Red Hat Enterprise Linux 9

スマートカード認証の管理

RHEL でのスマートカード認証の設定および管理

Last Updated: 2022-07-28

Red Hat Enterprise Linux 9 スマートカード認証の管理

RHEL でのスマートカード認証の設定および管理

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

法律上の通知

概要

本書は、RHEL でスマートカード認証を管理する方法を説明します。

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参ります。詳細は、弊社 の CTO、Chris Wright のメッセージ を参照してください。

Identity Management では、以下のような用語の置き換えが含まれます。

- **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**
- 単語 **マスター** は、コンテキストに応じて、より正確な言語に置き換えられます。
  - **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)

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
    1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
    2. マウスカーソルで、コメントを追加する部分を強調表示します。
    3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
    4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
    1. Bugzilla の Web サイトに移動します。
    2. Component で **Documentation** を選択します。
    3. **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記入してください。
    4. **Submit Bug** をクリックします。

第1章 スマートカード認証用の IDENTITY MANAGEMENT の設定

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。

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

このユーザーストーリーでは、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する方法を説明します。ユーザーストーリーでは、**smartcard_ca.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。

ユーザーストーリーには次のようなモジュールが含まれています。

- 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を使用した GDM アクセスの設定
- スマートカード認証を使用した su アクセスの設定

1.1. スマートカード認証用の IDM サーバーの設定

LDAP 識別名 (DN) が、**CN=Certificate Authority,DC=EXAMPLE,DC=ORG** である **EXAMPLE.ORG** ドメインの認証局により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、IdM サーバーが設定するスクリプトで実行できるように、認証局の証明書を取得する必要があります。たとえば、認証局により発行された証明書が含まれる Web ページから、その証明書を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 認証局が証明書を発行している IdM ユーザーに対してスマートカード認証を有効にするには、IdM CA が実行している IdM サーバーの **/etc/ipa/ca.crt** ファイルから CA 証明書を取得します。

本セクションでは、スマートカード認証に IdM サーバーを設定する方法を説明します。まず、PEM 形式の CA 証明書でファイルを取得してから、組み込みの **ipa-advise** スクリプトを実行します。最後に、システム設定を再読み込みます。

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 証明書とサブ CA 証明書がある。

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、**/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 /etc/ipa/ca.crt ~/SmartCard/
[root@server SmartCard]# cp /tmp/smartcard_ca.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 smartcard_ca.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 smartcard_ca.pem ca.crt
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。

1.2. スマートカード認証用の IDM クライアントの設定

本セクションでは、スマートカード認証に IdM クライアントを設定する方法を説明します。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、「Configuring SSH access using smart card authentication 」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。

- マシン - 場合によっては IdM ドメイン外にあります (ブラウザーが実行されている場合)
- **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。
- root CA証明書とサブ CA 証明書にアクセスできる。
- **--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する。
   - System Security Services Daemon (SSSD) を設定して、スマートカードのログインをデスクトップに許可する。

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:
```

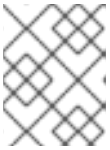
```
config-client-for-smart-card-auth.sh        100%  2419     3.5MB/s   00:00
```

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 {smartcard_ca.pem,ca.crt}
root@client.idm.example.com:/root/SmartCard/
Password:
smartcard_ca.pem                 100%  1237     9.6KB/s   00:00
ca.crt                           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 smartcard_ca.pem ca.crt
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。

1.3. IDM WEB UI のユーザーエントリーへの証明書の追加

この手順では、IdM Web UIのユーザーエントリーに外部証明書を追加する方法を説明します。

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、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。

注記

ユーザーの証明書が IdM 認証局により発行された場合、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、本セクションを省略できます。

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

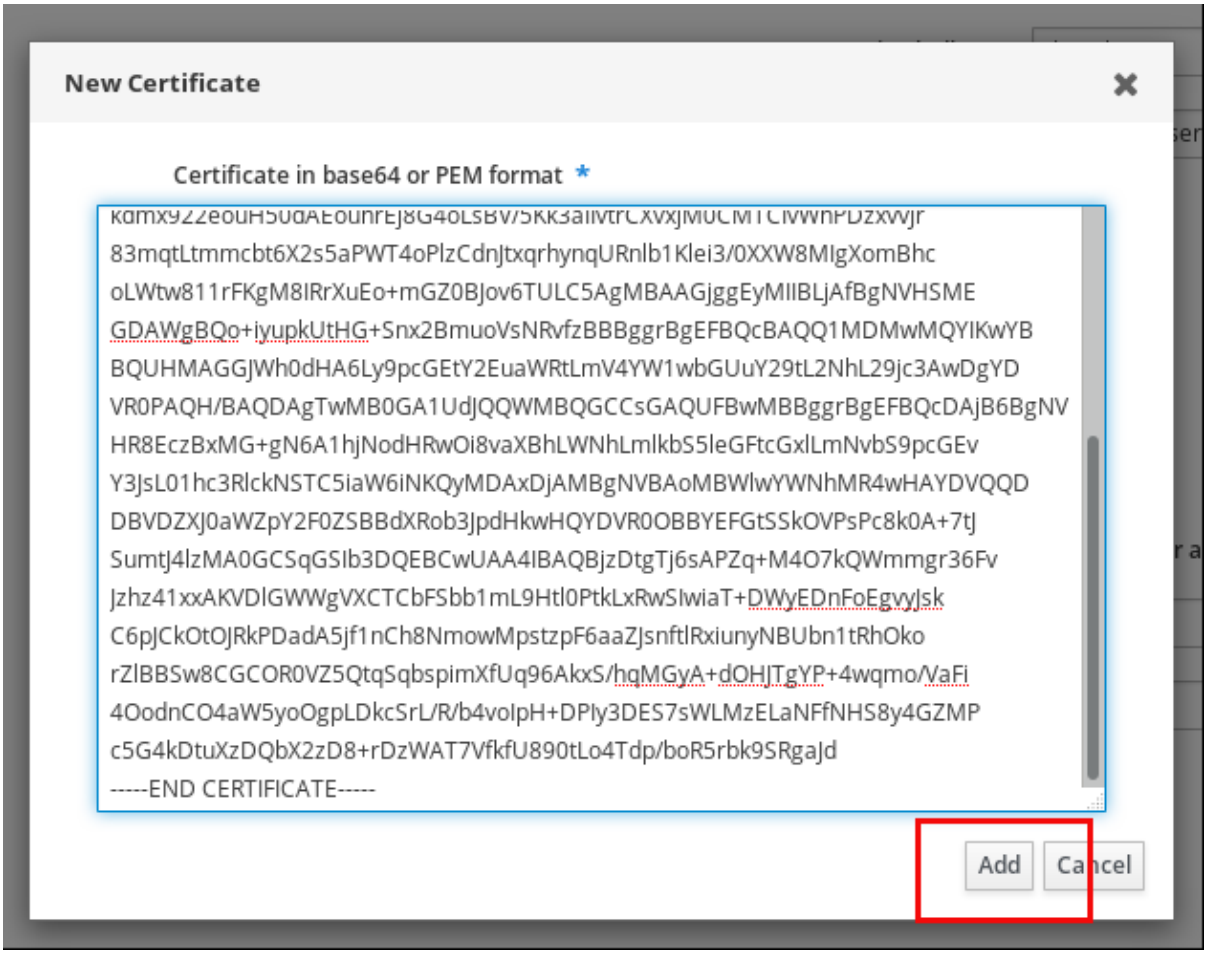
1. 別のユーザーに証明書を追加する場合は、管理者として IdM Web UI にログインします。独自のプロファイルに証明書を追加する場合は、管理者の認証情報が必要ありません。

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ース** で、**cat** ユーティリティーまたはテキストエディターで、**PEM** 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1.1 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。

1.4. IDM CLI でユーザーエントリーへの証明書の追加

この手順では、IdM CLIのユーザーエントリーに外部証明書を追加する方法を説明します。

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。詳細は以下を参照してください。

スマートカードで認証を設定するための証明書マッピングルール。

注記

ユーザーの証明書が IdM 認証局により発行された場合、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、本セクションを省略できます。

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。

1.5. 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

スマートカードを設定するには、証明書を生成し、スマートカードに保存するツールが必要になります。

以下を行う必要があります。

- 証明書管理に役立つ **gnutls-utils** パッケージをインストールする。
- スマートカードと連携するライブラリーおよびユーティリティーのセットを提供する **opensc** パッケージをインストールします。
- スマートカードリーダーと通信する **pcscd** サービスを開始する。

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。

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

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

pcscd サービスが稼働していることを確認します。

1.6. スマートカードでの証明書の保存

本セクションでは、設定に役立つ **pkcs15-init** によるスマートカードの設定を説明します。

- スマートカードの消去
- 新しい PIN およびオプションの 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
- スマートカード設定のロック (一部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になります)

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。この手順では、**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 PIN と PUK を設定します。また、セキュリティーオフィス 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、秘密鍵と証明書を保存する際に同じである必要があります。**--id** の値を指定しないと、ツールによりより複雑な値が計算されるため、独自の値の定義が容易になります。

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. (オプション) 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している場合は、その秘密鍵または証明書と同じ ID を指定する必要があります。

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るものもあります。

```
$ 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。

1.7. スマートカードを使用して IDM へのログイン

本セクションでは、IdM Web UI へのログインにスマートカードを使用する方法を説明します。

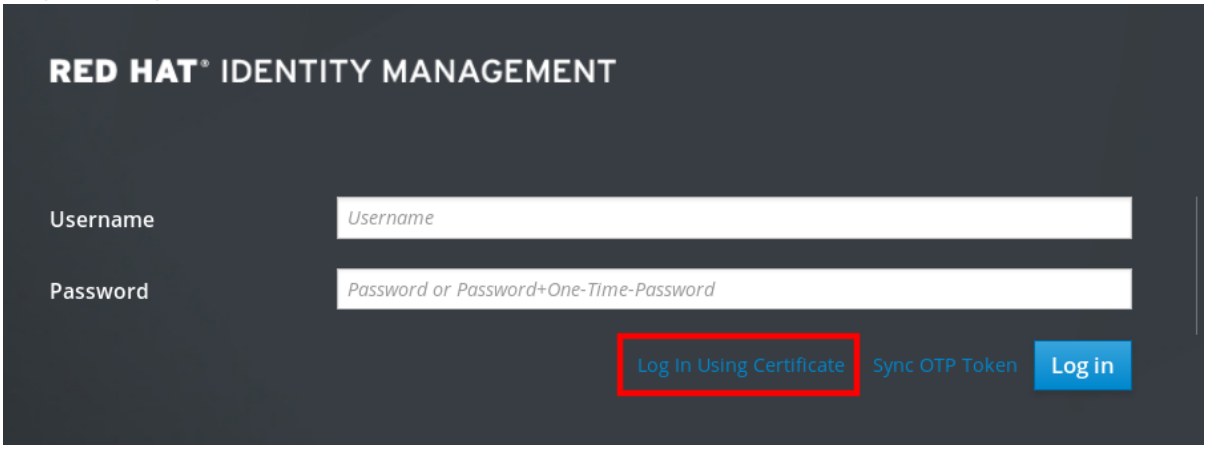
前提条件

- 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 サーバーが、スマートカード認証用に設定されている。
- IdM サーバーが、スマートカードにインストールした証明書を認識している。
- スマートカードのロックを解除するには PIN が必要です。
- スマートカードがリーダーにプラグインされている。

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。

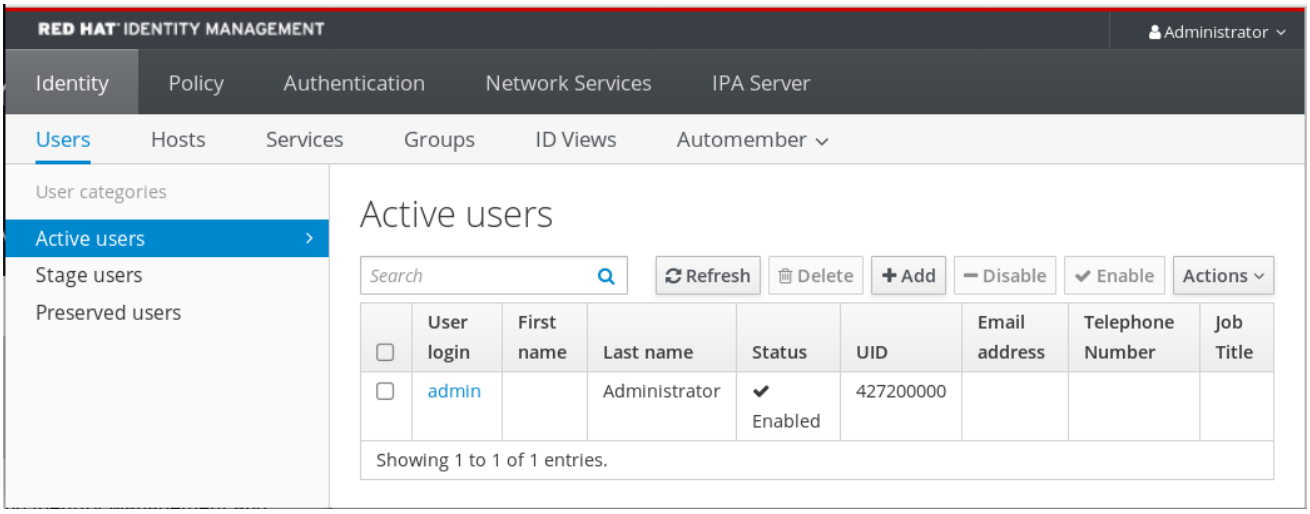


3. **Password Required** ダイアログボックスが開いたら、スマートカードのロックを解除する PIN を追加して、**OK** ボタンをクリックします。
User Identification Request ダイアログボックスが開きます。

   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。



1.8. スマートカード認証を使用した GDM アクセスの設定

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。

本セクションでは、GDM にアクセスするスマートカード認証を説明します。

スマートカード認証を使用する利点は、ユーザーアカウントが Identity Management ドメインの 一 部である場合に、TGT (Ticket-Granting Ticket) を取得することです。

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。
- 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
```

1.9. スマートカード認証を使用した SU アクセスの設定

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。本セクションでは、**su** コマンドでスマートカードを使用する方法を説明します。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

- 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。

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。

第2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

このシナリオでは、次の状況を説明します。

- 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

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構成
- 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。

2.1. 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。

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。

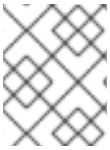
特定の IdM クライアントでスマートカード認証を機能させる方法を設定できます。

- 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。
- ユーザーは、自分のスマートカードで認証でき、パスワードは使用できません。
- ユーザーは、スマートカードを使用して、削除時に機能ロックを設定してログアウトできます。パスワードは使用できません。

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。

- 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。
- ユーザー証明書は、RHEL システムで生成されたローカル証明機関により発行されています。詳細は「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」を参照してください。
- 外部認証局が発行するユーザー証明書



注記

スマートカード認証の使用を開始する場合は、ハードウェアの要件を参照してください。RHEL9 でのスマートカードのサポート。

2.2. 信頼の構成と証明書の使用に必要な WINDOWS SERVER 設定

本セクションでは、Windows Server で設定する必要があるものを要約します。

- 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
- 以下の形式の証明書が必要になります。個人情報交換: **PKCS #12(.PFX)**
  - 証明書のプライバシーを有効にする

2.3. 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。

2.4. 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構成

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 サーバーとクライアントを構成して設定する
- CA 証明書を予想される場所にコピーする

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

この手順では以下を説明します。

- 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 クライアントの場合: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

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。

2.5. 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展開します。

```
[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展開します。

```
[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。

2.6. 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

スマートカードを設定するには、証明書を生成し、スマートカードに保存するツールが必要になります。

以下を行う必要があります。

- 証明書管理に役立つ **gnutls-utils** パッケージをインストールする。
- スマートカードと連携するライブラリーおよびユーティリティーのセットを提供する **opensc** パッケージをインストールします。
- スマートカードリーダーと通信する **pcscd** サービスを開始する。

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。

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

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

pcscd サービスが稼働していることを確認します。

2.7. スマートカードでの証明書の保存

本セクションでは、設定に役立つ **pkcs15-init** によるスマートカードの設定を説明します。

- スマートカードの消去
- 新しい PIN およびオプションの 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
- スマートカード設定のロック (一部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になります)

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。この手順では、**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 PIN と PUK を設定します。また、セキュリティーオフィス 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。

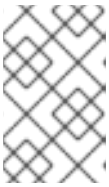
3. スロットのラベルと認証 ID を設定します。

```
$ 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、秘密鍵と証明書を保存する際に同じである必要があります。**--id** の値を指定しないと、ツールによりより複雑な値が計算されるため、独自の値の定義が容易になります。

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. (オプション) 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している場合は、その秘密鍵または証明書と同じ ID を指定する必要があります。

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るものもあります。

```
$ 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。

2.8. 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。

2.9. 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、「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。

第3章 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

証明書マッピングルールは、Identity Management (IdM) 管理者が特定のユーザーの証明書にアクセスしない場合に、シナリオで証明書を使用して認証できるため便利な方法です。通常、このようなアクセスがない理由は、証明書が外部認証局によって発行されたためです。特別なユースケースは、IdM ドメインが信頼関係にある Active Directory (AD) の証明書システムが発行した証明書によって表されます。

証明書マッピングルールは、スマートカードを使用するユーザーが多く、IdM 環境が大きい場合にも便利です。このような場合、完全な証明書を追加すると複雑になります。ほとんどの場合、発行先と発行者は予測可能であるため、完全な証明書よりも簡単に追加できます。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、証明書は一定間隔で更新する必要があるため、証明書マッピングルールは管理のオーバーヘッドを軽減します。ユーザーの証明書を更新する際に、管理者がユーザーエントリーを更新する必要がありません。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 (通常は発行者と題名)、または完全な証明書をユーザーアカウントに読み込む必要があります。

2. ユーザーが IdM へのログインを問題なく行えるようにするために、管理者が証明書マッピングルールを作成する必要があります。

   a. アカウントに、証明書マッピングデータエントリーが含まれる

   b. 証明書マッピングデータエントリーが、証明書の情報と一致する

   マッピングルールを構成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、Components of an identity mapping rule in IdM およびObtaining the issuer from a certificate for use in a matching ruleを参照してください。

その後、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際に適切に認証されます。

3.1. ACTIVE DIRECTORY ドメインとの信頼に対する証明書マッピングルール

本セクションでは、IdM デプロイメントが Active Directory (AD) ドメインと信頼関係にある場合に可能な、別の証明書マッピングのユースケースを簡単に説明します。

証明書マッピングルールは、信頼された AD 証明書システムが発行したスマートカード証明書を持つユーザーに対して、IdM リソースにアクセスするのに便利な方法です。AD 設定によっては、以下の状況が考えられます。

- 証明書が AD で発行され、ユーザーと証明書が IdM に保存されている場合、マッピングと、認証リクエストの全処理は IdM 側で行われます。このシナリオの設定に関する詳細は 「IdM に保存されたユーザーの証明書マッピングの設定」を参照してください。

- ユーザーが AD に保存されている場合は、認証要求の処理が AD で実行されます。サブケースは 3 つあります。
  - AD ユーザーエントリーに、証明書全体が含まれる場合。このシナリオで IdM を設定する方法は、「AD ユーザーエントリーに証明書全体が含まれるユーザー用の証明書マッピングの設定」を参照してください。
  - AD が、ユーザー証明書をユーザーアカウントにマップするように設定されている場合。この場合、AD ユーザーエントリーには証明書全体が含まれず、代わりに **altSecurityIdentities** と呼ばれる属性が含まれます。このシナリオで IdM を設定する方法は、「AD がユーザー証明書をユーザーアカウントにマッピングするように設定している場合は、証明書マッピングの設定」を参照してください。
  - AD ユーザーエントリーに、証明書全体またはマッピングデータが含まれない場合。この場合の解決策として、**ipa idoverrideuser-add** コマンドを使用して、IdM で AD ユーザーの ID オーバーライドに証明書全体を追加します。詳細は「AD ユーザーエントリーに証明書やマッピングデータが含まれていない場合に、証明書マッピングの設定」を参照してください。

3.2. IDM における ID マッピングルールのコンポーネント

本セクションでは、IdM の **ID マッピングルール** のコンポーネントと、その設定方法を説明します。各コンポーネントには、上書きできるデフォルト値があります。コンポーネントは、Web UI または CLI のいずれかで定義できます。CLI では、**ipa certmaprule-add** コマンドを使用して、ID マッピングルールが作成されます。

マッピングルール

マッピングルールコンポーネントでは、証明書を 1 人または複数のユーザーアカウントに関連付けます (または **マップ** します)。ルールは、証明書を目的のユーザーアカウントに関連付ける LDAP 検索フィルターを定義します。
さまざまな認証局 (CA) が発行する証明書にはさまざまなプロパティーがあり、さまざまなドメインで使用される可能性があります。そのため、IdM はマッピングルールを無条件に適用せず、適切な証明書にのみ適用されます。適切な証明書は、**マッチングルール** を使用して定義されます。

マッピングルールのオプションを空のままにすると、証明書は、DER でエンコードされたバイナリーファイルとして、**userCertificate** 属性で検索されることに注意してください。

--maprule オプションを使用して、CLI でマッピングルールを定義します。

マッチングルール

マッチングルールコンポーネントは、マッピングルールを適用する証明書を選択します。デフォルトのマッチングルールは、**digitalSignature 鍵** の使用と、**clientAuth 拡張鍵** の使用で、証明書と一致します。
--matchrule オプションを使用して、CLI にマッチングルールを定義します。

ドメインリスト

ドメイン一覧は、ID マッピングルールの処理時に IdM がユーザーを検索する ID ドメインを指定します。このオプションを指定しないと、IdM は、IdM クライアントが所属しているローカルドメイン内でのみユーザーを検索します。
--domain オプションを使用して CLI にドメインを定義します。

優先度

複数のルールが証明書に適用される場合は、最も優先度が高いルールが優先されます。その他のルールはすべて無視されます。

- 数値が低いほど、ID マッピングルールの優先度が高くなります。たとえば、優先度 1 のルールは、優先度 2 のルールよりも高く設定されています。
- ルールに優先度の値が定義されていないと、優先度が最も低くなります。

--priority オプションを使用して、CLI にマッピングルールの優先度を定義します。

証明書マッピングルールの例 1

CLI を使用して、その証明書の **Subject** が IdM のユーザーアカウントの **certmapdata** エントリーと一致している場合に限り、**EXAMPLE.ORG** 組織の **スマートカード CA** が発行する証明書を認証局が認証できるようにする証明書マッピングルール **simple_rule** を定義するには、次のコマンドを実行します。

```
# ipa certmaprule-add simple_rule --matchrule '<ISSUER>CN=Smart Card
CA,O=EXAMPLE.ORG' --maprule '(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>
{subject_dn!nss_x500})'
```

3.3. マッチングルールで使用する証明書から発行者の取得

この手順では、証明書から発行者情報を取得して、証明書マッピングルールのマッチングルールにコピーする方法を説明します。マッチングルールで必要な発行者の形式を取得するには、**openssl x509** ユーティリティーを使用します。

前提条件

- **.pem** 形式または **.crt** 形式のユーザー証明書がある。

手順

1. 証明書からユーザー情報を取得します。以下のように、**openssl x509** 証明書の表示および署名ユーティリティーを使用します。
   - リクエストのエンコードされたバージョンの出力を防ぐ **-noout** オプション
   - 発行者名を出力する **-issuer** オプション
   - 証明書を読み込む入力ファイル名を指定する **-in** オプション
   - **RFC2253** 値と共に **-nameopt** オプションを指定して、最初に最も具体的な相対識別名 (RDN) で出力を表示します。
     入力ファイルに Identity Management 証明書が含まれる場合は、コマンドの出力で、**Organization** 情報を使用して 発行者が定義されていることを示しています。

     ```
     # openssl x509 -noout -issuer -in idm_user.crt -nameopt RFC2253
     issuer=CN=Certificate Authority,O=REALM.EXAMPLE.COM
     ```

     入力ファイルに Active Directory 証明書が含まれる場合は、コマンドの出力で、**ドメインコンポーネント** の情報を使用して発行者が定義されていることを示しています。

     ```
     # openssl x509 -noout -issuer -in ad_user.crt -nameopt RFC2253
     issuer=CN=AD-WIN2012R2-CA,DC=AD,DC=EXAMPLE,DC=COM
     ```
2. 必要に応じて、証明書発行者が、**ad.example.com** ドメインから展開した **AD-WIN2012R2-CA**

であることを指定するマッチングルールに基づいて、CLI で新しいマッピングルールを作成する場合は、証明書の発行先が、IdM のユーザーアカウントにある **certmapdata** エントリーと一致する必要があります。

```
# ipa certmaprule-add simple_rule --matchrule '<ISSUER>CN=AD-WIN2012R2-CA,DC=AD,DC=EXAMPLE,DC=COM' --maprule '(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})'
```

3.4. 関連情報

- **sss-certmap(5)** のman ページを参照してください。

第4章 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

本章では、以下のシナリオを説明します。

- 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
- **authselect** を使用してスマートカードを設定する

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したい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する。
- 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 9 で対応しているスマートカードまたは USB デバイス
  詳しくは、Smart Card support in RHEL9 を参照してください。

4.1. ローカル証明書の作成

本セクションでは、以下のタスクを実行する方法を説明します。

- 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による手順に従います。

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。

4.2. SSSD ディレクトリーへの証明書のコピー

GNOME デスクトップマネージャー (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ます。

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

```
# 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。

4.3. 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

スマートカードを設定するには、証明書を生成し、スマートカードに保存するツールが必要になります。

以下を行う必要があります。

- 証明書管理に役立つ **gnutls-utils** パッケージをインストールする。
- スマートカードと連携するライブラリーおよびユーティリティーのセットを提供する **opensc** パッケージをインストールします。
- スマートカードリーダーと通信する **pcscd** サービスを開始する。

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。

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

2. **pcscd** サービスを開始します。

```
# systemctl start pcscd
```

pcscd サービスが稼働していることを確認します。

4.4. スマートカードでの証明書の保存

本セクションでは、設定に役立つ **pkcs15-init** によるスマートカードの設定を説明します。

- スマートカードの消去
- 新しい PIN およびオプションの 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
- スマートカード設定のロック (一部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になります)

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。この手順では、**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 PIN と PUK を設定します。また、セキュリティーオフィス 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。

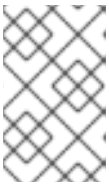
3. スロットのラベルと認証 ID を設定します。

```
$ 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、秘密鍵と証明書を保存する際に同じである必要があります。**--id** の値を指定しないと、ツールによりより複雑な値が計算されるため、独自の値の定義が容易になります。

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. (オプション) 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している場合は、その秘密鍵または証明書と同じ ID を指定する必要があります。

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るものもあります。

```
$ 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。

4.5. 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。本セクションでは、以下を説明します。

- スマートカードに保存されている証明書を使用して認証を有効にするのに必要な設定
- **authselect** ツールを使用した削除設定のロック

削除時にロックを設定すると、スマートカードの削除後に強制的にログアウトされます。

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。
- ユーザー名は、証明書の SUBJECT の CN (Common Name) または UID (User ID) と一致します。
- **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** 一覧に追加し、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 with-smartcard-lock-on-removal --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5章 AUTHSELECT でスマートカードの設定

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
- パスワードを無効にしてスマートカード認証を有効化
- 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 9 で対応しているスマートカードまたは USB デバイス
  詳しくは、Smart Card support in RHEL9 を参照してください。

5.1. 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。
- 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できます。
  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。

5.2. ユーザーパスワード認証を有効にしてスマートカード認証を設定

本セクションでは、システムでスマートカードとパスワードの両方認証を有効にする方法を説明します。

前提条件

- スマートカードに、証明書と秘密鍵が含まれる。
- このカードは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** ツールがシステム上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。

5.3. スマートカード認証を強制するための AUTHSELECT の設定

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することができます。**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。
- このカードは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。

5.4. 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することができます。**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

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。
- このカードは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6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

6.1. IDM での SUDO ルールの作成

この手順では、リモートホストで sudo を実行する権限を **ipauser1** に付与するために、IdM で sudo ルールを作成する方法を説明します。

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。

6.2. SUDO 用の PAM モジュールの設定

この手順では、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する方法を説明します。

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。

6.3. スマートカードを使用した SUDO へのリモート接続

この手順では、スマートカードを使用して sudo にリモートで接続するために SSH エージェントおよびクライアントを設定する方法を説明します。

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 転送を有効にして（**-A** オプションを使用して）SSH 経由で **sudo** をリモートで実行するシステムに接続します。

```
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
```

PIN またはパスワードの入力を要求されないはずです。

第7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

7.1. システムでのスマートカードアクセスのテスト

この手順では、スマートカードにアクセスできるかどうかをテストする方法を説明します。

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。
- **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 でテストおよび対応しているスマートカードおよびリーダーの詳細は、「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。

7.2. 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

この手順では、スマートカードを使用して SSSD を使用した認証のトラブルシューティングを行う方法を説明します。

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。
- **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 でのログインについては、Troubleshooting authentication with SSSD in IdM 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 でのログインについては、Troubleshooting authentication with SSSD in IdM 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
```

7.3. 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

この手順では、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を説明します。

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一覧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。

7.4. 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。

7.5. 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。

7.5.1. 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。この手順では、証明書マッピングルールを確認する方法を説明します。

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 を参照してください。
- スマートカード証明書を IdM ユーザーにマッピングしました。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。

7.5.2. 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 を参照してください。
- スマートカード証明書を IdM ユーザーにマッピングしました。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。