、スマートカードの内容を表示できる。Testing smart card access on the system を参照してください。

手順

1. **su** を使用して、スマートカードで認証できることを確認します。

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

スマートカードの PIN の入力を求められず、パスワードプロンプトまたは認証エラーが返された場合は、SSSD ログを確認してください。SSSD でのログインの詳細は、IdM で SSSD を使用した認証のトラブルシューティング を参照してください。認証の失敗の例を以下に示します。

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

以下のように、SSSD ログが **krb5_child** からの問題を示している場合は、CA 証明書に問題がある可能性があります。証明書に関する問題をトラブルシューティングするには、Verifying that IdM Kerberos KDC can use Pkinit and that the CA certificates are correctly located を参照してください。

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

SSSD ログに **p11_child** または **krb5_child** からのタイムアウトが示されている場合は、SSSD タイムアウトを増やして、スマートカードでの認証を再試行する必要があります。タイムアウトを増やす方法は、Increasing SSSD timeouts を参照してください。

2. GDM スマートカード認証の設定が正しいことを確認します。PAM 認証の成功メッセージは、以下のように返す必要があります。

```
# 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:
```

以下のような認証エラーが返された場合は、SSSD ログを確認して、問題の原因を特定します。SSSD でのログインの詳細は、IdM で SSSD を使用した認証のトラブルシューティング を参照してください。

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

PAM Environment:
 - no env -
```

PAM 認証に失敗したら、キャッシュをクリアしてコマンドを再度実行します。

```
# 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 KDC が PKINIT を使用でき、CA 証明書が正しく配置されていることの確認

この手順に従って、IdM Kerberos KDC が PKINIT を使用できることを検証する方法と、CA 証明書が正しく配置されていることを検証する方法についても説明します。

前提条件

- スマートカードで使用する IdM サーバーおよびクライアントをインストールして設定している。
- スマートカードリーダーを検出し、スマートカードの内容を表示できる。Testing smart card access on the system を参照してください。

手順

1. **kinit** ユーティリティーを実行し、スマートカードに保存されている証明書を使用して **idmuser1** として認証します。

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

2. スマートカード PIN を入力します。PIN の入力を求められない場合は、スマートカードリーダーを検出してスマートカードの内容を表示できることを確認してください。Testing smart card authentication を参照してください。
3. PIN が受け入れられ、パスワードの入力を求められた場合は、CA 署名証明書がない可能性があります。

a. **openssl** コマンドを使用して、CA チェーンがデフォルトの証明書バンドルファイルにリスト表示されていることを確認します。

```
$ 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. 証明書の有効性を確認します。

i. **idmuser1** のユーザー認証証明書 ID を見つけます。

```
$ 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 形式で、スマートカードからユーザー証明書情報を読み取ります。

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

iii. DER 証明書を PEM 形式に変換します。

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

iv. CA までの有効な発行者署名が証明書にあることを確認します。

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

4. スマートカードに複数の証明書が含まれている場合、**kinit** は認証用の正しい証明書を選択できない可能性があります。この場合は、**certid=<ID>** オプションを使用して、**kinit** コマンドの引数として証明書 ID を指定する必要があります。

a. スマートカードに保存されている証明書の数を確認し、使用している証明書の ID を取得します。

```
$ 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. 証明書 ID 01 で **kinit** を実行します。

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

5. **klist** を実行して、Kerberos 認証情報キャッシュの内容を表示します。

```
$ 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. 終了したら、アクティブな Kerberos チケットを破棄します。

```
$ kdestroy -A
```

関連情報

- **kinit** の man ページを参照してください。
- **kdestroy** の man ページを参照してください。

9.4. SSSD タイムアウトの増加

スマートカードでの認証に問題がある場合は、**krb5_child.log** および **p11_child.log** ファイルに以下のようなタイムアウトエントリーがないか確認してください。

krb5_child: Timeout for child [9607] reached…..consider increasing value of krb5_auth_timeout.

ログファイルにタイムアウトエントリーがある場合は、この手順で説明されているように SSSD タイムアウトを増やしてください。

前提条件

- スマートカード認証用に IdM サーバーおよびクライアントを設定している。

手順

1. IdM クライアントで **sssd.conf** ファイルを開きます。

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

2. ドメインセクション (**[domain/idm.example.com]** など) に、以下のオプションを追加します。

```
krb5_auth_timeout = 60
```

3. **[pam]** セクションに、以下を追加します。

```
p11_child_timeout = 60
```

4. SSSD キャッシュを削除します。

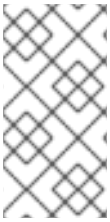
```
# sssctl cache-remove
SSSD must not be running. Stop SSSD now? (yes/no) [yes] yes
Creating backup of local data…
Removing cache files…
SSSD needs to be running. Start SSSD now? (yes/no) [yes] yes
```

タイムアウトを増やしたら、スマートカードを使用して再度認証を試行します。詳細は、Testing smart card authentication を参照してください。

9.5. 証明書マッピングとマッチングルールのトラブルシューティング

スマートカードでの認証に問題がある場合は、スマートカード証明書がユーザーに正しくリンクされていることを確認してください。デフォルトでは、ユーザーエントリーに **usercertificate** 属性の一部として完全な証明書が含まれる場合、証明書はユーザーに関連付けられます。ただし、証明書マッピングルールを定義している場合は、証明書をユーザーに関連付ける方法を変更している可能性があります。証明書マッピングとマッチングルールをトラブルシューティングするには、以下のセクションを参照してください。

- 証明書がユーザーにどのようにマッピングされているかを確認する
- スマートカード証明書に関連付けられたユーザーの確認



注記

スマートカードを使用して SSH で認証する場合は、Identity Management(IdM) のユーザーエントリーに完全な証明書を追加する必要があります。スマートカードを使用して SSH を使用した認証を行っていない場合は、**ipa user-add-certmapdata** コマンドを使用して証明書マッピングデータを追加できます。

9.5.1. 証明書がユーザーにどのようにマッピングされているかを確認する

デフォルトでは、ユーザーエントリーに **usercertificate** 属性の一部として完全な証明書が含まれる場合、証明書はユーザーに関連付けられます。ただし、証明書マッピングルールを定義している場合は、証明書をユーザーに関連付ける方法を変更している可能性があります。証明書マッピングルールを確認するには、次の手順に従ってください。

前提条件

- Identity Management(IdM) サーバーおよびクライアントを、スマートカードで使用するようにインストールおよび設定している。
- スマートカードリーダーを検出し、スマートカードの内容を表示できる。Testing smart card access on the system を参照すること。
- スマートカード証明書を IdM ユーザーにマッピングした。スマートカードにおける認証を設定するための証明書マッピングルール を参照してください。

手順

1. IdM 用に現在設定されている証明書マッピングルールを確認します。

```
# 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
----------------------------
```

以下のマッピングルールのいずれかが定義されていることが予想されます。

- **ipacertmapdata** は、IdM ユーザーエントリーの **certmapdata** 属性が使用されていることを示します。
- **altSecurityIdentities** は、Active Directory のユーザーエントリー名マッピング属性が使用されることを指定します。
- **userCertificate;binary=** は、IdM または AD のいずれかの証明書全体が使用されていることを示します。

多数のマッチングオプションを定義できますが、通常設定されたオプションの一部は以下のようになります。

- **<ISSUER>CN=[…]** は、使用されている証明書の発行者属性がこれと一致することを確認するためにチェックされることを指定します。
- **<SUBJECT>.*,DC=MY,DC=DOMAIN** は、証明書のサブジェクトがチェックされていることを示します。

2. IdM サーバーの **/etc/sssd/sssd.conf** ファイルに **debug_level = 9** を追加して、System Security Services Daemon (SSSD) ログを有効にします。

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

3. SSSD を再起動します。

```
# systemctl restart sssd
```

4. マッピングが正しく読み込まれる場合は、**/var/log/sssd/sssd_idm.example.com.log** ファイルに以下のエントリーが表示されるはずです。

```
[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. マッピングルールに無効な構文が含まれる場合は、以下のようなエントリーがログファイルに表示されます。

```
[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. マッピングルールの構文を確認してください。

```
# 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. 必要に応じて、証明書マッピングルールを変更します。

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

関連情報

- man ページの **sss-certmap** を参照してください。

9.5.2. スマートカード証明書に関連付けられたユーザーの確認

スマートカードでの認証に問題がある場合は、正しいユーザーがスマートカード証明書に関連付けられていることを確認してください。

前提条件

- Identity Management(IdM) サーバーおよびクライアントを、スマートカードで使用するようにインストールおよび設定している。
- スマートカードリーダーを検出し、スマートカードの内容を表示できる。Testing smart card access on the system を参照すること。
- スマートカード証明書を IdM ユーザーにマッピングした。スマートカードにおける認証を設定するための証明書マッピングルール を参照してください。
- PEM 形式のスマートカードからの証明書のコピー (例: **cert.pem**) がある。

手順

1. ユーザーがスマートカード証明書に関連付けられていることを確認します。

```
# ipa certmap-match cert.pem
--------------
```

```
1 user matched
--------------
  Domain: IDM.EXAMPLE.COM
  User logins: idmuser1
----------------------------
Number of entries returned 1
----------------------------
```

ユーザーまたはドメインが正しくない場合は、証明書がユーザーにどのようにマッピングされているかを確認してください。証明書がユーザーにどのようにマッピングされているかを確認する を参照してください。

2. ユーザーエントリーに証明書が含まれているかどうかを確認します。

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

3. ユーザーエントリーに証明書が含まれていない場合は、base-64 でエンコードされた証明書をユーザーエントリーに追加します。

   a. ヘッダーとフッターのある証明書を含む環境変数を作成し、1行に連結します。これは、**ipa user-add-cert** コマンドで必要な形式です。

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

   idmuser1.crt ファイルの証明書は PEM 形式である必要があることに注意してください。

   b. **ipa user-add-cert** コマンドを使用して、証明書を **idmuser1** のプロファイルに追加します。

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

   c. System Security Services Daemon (SSSD) キャッシュをクリアします。

   ```
   # 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** を再度実行して、ユーザーがスマートカード証明書に関連付けられていることを確認します。