



Red Hat Enterprise Linux 7 システムレベルの認証ガイド

認証および ID 管理に関するシステムレベルのサービス

Ella Deon Ballard

Tomáš Čapek

Aneta Petrová

認証および ID 管理に関するシステムレベルのサービス

Ella Deon Ballard
Red Hat Customer Content Services
dlackey@redhat.com

Tomáš Čapek
Red Hat Customer Content Services
tcapek@redhat.com

Aneta Petrová
Red Hat Customer Content Services
apetrova@redhat.com

法律上の通知

Copyright © 2015 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、ローカルシステム上で認証設定に利用可能な様々なアプリケーションとサービスを扱っています。

目次

第1章 システム認証について	2
1.1. ユーザー ID の確認	2
1.2. シングルサインオンのプランニング	2
1.3. 利用可能なサービス	3
パート I. システムログイン	5
第2章 システム認証の設定	6
2.1. authconfig ユーティリティーの使用	6
2.2. 認証用 ID ストアを選択	11
2.3. 認証メカニズムの設定	21
2.4. キックスタートと設定ファイルの管理	44
2.5. カスタムホームディレクトリーの有効化	44
2.6. 設定の保存と復元	46
パート II. ID および認証ストア	47
第3章 SSSD による認証情報の使用とキャッシング	48
3.1. SSSD 設定の基本	48
3.2. SSSD とシステムサービス	51
3.3. SSSD および ID プロバイダー (ドメイン)	68
3.4. SSSD でのローカルシステムユーザーの管理	111
3.5. SSSD のダウングレード	115
3.6. SSSD と NSCD の使用	116
3.7. SSSD のトラブルシューティング	116
第4章 realmd を使った ID ドメインへの接続	123
パート III. セキュアなアプリケーション	124
第5章 PAM (プラグ可能な認証モジュール) の使用	125
5.1. PAM について	125
5.2. PAM 設定ファイルについて	125
5.3. PAM と管理認証情報のキャッシング	129
第6章 Kerberos の使用	131
6.1. Kerberos について	131
6.2. Kerberos KDC の設定	135
6.3. Kerberos クライアントの設定	140
6.4. スマートカード用の Kerberos クライアントの設定	141
6.5. レルム間 Kerberos 信頼の設定	142
第7章 certmonger を使った作業	147
7.1. certmonger および認証局	147
7.2. certmonger を使った証明書のリクエスト	147
7.3. NSS データベースでの証明書の保存	148
7.4. certmonger を使った証明書の追跡	149
第8章 アプリケーションのシングルサインオン用の設定	150
8.1. Firefox でシングルサインオンに Kerberos を使用するよう設定する	150
8.2. トークン用の SSL をサポートするブラウザの設定	151
8.3. 証明書をメールクライアントのトークンに使用する	154
付録A 改訂履歴	157

第1章 システム認証について

セキュアなネットワーク環境を確立するための第一歩は、ネットワークへのアクセス権限を持つユーザーにアクセスを限定することです。アクセスが許可されるとユーザーはシステムに対して **認証** することができます。つまり、ユーザー自身の ID を実証できることとなります。

Red Hat Enterprise Linux システムでは、ユーザー ID を作成し、識別する多くの異なるサービスが利用可能になっています。これらはローカルシステムファイルや、Kerberos や Samba などの大型の ID ドメインに接続するサービスであったり、このようなドメインを作成するツールなどです。

本ガイドでは、ローカルシステムで管理者が認証および ID 管理に利用できる一般的なシステムサービスとアプリケーションについて説明しています。[creating Linux domains](#) や [integrating a Linux system into a Windows domain](#) についての詳細情報は、個別のガイドが用意されています。

1.1. ユーザー ID の確認

認証 とは、ID を確認するプロセスのことです。ネットワークの相互作用では、一方が他方を特定することで認証が行われます。ネットワーク上での認証には多くの方法があります。簡単なパスワードや証明書、ワンタイムパスワード (OTP) トークンや生体認証スキャンなどです。

一方で **承認** では、認証された関係者が許可される動作やアクセスを定義します。

認証においては、ユーザーが自身の ID を証明するためになんらかの **認証情報** を提示することが求められます。求められる認証情報の種類は、使用される認証メカニズムによって定義されます。システム上のローカルユーザーは、以下のような認証が利用できます。

- ※ **パスワードベースの認証**。ほとんどすべてのソフトウェアでは、ユーザーが提供する名前とパスワードによる認証を許可しています。これは **簡易認証** とも呼ばれます。
- ※ **証明書ベースの認証**。証明書をベースとしたクライアントの認証は、SSL プロトコルの一部です。クライアントはランダムに生成されたデータにデジタル処理で署名し、証明書と証明済みのデータをネットワーク経由で送信します。サーバーは署名を確認して証明書の有効性を確認します。
- ※ **Kerberos 認証**。Kerberos は **ticket-granting tickets (TGT)** と呼ばれる短期の認証情報システムを確立します。ユーザーがユーザー名とパスワードという認証情報を提示することでユーザーが特定され、このユーザーにチケットを発行可能であることをシステムに対して示します。TGT はその後、ウェブサイトや電子メールなどの他のサービスへのアクセスチケットを要求する際に繰り返し使用することができます。このように TGT 経由の認証では、ユーザーの認証プロセスは一度で済みます。
- ※ **スマートカードベースの認証**。これは証明書ベースの認証とわずかに異なるものです。スマートカード (またはトークン) にはユーザーの証明書が保存されています。ユーザーがトークンをシステムに挿入すると、システムは証明書を読み取り、アクセスを許可することができます。スマートカードを使ったシングルサインオンには、以下の 3 つのステップがあります。
 - ※ ユーザーがスマートカードをカードリーダーに差し込みます。PAM (プラグ可能な認証モジュール) が差し込まれたスマートカードを検出します。
 - ※ システムが証明書をユーザーエントリーにマッピングし、スマートカード上で提示された証明書とユーザーエントリーで保存されている証明書を比較します。前者は、証明書ベースの認証で説明されている秘密鍵で暗号化されています。
 - ※ 証明書がキー配布センター (KDC) に対して正常に確認されると、ユーザーはログインを許可されます。

スマートカードベースの認証は、Kerberos によって確立された簡易認証の層に物理的アクセス要件と認証情報を新たな識別メカニズムとして追加することで構築されます。

1.2. シングルサインオンのプランニング

「[ユーザー ID の確認](#)」で説明された認証では、セキュアなアプリケーションすべてでアクセスのために少なくともパスワードが必要になります。中央の ID ストアがなかったりアプリケーションが独自にユーザーと認証情報を維持していなければ、ユーザーはサービスやアプリケーションを開くたびにパスワードの入力が求められます。こうなると一日に何度も、場合によっては数分ごとにパスワードを入力しなくてはなりません。

複数のパスワードを維持してそれらを何度も入力することは、ユーザーおよび管理者にとって大変な負担です。**シングルサインオン**を使うと、管理者は単一のパスワードストアを作成できるので、ユーザーは単一のパスワードを使ってログインして、すべてのネットワークリソースに認証されることが可能になります。

Red Hat Enterprise Linux では、ワークステーションへのログインやスクリーンセーバーの解除、Mozilla Firefox を使った安全なウェブページへのアクセスなど、複数のリソースに対してシングルサインオンをサポートしています。PAM、NSS、および Kerberos などの他の利用可能なシステムサービスを使うと、他のシステムアプリケーションもこれらの ID ソースを使用するように設定できます。

シングルサインオンはユーザーの利便性を高めると同時に、サーバーおよびネットワークの新たなセキュリティ層の役割も果たします。シングルサインオンはセキュアで効果的な認証の要所となります。Red Hat Enterprise Linux では、シングルサインオンを有効にする以下の 2 つの認証メカニズムを提供しています。

- ※ Kerberos レalmと Active Directory ドメインの両方を使った Kerberos ベースの認証
- ※ スマートカードベースの認証

これらのメカニズムは両方とも (Kerberos レalmまたは公開鍵インフラストラクチャーの認証局により) 中央 ID ストアを作成します。ローカルシステムのサービスは複数のローカルストアを維持するのではなく、これらの ID ドメインを使用します。

1.3. 利用可能なサービス

すべての Red Hat Enterprise Linux システムには、ローカルシステム上のローカルユーザーの認証が設定可能なサービスがあります。以下のものが含まれます。

認証セットアップ

- ※ 認証設定ツール (**authconfig**) はシステム用に異なる ID バックエンドと認証方法 (パスワードや指紋、スマートカードなど) をセットアップします。

ID バックエンドセットアップ

- ※ Security System Services Daemon (SSSD) は複数の ID プロバイダー (主に Microsoft Active Directory や Red Hat Enterprise Linux IdM などの LDAP ベースのディレクトリー) をセットアップし、これをローカルシステムとアプリケーションの両方のユーザーに使用することができます。パスワードとチケットはキャッシュされるので、認証情報を再利用してオフライン認証とシングルサインオンの両方が可能になります。
- ※ **realmd** サービスはコマンドラインユーティリティーで、IdM 用の SSSD である認証バックエンドの設定を可能にします。**realmd** サービスは DNS レコードに基づいて利用可能な IdM ドメインを検出し、SSSD を設定してからドメインのアカウントとしてシステムに参加します。
- ※ NSS (ネームサーバースイッチ) は、ユーザー、グループ、またはホストの情報を返信する低レベルのシステムコール用のメカニズムです。NSS は、必要な情報を取得するためにどのソース、つまりどのモジュールを使用すべきか判断します。たとえば、ユーザー情報は `/etc/passwd` ファイルなどの従来の UNIX ファイルか LDAP ベースのディレクトリーで見つかりますが、ホストアドレスは `/etc/hosts` ファイルなどのファイルか DNS レコードから読み込みます。NSS は情報の格納場所を見つけます。

認証メカニズム

- ※ PAM (プラグ可能な認証モジュール) は、認証ポリシーをセットアップするシステムを提供します。認証

に PAM を使用するアプリケーションは、認証における異なる要素を制御する異なるモジュールを読み込みます。アプリケーションがどの PAM モジュールを使用するかは、そのアプリケーションの設定方法に基づきます。利用可能な PAM には、Kerberos、Winbind、ローカルの UNIX ファイルベースの認証などがあります。

他のサービスやアプリケーションも利用可能ですが、本ガイドでは上記のものが中心になります。

パート I. システムログイン

第2章 システム認証の設定

認証とは、ユーザーがシステムに対して識別され、確認される手段です。認証プロセスでは、ユーザー名とパスワードなど、なんらかの ID と認証情報が必要になります。この認証情報は、システム上のデータストアに保存してある情報と比較されます。Red Hat Enterprise Linux では、**authconfig** ツールが LDAP などのユーザー認証情報に使用するデータストアの設定を支援します。

利便性とシングルサインオンの一部のために、Red Hat Enterprise Linux は System Security Services Daemon (SSSD) を中心となるデーモンとして使用し、異なる ID バックエンドに対してユーザーを認証したり、ユーザー用に ticket-granting ticket (TGT) を依頼することができます。SSSD は、LDAP や Kerberos、外部のアプリケーションと対話して、ユーザーの認証情報を確認することができます。**authconfig** ツールはシステムが異なる形式の認証メカニズムとともに、SSSD、NIS、Winbind、LDAP などの特定サービスを使用するよう設定することができます。

ユーザーが Red Hat Enterprise Linux システムにログインする際は、なんらかの **認証情報** を提示してユーザーの ID を確立します。システムはその認証情報を設定済みの認証サービスに照らし合わせてチェックします。認証情報が一致してユーザーのアカウントがアクティブの場合、ユーザーは**認証されます**。(ユーザーが認証されると、その情報はアクセス制御サービスに渡され、そのユーザーに何が許可されているかを決定します。これが、ユーザーがアクセスする **認証済み** リソースです。)

ユーザーを確認するための情報はローカルシステム上に位置するか、ローカルシステムが LDAP や Kerberos などのリモートシステム上のユーザーデータベースを参照する場合があります。

システムはユーザー認証をチェックするために、有効なアカウントデータベースの設定済み一覧を保有している必要があります。Red Hat Enterprise Linux では、**authconfig** ツールが GUI とコマンドラインの両方のオプションを保有していて、すべてのユーザーデータストアを設定します。

ローカルシステムでは、Lightweight Directory Access Protocol (LDAP)、Network Information Service (NIS)、および Winbind などの様々な異なるデータストアをユーザー情報に使用することができます。さらに LDAP と NIS の両データストアは Kerberos を使ってユーザーを認証することができます。



重要

インストール中またはセキュリティレベル設定ツールで中レベルまたは高レベルのセキュリティが設定されると、ファイアウォールは NIS 認証を妨げます。ファイアウォールの詳細については『セキュリティガイド』の「ファイアウォール」のセクションを参照してください。

2.1. authconfig ユーティリティの使用

認証セッティングの設定には、以下の 3 つのユーティリティが使用できます。

- ✦ **authconfig-gtk** は、完全なグラフィカルインターフェイスを提供します。
- ✦ **authconfig** は、手動での設定に使用するコマンドラインインターフェイスを提供します。
- ✦ **authconfig-tui** は、シンプルなテキストベースの UI を提供します。このユーティリティは非推奨であることに注意してください。



注記

認証設定ユーティリティはすべて、**root** で実行する必要があります。

**重要**

IdM の設定には、制限があり柔軟性に欠ける `authconfig` ではなく、`ipa-client-install` または `realmd` の使用が推奨されます。

2.1.1. authconfig CLI 使用時のヒント

`authconfig` コマンドラインツールは、スクリプトに渡されたセッティングにしたがい、システム認証に必要な設定ファイルとサービスのすべてを更新します。UI で設定可能な ID と認証設定オプションよりもさらに多くを提供するとともに、`authconfig` ツールを使うとバックアップファイルとキックスタートファイルも作成できます。

`authconfig` オプションの完全なリストについては、ヘルプの出力と `man` ページを参照してください。

`authconfig` の実行に際しては、以下の点に留意してください。

- ※ すべてのコマンドで `--update` または `--test` のオプションを使用してください。コマンドが正常に実行されるには、これらのオプションのいずれかが必要になります。`--update` を使用すると設定の変更が書き込まれます。`--test` は標準出力に変更をプリントしますが、設定への変更は適用されません。

`--update` オプションを使用しないと、システム設定ファイルに変更が書き込まれません。

- ※ コマンドラインは、新規設定と既存設定の更新の両方に使用することができます。このためコマンドラインは、特定の呼び出しに必須の属性が使用されることを強制しません (強制すると、コマンドが設定すべてを更新する可能性があるため)。

認証設定を編集する際は、設定が完全かつ正確であることを確認してください。認証設定を不完全なものまたは間違った値に変更すると、ユーザーがシステムからロックアウトされてしまう可能性があります。`--update` オプションを使用して変更を書き込む前に、`--test` オプションで設定が適切であることを確認してください。

- ※ それぞれの「enable」オプションには対応する「disable」オプションがあります。

2.1.2. authconfig UI をインストールする

`authconfig` UI は、デフォルトではインストールされませんが、管理者が認証設定に簡単な変更を加える際には役に立ちます。

UI をインストールするには、`authconfig-gtk` パッケージをインストールします。このパッケージには、`authconfig` コマンドラインツールや Bash、Python など一般的なシステムパッケージへの依存関係があります。これらのほとんどは、デフォルトでインストールされます。

```
[root@server ~]# yum install authconfig-gtk
Loaded plugins: langpacks, product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package authconfig-gtk.x86_64 0:6.2.8-8.el7 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package                Arch          Version      Repository    Size
=====
Installing:
authconfig-gtk         x86_64        6.2.8-8.el7  RHEL-Server  105 k
=====
```

```
Transaction Summary
=====
```

```
Install 1 Package
```

```
... 8< ...
```

2.1.3. authconfig UI を起動する

1. 端末を開いて、root でログインします。
2. `system-config-authentication` コマンドを実行します。



重要

`authconfig` UI を閉じると、すぐに変更が反映されます。

認証の設定 ダイアログボックスには以下の 3 つの設定タブがあります。

- ※ **識別と認証**。ID ストア (ユーザー ID と対応する認証情報が保存されるデータレポジトリ) として使用されるリソースを設定します。
- ※ **高度なオプション**。スマートカードや指紋など、パスワードや認証情報以外の認証方法が可能になります。
- ※ **パスワードオプション**。パスワード認証方法を設定します。

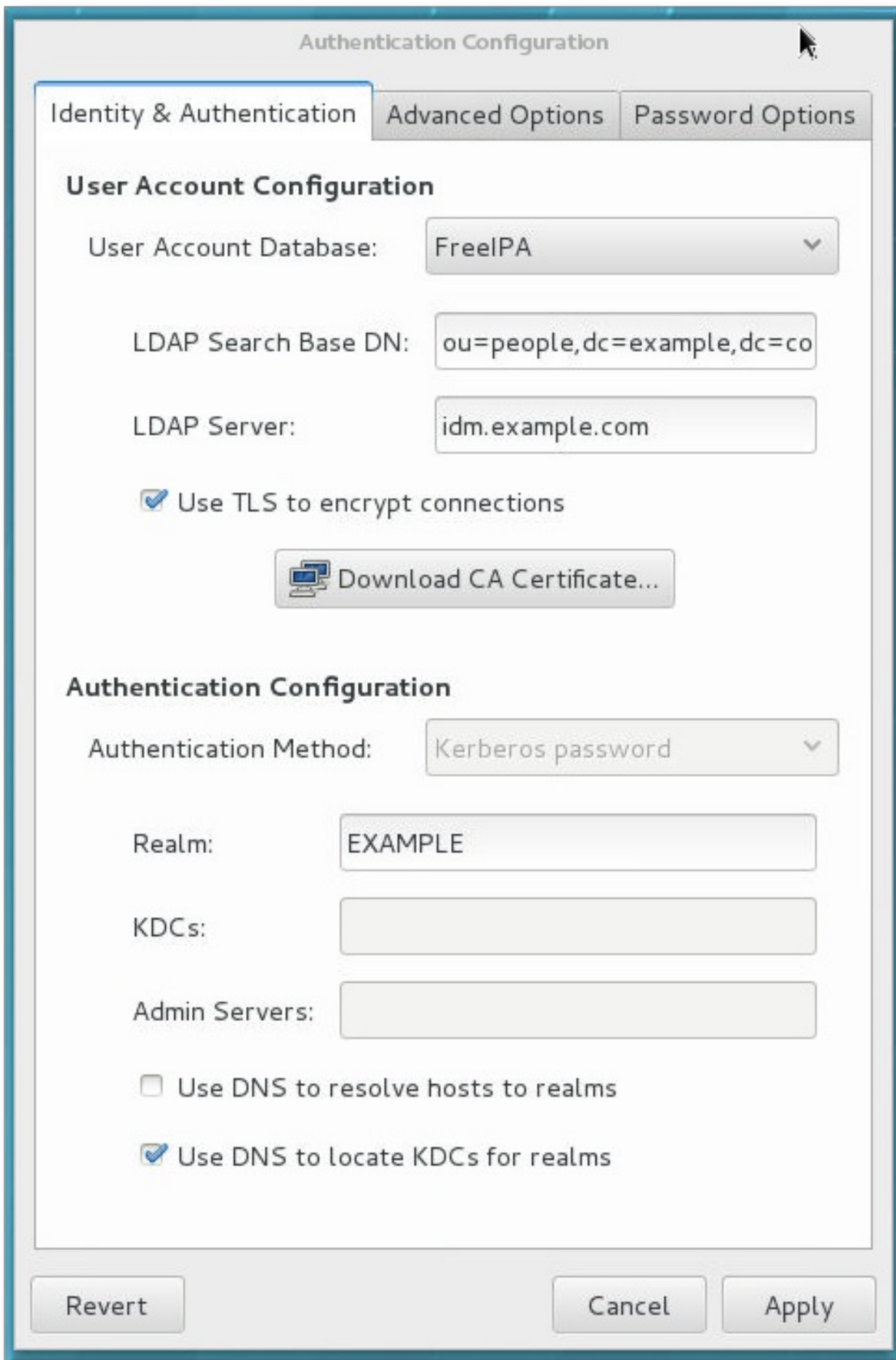


図2.1 authconfig ウィンドウ

2.1.4. 認証設定のテスト

認証設定が完全かつ適切に行われることは重要です。これが行われないと、最悪の場合、すべてのユーザー (root さえも) システムからロックアウトされてしまいます。ここまでひどくなくても、何人かのユーザーがロックアウトされてしまったり、間違った ID が使われてしまう恐れがあります。

`--test` オプションは、システム用のすべての ID および認証メカニズムに関する認証設定をプリントします。これには、有効なものおよび無効なエリアの設定の両方が表示されます。

`test` オプションはそれのみで使用して完全な現行設定を表示するか、`authconfig` コマンドと使用して **設定がどのように変更されるか** (実際に変更せずに) を表示することができます。これは、提示された認証設定が完全かつ正確かどうかを確認する上で非常に便利なものです。

```
[root@server ~]# authconfig --test
caching is disabled
nss_files is always enabled
nss_compat is disabled
nss_db is disabled
nss_hesiod is disabled
  hesiod LHS = ""
  hesiod RHS = ""
nss_ldap is disabled
  LDAP+TLS is disabled
  LDAP server = ""
  LDAP base DN = ""
nss_nis is disabled
  NIS server = ""
  NIS domain = ""
nss_nisplus is disabled
nss_winbind is disabled
  SMB workgroup = "MYGROUP"
  SMB servers = ""
  SMB security = "user"
  SMB realm = ""
  Winbind template shell = "/bin/false"
  SMB idmap range = "16777216-33554431"
nss_sss is enabled by default
nss_wins is disabled
nss_mdns4_minimal is disabled
DNS preference over NSS or WINS is disabled
pam_unix is always enabled
  shadow passwords are enabled
  password hashing algorithm is sha512
pam_krb5 is disabled
  krb5 realm = "#"
  krb5 realm via dns is disabled
  krb5 kdc = ""
  krb5 kdc via dns is disabled
  krb5 admin server = ""
pam_ldap is disabled
  LDAP+TLS is disabled
  LDAP server = ""
  LDAP base DN = ""
  LDAP schema = "rfc2307"
pam_pkcs11 is disabled
  use only smartcard for login is disabled
```

```
smartcard module = ""
smartcard removal action = ""
pam_fprintd is disabled
pam_ecryptfs is disabled
pam_winbind is disabled
SMB workgroup = "MYGROUP"
SMB servers = ""
SMB security = "user"
SMB realm = ""
pam_sss is disabled by default
credential caching in SSSD is enabled
SSSD use instead of legacy services if possible is enabled
IPAv2 is disabled
IPAv2 domain was not joined
IPAv2 server = ""
IPAv2 realm = ""
IPAv2 domain = ""
pam_pwquality is enabled (try_first_pass local_users_only retry=3 authtok_type=)
pam_passwdqc is disabled ()
pam_access is disabled ()
pam_mkhome or pam_oddjob_mkhome is disabled (umask=0077)
Always authorize local users is enabled ()
Authenticate system accounts against network services is disabled
```

2.2. 認証用 ID ストアの選択

識別と認証 タブは、ユーザーの認証方法を設定します。デフォルトではローカルシステムの認証を使用します。つまり、ユーザーとユーザーのパスワードがローカルシステムのアカウントに対してチェックされます。Red Hat Enterprise Linux のマシンは、ユーザーと認証情報を含む LDAP、NIS、および Winbind などの外部リソースを使用することもできます。

2.2.1. IPAv2

Identity Management サーバーを ID バックエンドとして設定するには、2 つの方法があります。IdM のバージョン 2 (Red Hat Enterprise Linux バージョン 6.3 およびそれ以前)、バージョン 3 (Red Hat Enterprise Linux 6.4 およびそれ以降)、およびバージョン 4 (Red Hat Enterprise Linux 7.1) では、**authconfig** で IPAv2 プロバイダーとして設定されます。これよりも旧式の IPA バージョンおよびコミュニティの FreeIPA サーバーの場合は、LDAP プロバイダーとして設定されます。

2.2.1.1. UI で IdM を設定する

1. **authconfig** UI を開きます。
2. **ユーザーアカウントデータベース** のドロップダウンメニューで **IPAv2** を選択します。

Authentication Configuration

Identity & Authentication | Advanced Options | Password Options


User Account Configuration

User Account Database:

LDAP Search Base DN:

LDAP Server:

Use TLS to encrypt connections

 Download CA Certificate...

Authentication Configuration

Authentication Method:

Realm:

KDCs:

Admin Servers:

Use DNS to resolve hosts to realms

Use DNS to locate KDCs for realms

Revert | Cancel | Apply

3. IdM サーバーへの接続に必要な情報を設定します。

※ IPA ドメイン には、IdM ドメインの DNS ドメインを入力します。

- ※ **IPA レルム** には、IdM ドメインの Kerberos ドメインを入力します。
- ※ **IPA サーバー** には、IdM ドメインポロジ内のいずれかの IdM サーバーのホスト名を入力します。
- ※ **NTP を設定しない** チェックボックスを選択すると、クライアント設定時に NTP サービスを無効にします。IdM サーバーとすべてのクライアントは、Kerberos 認証と認証情報が正常に機能するために同期されたクロックを必要とするため、この設定は通常推奨されません。IdM サーバーがドメイン内でホスティングしている NTP サーバー以外のものを使用している場合は、これを無効にすることができます。

4. ドメインへ参加 ボタンをクリックします。

これで `ipa-client-install` コマンドが実行され、必要な場合は `ipa-client` パッケージがインストールされます。インストールスクリプトは、ローカルシステムに必要となるすべてのシステムファイルを自動的に設定し、ドメイン情報更新のためにドメインサーバーに接続します。

2.2.1.2. コマンドラインを使って IdM を設定する

IdM ドメインは、単一階層に一般的かつ必須のサービスをいくつか集中させます。その最たるものは DNS と Kerberos です。

([4章 `realm` を使った ID ドメインへの接続](#) の `realm` のように) `authconfig` を使うとそのドメインにシステムを登録することができます。このコマンドは `ipa-client-install` コマンドを実行し、必要な場合は `ipa-client` パッケージをインストールします。インストールスクリプトは、ローカルシステムに必要となるすべてのシステムファイルを自動的に設定し、ドメイン情報更新のためにドメインサーバーに接続します。

ドメインへの参加には、DNS ドメイン名 (`--ipav2domain`)、Kerberos レルム名 (`--ipav2realm`)、および接続する IdM サーバー (`--ipav2server`) という 3 つの情報が必要になります。`--ipav2join` オプションは、IdM サーバーへの接続に管理者が使用するユーザー名を指定し、通常は `admin` とします。

```
[root@server ~]# authconfig --enableipav2 --ipav2domain=IPAEXAMPLE --
ipav2realm=IPAEXAMPLE --ipav2server=ipaexample.com --ipav2join=admin
```

IdM ドメインが独自の NTP サービスを実行していない場合、`--disableipav2nntp` オプションを使って設定スクリプトが NTP サーバーに IdM サーバーを使用しないようにすることが可能です。IdM サーバーとすべてのクライアントは、Kerberos 認証と認証情報が正常に機能するために同期されたクロックを必要とするため、この設定は通常推奨されません。

2.2.2. LDAP と FreeIPA

(OpenLDAP や Red Hat Directory Server などの) LDAP ディレクトリーは、LDAP ID プロバイダーとして使用することができます。また、古いバージョンの IPA と FreeIPA は、それらに関連する Kerberos サーバーと LDAP プロバイダーとして設定することで、ID プロバイダーとすることができます。

ユーザーデータベースの LDAP サーバー設定には、`openldap-clients` パッケージか `sssd` パッケージを使用します。両パッケージともデフォルトでインストールされています。

2.2.2.1. UI で LDAP 認証を設定する

1. [「authconfig UI を起動する」](#)にあるように、`authconfig` UI を開きます。
2. ユーザーアカウントデータベース のドロップダウンメニューで **LDAP** を選択します。

Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: ou=people,dc=example,dc=co

LDAP Server: ldap://idm.example.com/

Use TLS to encrypt connections

Download CA Certificate...

Authentication Configuration

Authentication Method: Kerberos password

Realm: EXAMPLE

KDCs:

Admin Servers:

Use DNS to resolve hosts to realms

Use DNS to locate KDCs for realms

Revert | Cancel | Apply

3. LDAP サーバーへの接続に必要な情報を設定します。

- ※ **LDAP 検索ベース DN** には、root 接尾辞またはユーザーディレクトリー用の**識別名 (DN)** を入力します。識別/認証に使用されるすべてのユーザーエントリーは、この親エントリー下に存在します。たとえ

ば、`ou=people,dc=example,dc=com` となります。

このフィールドはオプションです。指定されない場合、System Security Services Daemon (SSSD) は、LDAP サーバーの設定エントリーにある `namingContexts` および `defaultNamingContext` 属性を使って検索ベースの検出を試みます。

- ※ **LDAP サーバー** には、LDAP サーバーの URL を入力します。これには通常、`ldap://ldap.example.com:389` のように、LDAP サーバーのホスト名とポート番号の両方が必要になります。

安全なプロトコルを `ldaps://` に入力すると、**CA 証明書をダウンロードする** ボタンが有効になります。

- ※ **TLS を使用して接続を暗号化する** は、LDAP サーバーへの接続の暗号化に Start TLS を使用するかどうかを設定します。これにより、標準ポート上での安全な接続を可能にします。

TLS を選択すると **CA 証明書をダウンロードする** ボタンが有効になり、認証局から発行された LDAP サーバー用の CA 証明書がダウンロードされます。CA 証明書は PEM (プライバシー強化メール) 形式である必要があります。



重要

サーバー URL が安全なプロトコル (`ldaps`) を使用する場合は、**TLS を使用して接続を暗号化する** を選択しないでください。このオプションは Start TLS を使用し、これは標準ポートでの安全な接続を開始します。つまり、安全なポートが指定されている場合、Start TLS ではなく、SSL のようなプロトコルを使用する必要があります。

4. 認証の方法を選択します。LDAP は、単純なパスワード認証または Kerberos 認証を受け付けます。

Kerberos の使用方法は、[「UI で Kerberos 認証を設定する」](#) で説明しています。

LDAP パスワード のオプションは、PAM アプリケーションを使用して LDAP 認証を行います。このオプションは LDAP サーバーへの接続に、安全な (`ldaps://`) URL もしくは TLS オプションを必要とします。

2.2.2.2. コマンドラインで LDAP ユーザストアを設定する

LDAP ID ストアを使用するには、`--enableldap` を使用します。LDAP を認証ソースとして使用するには、`--enableldapauth` を使用して、それから LDAP サーバー名、ユーザー接尾辞のベース DN、および (オプションとして) TLS を使用するかどうか、などの必須となる接続情報を使用します。`authconfig` コマンドには、ユーザーエントリーの RFC 2307bis スキーマを有効または無効にするオプションもあります。これは、`authconfig` UI では利用できません。

プロトコル (`ldap` または `ldaps`) とポート番号を含む完全な LDAP URL を使うようにしてください。`--enableldaptls` オプションとセキュアな LDAP URL (`ldaps`) を一緒に使用しないでください。

```
authconfig --enableldap --enableldapauth --
ldapserver=ldap://ldap.example.com:389, ldap://ldap2.example.com:389 --
ldapbasedn="ou=people,dc=example,dc=com" --enableldaptls --
ldaploadcacert=https://ca.server.example.com/caCert.crt --update
```

LDAP のパスワード認証に `--ldapauth` ではなく、LDAP ユーザストアで Kerberos を使用することもできます。これらのオプションは [「コマンドラインを使って Kerberos 認証を設定する」](#) で説明されています。

2.2.3. NIS



重要

NIS を ID ストアとして設定する前に、NIS 自体を環境にあわせて設定する必要があります。

- ※ NIS サーバーは、ユーザーアカウント設定で完全に設定する必要があります。
- ※ ローカルシステムに `ypbind` パッケージをインストールする必要があります。これは NIS サービスに必要なものですが、デフォルトではインストールされません。
- ※ `portmap` と `ypbind` サービスが起動されており、ブート時に起動するように設定します。これは、`ypbind` パッケージのインストール時に設定します。

2.2.3.1. UI で NIS 認証を設定する

1. [「authconfig UI を起動する」](#)にあるように、`authconfig` UI を開きます。
2. ユーザーアカウントデータベース のドロップダウンメニューで **NIS** を選択します。

The image shows a 'Authentication Configuration' dialog box with three tabs: 'Identity & Authentication' (selected), 'Advanced Options', and 'Password Options'. The 'User Account Configuration' section includes a dropdown for 'User Account Database' set to 'NIS', a text field for 'NIS Domain' with 'NISEXAMPLE', and a text field for 'NIS Server' with 'nis.example.com'. The 'Authentication Configuration' section includes a dropdown for 'Authentication Method' set to 'Kerberos password', a text field for 'Realm' with 'EXAMPLE', and empty text fields for 'KDCs' and 'Admin Servers'. There are two checkboxes: 'Use DNS to resolve hosts to realms' (unchecked) and 'Use DNS to locate KDCs for realms' (checked). At the bottom are 'Revert', 'Cancel', and 'Apply' buttons.

3. NIS サーバーに接続するための情報として、NIS ドメイン名とサーバーのホスト名を設定します。NIS サーバーが指定されない場合は、`authconfig` デーモンが NIS サーバーをスキャンして探します。
4. 認証の方法を選択します。NIS は、単純なパスワード認証または Kerberos 認証を受け付けます。
Kerberos の使用方法は、[「UI で Kerberos 認証を設定する」](#) で説明しています。

2.2.3.2. コマンドラインを使って NIS を設定する

NIS ID ストアを使用するには、`--enablenis` を使います。Kerberos パラメーターが明示的に設定されている場合 ([「コマンドラインを使って Kerberos 認証を設定する」](#)) を除き、ここでは自動的に NIS 認証が使用されます。唯一のパラメーターは、NIS サーバーと NIS ドメインを特定するためのものです。これらが使用されない場合は、`authconfig` サービスはネットワークをスキャンして NIS サーバーを探します。

```
[root@server ~]# authconfig --enablenis --nisdomain=EXAMPLE --
nisserver=nis.example.com --update
```

2.2.4. Winbind

Winbind をシステムに ID ストアとして設定する前に、Samba を設定する必要があります。Samba サーバーはユーザーアカウント用にセットアップするか、バックエンドの ID ストアとして Active Directory を使用するように設定する必要があります。

Samba の設定については、[Samba project documentation](#) で説明されています。Samba を Active Directory との統合ポイントとして設定する方法については、[Red Hat Enterprise Linux Windows Integration Guide](#) で説明しています。

2.2.4.1. authconfig GUI で Winbind を有効にする

1. `samba-winbind` パッケージをインストールします。これは、Samba サービスの Windows 統合機能に必要なものですが、デフォルトではインストールされていません。

```
[root@server ~]# yum install samba-winbind
```

2. `authconfig` UI を開きます。

```
[root@server ~]# authconfig-gtk
```

3. **識別と認証** タブの **ユーザーアカウントデータベース** ドロップダウンメニューで **Winbind** を選択します。

Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

User Account Configuration

User Account Database: Winbind

Winbind Domain: MYGROUP

Security Model: ads

Winbind ADS Realm: AEXAMPLE

Winbind Domain Controllers: adexample.com

Template Shell: /bin/false

Allow offline login

Join Domain...

Authentication Configuration

Authentication Method: Winbind password

Revert | Cancel | Apply

4. Microsoft Active Directory ドメインコントローラーへ接続するために必要な情報を設定します。

※ **Winbind ドメイン** には、接続先の Windows ドメインを入力します。

これは、**DOMAIN** のような Windows 2000 形式にします。

※ **セキュリティモデル** では、Samba クライアントに使用するセキュリティモデルを設定します。**authconfig** は、以下の 4 つのタイプのセキュリティモデルをサポートしています。

- **ads** は、Samba が Active Directory Server (ADS) レルムのドメインメンバーとして機能するように設定します。このモードで操作するには、**krb5-server** パッケージがインストールされ、Kerberos が適切に設定されている必要があります。
- **domain** では、Windows Server と同様の方法で、Windows のプライマリまたはバックアップドメインコントローラーがユーザー名/パスワードを認証することで、Samba がそれらを確認します。
- **server** では、別のサーバー (例えば Windows Server) で認証することにより、ローカルの Samba サーバーがユーザー名/パスワードを確認します。サーバー認証に失敗した場合には、システムは **user** モードで認証を試みます。
- **user** では、クライアントが有効なユーザー名とパスワードでログインする必要があります。このモードは暗号化されたパスワードをサポートします。

ユーザー名の形式は、**EXAMPLE¥jsmith** のように **ドメイン¥ユーザー** とする必要があります。



注記

あるユーザーが Windows ドメイン内に存在することを検証する際は、常に Windows 2000 スタイルの形式を使い、バックスラッシュ文字 (¥) でエスケープしてください。例を示します。

```
[root@server ~]# getent passwd domain¥user
DOMAIN¥user :*:16777216:16777216:Name
Surname:/home/DOMAIN/user:/bin/bash
```

これがデフォルトのオプションになります。

- ※ **Winbind ADS レルム** には、Samba サーバーが参加する Active Directory レルムを入力します。これは **ads** セキュリティモデルの場合にのみ、使用されます。
- ※ **Winbind ドメインコントローラー** には、システム登録に使用するドメインコントローラーのホスト名または IP アドレスを入力します。
- ※ **テンプレートシェル** では、Windows のユーザーアカウント設定に使用するログインシェルを設定します。
- ※ **オフラインログインを許可** は、ローカルキャッシュでの認証情報の保存を許可します。システムがオフラインの時にユーザーがシステムリソースに認証を試みると、キャッシュが参照されます。

2.2.4.2. コマンドラインで Winbind を有効にする

Windows のドメインにはいくつかの異なるセキュリティモデルがあり、ドメインで使用されるセキュリティモデルがローカルシステムの認証設定を決定します。ユーザーとサーバーのセキュリティモデルでは、Winbind 設定で必要となるのはドメイン (またはワークグループ) の名前とドメインコントローラーのホスト名のみです。

--winbindjoin パラメーターは Active Directory ドメイン接続に使用するユーザーを設定し、**--enablelocalauthorize** はローカルの権限付与操作で **/etc/passwd** ファイルを確認するよう設定します。

authconfig コマンドの実行後に、Active Directory ドメインに参加します。

```
[root@server ~]# authconfig --enablewinbind --enablewinbindauth --
smbsecurity=user|server --enablewinbindoffline --smbservers=ad.example.com --
smbworkgroup=EXAMPLE --update --enablelocalauthorize --winbindjoin=admin
[root@server ~]# net join ads
```




注記

ユーザー名の形式は、`EXAMPLE¥jsmith` のように `ドメイン¥ユーザー` とする必要があります。

あるユーザーが Windows ドメイン内に存在することを検証する際は、常に Windows 2000 スタイルの形式を使い、バックスラッシュ文字 (¥) でエスケープしてください。例を示します。

```
[root@server ~]# getent passwd domain¥user
DOMAIN¥user:*:16777216:16777216:Name Surname:/home/DOMAIN/user:/bin/bash
```

ads と domain ドメインのセキュリティモデルでは、Winbind 設定はテンプレートシェルとレルム (ads のみ) の追加設定を許可します。例を示します。

```
[root@server ~]# authconfig --enablewinbind --enablewinbindauth --smbsecurity ads --
enablewinbindoffline --smbservers=ad.example.com --smbworkgroup=EXAMPLE --smbrealm
EXAMPLE.COM --winbindtemplateshell=/bin/sh --update
```

Windows ベースの認証と Windows ユーザーアカウント情報の設定では、名前形式、ドメイン名をユーザー名と一緒に要求するかどうか、UID の範囲など、数多くの他のオプションがあります。これらのオプションは `authconfig` ヘルプ内に記載されています。

2.3. 認証メカニズムの設定

`authconfig` ユーティリティーは、ID ストアとは別に認証動作に関連したセッティングも設定します。これには、まったく異なる認証方式 (指紋スキャンやスマートカード) およびローカル認証ルールが含まれます。

2.3.1. ローカルアカウント

ローカル認証のオプションでは、バックエンドに保存されているユーザーではなく、ローカルシステムのアカウントの設定を定義します。この設定では、システムサービスへのユーザーベースの承認を定義します (`/etc/security/access.conf` で定義される)。そうでない場合は、承認ポリシーは ID プロバイダー内もしくはサービス自体で定義できます。

2.3.1.1. UI でローカルアクセス制御を有効にする

ローカルアクセス制御を有効にする は、ローカルユーザーの承認ルールをシステムが `/etc/security/access.conf` ファイルで確認するように設定します。

The image shows a dialog box titled "Authentication Configuration" with three tabs: "Identity & Authentication", "Advanced Options" (which is selected), and "Password Options".

Local Authentication Options

- Enable fingerprint reader support
- Enable local access control

Tip: This is managed via `/etc/security/access.conf`.

Password Hashing Algorithm: SHA512 (dropdown menu)

Other Authentication Options

- Create home directories on the first login

Smart Card Authentication Options

- Enable smart card support

Tip: Smart cards support logging into both local and centrally managed accounts.

Card Removal Action: Ignore (dropdown menu)

- Require smart card for login

Buttons: Revert, Cancel, Apply

図2.2 ローカルアカウントのフィールド

2.3.1.2. コマンドラインでローカルアクセス制御を設定する

`authconfig` ではローカルアクセス制御を有効にする 2 つのオプションがあります。`--enable locauthorize` はネットワーク権限を省略して、システムユーザーに関してローカルファイルのみを確認します。`--enable pamaccess` は、システムが `/etc/security/access.conf` でシステム承認ポリシーを検索するように設定します。

```
[root@server ~]# authconfig --enable locauthorize --enable pamaccess --update
```

2.3.2. システムパスワード

2.3.2.1. パスワードの暗号化

パスワードはデフォルトではプレーンテキスト形式で保存されるので、ハッキングや未承認のアクセス、改ざんなどに対して脆弱になります。保存済みのパスワードは、暗号化アルゴリズムを使うことで暗号化することができます。

IdM でサポートされる主な暗号化アルゴリズムは、以下のとおりです。

- ▶ SHA-256 (sha256)
- ▶ SHA-512 (sha512)

後方互換性を確保するために、以下のアルゴリズムもサポートされています。

- ▶ DES (descrypt)
- ▶ BigCrypt (bigcrypt)、これは DES 暗号化にソルトを使用したものです。
- ▶ MD5 (md5)



警告

後方互換性が不要な場合は、SHA-256 または SHA-512 のみを使用してください。これらの方がよりセキュアになります。

2.3.2.1.1. UI を使ってパスワード暗号化を設定する

ローカル認証のオプション タブでは、ローカルパスワードをシステムに保存する方法を設定します。パスワードハッシュアルゴリズム オプションでは、ローカルに保存するパスワードの暗号化に使用するハッシュアルゴリズムを設定します。

1. [「authconfig UI を起動する」](#)にあるように、`authconfig` UI を開きます。
2. 高度なオプション タブを開きます。
3. パスワードハッシュアルゴリズム のドロップダウンメニューから使用するアルゴリズムを選択します。

Authentication Configuration

Identity & Authentication **Advanced Options** Password Options

Local Authentication Options

Enable fingerprint reader support

Enable local access control

Tip: This is managed via /etc/security/access.conf.

Password Hashing Algorithm: SHA512

Other Authentication Options

Create home directories on the first login

Smart Card Authentication Options

Enable smart card support

Tip: Smart cards support logging into both local and centrally managed accounts.

Card Removal Action: Ignore

Require smart card for login

Revert Cancel Apply

4. **適用** ボタンをクリックします。

2.3.2.1.2. コマンドラインでパスワード暗号化を設定する

ユーザーのパスワードの暗号化に使用されているハッシュアルゴリズムを設定もしくは変更するには、`--passalgo` オプションとハッシュの短い名前を使用します。たとえば、SHA-512 アルゴリズムには以下を使用します。

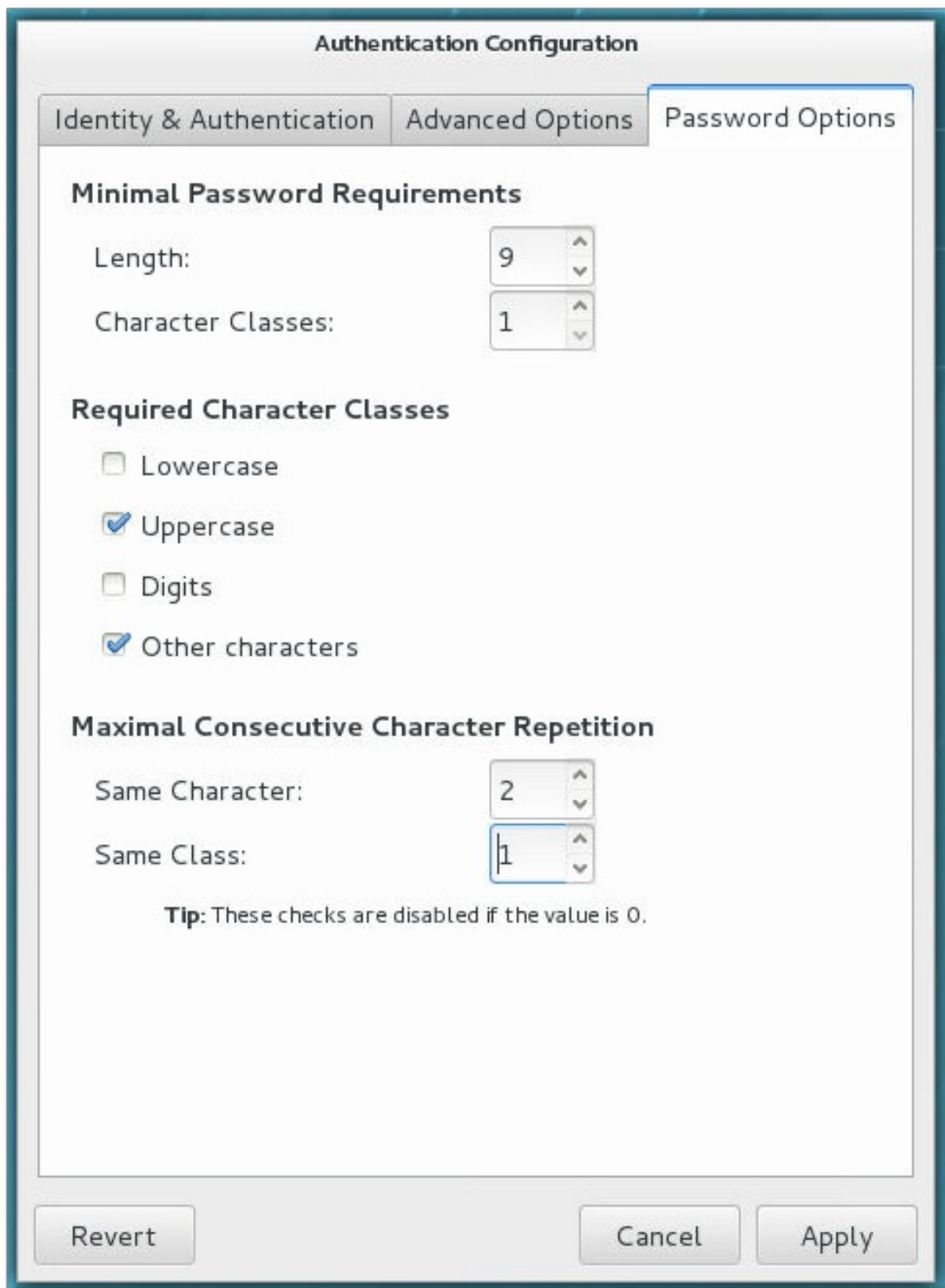
```
[root@server ~]# authconfig --passalgo=sha512 --update
```

2.3.2.2. パスワードの複雑性

パスワードの複雑性は、ローカルユーザーアカウントの設定に使用可能なパスワードの強度を設定します。複雑性は、長さや文字クラスの多様性で決められます。パスワードの複雑性を決定するポリシーは、2つの要素から構成されます。ひとつはパスワードで使用可能な文字タイプの特定です (大文字と小文字や特別な文字など)。もうひとつはパスワード内でこれらの文字がどのように使用できるかです (長さおよび連続文字数)。

2.3.2.2.1. UI を使ってパスワードの複雑性を設定する

1. [「authconfig UI を起動する」](#)にあるように、`authconfig` UI を開きます。
2. `パスワードオプション` タブを開きます。



- 以下の **最小パスワード要件** を設定します。
 - ※ パスワードの最低限の長さ
 - ※ パスワードで使用する文字クラスの最低数
- パスワードに **必要な文字クラス** を有効にします。たとえば、パスワードには大文字を使用することができませんが、**大文字** チェックボックスを選択すると、すべてのパスワード必ず大文字を使用する必要があります。

5. 同じ文字もしくは同じクラスを連続させることができる最大数を設定します (これをゼロに設定すると、連続する数は制限されません)。

同じ文字 フィールドでは、単一文字の連続可能な最大数を設定します。たとえばこれを 2 に設定すると、`ssecret` は許可されますが `sssecret` は許可されません。

同様に **同じクラス** では、(大文字、特別文字といった) 文字クラスからの文字の連続可能な最大数を設定します。たとえばこれを 3 に設定すると、`secret!!` は許可されますが、`secret!!@` や `secret1234` は許可されません。

6. **適用** ボタンをクリックします。

2.3.2.2.2. コマンドラインでパスワードの複雑性を設定する

パスワードの複雑性をコマンドラインで定義する際には、設定要件は 2 つの部分で構成されます。ひとつ目は、パスワードの構成方法です。つまり、長さや連続文字数、必要な文字クラスの数などです。

- ※ パスワードの最低限の長さ (`--passminlen`)。
- ※ パスワードで使用する文字クラスの最低数 (`--passminclass`)。
- ※ 同じ文字を連続させることができる最大数 (`--passmaxrepeat`)。これをゼロに設定すると、連続する数は制限されません。
- ※ 同じクラスの文字 (数字など) を連続させることができる最大数 (`--passmaxclassrepeat`)。これをゼロに設定すると、連続する数は制限されません。

2 つ目の部分では、パスワードに使用できる文字クラスを定義します。すべてのクラスが暗示的に使用可能ですが、`--enablereqType` オプションを使うと、特定のクラスが必須となり、そのクラスがないパスワードは許可されません。(反対に、クラスを明示的に拒否する設定も可能です。)

- ※ 大文字 (`--enablerequpper`)
- ※ 小文字 (`--enablereqlower`)
- ※ 数字 (`--enablereqdigit`)
- ※ 特殊文字 (`--enablereqother`)

たとえば、以下の設定では最小文字数を 9 文字とし、文字およびクラスの最大連続数は 2 となり、大文字と特殊文字の両方を必須とします。

```
[root@server ~]# authconfig --passminlen=9 --passminclass=3 --passmaxrepeat=2 --passmaxclassrepeat=2 --enablerequpper --enablereqother --update
```

2.3.3. Kerberos (LDAP または NIS 認証)

LDAP と NIS の認証ストアは両方とも Kerberos の認証メソッドをサポートします。Kerberos を使用すると以下の利点があります。

- ※ 標準ポート上での接続を許可しながら、通信にセキュリティ層を使用します。
- ※ オフラインログインを可能にする SSSD を使用した認証情報キャッシングを自動的に使用します。



注記

Kerberos 認証の使用には、`krb5-libs` と `krb5-workstation` の両パッケージが必要になります。

2.3.3.1. UI で Kerberos 認証を設定する

認証の方法 のドロップダウンメニューから **Kerberos パスワード** を選ぶと、自動的に Kerberos レルムへの接続に必要なフィールドが開きます。

The image shows a screenshot of the 'Authentication Configuration' dialog box. It has three tabs: 'Identity & Authentication' (selected), 'Advanced Options', and 'Password Options'. Under the 'Identity & Authentication' tab, there is a section titled 'User Account Configuration'. This section includes a dropdown menu for 'User Account Database' set to 'LDAP', a text field for 'LDAP Search Base DN' containing 'ou=people,dc=example,dc=co', and another text field for 'LDAP Server' containing 'ldap://idm.example.com/'. Below these fields is a checked checkbox for 'Use TLS to encrypt connections' and a button labeled 'Download CA Certificate...'. A red rectangular box highlights a sub-section of the dialog, also titled 'Authentication Configuration'. This sub-section contains a dropdown menu for 'Authentication Method' set to 'Kerberos password', a text field for 'Realm' containing 'EXAMPLE', and two empty text fields for 'KDCs' and 'Admin Servers'. At the bottom of this sub-section are two checkboxes: 'Use DNS to resolve hosts to realms' (unchecked) and 'Use DNS to locate KDCs for realms' (checked). At the very bottom of the main dialog box are three buttons: 'Revert', 'Cancel', and 'Apply'.

Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: ou=people,dc=example,dc=co

LDAP Server: ldap://idm.example.com/

Use TLS to encrypt connections

Download CA Certificate...

Authentication Configuration

Authentication Method: Kerberos password

Realm: EXAMPLE

KDCs:

Admin Servers:

Use DNS to resolve hosts to realms

Use DNS to locate KDCs for realms

Revert | Cancel | Apply

図2.3 Kerberos フィールド

- ※ **レルム** には、Kerberos サーバー用のレルム名を入力します。レルムとは、1 つまたはそれ以上の **キー配布センター (KDC)** と多数のクライアントで構成される、Kerberos を使用するネットワークのことです。
- ※ **KDCs** には、Kerberos チケットを発行するサーバーのコンマ区切りの一覧を入力します。
- ※ **管理サーバー** には、レルム内で **kadmin** プロセスを実行している管理サーバーの一覧を入力します。
- ※ オプションとして、DNS を使用してサーバーのホスト名を解決し、レルム内の新たな KDC を見つけることができます。

2.3.3.2. コマンドラインを使って Kerberos 認証を設定する

LDAP と NIS は両方とも、それらのネイティブな認証メカニズムの代わりに Kerberos 認証を使用することができます。Kerberos 認証を使用する場合、最低でもレルム、KDC、および管理サーバーを指定する必要があります。DNS を使用してクライアント名を解決、追加の管理サーバーを見つけるオプションもあります。

```
[root@server ~]# authconfig NIS or LDAP options --enablekrb5 --krb5realm EXAMPLE --krb5kdc kdc.example.com:88,server.example.com:88 --krb5adminserver server.example.com:749 --enablekrb5kdcdns --enablekrb5realmdns --update
```

2.3.4. スマートカード

適切なスマートカードリーダーが利用可能な場合は、システムは認証のために他のユーザー認証情報の代わりにスマートカード (トークン とも呼ぶ) を受け付けることができます。

スマートカードサポートを有効にする オプションを選択すると、スマートカードの設定動作を追加で制御できるようになります。

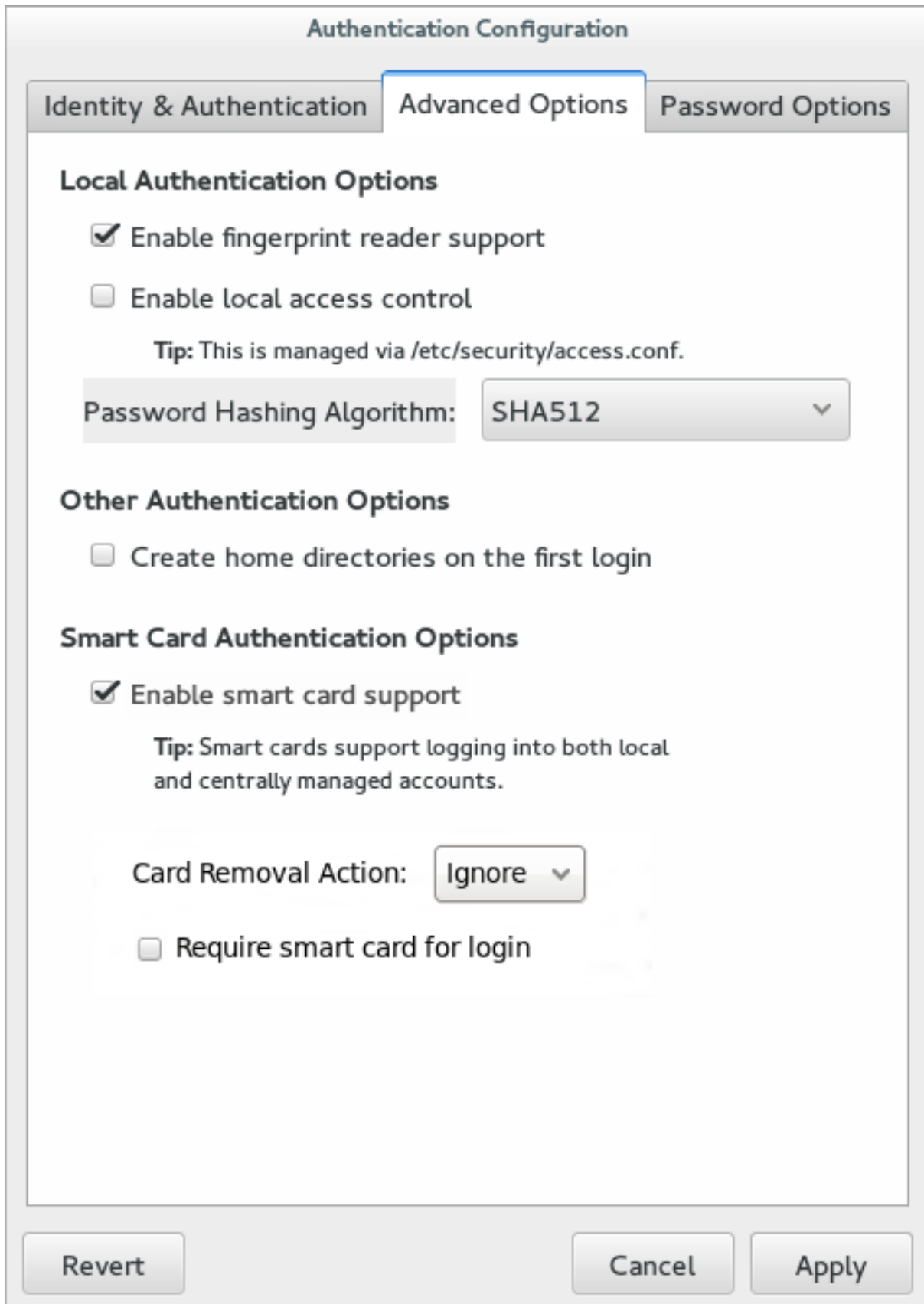


図2.4 スマートカード認証のオプション

Red Hat Enterprise Linux サーバーおよびワークステーションにおけるスマートカードでのログインはデフォルトでは有効になっておらず、システム設定で有効にする必要があります。


注記

Red Hat Enterprise Linux へのログイン時にシングルサインオンを使用する場合は、以下のパッケージが必要になります。

- ※ nss-tools
- ※ nss-pam-ldapd
- ※ esc
- ※ pam_pkcs11
- ※ pam_krb5
- ※ coolkey
- ※ ccid
- ※ gdm
- ※ authconfig
- ※ authconfig-gtk
- ※ krb5-libs
- ※ krb5-workstation
- ※ krb5-auth-dialog
- ※ krb5-pkinit-openssl
- ※ pcsc-lite
- ※ pcsc-lite-libs

2.3.4.1. UI でスマートカード認証を有効にする

1. root でシステムにログインします。
2. ネットワーク用の root CA 証明書をベース 64 形式でダウンロードし、サーバーにインストールします。`certutil` コマンドを使うと、証明書は適切なシステムデータベースにインストールされます。例を示します。

```
[root@server ~]# certutil -A -d /etc/pki/nssdb -n "root CA cert" -t "CT,C,C" -i /tmp/ca_cert.crt
```


注記

このプロセスの後半で `authconfig` UI にインポートされた証明書が表示されなくても、心配は要りません。UI では証明書は表示されません。認証時に `/etc/pki/nssdb/` ディレクトリーから取得されます。

3. トップメニューで **アプリケーション** から **諸ツール** を選択し、**認証** をクリックします。
4. **高度なオプション** タブを開きます。
5. **スマートカードサポートを有効にする** チェックボックスをクリックします。
6. スマートカードでは 2 つの動作が設定可能です。
 - ※ **カード削除のアクション** では、アクティブセッション中にカードが取り出された時のシステムの対応方法を設定します。**無視する** オプションの場合、カードが取り出されてもシステムは通常の機能を続けます。**ロックする** の場合は直ちに画面をロックします。

- ※ **スマートカードログインを要求** は、スマートカードがログインで必要かどうかを設定します。このオプションが選択されると、他の認証メソッドはすべてブロックされます。



警告

スマートカードを使用してシステムに正常にログインするまで は、このオプションは選択しないでください。

- デフォルトでは、証明書が失効となったかどうかを確認するメカニズム (オンライン証明書ステータスプロトコル、OCSP の反応) は、無効になっています。有効期限が切れる前に証明書が失効したかどうかを検証するには、`cert_policy` ディレクティブに `ocsp_on` オプションを追加して OCSP のチェックを有効にします。

- `pam_pkcs11.conf` ファイルを開きます。

```
vim /etc/pam_pkcs11/pam_pkcs11.conf
```

- `cert_policy` 行すべてに `ocsp_on` オプションを追加します。

```
cert_policy = ca, ocsp_on, signature;
```



注記

このファイルを解析する方法が理由で、`cert_policy` とイコール記号の間には空白が **必要** になります。これがないと、パラメーターの解析が失敗します。

- (個人証明書とキーによる設定で) スマートカードが登録されていない場合、スマートカードを登録します。
- スマートカードが CAC カードの場合、CAC ユーザーのホームディレクトリーに `.k5login` ファイルを作成します。`.k5login` ファイルは、CAC カード上に Microsoft Principal Name を記載するために必要となります。
- 以下の行を `/etc/pam.d/smartcard-auth` と `/etc/pam.d/system-auth` の各ファイルに追加します。

```
auth optional pam_krb5.so use_first_pass no_subsequent_prompt
preauth_options=X509_user_identity=PKCS11:/usr/lib64/pkcs11/libcoolkeypk11.so
```

- `/etc/krb5.conf` ファイルを設定します。この設定は、CAC カードか Gemalto 64K カードを使っているかによって異なります。

- ※ CAC カードの場合、CAC カード使用に関連するすべての root 証明書を `pkinit_anchors` で指定します。以下の `/etc/krb5.conf` ファイルで CAC カードを設定する例では、`EXAMPLE.COM` が CAC カードのレルム名になり、`kdc.server.hostname.com` が KDC サーバーのホスト名になります。

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
dns_lookup_realm = false
```

```

dns_lookup_kdc = false
ticket_lifetime = 1h
renew_lifetime = 6h
forwardable = true

default_realm = EXAMPLE.COM
[realms]
EXAMPLE.COM = {
    kdc = kdc.server.hostname.com
    admin_server = kdc.server.hostname.com
    pkinit_anchors = FILE:/etc/pki/nssdb/ca_cert.pem
    pkinit_anchors = FILE:/etc/pki/nssdb/CAC_CA_cert.pem
    pkinit_anchors = FILE:/etc/pki/nssdb/CAC_CA_email_cert.pem
    pkinit_anchors = FILE:/etc/pki/nssdb/CAC_root_ca_cert.pem
    pkinit_cert_match = CAC card specific information
}

[domain_realm]
EXAMPLE.COM = EXAMPLE.COM
.EXAMPLE.COM = EXAMPLE.COM

.kdc.server.hostname.com = EXAMPLE.COM
kdc.server.hostname.com = EXAMPLE.COM

[appdefaults]
pam = {
    debug = true
    ticket_lifetime = 1h
    renew_lifetime = 3h
    forwardable = true
    krb4_convert = false
    mappings = username on the CAC card      Principal name on the card
}

```

- ※ Gemalto 64K カードを設定する以下の `/etc/krb5.conf` ファイルの場合、`EXAMPLE.COM` は KDC サーバー上で作成されたレルムになり、`kdc-ca.pem` は CA 証明書、`kdc.server.hostname.com` が KDC サーバーのホスト名になります。

```

[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadm ind.log

[libdefaults]
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 15m
renew_lifetime = 6h
forwardable = true

default_realm = EXAMPLE.COM
[realms]
EXAMPLE.COM = {
    kdc = kdc.server.hostname.com
    admin_server = kdc.server.hostname.com
    pkinit_anchors = FILE:/etc/pki/nssdb/kdc-ca.pem
}

```

```

pkinit_cert_match = <KU>digitalSignature
pkinit_kdc_hostname = kdc.server.hostname.com
}

[domain_realm]
EXAMPLE.COM = EXAMPLE.COM
.EXAMPLE.COM = EXAMPLE.COM

.kdc.server.hostname.com = EXAMPLE.COM
kdc.server.hostname.com = EXAMPLE.COM

[appdefaults]
pam = {
    debug = true
    ticket_lifetime = 1h
    renew_lifetime = 3h
    forwardable = true
    krb4_convert = false
}

```

注記

スマートカードが挿入されると、**pklogin_finder** ユーティリティがデバッグモードで実行されている場合、まずログイン ID をカード上の証明書にマッピングし、証明書の有効性についての情報の出力を試みます。

```
pklogin_finder debug
```

このコマンドは、スマートカードを使ってシステムにログインする際の問題診断で役に立ちます。

2.3.4.2. コマンドラインでスマートカード認証を設定する

システムでスマートカードを使用する際に必要となるのは **--enablesmartcard** オプションの設定のみです。

```
[root@server ~]# authconfig --enablesmartcard --update
```

スマートカードには、デフォルトのスマートカードモジュールの変更、スマートカードが取り出された時のシステムの動作のセッティング、ログイン時にスマートカードを要求するなど、他の設定オプションがあります。

以下のコマンドの値 **0** は、スマートカードが取り出された場合、システムがユーザーをロックアウトするように指示します。設定が **1** の場合は、スマートカードが取り出されても、これを無視します。

```
[root@server ~]# authconfig --enablesmartcard --smartcardaction=0 --update
```

スマートカード認証が正常に設定されテストが終わると、単純なパスワードベースの認証ではなく、スマートカード認証をユーザーに要求するようにシステムを設定することができます。

```
[root@server ~]# authconfig --enablerequiresmartcard --update
```

**警告**

スマートカードを使用して正常にシステム認証が終了するまで、`--enablerequiresmartcard` オプションは使用しないでください。終了前に使用すると、ユーザーがシステムにログインできなくなる可能性があります。

2.3.5. ワンタイムパスワード

ワンタイムパスワード (OTP) は、1 回の認証セッションのみで有効なパスワードです。これは一度使用すると、無効になります。長期間に渡り変わることがない従来の静的なパスワードと違って、OTP は常に変更されます。OTP は 2 要素認証の一部として使用されます。まず、ユーザーは従来の静的パスワードで認証を行い、次に認識された認証トークンが発行する OTP が求められます。

静的パスワードと OTP を組み合わせた認証は、静的パスワードのみの認証よりも安全だと考えられています。これは OTP は認証の成功に 1 度しか使えないため、ログイン中に侵入者が OTP を傍受したとしても、その OTP はその時点で既に無効になっているためです。

IdM は、2 つの標準 OTP メカニズムをサポートしています。IdM のネイティブ OTP サポート内で使用されるトークンはすべて、以下のいずれかを実装している必要があります。

- ※ HMAC ベースのワンタイムパスワード (HOTP) アルゴリズムは、カウンターに基づいています。HMAC は、Hashed Message Authentication Code (ハッシュメッセージ認証コード) を表しています。
- ※ 時間ベースのワンタイムパスワード (TOTP) アルゴリズムは、時間ベースの移動要素をサポートする HOTP の拡張機能です。

**警告**

以下のセキュリティとその他の制限は現在、IdM ネイティブの OTP サポートに関連しています。

- ※ セキュリティ制限で最も重要性が高いものは、システム全体に渡る潜在的なりレー攻撃への脆弱性です。レプリケーションは非同期的なので、OTP コードはレプリケーション期間に再利用することが可能です。ユーザーは同時に 2 つのサーバーにログオンすることが可能です。ただし、この脆弱性は包括的な暗号化により、通常は悪用が困難なものです。
- ※ IdM OTP サポートは、GNOME キーリングサービスと統合されていません。
- ※ オフラインの OTP ログインはサポートされていません。セッションをオフラインでロック解除することもできません。システムが認証サーバーに連絡できないと、ユーザーは OTP を使ってログインしたりコンピューター画面をロック解除したりすることができません。
- ※ OTP 認証をサポートしていないクライアント経由では、ticket-granting ticket (TGT) を取得することができません。これは、`mod_auth_kerb` モジュールまたは Generic Security Services API (GSSAPI) を使用した認証などのユースケースに影響する可能性があります。

2.3.5.1. IdM で OTP 認証を有効にする

OTP サポートの有効化、無効化ができるのは、管理者のみです。ユーザーには許可されていません。管理者は OTP サポートを特定のユーザーのみに有効とするか、すべてのユーザーに有効とすることができます。

管理者は、どのユーザーにどの認証方法を利用可能にするかを管理することができます。許可される認証方法をすべてのユーザーにグローバルで設定するか、ユーザーごとに個別に設定することができます。IdM では、以下の認証方法が利用できます。

- ※ パスワード認証

- ※ RADIUS プロキシサーバー認証
- ※ 2 要素認証 (パスワード + OTP)

複数のオプションを同時に設定することも可能です。その場合、どれか 1 つが成功すれば認証されます。

ユーザーは IdM に対して Kerberos と LDAP という 2 つのプロトコルで認証されます。パスワードベースの単一要素認証の場合、ユーザーは 2 つのプロトコルのいずれかで同じパスワードを使って認証されます。OTP ベースの 2 要素認証の場合、どちらのプロトコルを使用するかによって若干の違いがあります。

パスワードと 2 要素認証のタイプを同時に選んだ場合でも、Kerberos はパスワードと OTP の両方での認証を強制します。LDAP の場合は、いずれかの認証タイプでの認証が許可されます。

注記

ユーザーによる 2 要素認証を義務付けるには、IdM と統合されるアプリケーションから Kerberos を使用します。義務付けない場合は、LDAP を使用してユーザーがパスワードのみで認証できるようにします。

RADIUS 認証を別の認証タイプと合わせて選択した場合、Kerberos は常に RADIUS を使用しますが LDAP は RADIUS を使用しません。LDAP が認識するのは、パスワードと 2 要素認証のオプションのみです。

注記

外部の 2 要素認証プロバイダーを使用する場合は、使用しているアプリケーションから Kerberos を使用します。ユーザーがパスワードのみで認証するようにするには、LDAP を使用します。アプリケーションが Apache モジュールと SSSD を活用するようにすることが推奨されます。そうすることで、Kerberos か LDAP のいずれかを設定することができます。

2.3.5.1.1. 認証方法の定義

IdM ウェブ UI からグローバルで認証方法を設定するには、**IdM server** メインタブ下にある **Configuration** のサブタブから **Default user authentication types** オプションを使用します。

IdM ウェブ UI からユーザーごとの認証方法を設定するには、**Identity** メインタブ下にある **Users** のサブタブから該当するユーザーの詳細ページで **User authentication types** オプションを使用します。

コマンドラインでグローバルの認証方法を設定するには、**ipa config-mod** コマンドを **--user-auth-type** オプションと実行し、認証方法を定義します。このオプションで認識される属性は、**password**、**radius**、および **otp** になります。たとえば、認識方法を 2 要素認証に設定するには、以下のコマンドを実行します。

```
[root@server ~]# ipa config-mod --user-auth-type=otp
```

コマンドラインでユーザーごとの認証方法を設定するには、**ipa config-mod** コマンドを **--user-auth-type** オプションと実行し、認証方法を定義します。たとえば、**employee** ユーザーがパスワードで認識されるように定義するには、以下のコマンドを実行します。

```
[root@server ~]# ipa user-mod employee --user-auth-type=password
```

複数の認証方法を設定するには、複数の **--user-auth-type** オプションを **ipa config-mod** または **ipa user-mod** と実行します。

 注記

ユーザーの認証方法を変更できるのは、管理者のみです。

2.3.5.2. ハードウェアおよびソフトウェアトークン

OTP の発行には、ハードウェアトークンとソフトウェアトークンの両方が使用できます。ハードウェアトークンは、専用の物理デバイスに保存されます。ソフトウェアトークンは、通常、スマートフォンやタブレットなどのユーザーのモバイルデバイスに保存されます。

ハードウェアトークンは通常管理者が管理しますが、いつもそうとは限りません。たとえば、Yubikey トークンのようなハードウェアトークンは、ユーザーが管理しています。管理者はハードウェアトークンをバルク購入して、ユーザーに配布することができます。

同様に、ソフトウェアトークンは通常ユーザーが管理していますが、常にそうとは限りません。たとえば、従業員にモバイルデバイスを割り当てている企業では、管理者が管理するソフトウェアトークンを使用できます。

2.3.5.3. ユーザー管理のトークンと管理者管理のトークン

ユーザーはユーザー管理のトークンの詳細に関して完全な制御が可能で、トークンの作成、変更、削除が許可されます。ユーザー自身によるトークン管理を許可する際には、ユーザーまたはグローバルですべてのユーザーに対してトークンサポートが有効となっていることを確認してください。その後、ユーザーがトークンの設定をするようにします。

ユーザーが管理者管理のトークンについて許可されるのは、読み取り専用のアクセスです。トークンの管理または修正は許可されず、元々設定を求められることもありません。管理者としてユーザーにトークンを割り当てる際には、ユーザーまたはグローバルですべてのユーザーに対してトークンサポートが有効となっていることを確認してください。その後、ユーザーのアカウントにトークンを追加します。

 注記

ユーザーは常に、アクティブなトークンを少なくとも 1 つ持つことが求められます。その時点においてアクティブなトークンが 1 つしかない場合、そのトークンを削除したり非アクティブにすることは許可されません。同様に、管理者も、ユーザーの最後のアクティブなトークンを削除したり非アクティブにすることはできません。

2.3.5.3.1. ユーザー管理のソフトウェアトークンを追加する

ユーザー管理のソフトウェアトークンを追加するには、標準パスワードでユーザーとしてログインし、以下の手順にしたがいます。

1. モバイルデバイスに Android 用の **FreeOTP Authenticator** アプリケーションがインストールされていることを確認します。 [1]
2. IdM ウェブ UI またはコマンドラインでソフトウェアトークンを作成します。
 - ※ ウェブ UI でトークンを作成するには、**OTP Tokens** タブをクリックし、OTP トークンのリスト上部にある **Add** をクリックします。管理者としてログインしている場合は、**Authentication** のメインタブから **OTP Tokens tab** にアクセスできます。

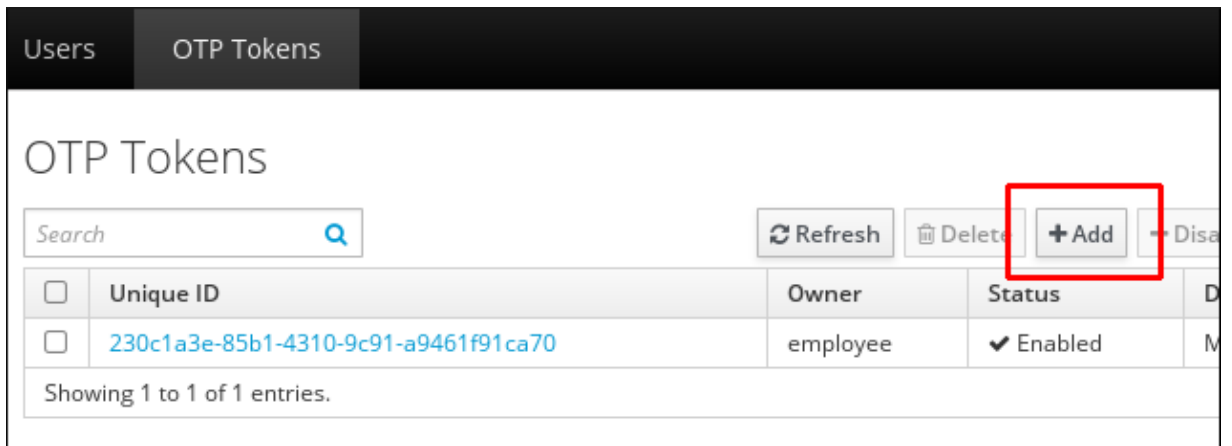


図2.5 ユーザー用の OTP トークンを追加する

表示されるフォームに記入して、フォーム下部にある **Add** をクリックします。

※ コマンドラインからトークンを作成するには、`ipa otptoken-add` を実行します。

3. ウェブ UI またはコマンドラインに QR コードが表示されます。QR コードを **FreeOTP Authenticator** でスキャンします。これでスマートフォンまたはタブレットにトークンが提供されます。

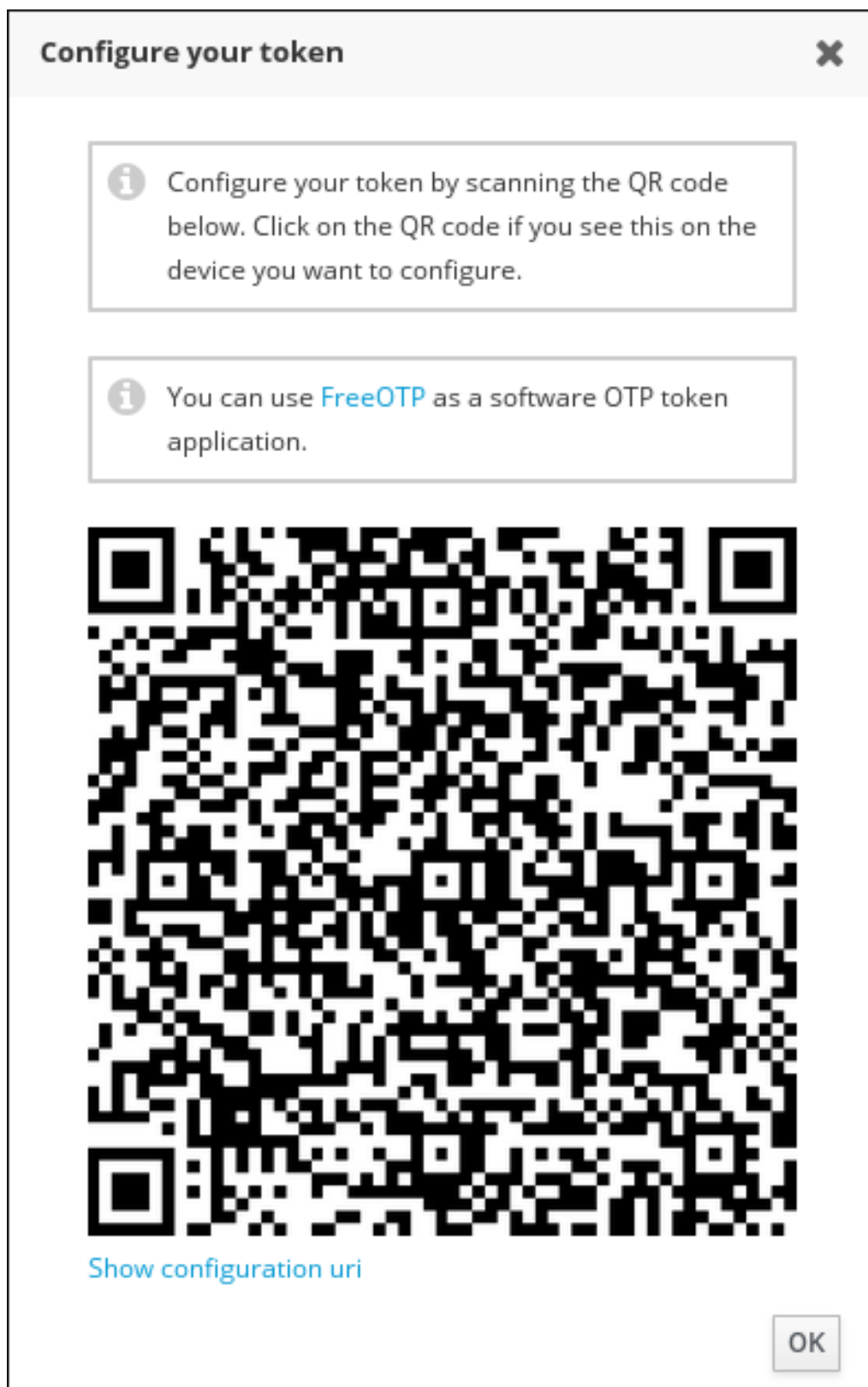


図2.6 ウェブ UI の QR コード

2.3.5.3.2. ユーザー管理の YubiKey ハードウェアトークンを追加する

ブラウザの制限により、YubiKey のようなプログラム可能なハードウェアトークンは、コマンドライン上でのみ追加できます。ユーザーが所有するトークンとして YubiKey ハードウェアトークンを追加するには、ユーザーの標準のパスワードでログインし、以下の手順にしたがいます。

1. YubiKey トークンを挿入します。

2. `ipa otptoken-add-yubikey` コマンドを実行します。YubiKey に未使用のスロットがあると、コマンドがこれを自動で見つけます。未使用のスロットがない場合は、コマンドに `--slot` オプションを追加して、上書きするスロットを選択します。例を示します。

```
[user@server ~]$ ipa otptoken-add-yubikey --slot=2
```

2.3.5.3.3. 管理者としてユーザー用のトークンを追加する

管理者はユーザーの代わりにトークンを作成することができます。管理者としてソフトウェアトークンを追加するには、以下の手順にしたがいます。

1. 管理者としてログインしていることを確認します。
2. [「ユーザー管理のソフトウェアトークンを追加する」](#)にある手順にしたがい、トークンを所有するユーザーを指定します。
 - ※ ウェブ UI からトークンを追加して所有者を指定するには、**Owner** フィールドを使用して、トークンを追加するフォームで直接ユーザーを選択します。
 - ※ コマンドラインからトークンを追加して所有者を指定するには、`ipa otptoken-add` コマンドで `--owner` オプションを使用します。例を示します。

```
[root@server ~]# ipa otptoken-add --owner=employee
```

管理者として、YubiKey のようなプログラム可能なハードウェアトークンを追加するには、以下の手順にしたがいます。

1. 管理者としてログインしていることを確認します。
2. [「ユーザー管理の YubiKey ハードウェアトークンを追加する」](#)にある手順にしたがい、`ipa otptoken-add-yubikey` コマンドに `--owner` オプションを追加してトークンを所有するユーザーを指定します。例を示します。

```
[root@server ~]# ipa otptoken-add-yubikey --owner=employee
```

2.3.5.4. ノートパソコンユーザーの制限

IdM 側で OTP を使用して認証するように設定され、`ipa-client-install` コマンド実行後にデフォルト設定に SSSD があるノートパソコンのユーザーは、2 要素認証しか使用できません。ただし、IdM の現行バージョンは、オフラインでの 2 要素認証をサポートしていません。



重要

デフォルト設定のノートパソコンのユーザーの場合、OTP 認証を使用しないでください。ノートパソコンのユーザーの場合は OTP を有効にしないか、LDAP に切り替えることが推奨されます。LDAP に切り替えると、シングルサインオンが無効になることに注意してください。

2.3.5.5. 商用 OTP ソリューションから移行する

大規模な商用 OTP ソリューションのデプロイメントを統合された OTP サポートのある IdM に移行するために、IdM では OTP 確認をサードパーティの RADIUS サーバーにアンロードするための方法をユーザーに提供しています。管理者は RADIUS プロキシのセットを作成します。各プロキシには、複数の個別 RADIUS サーバーが含まれます。管理者は、これらプロキシセットのひとつをユーザーに割り当てます。ユーザーに RADIUS プロキシのセットが割り当てられている限り、IdM は他のすべての認証メカニズムを迂回します。



注記

IdM は、サードパーティシステム内におけるトークン管理やトークンの同期サポートを提供していません。

OTP 確認用の RADIUS サーバーを設定し、ユーザーをプロキシサーバーに追加するには、以下の手順にしたがいます。

1. `radius` ユーザー認識方法が有効になっていることを確認します。[「認証方法の定義」](#)を参照してください。
2. `ipa radiusproxy-add testproxy` を実行し、その後の指示にしたがって RADIUS プロキシを追加します。
3. `ipa user-mod radiususer --radius=testproxy` を実行してユーザーをこのプロキシに割り当てます。
4. 必要に応じて、`ipa user-mod radiususer --radius-username=myradiususer` を実行し、RADIUS に送信されるユーザー名を設定します。
5. これでユーザー OTP 認証が RADIUS プロキシサーバーで処理されるようになります。

ユーザーが IdM ネイティブの OTP システムに移行する準備ができたなら、そのユーザーへの RADIUS プロキシの割り当てを削除します。

2.3.5.6. トークンの同期

トークンが非同期状態になると、正常な認証に使用できなくなります。トークンを再同期するには、IdM ウェブ UI のログインページで **Sync OTP Token** ボタンをクリックするか、コマンドラインで `ipa otptoken-sync` を実行します。パスワードと 2 つのトークンコードを続けて入力することが求められます。



注記

トークンのタイプや、ユーザーがトークン設定を修正するパーミッションがあるかどうかに関係なく、ユーザーはトークンの再同期ができます。

2.3.6. 指紋

2.3.6.1. UI で指紋認証を使用する

適切なハードウェアが利用可能である場合は、他の認証情報に加えて **指紋リーダーサポートを有効にする** オプションでローカルユーザーの認証に指紋スキャンを使用することができます。

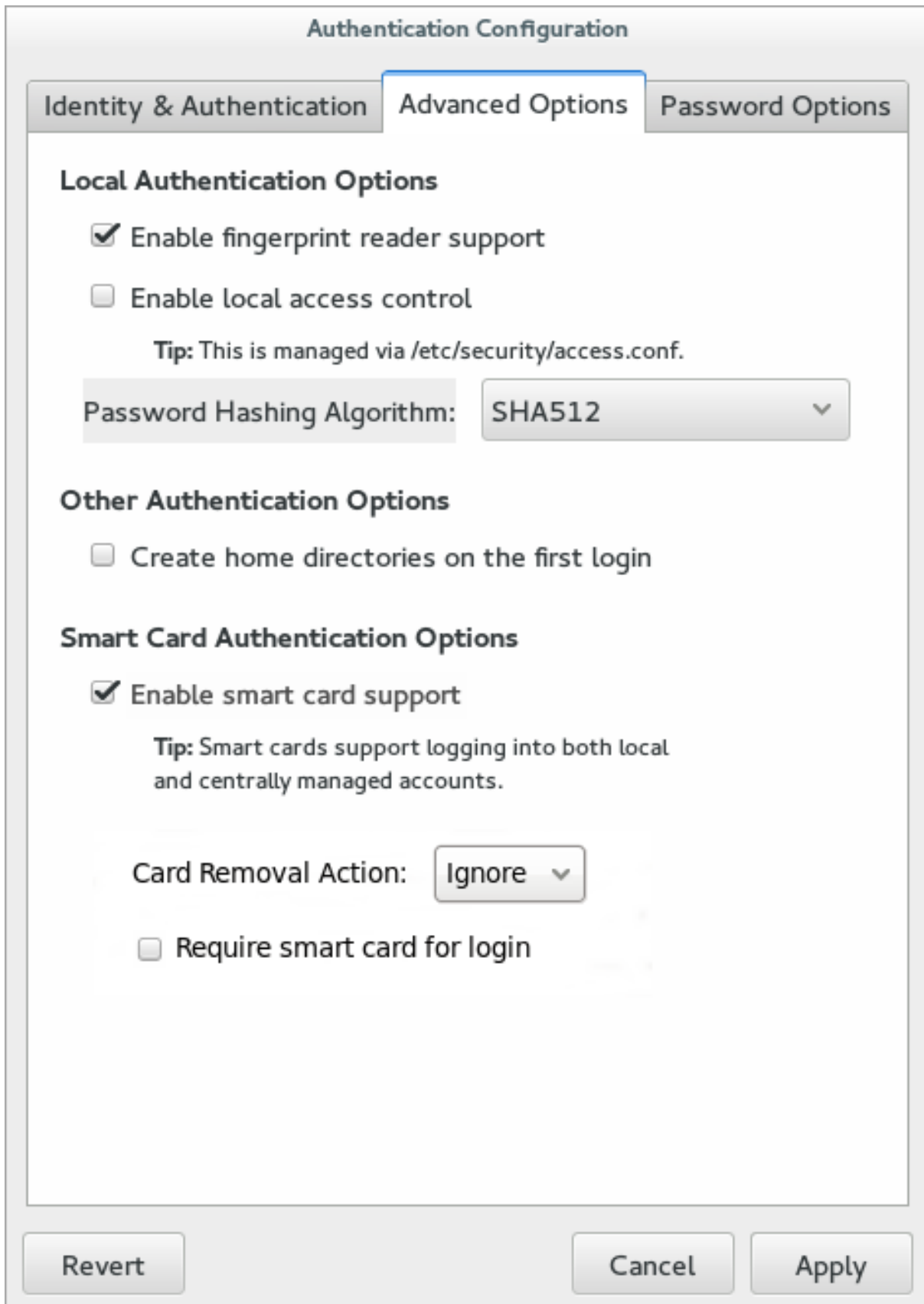


図2.7 指紋オプション

2.3.6.2. コマンドラインで指紋認証を設定する

指紋リーダーのサポートを有効にするには、オプションが1つあります。このオプションは単独でも、またはLDAPユーザストアのような他の `authconfig` 設定と合わせて使用することもできます。

```
[root@server ~]# authconfig --enablefingerprint --update
```

2.4. キックスタートと設定ファイルの管理

`--update` オプションは、変更されたすべての設定ファイルを更新します。以下の代替オプションでは、動作が多少異なります。

- ※ `--kickstart` は、更新した設定をキックスタートファイルに書き込みます。
- ※ `--test` は、変更を含むの設定すべてを標準出力にプリントしますが、設定ファイルを編集することはありません。

また、`authconfig` を使用してバックアップを作成したり、以前の設定に復元することもできます。すべてのアーカイブは `/var/lib/authconfig/` ディレクトリー内の一意のサブディレクトリーに保存されます。たとえば、`--savebackup` オプションを使って `2011-07-01` といったバックアップディレクトリーを作成します。

```
[root@server ~]# authconfig --savebackup=2011-07-01
```

これにより、`/var/lib/authconfig/backup-2011-07-01` ディレクトリーに認証設定ファイルすべてのバックアップが作成されます。

保存されたバックアップはいずれも、`--restorebackup` オプションで手動保存した設定の名前を入力することで復元されます。

```
[root@server ~]# authconfig --restorebackup=2011-07-01
```

また、`authconfig` は、変更を (`--update` オプションで) 適用する前に、自動的に設定のバックアップを作成します。`--restorelastbackup` オプションを使用すると、バックアップを指定しない場合は最新の自動バックアップから設定を復元できます。

2.5. カスタムホームディレクトリーの有効化

LDAP ユーザーのホームディレクトリーが `/home` 以外の場所にあり、かつユーザーの初回ログイン時にシステムがホームディレクトリーを作成するように設定されている場合は、これらのディレクトリーは間違ったパーミッションで作成されてしまいます。

1. `/home` ディレクトリーから正しい SELinux コンテキストとパーミッションをローカルシステム上で作成されたホームディレクトリーに適用します。例を示します。

```
[root@server ~]# semanage fcontext -a -e /home /home/locale
```

2. `oddjob-mkhomedir` パッケージをシステムにインストールします。

このパッケージは、`pam_oddjob_mkhomedir.so` ライブラリーを提供し、`authconfig` コマンドはこのライブラリーを使用してホームディレクトリーを作成します。デフォルトの `pam_mkhomedir.so` ライブラリーとは異なり、`pam_oddjob_mkhomedir.so` ライブラリーは SELinux ラベルを作成できます。

`authconfig` コマンドは、`pam_oddjob_mkhomedir.so` ライブラリーが利用可能な場合には、自動的にこれを使用します。利用可能でない場合には、デフォルトの `pam_mkhomedir.so` ライブラリーを使用します。

3. `oddjobd` サービスが実行されていることを確認します。
4. `authconfig` コマンドを再度実行して、ホームディレクトリーを有効にします。コマンドラインでは、`--enablemkhomedir` オプションを使うとこれが実行できます。


```
[root@server ~]# authconfig --enablemkhomedir --update
```

UIでは、高度なオプションタブ (利用者の最初のログイン時にホームディレクトリーを作成する) オプションを選択するとユーザーの初回ログイン時に自動的にホームディレクトリーが作成されます。

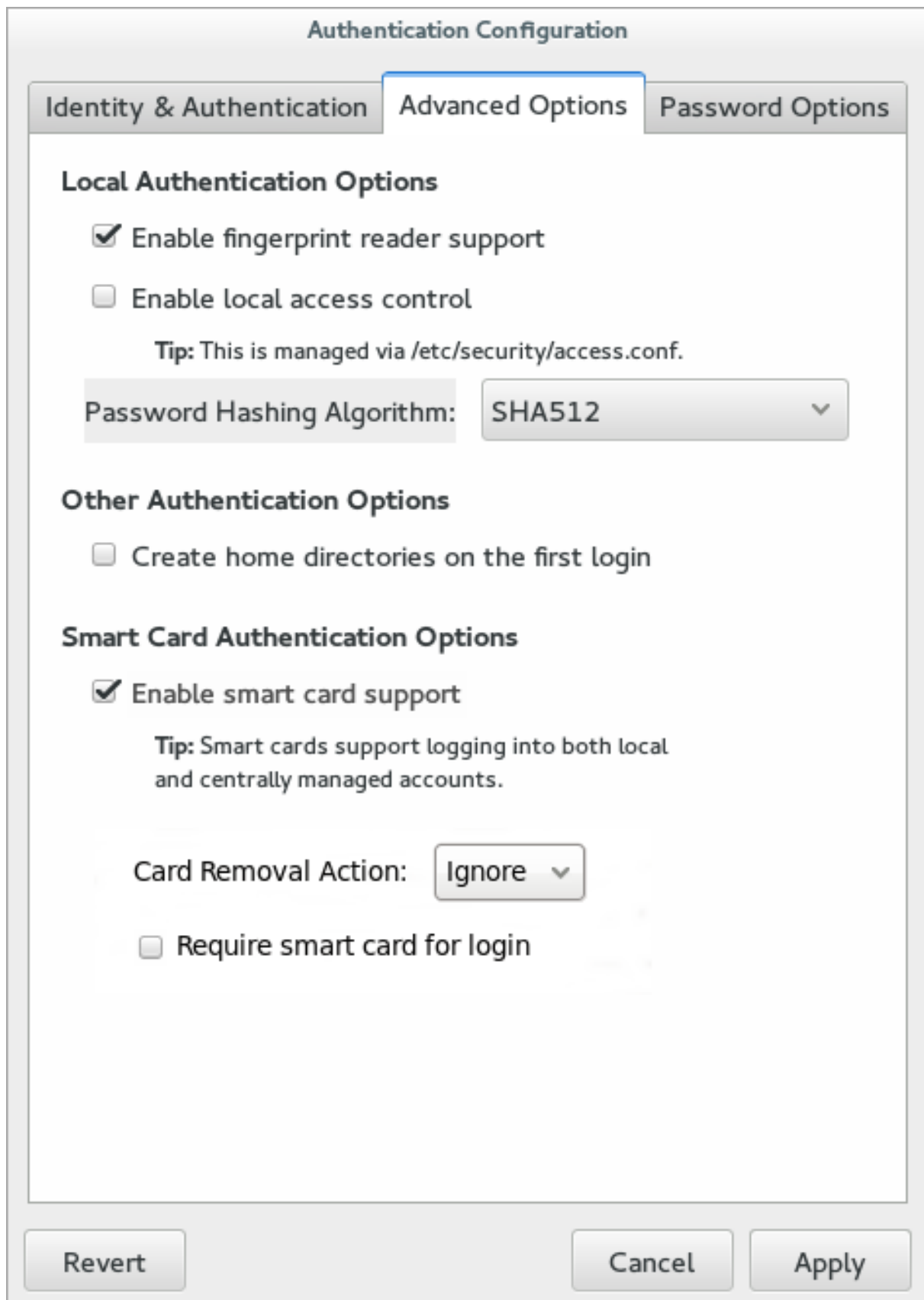


図2.8 ホームディレクトリーオプション

このオプションは、LDAP などアカウントを集中的に管理している場合に便利です。しかし、ユーザーのホームディレクトリー管理に automount のようなシステムを使用している場合は、このオプションを選択しないでください。

ホームディレクトリーの設定変更前にホームディレクトリーが作成された場合は、パーミッションとSELinux コンテキストを訂正します。例を示します。

```
[root@server ~]# semanage fcontext -a -e /home /home/locale
# restorecon -R -v /home/locale
```

2.6. 設定の保存と復元

認証設定を変更すると、問題が発生する場合があります。不適切な変更を行うと、アクセスがあるべきユーザーを除外したり、ID ストアへの接続が失敗したり、システムへの全アクセスがロックアウトされてしまうことすらあります。

管理者は認証設定を変更する前に、全設定ファイルのバックアップを作成することが強く推奨されます。これは `--savebackup` オプションで実行できます。

```
[root@server ~]# authconfig --savebackup=/backups/authconfigbackup20150701
```

`--restorebackup` オプションと適用するバックアップ名を使用することで、以前に保存されたいかなるバージョンの認証設定をも復元することができます。

```
[root@server ~]# authconfig --restorebackup=/backups/authconfigbackup20150701
```

`authconfig` コマンドは、設定が変更されるたびに自動的にバックアップを保存します。`--restorelastbackup` オプションを使用すると、最新の自動バックアップを復元できます。

```
[root@server ~]# authconfig --restorelastbackup
```

[1] `FreeOTP Authenticator` のダウンロードについては、[the FreeOTP source page](#) を参照してください。

パート II. ID および認証ストア

第3章 SSSD による認証情報の使用とキャッシング

System Security Services Daemon (SSSD) は、異なる ID および認証プロバイダーへのアクセスを提供します。このサービスは、ローカルシステムと大規模バックエンドシステムを紐付けます。たとえば、LDAP ディレクトリーや Red Hat Enterprise Linux の Active Directory や IdM ドメイン、または Kerberos レルムなどです。

SSSD は、ID ストアに接続して認証情報を取得する方法を設定し、その情報を使用してユーザーと認証情報のローカルキャッシュを作成します。Active Directory を含む ID プロバイダーでは、SSSD は承認情報も取得することができます。

SSSD は、ローカルクライアントと設定済みのデータストアとの仲介役です。この関係は、管理者に多くの利益をもたらします。

- ※ **ID / 認証サーバー上の負荷の軽減** すべてのクライアントサービスが認証サーバーに直接連絡するのではなく、すべてのローカルクライアントが SSSD に連絡して、SSSD が ID サーバーに接続するか、そのキャッシュをチェックします。
- ※ **オフライン認証の許可** デフォルトでは、SSSD はユーザー ID (ユーザー名、UID、GID) のキャッシュを保持します。また、SSSD がリモートサービスから取得したハッシュされたパスワードという認証情報のキャッシュを SSSD が保持できるように明示的に有効化します。これらのキャッシュを保持することで、ユーザーは ID サーバーがオフラインであったり、ローカルマシンがオフラインであってもリソースに正常に認証することができます。
- ※ **単一のユーザーアカウントの使用** リモートユーザーは多くの場合、2 つ (またはそれ以上) のユーザーアカウントを持っています。1 つが自身のローカルマシン用で、もう 1 つが組織のシステム用などです。仮想プライベートネットワーク (VPN) に接続するために、これが必要になります。SSSD がキャッシングとオフライン認証をサポートするので、リモートユーザーはローカルマシンに認証するだけで SSSD がそのネットワーク認証情報を維持し、ネットワークリソースに接続できるようになります。

SSSD はこれらのユーザーや認証情報をキャッシュするので、ローカルシステムまたは ID プロバイダーがオフラインになっても、サービスはユーザー認証情報を利用して確認することができます。

3.1. SSSD 設定の基本

SSSD は、システムがより大型の外部 ID サービスに接続できるようにするローカルサービスです。これは、SSSD 設定ファイルの **ドメイン** を設定することで実行されます。ドメインは常に、ユーザー情報を提供する **ID プロバイダー** を表します。またドメインはオプションで、認証やパスワード変更などの操作の種類が異なる他のプロバイダーを定義します (すべて操作が単一ドメインまたはサーバーで実行される場合、ID プロバイダーはその操作に使用することもできます)。



注記

SSSD を使うと、すべてのユーザー ID を別個の外部 ID ソースで作成、保持することができます。Windows 統合では、ユーザーアカウントの管理に Active Directory ドメインを使用することができます (ほとんどの環境の場合と同様)。ローカルシステムのユーザーは、Active Directory 内のユーザーアカウントで作成したり同期したりする必要はありません。SSSD はこれらの Windows ID を使用して、Windows ユーザーがローカルシステムとローカルサービスにアクセスできるようにします。

SSSD は、システム上のどのサービスが認証情報のキャッシングとユーザーアカウントに SSSD を使用するかも定義します。これらは Name Service Switch (NSS) やプラグ可能な認証モジュール (PAM) などの基礎的なセキュリティサービスに関連してきます。これらは、高レベルサービスが後で使用します。

3.1.1. sssd.conf ファイルを設定する

SSSD のサービスとドメインは、`.conf` ファイルで設定されます。デフォルトでは `/etc/sss/sss.conf` ですが、SSSD はインストール後に設定されないため、このファイルは手動で作成、設定する必要があります。

3.1.1.1. sssd.conf ファイルを作成する

SSSD 設定ファイルには、以下の 3 つのパートがあります。

- ※ **[sss]** これは全般的な SSSD プロセスとオペレーションの設定のためのものです。基本的に、設定済みサービスとドメイン、それぞれの設定パラメーターを一覧表示します。
- ※ **[service_name]** これは「[SSSD とシステムサービス](#)」にある、対応しているシステムサービスの設定オプション用です。
- ※ **[domain_type/DOMAIN_NAME]** これは設定済み ID プロバイダーの設定オプション用です。



重要

サービスはオプションですが、SSSD サービスの開始前に少なくとも 1 つの ID プロバイダードメインを設定する必要があります。

例3.1 シンプルな sssd.conf ファイル

```
[sss]
domains = LOCAL
services = nss
config_file_version = 2

[nss]
filter_groups = root
filter_users = root

[domain/LOCAL]
id_provider = local
auth_provider = local
access_provider = permit
```

[sss] セクションには、以下の重要な 3 つのパラメーターがあります。

- ※ **domains** には、`sss.conf` で設定された全ドメインを記載します。これは、SSSD が ID プロバイダーとして使用するものです。`domains` キーにドメインがない場合、設定セクションがあっても、SSSD は使用しません。
- ※ **services** には、`sss.conf` で設定され、SSSD を使用する全システムサービスを記載します。SSSD が開始されると、対応する SSSD サービスが設定済みの各システムサービス用に開始されます。`services` キーにサービスがない場合、設定セクションがあったとしても、SSSD は使用しません。
- ※ **config_file_version** は、ファイル形式の期待値を設定する設定ファイルのバージョンを設定します。ここではバージョン 2 で、最近の SSSD 全バージョン用になっています。

**注記**

`sssd.conf` ファイルでサービスやドメインが設定されても、それらが `[sssd]` セクションの `services` もしくは `domains` パラメーターに記載されていないと、SSSD はそのサービスやドメインと対話しません。

他の設定パラメーターは、`sssd.conf man` ページに記載されています。

サービスおよびドメインの各パラメーターは、本章の該当設定セクションとそれらの `man` ページで説明されています。

3.1.1.2. カスタム設定ファイルを使用する

デフォルトでは、`sssd` プロセスは設定ファイルが `/etc/sssd/sssd.conf` であると想定しています。

代替ファイルは、`sssd` コマンドで `-c` オプションを使うと SSSD に渡すことができます。

```
[root@server ~]# sssd -c /etc/sssd/customfile.conf --daemon
```

3.1.1.3. その他のリソース

本章では、SSSD のサービスおよびドメインの基本的な設定を説明していますが、包括的なリソースではありません。SSSD の各機能エリアにはそれぞれ利用可能な他の設定オプションがあります。特定の機能エリアの `man` ページをチェックして、完全なオプション一覧を取得してください。

一般的な `man` ページの一覧は [表3.1「SSSD Man ページのサンプル」](#)に記載されています。SSSD `man` ページの全一覧は、`sssd(8)` `man` ページの "See Also" セクションを参照してください。

表3.1 SSSD Man ページのサンプル

機能エリア	Man ページ
設定全般	<code>sssd.conf(8)</code>
sudo サービス	<code>sssd-sudo</code>
LDAP ドメイン	<code>sssd-ldap</code>
アクティブディレクトリドメイン	<code>sssd-ad</code>
	<code>sssd-ldap</code>
ID 管理 (IdM または IPA) ドメイン	<code>sssd-ipa</code>
	<code>sssd-ldap</code>
ドメインの Kerberos 認証	<code>sssd-krb5</code>
オープン SSH キー	<code>sss_ssh_authorizedkeys</code>
	<code>sss_ssh_knownhostsproxy</code>
キャッシュのメンテナンス	<code>sss_cache</code> (クリーンアップ)
	<code>sss_useradd</code> , <code>sss_usermod</code> , <code>sss_userdel</code> , <code>sss_seed</code> (ユーザーキャッシュエントリ管理)

3.1.2. SSSD の起動と停止



重要

SSSD の初回起動時前に、ドメインを少なくとも 1 つ設定してください。[「SSSD および ID プロバイダー \(ドメイン\)」](#)を参照してください。

SSSD の起動と停止には、`systemctl` ユーティリティを使用します。

```
[root@server ~]# systemctl start sssd.service
```

```
[root@server ~]# systemctl stop sssd.service
```

デフォルトでは、SSSD は自動的に起動するようには設定されていません。システムブート時に SSSD が自動で起動するよう設定するには、`systemctl enable` コマンドを実行します。

```
[root@server ~]# systemctl enable sssd.service
```

`systemctl` を使用したサービス管理についての詳細情報は、[Red Hat Enterprise Linux 7 システム管理者のガイドの「システムサービスの管理」のセクション](#)を参照してください。

3.2. SSSD とシステムサービス

SSSD とその関連サービスは `/etc/sss/sss.conf` ファイル内で設定されています。`[sss]` セクションにはアクティブなサービスも記載され、それらは `sss` が `services` ディレクティブの中で開始する時に起動されます。

SSSD は、以下のシステムサービスに認証情報のキャッシュを提供できます。

- ※ `sss_nss` モジュールからのネームサービス要求に応答する Name Service Switch (NSS) プロバイダーサービス。これは、SSSD 設定の `[nss]` セクション内で設定されます。

[「サービスの設定: NSS」](#)で説明されています。

- ※ `sss_pam` モジュールを介して PAM 会話を管理する PAM プロバイダーサービス。これは設定の `[pam]` セクション内で設定されます。

[「サービスの設定: PAM」](#)で説明されています。

- ※ SSSD が `known_hosts` ファイルや他のキー関連の設定を管理する方法を定義する SSH プロバイダーサービス。OpenSSH による SSSD の使用については、[「サービスの設定: OpenSSH およびキャッシュ済みの鍵」](#)で説明されています。

- ※ 設定済みマウントの場所を取得するために LDAP サーバーに接続する `autofs` プロバイダーサービス。基本的な設定は、設定ファイル内の `[autofs]` セクションで設定します。`[autofs]` のオプションは、`sss-ldap(5) man` ページの "Autofs オプション" のセクションで説明されています。他の設定オプションは、設定ファイル内の `[domain/NAME]` セクションで、LDAP ID プロバイダーの一部として設定されます。

[「サービスの設定: autofs」](#)で説明されています。

- ※ 設定済み `sudo` ポリシーを取得するために LDAP サーバーに接続する `sudo` プロバイダーサービス。基本的な設定は、設定ファイル内の `[sudo]` セクションで設定します。`[sudo]` のオプションは、`sss-ldap(5) man` ページの "Sudo オプション" のセクションで説明されています。他の設定オプションは、設定ファイル内の `[domain/NAME]` セクションで、LDAP ID プロバイダーの一部として設定されます。`sss-sudo(5) man` ページにも `sudo` と `sss` が機能する設定方法に関する便利な情報が含まれています。

[「サービスの設定: sudo」](#)で説明されています。

- ※ SSSD が Kerberos と動作してアクティブディレクトリーのユーザーおよびグループを管理する方法を定義する PAC レスポンダーサービス。これは、ドメインのアクティブディレクトリー ID プロバイダー管理の一部です。

3.2.1. サービスの設定: NSS

SSSD は、NSS モジュールの 1 つである `sssd_nss` を提供します。これはシステムに SSSD を使用してユーザー情報を取得するよう指示します。NSS 設定には SSSD モジュールへの参照が含まれている必要があり、そうすることで SSSD 設定が SSSD と NSS の対話方法を設定します。

3.2.1.1. NSS サービスマップと SSSD

Name Service Switch (NSS) は、多くの設定および名前解決サービスをルックアップするサービスのために集中化した設定を提供します。NSS は、システム ID およびサービスと設定ソースとのマッピング方法の 1 つを提供します。

SSSD は数種類の NSS マップのプロバイダーサービスとして NSS と機能します。

- ※ パスワード (`passwd`)
- ※ ユーザーグループ (`shadow`)
- ※ グループ (`groups`)
- ※ ネットグループ (`netgroups`)
- ※ サービス (`services`)

3.2.1.2. SSSD を使用する NSS サービスの設定

NSS は、サービスマップの一部またはすべてにおいて複数の ID および設定プロバイダーを使用できます。デフォルトでは、サービス用のシステムファイルを使用します。SSSD を含めるには、希望するサービスタイプに `nss_sss` モジュールを含める必要があります。

1. 認証設定ツールを使って SSSD を有効にします。これで `nsswitch.conf` ファイルは自動的に SSSD をプロバイダーとして使用するよう設定されます。

```
[root@server ~]# authconfig --enablesssd --update
```

これでパスワード、シャドウ、グループ、およびネットグループサービスマップが SSSD モジュールを使用するよう自動的に設定されます。

```
passwd:      files sss
shadow:      files sss
group:       files sss

netgroup:    files sss
```

2. `authconfig` で SSSD が有効化されると、サービスマップはデフォルトでは有効化されません。このマップを含めるには、`nsswitch.conf` ファイルを開いて `services` マップに `sss` モジュールを追加します。

```
[root@server ~]# vim /etc/nsswitch.conf

...
services: file sss
...
```


3.2.1.3. NSS と機能するように SSSD を設定する

SSSD が NSS リクエストに応えるために使用するオプションおよび設定は、SSSD 設定ファイルの `[nss]` サービスセクションで設定します。

1. `sssd.conf` ファイルを開きます。

```
[root@server ~]# vim /etc/sss/sss.conf
```

2. NSS が、SSSD と一緒に機能するサービスの 1 つとして記載されていることを確認します。

```
[sss]
config_file_version = 2
sbus_timeout = 30
services = nss, pam
```

3. `[nss]` セクションで NSS パラメーターを変更します。[表3.2「SSSD \[nss\] 設定のパラメーター」](#)に一覧表示されています。

```
[nss]
filter_groups = root
filter_users = root
entry_cache_timeout = 300
entry_cache_nowait_percentage = 75
```

4. SSSD を再起動します。

```
[root@server ~]# systemctl restart sss.service
```

表3.2 SSSD [nss] 設定のパラメーター

パラメーター	値の形式	説明
--------	------	----

パラメーター	値の形式	説明
entry_cache_nowait_percentage	整数	<p><code>sssd_nss</code> がエントリーキャッシュの自動リフレッシュを開始するまでの時間を指定します。このオプションをゼロ (0) に設定すると、エントリーキャッシュのリフレッシュが無効になります。</p> <p>この時間は、ドメインの <code>entry_cache_timeout</code> の値の割合として指定されます。たとえば、<code>entry_cache_timeout</code> を 300 秒に設定し、<code>entry_cache_nowait_percentage</code> を 75 に設定した場合、タイムアウト間隔の 75% である 225 秒で要求が来ると、エントリーキャッシュのリフレッシュが開始されます。自動リフレッシュによりキャッシュ更新を待たずに済むので、次の要求がブロックされません。</p> <p>このオプションの有効な値は 0 から 99 で、<code>entry_cache_timeout</code> の値をベースにパーセンテージを設定します。デフォルト値は 50 % です。</p>
entry_negative_timeout	整数	<p><code>sssd_nss</code> が ネガティブ キャッシュヒットをキャッシュすべき時間の長さを秒単位で指定します。ネガティブキャッシュヒットとは、存在しないエントリーも含む無効なデータベースエントリーに対するクエリです。</p>
filter_users、filter_groups	文字列	<p>特定のユーザーを NSS データベースからフェッチ対象から除外するように SSSD に指示します。これは、特に <code>root</code> のようなシステムアカウントに有用です。</p>
filter_users_in_groups	ブール値	<p>グループルックアップの実行時に、<code>filter_users</code> リストに記載されているユーザーがグループメンバーシップに表示されるかどうかを設定します。これを <code>FALSE</code> にすると、グループルックアップは、そのグループのメンバーである全ユーザーを返します。指定されない場合は、値はデフォルトの <code>true</code> となり、グループメンバーリストをフィルターにかけます。</p>

パラメーター	値の形式	説明
override_homedir	文字列	<p>ユーザーのホームディレクトリーを上書きします。これは絶対値にすることもでき、テンプレートにすることもできます。テンプレートの場合は、以下の変数をとります。</p> <ul style="list-style-type: none"> %u ログイン名 %U UID 番号 %d ドメイン名 %f 完全修飾ユーザー名 (user@domain) %o ID プロバイダーから取得したオリジナルのホームディレクトリー %% 文字通りの '%' <p>このオプションは、グローバルまたはドメインごとに設定することができます。</p> <p>例を示します。</p> <div style="border: 1px solid gray; border-radius: 10px; padding: 5px; width: fit-content; margin: 10px auto;"> <pre>override_homedir = /home/%u</pre> </div> <p>これはデフォルトでは設定されません。上記のケースでは、ID プロバイダーである LDAP ディレクトリーから値が取得されます。</p>

パラメーター	値の形式	説明
fallback_homedir	文字列	<p>ドメインのデータプロバイダーが明示的にユーザーのホームディレクトリを指定していない場合、デフォルトのテンプレートを設定します。</p> <p>このオプションの有効な値は、<code>override_homedir</code>と同じです。</p> <p>例を示します。</p> <div style="border: 1px solid black; border-radius: 10px; padding: 5px; margin: 10px 0;"> <pre>override_homedir = /home/%u</pre> </div> <p>これはデフォルトでは設定されず、ユーザーのホームディレクトリが設定されていない場合は、代替値はありません。</p>
override_shell	文字列	<p>全ユーザーのログインシェルを上書きします。このオプションは、NSS サービスセクションでグローバルに指定するか、ドメインごとに指定することができます。</p> <p>これはデフォルトでは設定されず、SSSD は LDAP ID プロバイダーから取得した値を使用します。</p>
allowed_shells	文字列	<p>ユーザーシェルをリストの値のいずれかに制限します。リストは厳密に以下の順序で評価されます。</p> <ol style="list-style-type: none"> 1. <code>/etc/shells</code> にあるシェル。 2. <code>allowed_shells</code> リストにあるシェル。このリストは、SSSD 用に <code>shell_fallback</code> パラメーターで指定されます。 3. <code>/etc/shells</code> や <code>allowed_shells</code> のリストにシェルがない場合は、<code>nologin</code> シェルになります。 4. 空白の文字列は、そのまま <code>libc</code> に渡されます。 <p>「<code>/etc/shells</code>」は SSSD の起動時にのみ読み込まれるので、新しいシェルがインストールされた場合は、SSSD の再起動が必要になります。</p> <p>これはデフォルトでは設定されず、ユーザーのシェルが使用されます。</p>

パラメーター	値の形式	説明
vetoed_shells	文字列	記載されたシェルのインスタンスが <code>shell_fallback</code> パラメーターで指定されたシェルで置換されます。
shell_fallback	文字列	マシンに許可されたシェルがインストールされていない場合に使用するデフォルトのシェルです。 デフォルト値は <code>/bin/sh</code> です。
default_shell	文字列	ルックアップ中にプロバイダーがシェルを返さない場合のデフォルトのシェルを設定します。このオプションが有効な場合は、他のいずれのシェルオプションよりもこれが優先され、NSS セクションないまたはドメインごとに設定することができます。 これはデフォルトでは設定されず、シェルが指定されていない場合は <code>NULL</code> が返されます。 <code>/bin/sh</code> といった値の提供は、 <code>libc</code> に依存します。
get_domains_timeout	整数	サブドメインのリストが有効とみなされる時間を秒単位で指定します。 デフォルト値は 60 秒です。
memcache_timeout	整数	メモリー内のキャッシュのレコードが有効である時間を秒単位で指定します。 デフォルト値は 300 秒です。
debug_level	0 から 9 の整数	デバッグログレベルを設定します。
reconnection_retries	整数	データプロバイダーがクラッシュした場合または再起動した場合に、サービスが再接続を試行する回数を設定します。 デフォルト値は 3 です。

3.2.2. サービスの設定: PAM



警告

PAM 設定ファイルに誤りがあると、ユーザーはシステムから完全にロックアウトされてしまいます。設定ファイルは変更を行う前に必ずバックアップを作成し、変更を元に戻すことができるようにセッションをオープンのままにしてください。

SSSD は PAM モジュールの1つである `sssd_pam` を提供します。これが SSSD を使用してユーザー情報を取り込むようにシステムに指示します。PAM 設定には SSSD モジュールへの参照が含まれている必要があり、そうすることで SSSD 設定が SSSD と PAM との対話方法を設定します。

PAM サービスは以下の手順で設定します。

1. `authconfig` を使用してシステム認証のために SSSD を有効にします。

```
# authconfig --enablesssd --enablesssdauth --update
```

これで PAM 設定が自動更新され、すべての SSSD モジュールを参照するようにします。

```
##%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      pam_env.so
auth      sufficient    pam_unix.so nullok try_first_pass
auth      requisite     pam_succeed_if.so uid >= 500 quiet
auth sufficient pam_sss.so use_first_pass
auth      required      pam_deny.so

account   required      pam_unix.so
account   sufficient    pam_localuser.so
account   sufficient    pam_succeed_if.so uid < 500 quiet
account [default=bad success=ok user_unknown=ignore] pam_sss.so
account   required      pam_permit.so

password  requisite     pam_pwquality.so try_first_pass retry=3
password  sufficient    pam_unix.so sha512 shadow nullok try_first_pass
use_authok
password sufficient pam_sss.so use_authok
password  required      pam_deny.so

session   optional      pam_keyinit.so revoke
session   required     pam_limits.so
session   [success=1 default=ignore] pam_succeed_if.so service in crond quiet
use_uid
session sufficient pam_sss.so
session   required     pam_unix.so
```

これらのモジュールは必要に応じて `include` ステートメントを設定できます。

2. `sssd.conf` ファイルを開きます。

```
# vim /etc/sss/sss.conf
```

3. PAM が SSSD と一緒に機能するサービスの 1 つとして記載されていることを確認します。

```
[sss]
config_file_version = 2
sbus_timeout = 30
services = nss, pam
```

4. `[pam]` セクションで PAM パラメーターのいずれかを変更します。これらは、[表3.3「SSSD \[pam\] 設定のパラメーター」](#) 内に一覧表示されています。

```
[pam]
offline_credentials_expiration = 2
offline_failed_login_attempts = 3
offline_failed_login_delay = 5
```

5. SSSD を再起動します。

```
[root@server ~]# systemctl restart sssd.service
```

表3.3 SSSD [pam] 設定のパラメーター

パラメーター	値の形式	説明
offline_credentials_expiration	整数	認証プロバイダーがオフラインの場合に、キャッシュしたログインが使用可能な時間を日数で設定します。この値は、最後に正常にログインした時から測られます。指定されない場合は、デフォルト値はゼロ (0) となり、制限なしになります。
offline_failed_login_attempts	整数	認証プロバイダーがオフラインの場合に、許可されるログイン試行の失敗回数を設定します。指定されない場合は、デフォルト値はゼロ (0) になり、制限なしとなります。
offline_failed_login_delay	整数	ユーザーがログイン試行の失敗限度に到達した場合に、ログイン試行を阻止する長さを設定します。ゼロ (0) に設定されている場合は、ユーザーが試行の失敗限度に達するとプロバイダーがオフラインの間は認証ができません。正常にオンライン認証を行った場合のみ、オフライン認証を再度有効にできます。指定されていない場合は、デフォルトは 5 になります。
reconnection_retries	整数	データプロバイダーがクラッシュした場合または再起動した場合に、サービスが再接続を試行する回数を設定します。 デフォルト値は 3 です。
pam_verbosity	0 から 3 の整数	認証中にユーザーに表示される情報の詳細度を制御します。値が高いと、より多くの情報を SSSD が表示します。デフォルト値は 1 です。
pam_id_timeout	整数	クライアントのアプリケーションベースで、SSSD がユーザー ID 情報をキャッシュする長さを秒単位で制御するオプションです。これは ID プロバイダーへの過剰なラウンドトリップを回避するためのものです。デフォルト値は 5 です。

パラメーター	値の形式	説明
pam_pwd_expiration_warning	整数	パスワードの有効期限が切れる何日前にユーザーに知らせるかを設定します。デフォルトは 0 で、その場合は SSSD がバックエンドサーバーから有効期限切れの時期についての情報を受け取ると、SSSD は自動的に警告を表示します。

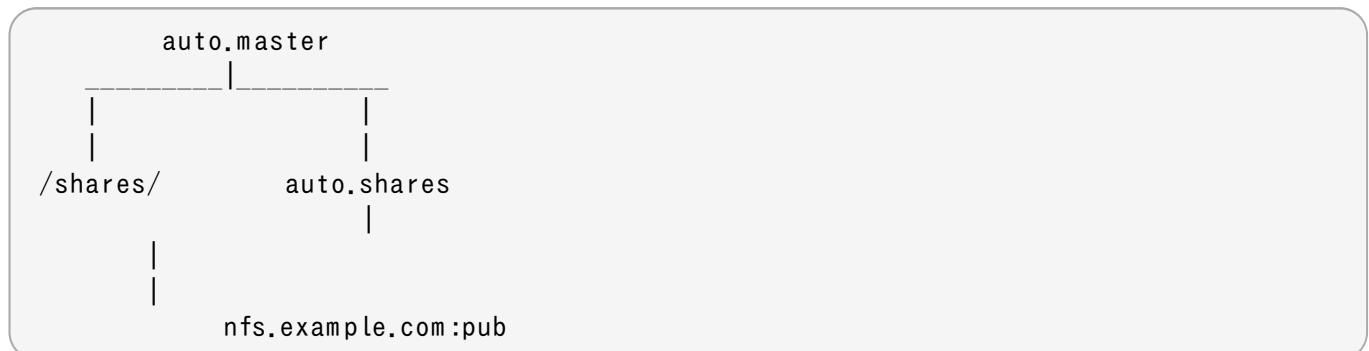
sssd.conf 内の SSSD [pam] 設定パラメーターについての詳細情報は、sssd.conf(5) man ページを参照してください。

3.2.3. サービスの設定: autofs

3.2.3.1. 自動マウント、LDAP、SSSD について

自動マウントマップは一般的にフラットファイルで、マップとマウントディレクトリー、ファイルサーバーの関係を定義します。(自動マウントは、[ストレージ管理ガイド](#)で説明されています。)

たとえば、**nfs.example.com** というファイルサーバーがあったとして、これがディレクトリー **pub** をホストし、自動マウントが **/shares/** ディレクトリーにディレクトリーをマウントするように設定されているとします。すると、マウントする場所は **/shares/pub** になります。すべてのマウントは **auto.master** ファイルに記載されており、これはマウントを設定する異なるマウントディレクトリーとファイルを特定します。**auto.shares** マップは各ファイルサーバーとマウントディレクトリーを特定し、自動マウントがこれらを **/shares/** ディレクトリーにマウントします。この関係を図にすると以下のようにになります。



すると、すべてのマウントポイントは、**auto.master** ファイルと **auto.whatever** ファイルという少なくとも 2 つのファイルで定義されることになり、これらのファイルは各ローカルの自動マウントプロセスで利用可能となっている必要があります。

大規模な環境で管理者がこれを管理する方法の一つは、中央 LDAP ディレクトリーに自動マウント設定を保存し、各ローカルシステムをその LDAP ディレクトリーに向けるように設定することです。こうすることで、更新が必要になるのは 1 カ所だけとなり、ローカルシステムが新しいマップを自動的に認識するようになります。

自動マウント - LDAP の設定では、自動マウントファイルは LDAP エントリーとして保存され、その後に必須の自動マウントファイルに変換されます。すると各要素は LDAP 属性に変換されます。

LDAP エントリーは以下のようにになります。

```

# container entry
dn: cn=automount,dc=example,dc=com
objectClass: nsContainer
objectClass: top
cn: automount
  
```



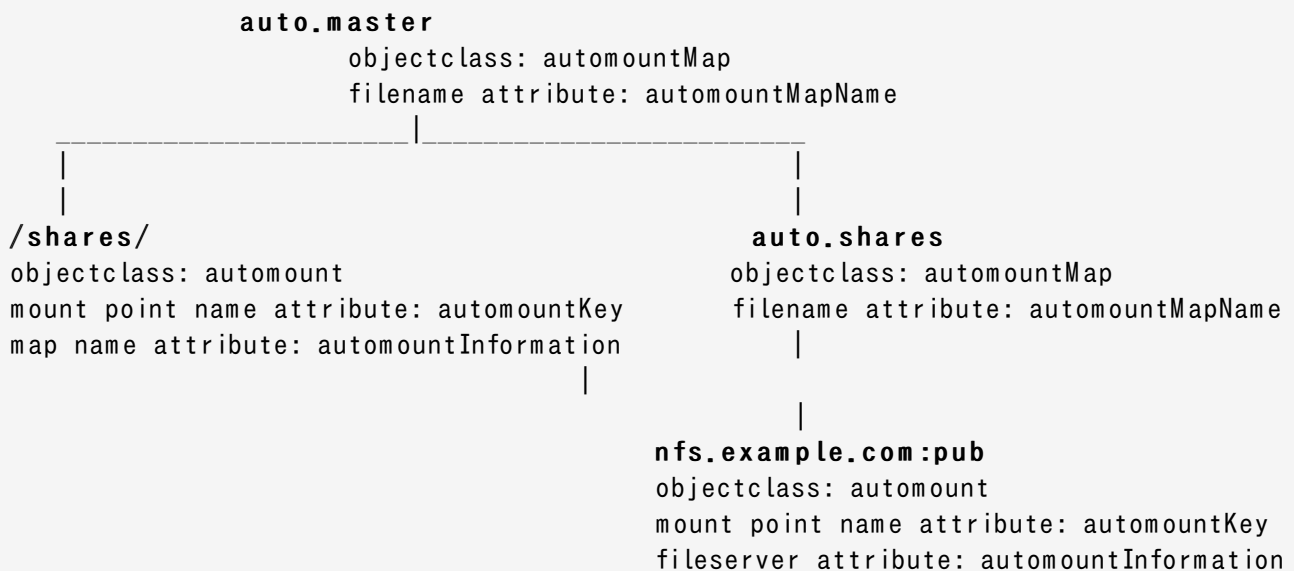
```
# master map entry
dn: automountMapName=auto.master,cn=automount,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.master

# shares map entry
dn: automountMapName=auto.shares,cn=automount,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.shares

# shares mount point
dn:
automountKey=/shares,automountMapName=auto.master,cn=automount,dc=example,dc=com
objectClass: automount
objectClass: top
automountKey: /shares
automountInformation: auto.shares

# pub mount point
dn: automountKey=pub,automountMapName=auto.shares,cn=automount,dc=example,dc=com
objectClass: automount
objectClass: top
automountKey: pub
automountInformation: filer.example.com:/pub
description: pub
```

スキーマ要素は、以下の図のようになります (RFC 2307 スキーマ)。



autofs は、これらのスキーマ要素を使用して自動マウント設定を取得します。`/etc/sysconfig/autofs` ファイルは、自動マウントエントリーに使用される LDAP サーバー、ディレクトリーの位置、スキーマ要素を特定します。

```
LDAP_URI=ldap://ldap.example.com
SEARCH_BASE="cn=automount,dc=example,dc=com"
MAP_OBJECT_CLASS="automountMap"
ENTRY_OBJECT_CLASS="automount"
MAP_ATTRIBUTE="automountMapName"
ENTRY_ATTRIBUTE="automountKey"
```

```
VALUE_ATTRIBUTE="automountInformation"
```

自動マウント設定を LDAP ディレクトリーではなく、SSSD に向けるように設定することができます。すると SSSD は自動マウントが必要とする全情報を保存し、ユーザーがディレクトリーをマウントしようとする、この情報は SSSD にキャッシュされます。これによりフェイルオーバーやサービスディレクトリー、タイムアウトなどの設定が便利になり、LDAP サーバーへの接続回数が減ることでパフォーマンスも改善されます。最も重要なのは、SSSD を使用することでマウント情報がすべてキャッシュされるので、LDAP サーバーがオフラインになっても クライアントはディレクトリーを正常にマウントできるという点です。

3.2.3.2. SSSD での autofs サービスの設定

1. **autofs** パッケージがインストールされていることを確認します。
2. **sssd.conf** ファイルを開きます。

```
[root@server ~]# vim /etc/sss/sss.conf
```

3. SSSD が管理するサービス一覧に **autofs** サービスを追加します。

```
[sss]
services = nss,pam,autofs
....
```

4. 新たな **[autofs]** サービス設定セクションを作成します。このセクションは空白のままでもかまいません。唯一設定可能なオプションは、ネガティブキャッシュヒットのタイムアウトです。

ただしこのセクションは、SSSD が **autofs** サービスを認識し、デフォルト設定を提供するために必須のものとなります。

```
[autofs]
```

5. 自動マウント情報は、SSSD 設定の設定済み LDAP ドメインから読み取られるので、LDAP ドメインが利用可能である必要があります。追加設定がない場合、自動マウント情報は RFC 2307 スキーマと LDAP 検索ベース (**ldap_search_base**) がデフォルト設定となります。これはカスタマイズが可能です。

- ※ **autofs_provider** はディレクトリーのタイプになります。デフォルトは、**id_provider** の値です。値が **none** の場合は、ドメインの **autofs** を明示的に無効とします。
- ※ **ldap_autofs_search_base** は検索ベースです。
- ※ **ldap_autofs_map_object_class** は、マップエントリーの認識に使用するオブジェクトクラスです。
- ※ **ldap_autofs_map_name** は、マップ名の認識に使用する属性です。
- ※ **ldap_autofs_entry_object_class** は、マウントポイントエントリーの認識に使用するオブジェクトクラスです。
- ※ **ldap_autofs_entry_key** は、マウントポイント名の認識に使用する属性です。
- ※ **ldap_autofs_entry_value** は、マウントポイントの新たな設定情報に使用する属性です。

例を示します。

```
[domain/LDAP]
...
autofs_provider=ldap
ldap_autofs_search_base=cn=automount,dc=example,dc=com
ldap_autofs_map_object_class=automountMap
ldap_autofs_entry_object_class=automount
ldap_autofs_map_name=automountMapName
ldap_autofs_entry_key=automountKey
ldap_autofs_entry_value=automountInformation
```

6. `sssd.conf` ファイルを保存し、閉じます。
7. `nsswitch.conf` ファイルを編集して、場所を `ldap` から `sss` に変更すると、`autofs` は自動マウントマップ情報を SSSD で探すように設定されます。

```
[root@server ~]# vim /etc/nsswitch.conf

automount: files sss
```

8. SSSD を再起動します。

```
[root@server ~]# systemctl restart sssd.service
```

3.2.4. サービスの設定: `sudo`

3.2.4.1. `sudo`、LDAP、SSSD について

`sudo` ルールは `sudoers` ファイルで定義され、これは一貫性の維持のために各マシンに個別に配布される必要があります。

大規模な環境で管理者がこれを管理する方法の一つは、中央 LDAP ディレクトリーに `sudo` 設定を保存し、各ローカルシステムをその LDAP ディレクトリーに向けるように設定することです。こうすることで、更新が必要になるのは 1 カ所だけとなり、ローカルシステムが新しいルールを自動的に認識するようになります。

`sudo`-LDAP 設定では、各 `sudo` ルールは LDAP エントリーとして保存され、`sudo` ルールの各コンポーネントは LDAP 属性で定義されます。

`sudoers` ルールは以下ようになります。

```
Defaults    env_keep+=SSH_AUTH_SOCK
...
%wheel     ALL=(ALL)    ALL
```

LDAP エントリーは以下ようになります。

```
# sudo defaults
dn: cn=defaults,ou=SUDOers,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: defaults
description: Default sudoOptions go here
sudoOption: env_keep+=SSH_AUTH_SOCK

# sudo rule
```

```
dn: cn=%wheel,ou=SUDOers,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: %wheel
sudoUser: %wheel
sudoHost: ALL
sudoCommand: ALL
```

注記

SSSD は、*sudoHost* 属性の値によって、ローカルシステムに適用される **sudo** ルールのみをキャッシュします。つまり、*sudoHost* の値は ALL に設定され、ホスト名かシステムネットグループ、システムホスト名か完全修飾ドメイン名または IP アドレスに合致する正規表現を使用します。

sudo サービスは、LDAP サーバーに向けて、それらの LDAP エントリーからルール設定をプルするように設定できます。**sudo** 設定を LDAP ディレクトリーではなく、SSSD に向けることもできます。すると SSSD は **sudo** が必要とする全情報を保存し、ユーザーが **sudo** 関連の操作を行おうとすると、最新の **sudo** 設定が LDAP ディレクトリーから SSSD 経由でプルされます。しかし、SSSD は **sudo** ルールをすべてキャッシュするので、**LDAP サーバーがオフラインになっても**、ユーザーは中央 LDAP 設定を使用してタスクを実行することができます。

3.2.4.2. SSSD を使った **sudo** の設定

SSSD **sudo** 設定オプションはすべて、`sssd-ldap(5) man` ページに記載されています。

sudo サービスは、以下の手順で設定します。

1. `sssd.conf` ファイルを開きます。

```
[root@server ~]# vim /etc/sss/sss.conf
```

2. **sudo** サービスを SSSD が管理するリストに追加します。

```
[sss]
services = nss, pam, sudo
....
```

3. 新たな [**sudo**] サービス設定セクションを作成します。このセクションは空白のままかまいません。唯一設定可能なオプションは、`sudo not before/after` 時期の評価です。

ただしこのセクションは、SSSD が **sudo** サービスを認識し、デフォルト設定を提供するために必須のものとなります。

```
[sudo]
```

4. **sudo** 情報は、SSSD 設定の設定済み LDAP ドメインから読み取られるので、LDAP ドメインが利用可能である必要があります。LDAP プロバイダーには、以下のパラメーターが必要です。

- ※ `sudo_provider` はディレクトリーのタイプです。これは常に `ldap` です。
- ※ `ldap_sudo_search_base` は検索ベースです。
- ※ `ldap_uri` は LDAP サーバーの URI です。

例を示します。

```
[domain/LDAP]
id_provider = ldap

sudo_provider = ldap
ldap_uri = ldap://example.com
ldap_sudo_search_base = ou=sudoers,dc=example,dc=com
```

IdM を ID プロバイダーとして設定すると、sudo プロバイダーは自動的に有効となるので、設定ファイルで **sudo_provider = ipa** を指定する必要はありません。

```
[domain/IDM]
id_provider = ipa
ipa_domain = example.com
ipa_server = ipa.example.com
```

5. sudo ルールキャッシュをリフレッシュする間隔を設定します。

特定のシステムユーザー向け キャッシュは、そのユーザーがタスクを実行すると毎回必ずチェックされ、更新されます。ただし、SSSD はローカルシステムに関連するルールをすべてキャッシュします。この完全なキャッシュは、以下の 2 つの方法で更新されます。

- ※ 増分、つまり最後の完全更新からルールになされた変更のみ (`ldap_sudo_smart_refresh_interval`、秒単位)。デフォルト値は 15 分間です。
- ※ 完全、つまりキャッシュ全体をダンプし、LDAP サーバー上の最新ルールすべてをプルします (`ldap_sudo_full_refresh_interval`、秒単位)。デフォルト値は 6 時間です。

これら 2 つのリフレッシュ間隔は、以下のように別個に設定されます。

```
[domain/LDAP]
...
ldap_sudo_full_refresh_interval=86400
ldap_sudo_smart_refresh_interval=3600
```



注記

SSSD は、ローカルシステムに適用される sudo ルールのみをキャッシュします。つまり、`sudoHost` の値は ALL に設定され、ホスト名かシステムネットグループ、システムホスト名か完全修飾ドメイン名または IP アドレスに合致する正規表現を使用します。

6. オプションでは、sudo ルールに使われるスキーマを任意の値を設定して変更します。

スキーマ要素は、`ldap_sudorule_*` 属性で設定されています。デフォルトでは、すべてのスキーマ要素が [sudoers.ldap](#) で定義されたスキーマを使用します。これらのデフォルトがほとんどすべての導入で使用されます。

7. sssd.conf ファイルを保存し、閉じます。

8. nsswitch.conf ファイルを編集し、sss の場所を加えることで sudo はルール設定を SSSD で探すように設定されます。

```
[root@server ~]# vim /etc/nsswitch.conf  
sudoers: files sss
```

9. SSSD を再起動します。

```
[root@server ~]# systemctl restart sssd.service
```

3.2.5. サービスの設定: OpenSSH およびキャッシュ済みの鍵

OpenSSH は、2 つのシステム間に安全で暗号化された接続を作成します。一方のマシンが他方のマシンを認証してアクセスを許可します。認証は、マシン自体のサーバー接続に対するものである場合と、該当マシン上のユーザーに対する場合があります。

この認証は、認証を行なっているユーザーもしくはマシンを特定する **公開鍵と秘密鍵のペア** で実行されます。マシンにアクセスしようとしているリモートのマシンもしくはユーザーは、鍵のペアを提示します。するとローカルマシンは、そのリモートのエンティティを信頼するかどうかを決定します。信頼できる場合は、そのリモートマシン用の公開鍵は **known_hosts** ファイルに、リモートユーザー用の公開鍵は **authorized_keys** ファイルに保存されます。このリモートマシンもしくはユーザーが再度認証を試みる際は常に、ローカルシステムは単にこの **known_hosts** か **authorized_keys** ファイルを最初にチェックして、このリモートのエンティティが認証され、信頼できるものかを確認した後アクセスを許可します。

最初の問題は、これらの ID を信頼性を持って確認することになります。

known_hosts ファイルは、マシン名、IP アドレス、公開鍵の 3 つで構成されています。

```
server.example.com, 172.16.0.1 ssh-rsa  
AbcdEfg1234ZYX098776/AbcdEfg1234ZYX098776/AbcdEfg1234ZYX098776=
```

known_hosts ファイルは、様々な理由ですぐに古いものとなってしまいます。IP アドレス経由の DHCP サイクルを使用しているシステムでは、新しい鍵が定期的に再発行されたり、仮想マシンやサービスがオンラインで持ち込まれたり削除されたりする場合があります。このようなことがあると、ホスト名、IP アドレス、鍵の構成が変わってしまいます。

管理者は、現行の **known_hosts** ファイルをクリーンに維持して、セキュリティを保つ必要があります。(それが行われないと、システムユーザーは単に提示されたマシンや鍵をどれでも受け入れるようになってしまい、鍵ベースのセキュリティの利点がなくなってしまいます。)

さらには、マシンとユーザーにはスケーラブルな方法で鍵を配布するという問題があります。マシンは暗号化セッション確立の一部として鍵を送信できますが、ユーザーは事前に鍵を提供する必要があります。単に鍵を配布して継続的に更新するという管理タスクは、困難なものです。

最後に、SSH 鍵およびマシン情報はローカルでのみ維持されます。**known_hosts** ファイル全体が更新されないと、ネットワーク上でマシンやユーザーが一部のシステムには認識、信頼されるものの、別のシステムには認識、信頼されないということが起こり得ます。

SSSD の目標は、認証情報キャッシュとして機能することです。これには、マシンおよびユーザーの SSH 公開鍵の認証情報キャッシュとして機能することが含まれます。OpenSSH は、キャッシュされた鍵の確認に SSSD を参照するように設定されています。SSSD は、Red Hat Linux の Identity Management (IPA) ドメインを ID として使用し、Identity Management は実際に公開鍵およびホスト情報を保存します。



注記

SSSD を OpenSSH の鍵のキャッシュとして使用できるのは、Identity Management ドメインに登録もしくは参加している Linux マシンのみです。他の Unix マシンや Windows マシンは、`known_hosts` ファイルで通常の認証メカニズムを使用する必要があります。

3.2.5.1. ホスト鍵に SSSD を使用するよう OpenSSH を設定する

OpenSSH は、ユーザー固有の設定ファイル (`~/.ssh/config`) か、システムワイドの設定ファイル (`/etc/ssh/sshd_config`) で設定されます。ユーザーファイルはシステム設定より優先され、パラメーターには最初に取得された値が使用されます。

SSSD にはホスト鍵を管理するためのツール `sss_ssh_knownhostsproxy` があり、以下の 3 つの操作を実行します。

1. 登録済み Linux システムから公開ホスト鍵を取得します。
2. カスタムホストファイル `/var/lib/sss/pubconf/known_hosts` にホスト鍵を保存します。
3. ソケット (デフォルト) もしくは安全な接続でホストマシンとの接続を確立します。

このツールは以下の形式になります。

```
sss_ssh_knownhostsproxy [-d sssd_domain] [-p ssh_port] HOSTNAME [PROXY_COMMAND]
```

表3.4 `sss_ssh_knownhostsproxy` のオプション

短い引数	長い引数	説明
	<code>HOSTNAME</code>	チェックおよび接続するホストのホスト名を記入します。OpenSSH 設定ファイルでは、 <code>%h</code> のトークンにすることができます。
	<code>PROXY_COMMAND</code>	SSH クライアント接続に使用するプロキシコマンドを渡します。これは、 <code>ssh -o ProxyCommand=値</code> の実行に類似したものです。このオプションは、コマンドラインもしくは別のスクリプトで <code>sss_ssh_knownhostsproxy</code> を実行する際に使われますが、OpenSSH 設定ファイルでは不要です。
<code>-d <i>sssd_domain</i></code>	<code>--domain <i>sssd_domain</i></code>	指定されたドメインのエンティティ内の公開鍵のみを検索します。これがない場合は、SSSD はすべての設定済みドメインで鍵を検索します。
<code>-p <i>port</i></code>	<code>--port <i>port</i></code>	SSH クライアントへの接続にこのポートを使用します。デフォルトでは、 <code>port 22</code> です。

この SSSD ツールを使用するには、`/etc/ssh/sshd_config` もしくは `~/.ssh/config` ファイルで 2 つのパラメーターを追加または編集します。

- ※ SSH クライアント接続に使用するコマンドに指定します (**ProxyCommand**)。これは、`sss_ssh_knownhostsproxy` に希望する引数とホスト名を加えたものです。
- ※ デフォルトの `known_hosts` ファイルではなく、SSSD ホストファイルの場所を指定します。 (**GlobalKnownHostsFile**)。SSSD ホストファイルは、`/var/lib/sss/pubconf/known_hosts` です。

たとえば、以下では SSSD ドメイン内で公開鍵を検索して、提供されるポートとホストで接続します。

```
ProxyCommand /usr/bin/sss_ssh_knownhostsproxy -p %p %h
GlobalKnownHostsFile /var/lib/sss/pubconf/known_hosts
```

3.2.5.2. ユーザー鍵に SSSD を使用するよう OpenSSH を設定する

OpenSSH では、ユーザー鍵はローカルシステムの `authorized_keys` ファイルに保存されています。ホストに関しては OpenSSH が参照するために、SSSD がユーザー公開鍵の別個のキャッシュを維持し、自動的に更新します。これは、`.ssh/sss_authorized_keys` ファイルに保存されます。

OpenSSH は、ユーザー固有の設定ファイル (`~/.ssh/config`) か、システムワイドの設定ファイル (`/etc/ssh/sshd_config`) で設定されます。ユーザーファイルはシステム設定より優先され、パラメーターには最初に取得された値が使用されます。

SSSD にはユーザー鍵を管理するためのツール `sss_ssh_authorizedkeys` があり、以下の 2 つの操作を実行します。

1. Identity Management (IPA) ドメインのユーザーエンティティからユーザーの公開鍵を取得します。
2. 標準的な認証鍵の形式でユーザー鍵をカスタムファイル `.ssh/sss_authorized_keys` に保存します。

このツールは以下の形式になります。

```
sss_ssh_authorizedkeys [-d sssd_domain] USER
```

表3.5 `sss_ssh_authorizedkeys` のオプション

短い引数	長い引数	説明
	<code>USER</code>	公開鍵を取得するユーザー名またはアカウント名を記入します。OpenSSH 設定ファイルでは、 <code>%u</code> のトークンとすることもできます。
<code>-d sssd_domain</code>	<code>--domain sssd_domain</code>	指定されたドメインのエンティティ内の公開鍵のみを検索します。これがない場合は、SSSD はすべての設定済みドメインで鍵を検索します。

OpenSSH がユーザー鍵に SSSD を使用するよう設定するには、認証鍵コマンドを使用します。ユーザー鍵とそれが実行されるアカウントのユーザーを取得するようにコマンドを指定します。例を示します。

```
AuthorizedKeysCommand /usr/bin/sss_ssh_authorizedkeys
AuthorizedKeysCommandUser nobody
```

3.3. SSSD および ID プロバイダー (ドメイン)

SSSD はドメインを認識します。ドメインは、異なる外部のデータソースに関連付けられた SSSD 設定ファイル内のエントリーです。ドメインは ID プロバイダー (ユーザー情報用) と、オプションで (認識要求の) 認証プロバイダーやパスワード変更などの他の操作のプロバイダーを組み合わせたものです (すべて操作が単一ドメインまたはサーバーで実行される場合、ID プロバイダーはそれらの操作に使用することもできます)。

SSSD は異なる LDAP ID プロバイダーと機能し (OpenLDAP、Red Hat Directory Server、および Microsoft Active Directory を含む)、ネイティブの LDAP 認証、Kerberos 認証、またはプロバイダー固有の認証プロトコル (アクティブディレクトリーなど) を使用することができます。

ドメインの設定は、ID プロバイダー、認証プロバイダー、およびこれらのプロバイダー内の情報にアクセスするための特定の設定を定義します。ID および認証プロバイダーには以下のようなタイプがあります。

- ※ LDAP、一般的な LDAP サーバー用
- ※ Active Directory (LDAP プロバイダータイプの機能拡張)
- ※ Identity Management (LDAP プロバイダータイプの機能拡張)
- ※ Local、ローカルの SSSD データベース用
- ※ Proxy
- ※ Kerberos (認証プロバイダーのみ)

ID および認証プロバイダーは、ドメインエントリー内で異なる組み合わせで設定が可能です。可能な組み合わせは [表 3.6「ID ストアと認証タイプの組み合わせ」](#) のとおりです。

表3.6 ID ストアと認証タイプの組み合わせ

ID プロバイダー	認証プロバイダー
Identity Management (LDAP)	Identity Management (LDAP)
Active Directory (LDAP)	Active Directory (LDAP)
Active Directory (LDAP)	Kerberos
LDAP	LDAP
LDAP	Kerberos
proxy	LDAP
proxy	Kerberos
proxy	proxy

SSSD がクエリするドメインの一覧には、ドメインエントリー自体と共にドメイン名を追加する必要があります。例を示します。

```
[sssd]
domains = LOCAL, Name
...

[domain/Name]
id_provider = type
auth_provider = type
provider_specific = value
global = value
```

global 属性は、キャッシュやタイムアウト設定など、全タイプのドメインで利用可能です。ID および認証の各プロバイダーには、独自の必須およびオプションの設定パラメーターセットがあります。

表3.7 一般的な [domain] 設定パラメーター

パラメーター	値の形式	説明
id_provider	文字列	<p>このドメインに使用するデータバックエンドを指定します。サポートされている ID バックエンドは以下の通りです。</p> <ul style="list-style-type: none"> ※ ldap ※ ipa (Red Hat Enterprise Linux 内の Identity Management) ※ ad (Microsoft Active Directory) ※ レガシー NSS プロバイダー用の <code>nss_nis</code> のようなプロキシ。プロキシ ID プロバイダーの使用にはレガシー NSS ライブラリーの指定も必要になります。これは、<code>proxy_lib_name</code> オプションで設定します。 ※ local、SSSD 内部ローカルプロバイダー
auth_provider	文字列	<p>ドメインに使用される認証プロバイダーを設定します。このオプションのデフォルト値は <code>id_provider</code> の値となります。サポートされている認証プロバイダーは ldap、ipa、ad、krb5 (Kerberos)、proxy、および none です。</p>
min_id、max_id	整数	<p>これはオプションになります。ドメイン用に UID と GID の範囲を指定します。ドメインがこの範囲外のエントリーを含んでいる場合、それらは無視されます。<code>min_id</code> のデフォルト値は 1 で、<code>max_id</code> のデフォルト値は 0 でこれは無制限となります。</p>



重要

デフォルトの `min_id` 値はすべてのタイプの ID プロバイダーに対して同じものになります。LDAP ディレクトリーが 1 から始まる UID 番号を使用している場合は、ローカルの `/etc/passwd` ファイル内のユーザーと競合原因となる可能性があります。この競合を回避するには、`min_id` を **1000** 以上のできるだけ高い値に設定します。

パラメーター	値の形式	説明
cache_credentials	ブール値	これはオプションになります。ユーザーの認証情報をローカルの SSSD ドメインデータベースキャッシュに保存するかどうかを指定します。このパラメーターのデフォルト値は <code>false</code> です。LOCAL ドメイン以外のドメイン用にこの値を <code>true</code> にセットすると、オフライン認証が有効になります。
entry_cache_timeout	整数	これはオプションになります。SSSD がポジティブキャッシュヒットをキャッシュする時間を秒単位で指定します。ポジティブキャッシュヒットとは、成功したクエリのことです。
use_fully_qualified_names	ブール値	これはオプションになります。このドメインへの要求が完全修飾型ドメイン名である必要があるかどうかを指定します。 <code>true</code> に設定すると、このドメインへの要求はすべて完全修飾型ドメイン名を使用する必要があります。また、要求からの出力は完全修飾名で表示されます。要求を完全修飾のユーザー名に限定すると、SSSD は競合するユーザー名を持つユーザーのドメイン間で区別ができるようになります。 <i>use_fully_qualified_names</i> が <code>false</code> に設定されると、要求で完全修飾名を使用できますが、出力で表示されるのは簡易バージョンのみになります。 SSSD が解析できるのは、レルム名ではなくドメイン名をベースにした名前だけです。ただし、ドメインとレルムの両方で同一名を使用できます。

3.3.1. LDAP ID プロバイダーを作成する

LDAP ドメインとは、単に SSSD が LDAP ディレクトリーを ID プロバイダーとして (オプションとして認証プロバイダーとしても) 使用するという意味です。SSSD は以下の主要なディレクトリーサービスをサポートします。

- ✧ Red Hat Directory Server
- ✧ OpenLDAP
- ✧ Identity Management (IdM または IPA)
- ✧ Microsoft Active Directory 2008 R2



注記

一般的な LDAP ID プロバイダーで利用可能なパラメーターはすべて、LDAP プロバイダーのサブセットである Identity Management および Active Directory ID プロバイダーでも利用できます。

3.3.1.1. LDAP ドメイン設定用のパラメーター

LDAP ディレクトリーは、ID プロバイダーとしても認証プロバイダーとしても機能することができます。この設定には、LDAP サーバー内のユーザーディレクトリーを特定し、これに接続するための十分な情報が必要ですが、これらの接続パラメーターの定義方法は柔軟なものです。

LDAP サーバーへの接続に使用するユーザーアカウントの指定や、パスワード操作に他の LDAP サーバーを使用するなど、よりきめ細かい制御を提供するための他のオプションも利用可能です。最も一般的なオプションは [表 3.8「LDAP ドメイン設定のパラメーター」](#)に記載してあります。



注記

LDAP ドメイン設定用の man ページ `sssd-ldap(5)` には、他にも多くのオプションが記載されています。

表3.8 LDAP ドメイン設定のパラメーター

パラメーター	説明
<code>ldap_uri</code>	SSSD の接続先となる LDAP サーバーの URI をコンマ区切りの一覧で記入します。このリストは優先順にするので、リスト内の最初のサーバーが最初に試行されます。リストにサーバーを追加しておく、フェイルオーバー保護ができます。これが記入されていない場合は、DNS SRV レコードから検出可能です。
<code>ldap_search_base</code>	LDAP ユーザー操作に使用するベース DN を記入します。
<code>ldap_tls_reqcert</code>	TLS セッションで SSL サーバー証明書をチェックする方法を指定します。以下の 4 つのオプションがあります。 <ul style="list-style-type: none"> ※ never は証明書の要求を無効にします。 ※ allow は証明書を要求しますが、証明書がなくても、または不正な証明書が与えられても通常通りに続行します。 ※ try は証明書を要求して証明書が与えられなくても通常通りに続行します。不正な証明書が与えられた場合は、セッションは終了します。 ※ demand と hard は同じオプションです。有効な証明書を要求して、それがないとセッションは終了します。 デフォルトは hard です。
<code>ldap_tls_cacert</code>	SSSD が認識するすべての CA の CA 証明書を含むファイルの完全なパスとファイル名を記入します。SSSD はこれらの CA が発行した証明書をすべて受理します。これが明示的に与えられていない場合は、OpenLDAP システムのデフォルトを使用します。

パラメーター	説明
ldap_referrals	<p>SSSD が LDAP 参照を使用するかどうかを設定します。LDAP 参照とは、ある LDAP データベースから別の LDAP データベースにクエリを転送することです。SSSD はデータベースレベルの参照とサブツリーの参照をサポートします。同一の LDAP サーバー内の参照には、SSSD はクエリされているエントリの DN を適応させます。異なる LDAP サーバーに転送される参照には、SSSD は DN 上で完全なマッチを行います。この値を true に設定すると参照が有効になり、これがデフォルトです。</p> <p>追跡を試みる時間のために、参照は全体的なパフォーマンスにマイナスの影響を与える場合があります。参照の確認を無効にすると、パフォーマンスは大幅に改善する可能性があります。</p>
ldap_schema	<p>ユーザーエントリーの検索に使用するスキーマのバージョンを設定します。rfc2307、rfc2307bis、ad、ipa が利用可能です。デフォルトは、rfc2307 です。</p> <p>RFC 2307 では、グループオブジェクトは複数値の属性である <i>memberuid</i> を使用しますが、これはそのグループに所属するユーザー名を一覧表示します。RFC 2307bis では、グループオブジェクトは <i>member</i> 属性を使用しますが、これはユーザーまたはグループエントリの完全識別名 (DN) を含んでいます。RFC 2307bis は、<i>member</i> 属性を使用している入れ子グループ (nested groups) を可能にします。これらの異なるスキーマがグループメンバーシップについて異なる定義を使用するため、SSSD で間違った LDAP スキーマを使用すると、適切な権限がある場合でもネットワークリソースの表示と管理の両方に影響します。</p> <p>たとえば RFC 2307bis では、入れ子グループあるいはプライマリ/セカンダリグループを使用すると、すべてのグループが返されます。</p> <div style="border: 1px solid gray; border-radius: 10px; padding: 10px; margin: 10px 0;"> <pre>\$ id uid=500(myserver) gid=500(myserver) groups=500(myserver), 510(myothergroup)</pre> </div> <p>SSSD が RFC 2307 スキーマを使用している場合は、プライマリグループのみが返されます。</p> <p>この設定が影響するのは、SSSD がグループメンバーを決定する方法のみです。実際のユーザーデータは変更されません。</p>
ldap_search_timeout	<p>LDAP 検索がキャンセルされ、キャッシュされた結果が返されるまでの LDAP 検索の時間を秒単位で設定します。LDAP 検索がタイムアウトになると、SSSD は自動的にオフラインモードに切り替わります。</p>

パラメーター	説明
ldap_rfc2307_fallback_to_local_users	LDAP グループメンバーが LDAP ディレクトリーで見つからない場合にローカルシステムユーザー (/etc/passwd) を確認するかどうかを設定します。これにより、LDAP グループにローカルシステムユーザーを追加できるようになります。 これを false に設定すると (デフォルト)、LDAP プロバイダーで id を実行した場合にローカルユーザーはすべて削除されます。SSSD は ID に LDAP ユーザーアカウントのみを使用するためです。
ldap_network_timeout	接続試行が失敗した後に SSSD が LDAP サーバーへ投票を試みる時間を秒単位で設定します。デフォルトは 6 秒です。
ldap_opt_timeout	サーバーから反応がない場合に同期 LDAP 操作を中止するまでの時間を秒単位で設定します。このオプションは、SASL バインドの場合に KDC との通信時のタイムアウトも制御します。デフォルトは 5 秒です。

3.3.1.2. LDAP ID プロバイダーを設定する

LDAP 設定は、ユーザー固有の環境と SSSD の動作によってかなり柔軟になります。以下は LDAP ドメインの一般的な例ですが、SSSD 設定はこれらの例に限定されるものではありません。



注記

ドメインエントリの作成と並行して、SSSD がクエリする新規ドメインを `sssd.conf` ファイル内のドメイン一覧に追加します。例を示します。

```
domains = LOCAL, LDAP1, AD, PROXYNIS
```

例3.2 基本的 LDAP ドメインの設定

LDAP ドメインには、以下の 3 つのものが必要になります。

- ✦ LDAP サーバー
- ✦ 検索ベース
- ✦ 安全な接続を確立する手段

最後の項目は LDAP 環境に左右されます。SSSD は機密情報を取り扱うので、安全な接続が必要になります。この接続は専用の TLS/SSL 接続とするか、Start TLS を使用することができます。

専用の TLS/SSL 接続の場合は、単に LDAPS 接続を使用してサーバーに接続します。そのため、`ldap_uri` オプションの一部として設定されます。

```
# An LDAP domain
[domain/LDAP]
cache_credentials = true
```

```
id_provider = ldap
auth_provider = ldap

ldap_uri = ldaps://ldap.example.com:636
ldap_search_base = dc=example,dc=com
```

Start TLS を使用する場合は、安全でないポート上で動的に安全な接続を確立するために証明書情報を入力する手段が必要になります。これは、`ldap_id_use_start_tls` オプションで Start TLS を使用するようにし、`ldap_tls_cacert` で SSL サーバー証明書を発行した CA 証明書を特定することで達成できます。

```
# An LDAP domain
[domain/LDAP]
cache_credentials = true

id_provider = ldap
auth_provider = ldap

ldap_uri = ldap://ldap.example.com
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
ldap_tls_reqcert = demand
ldap_tls_cacert = /etc/pki/tls/certs/ca-bundle.crt
```

3.3.2. Identity Management (IdM) ID プロバイダーを作成する

Identity Management (IdM または IPA) ID プロバイダーは、一般的な LDAP プロバイダーの拡張機能です。LDAP プロバイダーで利用可能な設定オプションはすべて、IdM プロバイダーでも利用可能です。SSSD を IdM ドメインのクライアントとして機能させ、IdM の機能拡張を可能にするパラメーターもいくつか利用できます。

Identity Management は、ID や認証、アクセス制御ルール、パスワードなど、ドメインの `*_provider` パラメーターすべてのプロバイダーとして機能します。さらに Identity Management のドメイン内には、SELinux ポリシーや自動マウント情報、ホストベースのアクセス制御を管理する設定オプションがあります。IdM ドメイン内のこれら全機能は SSSD 設定との関連付けが可能で、これらのセキュリティ関連のポリシーがシステムユーザーに適用、キャッシュできるようにしています。

例3.3 基本的な IdM プロバイダー

IdM プロバイダーは LDAP プロバイダーのように、ID や認証、アクセス制御などの異なるサービスに対応するように設定することが可能です。

IdM サーバーには、非常に便利な (ただし必須ではない) 以下の 2 つの追加設定があります。

- ※ `id_provider = ipa` 設定では、`ldap_schema = ipa` を使用します。`rfc2307` のデフォルトスキーマの値は、`id_provider = ldap` にのみ使用します。
- ※ クライアントが最初に IdM ドメインに接続する際に、このクライアントの IP アドレスで Identity Management ドメインの DNS サーバーを更新するように SSSD を設定します。

```
[sssd]
domains = local,example.com
...

[domain/example.com]
```

```
id_provider = ipa
ipa_server = ipaserver.example.com
ipa_hostname = ipa1.example.com
auth_provider = ipa
access_provider = ipa
chpass_provider = ipa

# set which schema to use
ldap_schema = ipa

# automatically update IdM DNS records
dyndns_update = true
```

Identity Management は、Linux ドメイン全体でのユーザーのセキュリティポリシーと ID を定義、維持します。これには、アクセス制御ポリシー、SELinux ポリシー、その他のルールが含まれます。IdM ドメインのこれらの要素のいくつかは、SSSD を IdM クライアントとして使用して SSSD と直接対話します。これらの機能は、`sssd.conf` の IdM ドメインエントリーで管理できます。

ほとんどの設定パラメーターは、スキーマ要素の設定 (IdM は固定スキーマを使用するのでほとんどの導入では関係なし) に関連し、変更の必要はありません。実際のところ、IdM の機能でクライアント側の設定を必要とするものは一つもありません。しかし、動作を微調整すると役立つ場合もあります。

例3.4 SELinux での IdM プロバイダー

IdM はシステムユーザー向けの SELinux ユーザーポリシーを定義できるので、SSSD 向けの SELinux プロバイダーとして機能することが可能です。これは、`selinux_provider` パラメーターで設定されます。プロバイダーは `id_provider` 値をデフォルトとしているので、SELinux ルールをサポートすると明示的に設定する必要はありません。しかし、SSSD 内で IdM プロバイダーに対する SELinux サポートを明示的に無効にすると便利な場合があります。

```
selinux_provider = ipa
```

例3.5 ホストベースのアクセス制御での IdM プロバイダー

IdM は、ユーザーが接続に使用しているホストやユーザーが接続しようとしているホストに基づいてサービスやシステム全体へのアクセスを制限して、ホストベースのアクセス制御を定義できます。このルールは、アクセスプロバイダー動作の一部として、SSSD による評価と強制が可能です。

ホストベースのアクセス制御を有効にするには、少なくとも Identity Management サーバーがアクセスプロバイダーである必要があります。

SSSD がホストベースのアクセス制御ルールを評価する方法を設定するには、以下の 2 つのオプションがあります。

- ※ SSSD は、ユーザーが IdM リソースへの接続に使用しているマシン (ソースホスト) を評価できます。これはデフォルトでは無効なので、ルールの対象ホスト部分のみが評価されます。
- ※ SSSD は、指定された間隔でキャッシュにあるホストベースのアクセス制御ルールをリフレッシュできます。

例を示します。


```
access_provider = ipa
ipa_hbac_refresh = 120

# check for source machine rules; disabled by default
```

例3.6 レルム間 Kerberos 信頼との Identity Management

Identity Management (IdM または IPA) は、Active Directory DNS ドメインと Kerberos レルムの信頼できる関係で設定することができます。これにより、Active Directory ユーザーは、Linux システム上のサービスとホストにアクセスできるようになります。

SSSD には、レルム間の信頼で使われる以下の 2 つの設定があります。

- ✦ Kerberos チケットに必要なデータを追加するサービス
- ✦ サブドメインをサポートする設定

Kerberos チケットデータ

Microsoft は、**privileged access certificates** または MS-PAC と呼ばれる特別な認証構造を使用しています。PAC は、Windows ドメイン内の他の Windows クライアントやサーバーへのエンティティを特定する方法として、Kerberos チケットに埋め込まれています。

SSSD には、Kerberos チケットに新たなデータを生成する特別な PAC サービスがあります。Active Directory ドメインを使用している際には、Windows ユーザー用に PAC データを含める必要がある場合があります。その場合には、SSSD で **pac** サービスを有効にします。

```
[sssd]
services = nss, pam, pac
...
```

Windows サブドメイン

通常は、SSSD のドメインエントリーは単一 ID プロバイダーに直接対応します。しかし、IdM のレルム間の信頼があると、IdM ドメインは別のドメインを信頼することができ、ドメインは互いに透明性を持つことになります。SSSD はこの信頼関係に従うことができるので、IdM ドメインが設定済みであれば、いかなる Windows ドメインも自動で SSSD が検索、サポートがすることになります。この場合、SSSD のドメインセクションで設定は不要です。

これは、IdM ドメインセクションに **subdomains_provider** パラメーターを追加することで設定されます。これは実際には、オプションのパラメーターです。サブドメインが発見されれば、SSSD はデフォルトで **ipa** プロバイダータイプを使用します。しかしこのパラメーターの値を **none** に設定することで、サブドメインのフェッチを無効にすることもできます。

```
[domain/IDM]
...
subdomains_provider = ipa
get_domains_timeout = 300
```

3.3.3. Active Directory ID プロバイダーを作成する

最も基本的なドメインのタイプは、LDAP ドメインです。LDAPv3 ディレクトリーサーバーはいかなるものでも、SSSD ドメイン用に LDAP ID プロバイダーとして設定することができます。特別の LDAP サービスの中には追加の特定設定があるものもあり、これらはサービス固有の設定を簡素化したり、サービス固有の機能を提供します。これらの ID プロバイダーの 1 つが Active Directory 向けになります。

[例3.1「シンプルな sssd.conf ファイル」](#)にあるように、SSSD 設定には 3 つの主要セクションがあります。1 つ目では SSSD サービス (`[sss]`) の設定、2 つ目では (`[nss]` や `[pam]` といった) SSSD を ID キャッシュとして使用するシステムサービスの設定、3 つ目では ID ドメイン (`[domain/NAME]`) の設定を行います。

デフォルトでは、本当に必要なのは ID プロバイダーの設定のみです。ID プロバイダーは、他のタイプやサーバーが特定されない場合に、認証、アクセス (権限)、およびパスワードプロバイダーに使用されます。Active Directory は、`ad` オプションを使用することでどんなタイプのプロバイダーとしても設定可能です。

```
[domain/ADEXAMPLE]
id_provider = ad
auth_provider = ad
access_provider = ad
chpass_provider = ad

ad_server = dc1.example.com
ad_hostname = client.example.com
ad_domain = example.com
```

使用する Active Directory サーバーを特定するには、接続情報が必要になります。

上記の基本的な設定に加え、Active Directory ID プロバイダーでは、ローカルシステム上で Windows SID やフェイルオーバーサーバー、ホームディレクトリーなどのアカウント情報用に POSIX 属性やマッピングを使用する方法といった Active Directory 固有の環境や特別な機能を設定することが可能です。

Active Directory プロバイダーでは、LDAP ドメインパラメーターのすべてと Active Directory 固有の設定パラメーターが利用可能です。完全な一覧は、[sssdlldap](#) と [sssad](#) の man ページに記載されています。

Active Directory プロバイダーの設定には、一般的な LDAP プロバイダー設定に多くの利用可能なオプションがあります。`ad` の値を使用すると、Active Directory 用に特定のプロバイダーを設定するためのパラメーターと値を自動的にプルできます。

たとえば、アクセスプロバイダーには以下のようにします。

```
access_provider = ad
```

一般的な LDAP パラメーターを使用すると、この設定は以下ようになります。

```
access_provider = ldap
ldap_access_order = expire
ldap_account_expire_policy = ad
```

`ad` プロバイダータイプを使用することで、上記の設定が暗黙的に設定されます。

3.3.3.1. ローカルシステム上の Active Directory ID について

Active Directory は、ローカルディレクトリーから **グローバルカタログ** にユーザーエントリーと属性を複製することができます。これにより、フォレスト内の他のドメインがその情報を利用できるようになります。SSSD はユーザーとグループの情報についてこのグローバルカタログをチェックするので、情報は単一の Active Directory ドメインやサブドメインに制限されません。SSSD モトポロジー内のすべてのドメインについて全ユーザーデータへのアクセスがあります。

すると SSSD は、ユーザーやグループ情報について Active Directory グローバルカタログにクエリする必要のあるアプリケーションが使用できるようになります。

Windows と Linux では、システムユーザーを処理する方法に固有の構造的違いがあります。また、Active Directory と標準 LDAPv3 ディレクトリーサービスで使用されるユーザースキーマにも構造的な違いがあります。SSSD でシステムユーザーの管理に Active Directory ID プロバイダーを使用する際は、Active Directory スタイルのユーザーを新たな SSSD ユーザーに整合させる必要があります。これには以下の 2 つの方法があります。

- ※ SSSD 上の ID マッピングを使用して、Active Directory セキュリティ ID (SID) と Linux 上の生成済み UID との間のマップを作成する。

ID マッピングは新たなパッケージや Active Directory 上での設定を必要としないため、ほとんどの環境で最も簡単なオプションです。

- ※ Services for Unix を使用して Windows ユーザーおよびグループエントリー上に POSIX 属性を挿入し、これらの属性を PAM/NSS にプルする。

これには Active Directory 環境内でさらなる設定と情報が必要になりますが、特定の UID/GID の値 (および他の POSIX 属性) についてより多くの管理者制御が可能になります。

3.3.3.1.1. セキュリティ ID マッピングについて

3.3.3.1.1.1. ID マッピングのメカニズム

Linux/Unix システムはローカルのユーザー ID 番号とグループ ID 番号を使ってシステム上のユーザーを識別します。これらの UID:GID 番号は、**501:501** といった単純な整数です。これらの番号は大規模な Linux/Unix ドメインの一部のシステムでも常にローカルで作成、管理されるので、単純なものになります。

Microsoft Windows と Active Directory は、異なるユーザー ID 構造を使ってユーザーやグループ、マシンを識別します。各 ID は、セキュリティバージョンや発行権限のタイプ、マシン、ID 自体を識別する異なるセグメントで構築されています。例を示します。

```
S-1-5-21-3623811015-3361044348-30300820-1013
```

3 つ目から 6 つ目のブロックは、マシン識別子です。

```
S-1-5-21-3623811015-3361044348-30300820-1013
```

最後のブロックは、特定エンティティを識別する **相対識別子 (RID)** です。

```
S-1-5-21-3623811015-3361044348-30300820-1013
```

SSSD には、常に ID 番号の範囲が割り当てられています。(これはローカルの範囲なので、すべてのマシンで同じものになります。)

```
|-----|
|               |
| minimum ID           | max ID
```

この範囲は (デフォルトで) 1 万セクションに分けられ、各セクションには 20 万の ID が割り当てられます。

```
| slice 1 | slice 2 | ... |
|-----|-----|-----|
|               |               |               |
| minimum ID           | max ID
```

新たな Active Directory ドメインが検出されると、SID がハッシュ化されます。すると SSSD はこのハッシュのモジュールと利用可能なセクションの数で、Active Directory ドメインにどの ID セクションを割り当てるかを決めます。これは、ID セクション割り当て方法としては一貫性のあるものなので、ほとんどのクライアントマシン上で同一の Active Directory ドメインに同一の ID 範囲が割り当てられます。

Active Directory domain 1	Active Directory domain 2	...
slice 1	slice 2	...
minimum ID		max ID

注記

ID セクションの割り当て方法には一貫性がありますが、ID マッピングは Active Directory ドメインがクライアントマシン上で発生する順序に基づいています。このため、すべての Linux クライアントマシン上で ID 範囲の割り当てに一貫性が保たれない場合もあります。一貫性が必要な場合は、ID マッピングを無効にし、明示的な POSIX 属性の使用を検討してください。

3.3.3.1.1.2. ID マッピングパラメーター

ID マッピングは 2 つのパラメーターで有効になります。1 つ目でマッピングを有効にし、2 つ目で適切な Active Directory ユーザースキーマを読み込みます。

```
ldap_id_mapping = True
ldap_schema = ad
```

注記

ID マッピングが有効になると、*uidNumber* および *gidNumber* 属性は無視されます。これにより、手動で割り当てられた値はすべて止められます。**何らかの** 値が手動で割り当てられる必要がある場合は、**すべての** 値を手動で割り当てて、ID マッピングを無効にする必要があります。

3.3.3.1.1.3. ユーザーのマッピング

Active Directory ユーザーのローカルシステムサービスへの初回ログイン時には、そのユーザーのエントリーが SSSD キャッシュに作成されます。リモートユーザーはシステムユーザーのように設定されます。

- ※ ユーザーの SID およびそのドメインの ID 範囲に基づいてシステム UID が作成されます。
- ※ ユーザーに GID が作成されます。これは UID と同一のものです。
- ※ ユーザーにプライベートグループが作成されます。
- ※ `sssd.conf` ファイルのホームディレクトリ形式に基づいて、ホームディレクトリが作成されます。
- ※ システムデフォルトまたは `sssd.conf` ファイル内の設定に基づいて、シェルが作成されます。
- ※ ユーザーが Active Directory 内のいずれかのグループに所属する場合、SID を使って SSSD が Linux 上のこれらのグループにユーザーを追加します。

3.3.3.1.2. SSSD および POSIX 属性について

Active Directory は、*uidNumber*、*gidNumber*、*unixHomeDirectory*、および *loginShell* といった POSIX 属性を作成、保存するように設定できます。すべてのユーザー属性と同じように、これらはまずローカルドメインで保存されますが、グローバルカタログに複製することが可能です。グローバルカタログに複製されると、属性は ID 情報に SSSD を使用する SSSD とアプリケーションで利用可能となります。



重要

SSSD が直接 POSIX 属性を使用する場合は、属性は Active Directory グローバルカタログに公開される必要があります。SSSD はユーザー情報についてグローバルカタログにクエリを行います。

POSIX 属性が Active Directory で既に定義されている場合は、「[セキュリティ ID マッピングについて](#)」で説明されている SID/UID マッピングの使用は推奨されません。UID と GID の番号は既に定義されており、マッピングをすると新たに異なる番号が作成されます。この状況における最善策は、Active Directory で定義されている UID と GID の番号を使用し、SSSD が管理するローカルの Linux アカウントにそれらを適用することです。

既存の POSIX 属性を使用するには、以下の 2 つを設定する必要があります。

- ✦ POSIX 属性を Active Directory のグローバルカタログに公開します。
- ✦ ID マッピング (Active Directory ドメインエントリ内の `ldap_id_mapping`) を SSSD 内で無効にします。

```
ldap_id_mapping = False
```

3.3.3.1.3. Active Directory ユーザーおよび範囲取得検索

Microsoft Active Directory には *MaxValRange* の属性があり、これは複数の値の属性に返す値の制限数を設定するものです。これは、**範囲取得** 検索の拡張機能です。基本的にこれは複数のミニ検索を実行し、すべての一致が返されるまでそれぞれの検索は特定の範囲内での結果のサブセットを返します。

たとえば、*member* 属性を検索する際に、各エントリに複数の値があり、その属性を持つ複数のエントリがある場合があります。2000 件 (またはそれ以上) の一致結果があった場合、*MaxValRange* は一度に表示される数を制限します。これが、値の範囲です。特定の属性には新たなフラグのセットが付けられ、その結果がセット内のどの範囲にあるかを示します。

```
attribute:range=low-high:value
```

たとえば、検索結果の 100 件から 500 件を表示するには、以下のようにします。

```
member;range=99-499: cn=John Smith...
```

これは Microsoft ドキュメンテーションの <http://msdn.microsoft.com/en-us/library/cc223242.aspx> で説明されています。

SSSD は、Active Directory プロバイダーを使った範囲取得を新たな設定なしにユーザーおよびグループ管理の一部としてサポートします。

しかし、`ldap_user_search_base` のような検索設定で利用可能な LDAP プロバイダー属性の中には、範囲取得との効率がよくないものもあります。Active Directory プロバイダードメインで検索ベースを設定する際には注意して、どの検索が範囲取得を開始するかを考慮してください。

3.3.3.1.4. Linux クライアントと Active Directory DNS サイト

SSSD は、ローカルの Linux システムを大規模 Active Directory 環境に接続します。これには、Active Directory フォレスト内で可能な設定を SSSD が認識しており、SSSD がこれと機能して Linux クライアントがスムーズに統合される必要があります。

Active Directory フォレストは非常に大型で、数多くの異なるドメインコントローラー、ドメインおよびサブドメイン、および物理サイトで構成される場合があります。クライアントのパフォーマンスを高めるために、Active Directory は特別の名前が付けられた DNS レコードを使って同一ドメイン内ではあるものの異なる物理的位置にあるドメインコントローラーを特定します。クライアントは最も近いドメインコントローラーに接続します。



注記

Microsoft では、DNS と Active Directory を連携させる方法を技術要約で説明しています。<http://technet.microsoft.com/es-es/library/cc759550%28v=ws.10%29.aspx>

Active Directory は通常の DNS SRV レコードを拡張して、ドメインコントローラーの特定の物理的位置またはサイトを特定します。(SSSD などの) クライアントは、それ自体のサイト設定をベースにどのドメインコントローラーを使用するか判断できます。

SSSD は、まず Active Directory ドメインにサイト設定についてクエリを行い、DNS レコードにドメインコントローラーについてクエリを行うことでどのドメインコントローラーを使用するか判断します。

1. SSSD は Active Directory ドメインへの接続を試行し、通常の DNS 発見で利用可能なドメインコントローラーをルックアップします。
2. プライマリおよびフォールバックサーバーの一覧を取得します。
3. SSSD が特別な CLDAP ping をドメインコントローラーに送信します。ping は実際には LDAP 検索で、DNS ドメイン、ドメイン SID、およびバージョンを検索します。

```
(&(&(DnsDomain=ad.domain)(DomainSid=S-1-5-21-1111-2222-3333))
(NtVer=0x01000016))
```

これはクライアントのサイトが設定されている場合、そのサイトについての情報を取得するために使用されません。

4. サイトがクライアント用に設定されている場合、返信にはプライマリサーバー用の拡張版の DNS SRV レコードが含まれ、これにはサイト名 (**site-name._sites.**) が含まれます。

```
_service._protocol.site-name._sites.domain.name
```

バックアップサーバーレコードも標準 SRV レコードとして送信されます。

```
_service._protocol.domain.name
```

サイトが設定されていない場合は、すべてのプライマリおよびバックアップサーバー用に標準 SRV レコードが送信されます。

3.3.3.2. ID マッピングを使って Active Directory ドメインを設定する

Active Directory ドメインを設定する際には、すべてのプロバイダー (ID、アクセス、パスワード) に **ad** の値を使用するのが最もシンプルな設定です。また、ユーザーおよびグループエントリ用にデフォルトの RFC 2307 を使用する代わりに、ネイティブの Active Directory スキーマを読み込みます。

他の設定は、一般的な LDAP プロバイダー設定 ([sssd-ldap](#)) と Active Directory 固有の設定 ([sssd-ad](#)) にあります。これには、特定のユーザーまたはグループサブツリー用の LDAP フィルターの設定、認証用のフィルター、およびアカウント設定用の値が含まれます。追加設定のいくつかは「[さらなる設定例](#)」で説明されています。

1. Active Directory と Linux システムの両方の環境が適切に設定されていることを確認します。
 - ※ サービス発見に SSSD が使用されている場合は特に、名前解決を適切に設定する必要があります。
 - ※ Kerberos が正常に機能するために、両方のシステムのクロックが同期されている必要があります。
2. Linux システムを Active Directory クライアントとして設定し、Active Directory ドメイン内で登録します。これは、Kerberos と Samba の両サービスを Linux システムで設定することで行います。
 - a. Kerberos が Active Directory Kerberos レルムを使用するように設定します。

- a. Kerberos クライアント設定ファイルを開きます。

```
[root@server ~]# vim /etc/krb5.conf
```

- b. **[logging]** および **[libdefaults]** セクションを設定し、Active Directory レルムに接続するようにします。

```
[logging]
default = FILE:/var/log/krb5libs.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = true
dns_lookup_kdc = true
ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
forwardable = true
```

SSSD で autodiscovery が使用されていない場合は、**[realms]** および **[domain_realm]** セクションを設定して Active Directory サーバーを明示的に定義します。

- b. Samba サーバーが Active Directory サーバーに接続するように設定します。
 - a. Samba 設定ファイルを開きます。

```
[root@server ~]# vim /etc/samba/smb.conf
```

- b. **[global]** セクションで Active Directory ドメイン情報を設定します。

```
[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads
```

c. Linux マシンを Active Directory ドメインに追加します。

a. Windows 管理ユーザー用の Kerberos 認証情報を取得します。

```
[root@server ~]# kinit Administrator
```

b. `net` コマンドでマシンをドメインに追加します。

```
[root@server ~]# net ads join -k  
Joined 'server' to dns domain 'example.com'
```

これで新規の keytab ファイル `/etc/krb5.keytab` が作成されます。

システム用の鍵を一覧表示し、ホストプリンシパルがあることを確認します。

```
[root@server ~]# klist -k
```

3. 必要な場合は `oddjob-mkhomedir` パッケージをインストールして、SSSD が Active Directory ユーザー用にホームディレクトリーを作成できるようにします。

```
[root@server ~]# yum install oddjob-mkhomedir
```

4. `authconfig` を使用してシステム認証のために SSSD を有効にします。`--enablemkhomedir` オプションを使用し、SSSD によるホームディレクトリー作成を有効にします。

```
[root@server ~]# authconfig --enablesssd --enablesssdauth --enablemkhomedir --update
```

5. SSSD 設定ファイルを開きます。

```
[root@rhel-server ~]# vim /etc/sss/sss.conf
```

6. Active Directory ドメインを設定します。

a. `[sss]` セクションで Active Directory ドメインをアクティブなドメイン一覧に追加します。これは、SSSD 設定ファイルの `[domain/NAME]` で設定されるドメインエントリー名です。

また、`pac` もサービス一覧に追加します。これにより、SSSD は Active Directory ドメインとの通信に使用されるチケットで MS-PAC 情報を設定、使用できるようになります。

```
[sss]  
config_file_version = 2  
domains = ad.example.com  
services = nss, pam, pac
```

b. Active Directory ドメイン用に、ファイルの下部で新規のドメインセクションを作成します。このセクションの形式は `domain/NAME` で、たとえば `domain/ad.example.com` となります。各プロバイダーの値は `ad` に設定し、特定の Active Directory インスタンスの接続先となる接続情報を提供します。

```
[domain/ad.example.com]  
id_provider = ad  
ad_server = adserver.example.com  
ad_hostname = client.example.com
```



```
auth_provider = ad
chpass_provider = ad
access_provider = ad
```

- c. 認証情報のキャッシュを有効にします。これでユーザーは、Active Directory ドメインが利用できない場合でも、キャッシュされた情報を使ってローカルシステムにログインできるようになります。

```
cache_credentials = true
```

- d. アクセス制御を設定します。

```
ldap_access_order = expire
ldap_account_expire_policy = ad
```

7. SSH サービスを再起動して新規の PAM 設定を読み込みます。

```
[root@server ~]# systemctl restart sshd.service
```

8. SSSD サービスを起動します。

```
[root@rhel-server ~]# systemctl start sssd.service
```

3.3.3.3. POSIX 属性を使って Active Directory ドメインを設定する

Active Directory 定義の POSIX 属性を SSSD で使用するには、属性をグローバルカタログに複製する必要があります。これには、Active Directory ドメインで追加設定が必要になります。さらに、ID マッピングを SSSD で無効にし、新たな設定をローカルで作成するのではなく、Active Directory からの POSIX 属性が使用されるようにする必要があります。

他の設定は、一般的な LDAP プロバイダー設定 ([sssd-ldap](#)) と Active Directory 固有の設定 ([sssd-ad](#)) にあります。これには、特定のユーザーまたはグループサブツリー用の LDAP フィルターの設定、認証用のフィルター、およびアカウント設定用の値が含まれます。追加設定のいくつかは「[さらなる設定例](#)」で説明されています。

- Active Directory と Linux システムの両方の環境が適切に設定されていることを確認します。
 - ※ サービス発見に SSSD が使用されている場合は特に、名前解決を適切に設定する必要があります。
 - ※ Kerberos が正常に機能するために、両方のシステムのクロックが同期されている必要があります。
- Active Directory ドメインで POSIX 属性がグローバルカタログに複製されるように設定します。
 - Identity Management for UNIX Components** をすべてのプライマリおよび子ドメインコントローラーにインストールします。詳細は、以下の Microsoft の資料で確認できます。<http://technet.microsoft.com/en-us/library/cc731178.aspx>
これで、POSIX 属性と関連スキーマがユーザーアカウントで利用可能になります。これらの属性は、**Properties** メニューの **UNIX Attributes** タブで利用できます。
 - Active Directory Schema Snap-in をインストールして、グローバルカタログに複製する属性を追加します。これは、以下の Microsoft 資料で説明されています。<http://technet.microsoft.com/en-us/library/cc755885%28v=ws.10%29.aspx>
 - スキーマ複製に関する詳細情報は、以下の Microsoft 資料で説明されています。<http://support.microsoft.com/kb/248717>

関連する POSIX 属性 (*uidNumber*、*gidNumber*、*unixHomeDirectory*、および *loginShell*) については、**Properties** メニューを開き、**Replicate this attribute to the Global Catalog** チェックボックスを選択して **OK** をクリックします。

3. Linux システムを Active Directory クライアントとして設定し、Active Directory ドメイン内で登録します。これは、Kerberos と Samba の両サービスを Linux システムで設定することで行います。

- a. Kerberos が Active Directory Kerberos レルムを使用するように設定します。

- a. Kerberos クライアント設定ファイルを開きます。

```
[root@server ~]# vim /etc/krb5.conf
```

- b. **[logging]** および **[libdefaults]** セクションを設定し、Active Directory レルムに接続するようにします。

```
[logging]
default = FILE:/var/log/krb5libs.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = true
dns_lookup_kdc = true
ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
forwardable = true
```

SSSD で *autodiscovery* が使用されていない場合は、**[realms]** および **[domain_realm]** セクションを設定して Active Directory サーバーを明示的に定義します。

- b. Samba サーバーが Active Directory サーバーに接続するように設定します。

- a. Samba 設定ファイルを開きます。

```
[root@server ~]# vim /etc/samba/smb.conf
```

- b. **[global]** セクションで Active Directory ドメイン情報を設定します。

```
[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads
```

- c. Linux マシンを Active Directory ドメインに追加します。

- a. Windows 管理ユーザー用の Kerberos 認証情報を取得します。

```
[root@server ~]# kinit Administrator
```

- b. `net` コマンドでマシンをドメインに追加します。

```
[root@server ~]# net ads join -k
Joined 'server' to dns domain 'example.com'
```

これで新規の `keytab` ファイル `/etc/krb5.keytab` が作成されます。

- c. システム用の鍵を一覧表示し、ホストプリンシパルがあることを確認します。

```
[root@server ~]# klist -ke
```

- d. `ldapsearch` を使用して、ユーザーがグローバルカタログを検索できるかテストします。

```
[root@server ~]# ldapsearch -H ldap://server.ad.example.com:3268
-Y GSSAPI -N -b "dc=ad,dc=example,dc=com" "(&(objectClass=user)
(sAMAccountName=aduser))"
```

4. `sssd-ad` パッケージをインストールします。

```
[root@server ~]# yum install sssd-ad
```

5. SSSD 設定ファイルを開きます。

```
[root@rhel-server ~]# vim /etc/sss/sss.conf
```

6. Active Directory ドメインを設定します。

- a. `[sss]` セクションで Active Directory ドメインをアクティブなドメイン一覧に追加します。これは、SSSD 設定ファイルの `[domain/NAME]` で設定されるドメインエントリー名です。

また、`pac` もサービス一覧に追加します。これにより、SSSD は Active Directory ドメインとの通信に使用されるチケットで MS-PAC 情報を設定、使用できるようになります。

```
[sss]
config_file_version = 2
domains = ad.example.com
services = nss, pam, pac
```

- b. Active Directory ドメイン用に、ファイルの下部で新規のドメインセクションを作成します。このセクションの形式は `domain/NAME` で、たとえば `domain/ad.example.com` となります。各プロバイダーの値は `ad` に設定し、特定の Active Directory インスタンスの接続先となる接続情報を提供します。

```
[domain/ad.example.com]
id_provider = ad
ad_server = adserver.example.com
ad_hostname = client.example.com
auth_provider = ad
chpass_provider = ad
access_provider = ad
```

- c. ID マッピングを無効にします。これは SSSD に Windows SID に基づいて UID:GID 番号を作成するのではなく、POSIX 属性をグローバルカタログで検索するよう指示します。

```
# disabling ID mapping
ldap_id_mapping = False
```

- d. ホームディレクトリーとログインシェルがユーザーアカウント内に設定されている場合は、これらの行をコメントアウトして、SSSD がテンプレートをベースに属性を作成するのではなく、POSIX 属性を使用するように設定します。

```
# Comment out if the users have the shell and home dir set on the AD
side
#default_shell = /bin/bash
#fallback_homedir = /home/%d/%u
```

- e. Microsoft Active Directory では、各アカウントは 2 つの Kerberos プリンシパルを持つことができます。(client.ad.example.com@AD.EXAMPLE.COM のような) ホストプリンシパルが利用できない場合、`ldap_sasl_authid` の行でコメント解除を行い、使用するホストプリンシパルを設定します。

```
# Uncomment and adjust if the default principal SHORTNAME$@REALM is not
available
# ldap_sasl_authid = host/client.ad.example.com@AD.EXAMPLE.COM
```

- f. Active Directory ユーザーに短い名前か完全修飾ユーザー名を使用するかを設定します。複雑なトポロジーでは、曖昧性を解消するために完全修飾名の使用が必要になる場合があります。

```
# Comment out if you prefer to user shortnames.
use_fully_qualified_names = True
```

- g. 認証情報のキャッシュを有効にします。これでユーザーは、Active Directory ドメインが利用できない場合でも、キャッシュされた情報を使ってローカルシステムにログインできるようになります。

```
cache_credentials = true
```

- h. アクセス制御を設定します。

```
ldap_access_order = expire
ldap_account_expire_policy = ad
```

7. SSSD 設定ファイルのファイルパーミッションと所有者を設定します。

```
[root@server ~]# chown root:root /etc/sssds/sssds.conf
[root@server ~]# chmod 0600 /etc/sssds/sssds.conf
[root@server ~]# restorecon /etc/sssds/sssds.conf
```

8. SSSD サービスを起動します。

```
[root@server ~]# systemctl start sssds.service
```

9. 必要な場合は `oddjob-mkhomedir` パッケージをインストールして、SSSD が Active Directory ユーザー用にホームディレクトリーを作成できるようにします。

```
[root@server ~]# yum install oddjob-mkhomedir
```

10. `authconfig` を使用してシステム認証のために SSSD を有効にします。`--enablemkhomedir` オプションを使用し、SSSD によるホームディレクトリー作成を有効にします。

```
[root@server ~]# authconfig --enablesssd --enablesssdauth --enablemkhomedir --update
```

11. SSH サービスを再起動して新規の PAM 設定を読み込みます。

```
[root@server ~]# systemctl restart sshd.service
```

上記の手順で `authconfig` を使用することで、NSS および PAM の設定ファイルが SSSD を ID ソースとして使用するよう自動的に設定されます。

たとえば、`nsswitch.conf` ファイルには SSSD (`sss`) がユーザー、グローバル、およびサービス上のソースとして追加されます。

```
passwd:      files sss
shadow:     files sss
group:      files sss
...
services:  files sss
...
netgroup:   files sss
```

別の `pam.d` ファイルが `/etc/pam.d/system-auth` と `/etc/pam.d/password-auth` のファイル内の `pam_unix.so` 行の下に `pam_sss.so` モジュールの行を追加します。

```
auth      sufficient    pam_sss.so use_first_pass
...
account   [default=bad success=ok user_unknown=ignore] pam_sss.so
...
password  sufficient    pam_sss.so use_authtok
...
session   optional      pam_mkhomedir.so
session   optional      pam_sss.so
```

3.3.3.4. LDAP ドメインとして Active Directory を設定する

Active Directory はタイプ固有の ID プロバイダーとして設定できますが、Kerberos 認証プロバイダーとともに純然たる LDAP ID プロバイダーとして設定することも可能です。

1. SSSD は、SASL を使って Active Directory サーバーに接続することが推奨されます。これにより、ローカルホストは Linux ホスト上で **Windows ドメイン** 用のサービス keytab を備えている必要があります。

この keytab は Samba を使って作成できます。

- a. Active Directory レルムを使用するよう `/etc/krb5.conf` ファイルを設定します。

```
[logging]
default = FILE:/var/log/krb5libs.log

[libdefaults]
default_realm = AD.EXAMPLE.COM
dns_lookup_realm = true
dns_lookup_kdc = true
```

```

ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
forwardable = true

[realms]
# Define only if DNS lookups are not working
# AD.EXAMPLE.COM = {
#   kdc = server.ad.example.com
#   admin_server = server.ad.example.com
# }

[domain_realm]
# Define only if DNS lookups are not working
# .ad.example.com = AD.EXAMPLE.COM
# ad.example.com = AD.EXAMPLE.COM

```

- b. Samba 設定ファイル `/etc/samba/smb.conf` を Windows Kerberos レalm に向けるよう設定します。

```

[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads

```

- c. `net ads` コマンドを実行して、管理者プリンシパルとしてログインします。この管理者アカウントにはマシンを Windows ドメインに追加することができる権限が必要ですが、ドメイン管理者権限は必要ありません。

```
[root@server ~]# net ads join -U Administrator
```

- d. `net ads` コマンドを再度実行して、ホストマシンをドメインに追加します。これは、ホストプリンシパル (`host/FQDN`) またはオプションで NFS サービス (`nfs/FQDN`) で実行できます。

```
[root@server ~]# net ads join createupn="host/rhel-server.example.com@AD.EXAMPLE.COM" -U Administrator
```

2. Services for Unix パッケージが Windows サーバー上にインストールされていることを確認します。
3. SSSD で使用する Windows ドメインを設定します。
 - a. Windows マシンで `サーバーマネージャー` を開きます。
 - b. Active Directory ドメインサービスの役割を作成します。
 - c. `ad.example.com` といった新規ドメインを作成します。
 - d. Identity Management for UNIX サービスを Active Directory ドメインサービスの役割に追加します。設定ではドメイン名に Unix NIS ドメインを使用してください。
4. Active Directory サーバー上で Linux ユーザーのグループを作成します。

- a. **管理ツール** を開いて、**Active Directory ユーザーとコンピューター** を選択します。
 - b. Active Directory ドメインの **ad.example.com** を選択します。
 - c. **ユーザー** タブで右クリックして、**新しいグループの作成** を選択します。
 - d. 新しいグループに **unixusers** と名前を付けて保存します。
 - e. **unixusers** グループエントリーをダブルクリックして、**ユーザー** タブを開きます。
 - f. **Unix 属性** タブを開きます。
 - g. **ad.example.com** 用に設定された NIS ドメインに NIS ドメインを設定し、オプションでグループ ID (GID) 番号を設定します。
5. ユーザーを Unix グループの一部となるよう設定します。
- a. **管理ツール** を開いて、**Active Directory ユーザーとコンピューター** を選択します。
 - b. Active Directory ドメインの **ad.example.com** を選択します。
 - c. **ユーザー** タブで右クリックして、**新しいユーザーの作成** を選択します。
 - d. 新しいユーザーに **aduser** と名前を付けて、**ユーザーは次回のログオン時にパスワード変更が必要** と **Lock account** のチェックボックスにチェックマークがついていないことを確認します。
- ユーザーを保存します。
- e. **aduser** ユーザーエントリーをダブルクリックして、**Unix 属性** タブを開きます。Unix 設定が、Active Directory ドメインと **unixgroup** グループのものと合致していることを確認します。
 - ※ NIS ドメインが Active Directory ドメイン用に作成されたものであること
 - ※ UID
 - ※ ログインシェルが **/bin/bash** であること
 - ※ ホームディレクトリーが **/home/aduser** であること
 - ※ プライマリーグループ名が **unixusers** であること

注記

大規模ディレクトリーでのパスワードのルックアップは、要求 1 件あたり数秒かかります。最初のユーザールックアップは、LDAP サーバーへのコールです。インデックス化されていない検索は、よりリソース集約的なのでインデックス化されている検索よりも時間が長かかります。これは、サーバーがディレクトリー内のすべてのエントリーに関して一致するかチェックするためです。ユーザールックアップを迅速化するには、SSSD が検索する以下の属性をインデックス化します。

- ※ uid
- ※ uidNumber
- ※ gidNumber
- ※ gecos

グループメンバーを無視することに関する情報は、[「グループメンバーを無視する」](#)を参照してください。

6. Linux システム上で SSSD ドメインを設定します。

```
[root@rhel-server ~]# vim /etc/sss/sss.conf
```

LDAP プロバイダーパラメーターの完全な一覧は、`sss-ldap(5)` の man ページを参照してください。

例3.7 Services for Unix による Active Directory 2008 R2 ドメイン

```
[sss]
config_file_version = 2
domains = ad.example.com
services = nss, pam

...

[domain/ad.example.com]
cache_credentials = true

# for performance
ldap_referrals = false

id_provider = ldap
auth_provider = krb5
chpass_provider = krb5
access_provider = ldap

ldap_schema = rfc2307bis

ldap_sasl_mech = GSSAPI
ldap_sasl_authid = host/rhel-server.example.com@AD.EXAMPLE.COM

#provide the schema for services for unix
ldap_schema = rfc2307bis

ldap_user_search_base = ou=user accounts,dc=ad,dc=example,dc=com
ldap_user_object_class = user
ldap_user_home_directory = unixHomeDirectory
ldap_user_principal = userPrincipalName

# optional - set schema mapping
# parameters are listed in sss-ldap
ldap_user_object_class = user
ldap_user_name = sAMAccountName

ldap_group_search_base = ou=groups,dc=ad,dc=example,dc=com
ldap_group_object_class = group

ldap_access_order = expire
ldap_account_expire_policy = ad

krb5_realm = AD-REALM.EXAMPLE.COM
# required
krb5_canonicalize = false
```

7. SSSD を再起動します。


```
[root@rhel-server ~]# systemctl restart sssd.service
```

3.3.3.5. さらなる設定例

3.3.3.5.1. アカウント設定

Linux ユーザーの場合、新規ユーザーにはデフォルトで特定のシステム設定がなされます。たとえば、`pam_oddjob_mkhomedir.so` ライブラリーは定義された場所に自動的にホームディレクトリーを作成します。

このようなシステム設定は Windows ユーザーアカウントでは設定されない場合や、Linux システムとは互換性のないものに設定される場合もあります。以下の 2 つのエリアがそれに該当します。

- ✦ ユーザーのホームディレクトリー
- ✦ デフォルトのユーザーシェル

3.3.3.5.1.1. ユーザーのホームディレクトリーを設定する

Red Hat Enterprise Linux には PAM ライブラリー (`pam_oddjob_mkhomedir.so`) があり、これはユーザーの初回ログイン時に自動的にユーザーディレクトリーを作成します。これには、Linux システムでの Active Directory ユーザーの初回ログイン時が含まれます。

SSSD では、ユーザーディレクトリーの形式は ID プロバイダーから取得されます。ID プロバイダーのホームディレクトリーの形式が Linux システムとは異なる場合や値が提供されない場合は、SSSD の設定で指定されているテンプレートを使ってホームディレクトリーを作成するように設定できます。テンプレートは NSS サービスでグローバルに設定するか、ドメインごとに設定することができます。

パラメーターには以下の 2 つが使用できます。

- ✦ `fallback_homedir` は、ID プロバイダーがテンプレートを提供しない場合にテンプレートを提供します。
- ✦ `override_homedir` は、ID プロバイダーで設定されている情報にかかわらず、使用するテンプレートを設定します。

パラメーターに関する詳細情報は、[表3.2「SSSD \[nss\] 設定のパラメーター」](#)を参照してください。

```
[nss]
fallback_homedir = /home/%u

...

[domain/AEXAMPLE]
id_provider = ad
ad_server = adserver.example.com
ad_hostname = client.example.com
auth_provider = ad
...
override_homedir = /home/%d/%u
```

3.3.3.5.1.2. ユーザーシェルを設定する

デフォルトでは、SSSD は ID プロバイダーからユーザーシェルに関する情報を取得しようとします。Active Directory と LDAPv3 スキーマでは、これは `loginShell` 属性で定義されます。ただしこれはオプションの属性なので、すべてのユーザーについて定義されない場合もあります。Active Directory ユーザーの場合、定義済みのログインシェルが Linux システム上で許可されない場合もあります。

SSSD 設定では、シェル処理には以下のような方法があります。

- ※ シェルが提供されない場合のフォールバックの値を設定する (`shell_fallback`)
- ※ 許可されたシェルまたはブラックリスト化されたシェルの一覧を設定する (`allowed_shells` と `vetoed_shells`)
- ※ デフォルト値を設定する (`default_shell`)
- ※ ID プロバイダーで別の値が提供されても使用する値を設定する (`override_shell`)



注記

`allowed_shells`、`vetoed_shells`、および `shell_fallback` の設定はドメインごとではなくグローバル設定のみが可能です。ただしこれらのパラメーターはローカルシステムユーザーには影響せず、SSSD ID プロバイダーから取得された外部のユーザーにのみ影響します。`/bin/bash` といった一般的な設定の使用がほとんどの外部ユーザーには有用です。

デフォルト値はドメインごとに設定可能ですが、(シェルに対するホワイトリストまたはブラックリストといった) 値のいくつかは NSS サービス設定でグローバル設定にする必要があります。例を示します。

```
[nss]
shell_fallback = /bin/sh
allowed_shells = /bin/sh, /bin/zsh, /bin/bash
vetoed_shells = /bin/ksh
...

[domain/AEXAMPLE]
id_provider = ad
ad_server = adserver.example.com
ad_hostname = client.example.com
auth_provider = ad
...
default_shell = /bin/zsh
```

3.3.3.5.2. ダイナミック DNS 更新を有効にする

IdM と Active Directory を使うと、クライアントは DNS レコードを自動的にリフレッシュできるようになります。また、クライアントの DNS レコードをアクティブに維持して、これらが確実に更新されるようにします。これには、タイムアウト (エイジング) や非アクティブなレコードの削除 (スキベンジング) が含まれます。

SSSD では、Linux システムは DNS レコードをリフレッシュすることで Windows クライアントを再現することができます。これにより、レコードが非アクティブとマークされ、DNS レコードから削除されることも防ぎます。動的 DNS 更新が有効になると、クライアントの DNS レコードは以下の場合にリフレッシュされます。

- ※ ID プロバイダーがオンラインになる時 (毎回)
- ※ Linux システムの再起動時 (毎回)
- ※ 定期的 (オプション設定)


注記

これは DHCP リースと同じ間隔に設定することができ、その場合は、リースの更新後に Linux クライアントが更新されることになります。

DNS は DNS 用 Kerberos/GSSAPI (GSS-TSIG) を使用して Active Directory サーバーに送信されます。つまり、セキュアな接続のみを有効にする必要があります。

動的 DNS 設定はドメインごとに設定されます。例を示します。

```
[domain/ad.example.com]
id_provider = ad
ad_server = adserver.example.com
ad_hostname = client.example.com
auth_provider = ad
chpass_provider = ad
access_provider = ad

ldap_schema = ad

dyndns_update = true
dyndns_refresh_interval = 43200
dyndns_update_ptr = true
dyndns_ttl = 3600
```

表3.9 動的 DNS 更新のオプション

オプション	説明	形式
dyndns_update	DNS サーバーをクライアント IP アドレスで動的に更新するかどうかを設定します。これには安全な更新が必要となります。有効とする他の動的 DNS 設定はすべて true に設定する必要があります。デフォルトは true です。	ブール値
dyndns_ttl	クライアントの DNS レコードの Time-to-Live (有効期限) を設定します。デフォルト値は 3600 秒です。	整数
dyndns_refresh_interval	プロバイダーがオンラインになる時に、更新のほかに自動 DNS 更新を実行する頻度を設定します。デフォルト値は 86400 秒 (24 時間) です。	整数
dyndns_update_ptr	クライアントが DNS レコードを更新する時に PTR レコードを更新するかどうかを設定します。デフォルト値は true です。	ブール値

オプション	説明	形式
dyndns_iface	その IP アドレスが動的 DNS 更新に使用されるインターフェイスを選択します。この設定が使用可能なのは dyndns_update が true の場合のみで、これはオプションになります。デフォルトでは、AD LDAP 接続の IP アドレスが使用されます。	文字列
dyndns_force_tcp	nsupdate ユーティリティが DNS サーバーとの通信にデフォルトで TCP プロトコルを使用するかどうかを指定します。デフォルトでは false に設定されており、 nsupdate は使用するプロトコルを選択します。	ブール値

3.3.3.5.3. アクセス制御にフィルターを使用する

Active Directory アクセスプロバイダーが存在することから、Active Directory は承認情報に関するソースとして使用されます。実際にはこれは、複数の一般的な LDAP パラメーターを単一の設定パラメーターにする簡単な方法になります。Active Directory プロバイダーを設定します。

```
access_provider = ad
```

これは、以下のように複数の異なる LDAP パラメーターを設定することと同じです。アカウントの有効期限を確認するアクセス順位の設定もこれに含まれます。

```
access_provider = ldap
ldap_access_order = expire
ldap_account_expire_policy = ad
```

LDAP フィルターに基づいてアクセスを付与するユーザーアカウントを特定する追加オプションもあります。アカウントはまずフィルターとマッチして、次に有効期限の確認をパスする必要があります。(access_provider = ad 設定で暗示されています)

たとえば、以下の設定では、管理者グループに所属して *unixHomeDirectory* 属性を持つユーザーのみがアクセス制御チェックに合致します。

```
access_provider = ad
ad_access_filter = (&(memberOf=cn=admin, ou=groups, dc=example, dc=com)
(unixHomeDirectory=*))
```

アクセス制御チェックでは、セキュアな接続 (GSS-API メカニズムによる SASL) が必要になります。一般的な LDAP パラメーターを用いて同一の機能を設定するには、SASL/GSS-API 接続、フィルター、有効期限チェックの定義が必要になります。

```
access_provider = ldap
ldap_access_order = filter, expire
ldap_account_expire_policy = ad
ldap_access_filter = (&(memberOf=cn=admin, ou=groups, dc=example, dc=com)
(unixHomeDirectory=*))
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = CLIENT_SHORTNAME$@EXAMPLE.COM
ldap_schema = ad
```

3.3.4. ID プロバイダーオプションの追加設定

3.3.4.1. ユーザー名形式を設定する

SSSD が実行する主要なアクションの 1 つは、ローカルシステムユーザーとリモート ID プロバイダーの ID とのマッピングです。SSSD はユーザー名とドメインバックエンド名の組み合わせを使ってログイン ID を作成します。

SSSD は、ユーザーが異なるドメインに属している限り、同一のユーザー名を持つ異なるユーザーを認識できます。例えば、SSSD は、`ldap.example.com` ドメイン内の `jsmith` と `ldap.otherexample.com` ドメイン内の `jsmith` の両方を問題なく認証できます。

完全なユーザー名の構築に使用する名前の形式は、(オプションで) 設定の `[sssd]` セクションでユニバーサルに定義され、その後に各ドメインセクションで個別に定義することができます。

LDAP や Samba、Active Directory、Identity Management、ローカルシステムなど、異なるサービスのユーザー名は、すべて異なる形式を使用しています。SSSD がユーザー名/ドメイン名の識別に使用する表現は、異なる形式の名前を解釈できる必要があります。この表現は、`re_expression` パラメーターで設定します。

グローバルのデフォルトでは、以下のフィルターが `name@domain` 形式の名前を構築します。

```
(?P<name>[^\@]+)@?(?P<domain>[^\@]*$)
```



注記

正規の表現形式は、Python 構文です。

ドメインの部分は、ID プロバイダーのドメイン名に基づいて自動的に提供される場合があります。つまり、ユーザーは `jsmith` としてログインして、このユーザーが例えば LOCAL ドメインに属しているとする、SSSD はこのユーザーのユーザー名を `jsmith@LOCAL` と解釈します。

しかし、別の ID プロバイダーは別の形式を使用している場合があります。例えば Samba は非常に厳格な形式を使用しているため、ユーザー名は `DOMAIN¥username` という形式に一致する必要があります。このため Samba の正規表現は以下のようになります。

```
(?P<domain>[^\¥]*?)¥?(?P<name>[^\¥]+$)
```

Active Directory のようなプロバイダーは、複数の異なる名前形式をサポートします。たとえば Active Directory と Identity Management は、デフォルトで以下の 3 つの異なる形式をサポートします。

- ✧ `username`
- ✧ `username@domain.name`
- ✧ `DOMAIN¥username`

このため Active Directory および Identity Management プロバイダーのデフォルト値は、3 つすべての名前形式を可能にするより複雑なフィルターとなります。

```
((?P<domain>[^\¥]+)¥(?P<name>.+))|((?P<name>[^\@]+)@(?P<domain>.+))|(^(?P<name>[^\@¥]+)$)
```



注記

`jsmith@ldap.example.com` といった完全修飾名で情報を要求すると、適切なユーザーアカウントが常に返されます。異なるドメイン内に同一ユーザー名を持つ複数のユーザーがいる場合、ユーザー名のみを特定すると、ルックアップの最初の順位のドメインのユーザーが返されます。

`re_expression` はユーザー名形式の設定方法で最も重要なものですが、別のアプリケーションで便利な 2 つのオプションもあります。

デフォルトのドメイン名の値

1 つ目は `default_domain_suffix` で、全ユーザーが使用するデフォルトのドメイン名を設定します。(これはグローバル設定で、`[sssd]` セクションのみで利用可能です。) 複数のドメインが設定されている場合があるかもしれませんが、ユーザーデータを保存するのは 1 つだけで、他はホストやサービス ID に使用されます。デフォルトのドメイン名を設定することで、ユーザーはドメイン名を特定せずにユーザー名だけでログインできます。ドメイン名は、プライマリードメイン外のユーザーの場合に必要となります。

```
[sssd]
...
default_domain_suffix = USERS.EXAMPLE.COM
```

出力用の完全名形式

もうひとつのパラメーターは `re_expression` に関連していますが、ユーザー名の **解釈** 方法を定義するのではなく、特定された名前の **プリント** 方法を定義します。`full_name_format` パラメーターは、ユーザー名とドメイン名(が決められた後に)の表示方法を設定します。

SSSD は常に、サブドメイン内のユーザー名を完全修飾として返します。デフォルト形式は `username@domain` としてプリントされます。`full_name_format` パラメーターは形式を `printf` 形式に設定するので、デフォルトは以下のように表現されます。

```
full_name_format = %1$s@%2$s
```

ユーザー名が引数 1、ドメインが引数 2、`$s` は値が文字列であることを意味します。

`%1$s` と `%2$s` の展開とは別に、`%3$s` の展開もサポートされています。これはドメインフラット名に展開し、通常直接設定された AD ドメインまたは IdM 信頼を使って発見された AD ドメインに使用されます。

完全修飾ユーザー名の形式は設定可能なものです。ただし、名前の設定によっては、SSSD が名前のドメイン要素を削除し、認証エラーが発生する場合があります。このため、`full_name_format` を標準以外の値に設定すると、より標準的な値への変更を促す警告が表示されます。

3.3.4.2. オフライン認証を有効にする

ドメインサービスについての情報のほかに、ユーザー ID は常にキャッシュされます。しかし、ユーザー **認証情報** は、デフォルトではキャッシュされません。このため、SSSD は常に認証要求をバックエンド ID プロバイダーでチェックすることになります。ID プロバイダーがオフラインだったり利用できない場合は、これらの認証要求を処理することができず、ユーザー認証に失敗することになります。

オフライン認証情報キャッシング は有効にすることができます。すると、(正常なログイン後に) 認証情報がユーザーアカウントの一部として SSSD キャッシュに保存されます。このため、もし ID プロバイダーが利用できない場合でも、ユーザーは保存された認証情報を使って認証ができることになります。オフライン認証情報キャッシングは本来、個別のドメインエントリーで設定されますが、PAM サービスセクションで設定可能なオプションもいくつかあります。これは、認証情報キャッシングがリモートドメインに加えて、ローカルの PAM サービスとも対話するためです。

```
[domain/EXAMPLE]
cache_credentials = true
```

認証情報の有効期限を設定するオプションのパラメーターがあります。有効期限が便利なのは、古くなったアカウントや認証情報を持つユーザーがローカルサービスに無期限にアクセスすることを防ぐことができるからです。

認証情報の有効期限そのものは PAM サービスで設定され、これがシステムへの認証要求を処理します。

```
[sssd]
services = nss,pam
...

[pam]
offline_credentials_expiration = 3
...

[domain/EXAMPLE]
cache_credentials = true
...
```

offline_credentials_expiration は、正常なログイン後に、ユーザーの単一認証情報エントリーがキャッシュに保存される日数を設定します。これを 0 に設定すると、エントリーは無期限に保存されます。この設定に関する詳細は、[表3.3「SSSD \[pam\] 設定のパラメーター」](#)を参照してください。

以下のオプションは認証情報キャッシュに特別に関連しているわけではありませんが、各ドメインには個別のユーザーとサービスキャッシュの有効期限に関する設定オプションがあります。

- ※ **account_cache_expiration** は、正常なログイン後に、ユーザーアカウントエントリー全体が SSSD キャッシュから削除される日数を設定します。これは、個別の認証情報キャッシュ有効期限と同じか、それより長く設定する必要があります。
- ※ **entry_cache_timeout** は、キャッシュに保存されている全エントリーについて、SSSD が ID プロバイダーから更新情報を要求するまでの有効期間 (秒単位) を設定します。また、グループやサービス、netgroup、sudo、autofs エントリーに関する個別のキャッシュタイムアウトのパラメーターもあり、それらは `sssd.conf man` ページに記載されています。デフォルトの時間は、5400 秒 (90 分) です。

例を示します。

```
[sssd]
services = nss,pam
...

[pam]
offline_credentials_expiration = 3
...

[domain/EXAMPLE]
cache_credentials = true
account_cache_expiration = 7
entry_cache_timeout = 14400
...
```

3.3.4.3. パスワードの有効期限を設定する

パスワードポリシーは一般的に、パスワードの有効期間と期限、変更時期を設定します。これらのパスワード有効期限ポリシーは ID プロバイダーを通してサーバー側で評価され、その後に PAM サービスを通して SSSD で警告が処理され、表示されます。

パスワードの警告には設定可能な以下の 2 つのエリアがあります。

- ※ 全ドメインについて、パスワードの有効期限切れの警告をどの程度前もって表示するかというグローバルデフォルト。これは、PAM サービス向けに設定されます。
- ※ パスワードの有効期限切れの警告をどの程度前もって表示するかというドメインごとの設定。

ドメインレベルのパスワード有効期限警告を使用する際には、認証プロバイダー (`auth_provider`) もドメイン向けに設定する必要があります。

例を示します。

```
[sssd]
services = nss,pam
...

[pam]
pam_pwd_expiration_warning = 3
...

[domain/EXAMPLE]
id_provider = ipa
auth_provider = ipa
pwd_expiration_warning = 7
```

パスワード有効期限の警告が表示されるには、サーバーから SSSD に警告が送信される必要があります。サーバーからパスワード警告が送信されないと、パスワード有効期限が SSSD で設定された期間内であっても、SSSD はメッセージを表示しません。

SSSD でパスワード有効期限警告が設定されていないか、0 に設定されていると、SSSD パスワード警告フィルターが適用されず、サーバー側のパスワード警告が自動的に表示されます。

注記

パスワード警告がサーバーから送信されていれば、PAM もしくはドメインパスワード有効期限は本来、バックエンド ID プロバイダーのパスワード警告設定に優先 (もしくは無視) します。

たとえば、バックエンド ID プロバイダーでは警告が 28 日間に設定されていて、SSSD の PAM サービスでは 7 日間に設定されていたとします。プロバイダーは 28 日前になると SSSD に警告を送信しますが、SSSD 設定のパスワード有効期限にしたがって、ローカルでは 7 日前になるまで警告は表示されません。

注記

同様のパラメーターは、Kerberos 認証情報をキャッシュする Kerberos 認証プロバイダー `krb5_store_password_if_offline` の使用時にも利用可能です。

3.3.4.4. LDAP グループにおけるローカルシステムユーザー

LDAP ID プロバイダー(LDAP または IdM) は、RFC 2307 か RC2307bis のスキーマを使うことができます。Active Directory LDAP プロバイダーは Active Directory 固有のスキーマを使用し、これは RFC 2307bis と互換性があります。これらのスキーマ要素を使用することで、SSSD はローカルユーザーを LDAP グループ内で管理できます。

新たな LDAP グループが作成されると、ローカルユーザー ID に設定された *memberUID* 属性の値を付けて、ローカルユーザーを追加することができます。

ローカルシステム上では、`getent group` を使うとローカルユーザーがグループメンバーに含まれます。

```
[root@server ~]# getent group example
example:x:3:jsmith,bjensen,landerson,mreynolds
```

これは、LDAP ディレクトリーにグループの情報をクエリします。このメンバーシップが処理されると、ユーザーは `/etc/passwd` のシステム設定に追加されます。

LDAP グループへのクエリおよびローカルユーザーの作成は SSSD の外で NSS (`nss_ldap`) によって実行されま

す。ただし、認証操作や `id` といった ID ツールは SSSD を通過し、SSSD 用に設定された LDAP ID プロバイダーにはローカルユーザーのレコードはありません。SSSD がローカルユーザーを処理するには、以下の 2 つの方法があります。

- ※ ユーザーが削除されたローカルアカウントの残りであるかのように、ローカル `passwd` ファイルからそのユーザーを削除します。
- ※ グループ内のユーザーが LDAP で見つからない場合は、フォールバックとしてローカルユーザーリスト (`passwd`) にクエリを行い、その後 LDAP ユーザーであるかのようにそのキャッシュにユーザーを追加します。

この動作は、ID プロバイダードメインの `ldap_rfc2307_fallback_to_local_users` パラメーターで設定します。デフォルトでは `false` となっており、LDAP プロバイダーで存在するユーザーのみが認識され、ローカルユーザーは LDAP グループに追加されると削除されます。これを `true` に設定すると、LDAP ディレクトリー内で LDAP グループメンバーが見つからない場合、フォールバックとしてローカルシステムユーザーにクエリを行います。

3.3.4.5. グループメンバーを無視する

LDAP グループについての情報をルックアップしている際には、デフォルトではそのグループのすべてのメンバーが返されます。大規模なグループやネスト化されたグループの場合、処理に時間がかかる場合もあります。メンバーシップ一覧自体は、ユーザーがグループに所属しているかどうかを評価する際には実際に使用されません。ほとんどのサービスでは、ユーザーがグループに所属しているかどうかを判断する際にメンバー一覧の UID をチェックするのではなく、`getent group` のようなものを使用しています。

全体的なパフォーマンス、特に ID ルックアップを改善するためには、グループメンバーシップのルックアップを無効にすることができます。こうすると実質的に、キャッシュするために空のグループが SSSD に返されます。

これはドメインエントリーごとに `ignore_group_members` パラメーターで設定されます。

```
[domain#ad.example.com]
id_provider = ad
ad_server = adserver.example.com
ad_hostname = client.example.com
...
ignore_group_members = true
```

3.3.4.6. DNS サービス検索を使用する

[RFC 2782](#) で定義されている DNS サービス検索は、アプリケーションが特定の種類の特定のサービスについてあるドメイン内の SRV 記録をチェックできるようにします。そして、この種類のサービスが発見されたサーバーを返します。

SSSD では、ID プロバイダーと認証プロバイダーは (IP アドレスかホスト名で) 明示的に定義されるか、サービス検索を使って動的に見つけることができます。`id_provider = ldap` が対応する `ldap_uri` パラメーターなしに設定されるなど、プロバイダーサーバーが一覧表示されない場合は、検索が自動的に使われます。

DNS 検索クエリは、以下の形式になります。

```
_service._protocol.domain
```

たとえば、`example.com` ドメインで TCP を使って LDAP サーバーをスキャンするには、以下のようになります。

```
_ldap._tcp.example.com
```

注記

サービス検索を使用するサービスには、特別の DNS レコードを DNS サーバーに追加します。

```
_service._protocol._domain TTL priority weight port hostname
```

SSSD では、サービスのタイプはデフォルトで LDAP となり、ほとんどすべてのサービスが TCP を使用します (UDP で開始する Kerberos を除く)。サービス検索を有効にするには、ドメイン名のみが必要です。デフォルトでは、マシンのホスト名のドメイン部分を使用しますが、別のドメインを指定することもできます (`dns_discovery_domain` パラメーターを使用)。

このため、1 つの例外を除いては、デフォルトではサービス検索に新たな設定を加える必要はありません。パスワード変更プロバイダーはデフォルトでサービス検索が無効になっているので、サービスのタイプを設定して明示的に有効にする必要があります。

```
[domain/EXAMPLE]
...
chpass_provider = ldap
ldap_chpass_dns_service_name = ldap
```

設定は不要ですが、異なる DNS ドメイン (`dns_discovery_domain`) を使ったり、スキャンするサービスタイプを異なるものに設定したりすることで、サービス検索はカスタマイズが可能です。例を示します。

```
[domain/EXAMPLE]
id_provider = ldap

dns_discovery_domain = corp.example.com
ldap_dns_service_name = ldap

chpass_provider = krb5
ldap_chpass_dns_service_name = kerberos
```

最後に、サービス検索はバックアップサーバーと使われることは決してありません。これは、プロバイダーのためにプライマリサーバーと使われるものです。つまり、検索はまずサーバーの位置を確認するために使われ、次に SSSD はバックアップサーバーの使用にフォールバックします。検索をプライマリサーバーに使用するには、`_srv_` をプライマリサーバーの値として使用し、その後バックアップサーバーを記載します。例を示します。

```
[domain/EXAMPLE]
id_provider = ldap
ldap_uri = _srv_
ldap_backup_uri = ldap://ldap2.example.com

auth_provider = krb5
krb5_server = _srv_
krb5_backup_server = kdc2.example.com

chpass_provider = krb5
ldap_chpass_dns_service_name = kerberos
ldap_chpass_uri = _srv_
ldap_chpass_backup_uri = kdc2.example.com
```

注記

サービス検索はバックアップサーバーでは使用できません。プライマリーサーバーのみとなります。

DNS ルックアップがホスト名の IPv4 アドレスを返すことに失敗した場合、SSSD は失敗を返す前に IPv6 アドレスのルックアップを試みます。これにより確実に非同期のリゾルバーが正しいアドレスを識別できるようになります。

ホスト名解決動作は、`sssd.conf` 設定ファイルの `lookup_family_order` オプションで設定します。

3.3.4.7. 証明書のサブジェクト名で IP アドレスを使用する (LDAP のみ)

`ldap_uri` オプションでサーバー名ではなく IP アドレスを使用すると、TLS/SSL 接続が失敗する場合があります。TLS/SSL 証明書は IP アドレスではなく、サーバー名を含んでいます。しかし、証明書の **subject alternative name (サブジェクトの別名)** フィールドにはサーバーの IP アドレスを含めることができるため、IP アドレスを使用して正常かつ安全な接続が可能になります。

1. 既存の証明書を証明書要求に変更します。証明書署名鍵 (CSR) は、証明書の対象となる LDAP サーバーの秘密鍵で署名される必要があります。`-signkey` オプションを使用して、秘密鍵を含んでいる PEM ファイルを渡します。

```
openssl x509 -x509toreq -in old_cert.pem -out req.pem -signkey key.pem
```

自己署名証明書の場合は以下ようになります。

```
openssl x509 -x509toreq -in old_cert.pem -out req.pem -signkey old_cert.pem
```

2. `/etc/pki/tls/openssl.cnf` 設定ファイルを編集して、`[v3_ca]` セクションの下にサーバーの IP アドレスを含めます。

```
subjectAltName = IP:10.0.0.10
```

3. 生成された証明書要求を使用して、指定した IP アドレスを持つ新規の自己署名証明書を生成します。

```
openssl x509 -req -in req.pem -out new_cert.pem -extfile ./openssl.cnf -
extensions v3_ca -signkey old_cert.pem
```

`-extensions` オプションは、証明書で使用する拡張子を設定します。上記の場合、適切なセクションを読み込む `v3_ca` とします。

4. `old_cert.pem` ファイルから秘密鍵ブロックを `new_cert.pem` ファイルにコピーして、すべての関連情報を 1 つのファイルに保存します。

`nss-utils` パッケージが提供する `certutil` ユーティリティーで証明書を作成する際は、`certutil` が証明書作成目的のみで DNS サブジェクトの別名をサポートすることに注意してください。

3.3.4.8. 別のタイプのアクセス制御を設定する

SSSD はドメイン設定用に基本的なアクセス制御を提供し、単純なユーザー allow/deny リストまたは LDAP バックエンド自体の使用を可能にします。

3.3.4.8.1. Simple Access Provider の使用

Simple Access Provider (シンプルアクセスプロバイダー) は、ユーザー名またはグループのリストを基にしてアクセスを許可または拒否します。

Simple Access Provider とは、特定のマシンへのアクセスを制限する手段です。たとえば、ある企業で複数のノートパソコンが使用されている場合、Simple Access Provider を使用すると、別のユーザーが同一の認識プロバイダーで認証されている場合でも、特定のユーザーもしくはグループのみにアクセスを限定することができます。

最も一般的なオプションは `simple_allow_users` と `simple_allow_groups` で、これらは明示的に特定のユーザー (該当するユーザーまたはグループメンバー) にアクセスを許可して他の人からのアクセスは拒否します。また、deny リストを作成することもできます (明示した人にアクセスを拒否し、他の人には暗黙に許可する)。

Simple Access Provider は、以下にあげる 4 つのルールにしたがって、どのユーザーにアクセスを許可する/しないかを決定します。

- allow (許可) と deny (拒否) の両方のリストが空の場合には、アクセスが許可されます。
- リストが提供されている場合は、allow ルールが最初に、その後に deny ルールが評価されます。実際には、deny ルールが allow ルールに優先することになります。
- allow リストが提供されている場合は、そのリストに記載されているユーザー以外はすべて、アクセスが拒否されます。
- deny リストのみが提供されている場合は、そのリストに記載されているユーザー以外はすべて、アクセスが許可されます。

以下の例では、2 人のユーザーと IT グループに所属する全員にアクセスが許可されます。暗黙的に、他のすべてのユーザーは拒否されます。

```
[domain/example.com]
access_provider = simple
simple_allow_users = jsmith,bjensen
simple_allow_groups = itgroup
```



注記

SSSD 内の LOCAL ドメインは `simple` をアクセスプロバイダーとしてサポートしません。

他のオプションは `sssd-simple man` ページに記載されていますが、あまり使用されることはありません。

3.3.4.8.2. LDAP アクセスフィルターの使用

LDAP、Active Directory、Identity Management サーバーは、ドメイン用のアクセス制御ルールを提供します。関連するフィルターオプション (`ldap_access_filter`) では、特定のホストへのアクセスを許可されるユーザーを指定します。ユーザーフィルターを使用しないと全ユーザーのアクセスが拒否されます。

例を示します。

```
[domain/example.com]
access_provider = ldap
ldap_access_filter = memberOf=cn=allowedusers,ou=Groups,dc=example,dc=com
```

注記

LDAP アクセスプロバイダーのオフラインキャッシングは、ユーザーの前のオンラインログイン試行が正常であったかどうかの判定に限定されます。前のログイン中にアクセスを許可されたユーザーは、オフライン中も継続してアクセスが許可されます。

SSSD は、エントリー内の *authorizedService* や *host* 属性で結果をチェックすることもできます。実際にはユーザーエントリーや設定によって、LDAP フィルター、*authorizedService*、*host* のすべての評価が可能です。`ldap_access_order` パラメーターには、使用するすべてのアクセス制御方法を評価方法の順番で記載します。

```
[domain/example.com]
access_provider = ldap
ldap_access_filter = memberOf=cn=allowedusers,ou=Groups,dc=example,dc=com
ldap_access_order = filter, host, authorized_service
```

認証済みサービスや許可されたホストの評価に使用するユーザーエントリー内の属性は、カスタマイズが可能です。追加のアクセス制御パラメーターは、`sssd-ldap(5) man` ページに記載されています。

3.3.4.9. プライマリーサーバーとバックアップサーバーの設定

ドメイン用の ID および認証プロバイダーは、自動フェイルオーバーの設定が可能です。SSSD は最初に、指定されたプライマリーサーバーへの接続を試みます。このサーバーに接続できない場合は、記載されたバックアップサーバーに順番に接続を試みます。

注記

SSSD は、プライマリーサーバーへの接続が再確立されるまで、30 秒ごとに接続を試み、接続ができるとバックアップからプライマリーに切り替えます。

主要なサービス領域にはすべて、プライマリーとバックアップサーバーのオプション設定があります [2]。

表3.10 プライマリーおよびセカンダリーサーバーのパラメーター

サービスエリア	プライマリーサーバーの属性	バックアップサーバーの属性
LDAP ID プロバイダー	<code>ldap_uri</code>	<code>ldap_backup_uri</code>
Active Directory ID プロバイダー	<code>ad_server</code>	<code>ad_backup_server</code>
Identity Management (IdM または IPA) ID プロバイダー	<code>ipa_server</code>	<code>ipa_backup_server</code>
Kerberos 認証プロバイダー	<code>krb5_server</code>	<code>krb5_backup_server</code>

サービスエリア	プライマリサーバーの属性	バックアップサーバーの属性
Kerberos パスワード変更プロバイダー	krb5_kpasswd	krb5_backup_kpasswd
パスワード変更プロバイダー	ldap_chpass_uri	ldap_chpass_backup_uri

プライマリサーバーとセカンダリーサーバーは、コンマ区切りのリストにします。プライマリサーバーのリストにあるサーバーが優先されます。SSSD がバックアップサーバーを検索するのは、プライマリサーバーに接続できなかった場合のみです。プライマリもセカンダリーも、優先順位でサーバーを記載します。つまり、最初に記載されているものが最初に試行されます。_srv_ を使用したサービス検索は、プライマリサーバーのみサポートされています。

```
[domain/EXAMPLE]
id_provider = ad
ad_server = ad.example.com, ad1.example.com
ad_backup_server = ad-backup.example.com, ad-backup1.example.com
```

フェイルオーバーのメカニズムに関する詳細情報は、`sssd-ldap(5) man` ページを参照してください。

3.3.5. プロキシ ID プロバイダーを作成する

SSSD におけるプロキシは、単なる中継点です。SSSD はそのプロキシサービスに接続して、それからそのプロキシが指定されたライブラリーを読み込みます。これにより SSSD はこの方法以外では使用できないリソースの一部を使用できるようになります。たとえば、SSSD は認証プロバイダーとしては LDAP と Kerberos のみをサポートしていますが、プロキシを使用すると SSSD は指紋スキャナーやスマートカードなど別の認証メソッドを使用できるようになります。

表3.11 プロキシドメイン設定のパラメーター

パラメーター	説明
proxy_pam_target	<p>PAM が認証プロバイダーとして委任する必要があるターゲットを指定します。ターゲットは PAM サービスです。これは PAM スタック情報を含むファイルで、デフォルトの <code>/etc/pam.d/</code> ディレクトリー内にあります。</p> <p>これは認証プロバイダーの代理に使用されます。</p>
proxy_lib_name	<p>ID 要求をプロキシする経路先の既存の NSS ライブラリーを指定します。これは ID プロバイダーの代理に使用されます。</p>



重要

プロキシ PAM スタックに `pam_sss.so` が再帰的に格納されていないことを確認してください。

例3.8 プロキシ ID と Kerberos 認証

プロキシライブラリーは、`proxy_lib_name` パラメータを使用して読み込まれます。このライブラリーは、任意の認証サービスと互換性があれば、いずれのものでも構いません。Kerberos 認証プロバイダーの場合は、NIS のように Kerberos 互換のライブラリーである必要があります。

```
[domain/PROXY_KRB5]
```

```

auth_provider = krb5
krb5_server = kdc.example.com
krb5_realm = EXAMPLE.COM

id_provider = proxy
proxy_lib_name = nis
cache_credentials = true

```

例3.9 LDAP ID とプロキシ認証

`proxy_pam_target` は PAM サービスを指定します。たとえば、以下では PAM 指紋モジュールに LDAP を使用します。

```

[domain/LDAP_PROXY]
id_provider = ldap
ldap_uri = ldap://example.com
ldap_search_base = dc=example,dc=com

auth_provider = proxy
proxy_pam_target = sssdpamproxy
cache_credentials = true

```

SSSD デーモンを設定した後に、指定した PAM ファイルが設定されていることを確認します。この例では、ターゲットは `sssdpamproxy` なので、`/etc/pam.d/sssdpamproxy` ファイルを作成して PAM/LDAP モジュールを読み込みます。

```

auth          required    pam_frprint.so
account       required    pam_frprint.so
password      required    pam_frprint.so
session       required    pam_frprint.so

```

例3.10 ID プロキシと認証プロキシ

SSSD は、ID プロキシと認証プロキシの両方を持つドメインを使用できます。そのため関与する唯一の設定はプロキシの設定です。認証 PAM モジュールの `proxy_pam_target` と、NIS や LDAP のようなサービスの `proxy_lib_name` の設定があります。

以下の例では考えられる設定を説明していますが、現実的なものではありません。ID と認証に LDAP が使われる場合は、ID プロバイダーと認証プロバイダーの両方がプロキシではなく、LDAP 設定とすべきです。

```

[domain/PROXY_PROXY]
auth_provider = proxy
id_provider = proxy
proxy_lib_name = ldap
proxy_pam_target = sssdproxyldap
cache_credentials = true

```

SSSD ドメインを追加したら、次にシステムのセッティングを更新してプロキシサービスを設定します。

1. `/etc/pam.d/sssdproxyldap` ファイルを作成します。これには `pam_ldap.so` モジュールが必要となります。

```
auth          required    pam_ldap.so
account       required    pam_ldap.so
password      required    pam_ldap.so
session       required    pam_ldap.so
```

2. `nss-pam-ldap` パッケージがインストールされていることを確認します。

```
[root@server ~]# yum install nss-pam-ldap
```

3. LDAP ネームサービスデーモン用の設定ファイルである `/etc/nslcd.conf` を編集し、LDAP ディレクトリー用の情報を含めます。

```
uid nslcd
gid ldap
uri ldaps://ldap.example.com:636
base dc=example,dc=com
ssl on
tls_cacertdir /etc/openldap/cacerts
```

3.3.6. ID プロバイダーを使った Kerberos 認証を設定する

LDAP とプロキシ ID のプロバイダーは、両方とも別個の Kerberos ドメインを使用して認証することができます。Kerberos 認証プロバイダーの設定には、*key distribution center* (キー配布センター:KDC) と Kerberos ドメインが必要になります。プリンシパル名はすべて、指定された ID プロバイダー内で利用可能である必要があります。利用可能でない場合は、SSSD が `username@REALM` 形式を使用してプリンシパルを構築します。



注記

Kerberos が提供できるのは、認証のみです。ID データベースは提供できません。

SSSD は、Kerberos KDC が Kerberos kadmin サーバーであることも想定しています。しかし実稼働環境では一般的に、KDC の読み込み専用のレプリカが複数あり、kadmin サーバーは 1 つのみとなっています。`krb5_kpasswd` オプションを使用してパスワード変更サービスが稼働する場所を指定するか、または同サービスがデフォルト以外のポートで稼働するかどうかを指定します。`krb5_kpasswd` オプションが指定されない場合は、SSSD は Kerberos KDC を使用してパスワード変更をします。

基本的な Kerberos 設定オプションは [表3.12「Kerberos 認証設定のパラメーター」](#) に記載されています。Kerberos 設定オプションについての詳細情報は `sssd-krb5(5)` の man ページを参照してください。

例3.11 基本的な Kerberos 認証

```
# A domain with identities provided by LDAP and authentication by Kerberos
[domain/KRBDOMAIN]
id_provider = ldap
chpass_provider = krb5
ldap_uri = ldap://ldap.example.com
ldap_search_base = dc=example,dc=com
ldap-tls_reqcert = demand
ldap_tls_cacert = /etc/pki/tls/certs/ca-bundle.crt
```



```

auth_provider = krb5
krb5_server = kdc.example.com
krb5_backup_server = kerberos.example.com
krb5_realm = EXAMPLE.COM
krb5_kpasswd = kerberos.admin.example.com
krb5_auth_timeout = 15
krb5_use_kdcinfo = true

```

例3.12 Kerberos チケット更新オプションの設定

Kerberos 認証プロバイダーのタスクの一つは、ユーザーとサービスのチケット保証チケット (TGT) を要求することです。このチケットは、チケットのプリンシパルであるユーザーがアクセスすると、特定サービスに動的に他のチケットを生成するために使用されます。

ユーザープリンシパルに最初に与えられた TGT は、そのチケットのライフタイムの間のみ有効です (デフォルトでは、設定済み KDC の設定)。その後は、チケットの更新や延長はできません。しかし、チケットを更新しないと、実行中のサービスにアクセスしようとする際やチケットの期限が切れた時に問題が起きるサービスもあります。

Kerberos チケットはデフォルトでは更新できませんが、`krb5_renewable_lifetime` や `krb5_renew_interval` パラメーターを使うとチケット更新を有効にできます。

チケットの有効期間は `krb5_lifetime` パラメーターを使って SSSD 内で設定します。ここでは単一チケットの有効期間を指定し、KDC の他のどの値にも優先します。

チケット更新自体は `krb5_renewable_lifetime` パラメーターで有効化されます。ここではすべての更新を勧奨して、チケットの最大有効期間を設定します。

たとえば以下では、チケットのライフタイムは 1 時間に、更新可能な有効期間は 24 時間に設定されます。

```

krb5_lifetime = 1h
krb5_renewable_lifetime = 1d

```

この設定では、チケットは 1 時間ごとに有効期限が切れ、最大 1 日間は継続して更新できることになります。

有効期間と更新可能な有効期間の値は、秒数 (s)、分数 (m)、時間 (h)、日数 (d) のどれでも使用できます。

チケット更新で設定が必要な別のオプションは、`krb5_renew_interval` パラメーターです。これは、チケット更新の必要性を SSSD がチェックする頻度を設定します。この設定に関係なく、チケットの有効期間が半分になった時点で、チケットは自動的に更新されます (この値は常に秒単位)。

```

krb5_lifetime = 1h
krb5_renewable_lifetime = 1d
krb5_renew_interval = 60s

```

注記

`krb5_renewable_lifetime` の値が設定されていなかったり、`krb5_renew_interval` パラメーターが設定されていないかゼロ (0) に設定されていたりすると、チケット更新は無効になります。チケット更新を有効にするには、`krb5_renewable_lifetime` と `krb5_renew_interval` の両方が必要です。

表3.12 Kerberos 認証設定のパラメーター

パラメーター	説明
chpass_provider	パスワード変換操作に使用するサービスを指定します。これは認証プロバイダーと同じものと想定されています。Kerberos を使用するには、これを <code>krb5</code> に指定します。
krb5_server krb5_backup_server	<p>SSSD が接続するプライマリ Kerberos サーバーを IP アドレスかホスト名で提供します。</p> <p>プライマリサーバーが利用できない場合に SSSD が接続する Kerberos サーバーの IP アドレスまたはホスト名をコンマ区切りのリストで提供します。このリストは設定順に提供されるため、最初のサーバーが最初に試行されます。SSSD は 1 時間後に <code>krb5_server</code> パラメーターで指定されているプライマリサービスに再接続を試みます。</p> <p>KDC サーバーまたは <code>kpasswd</code> サーバーにサービス検索を使用している場合、SSSD は最初に UDP を接続プロトコルとして指定する DNS エントリを検索し、次に TCP にフォールバックします。</p>
krb5_realm krb5_lifetime	<p>KDC が実行する Kerberos レalmを特定します。</p> <p>秒 (s)、分 (m)、時間 (h)、または日 (d) で指定した有効期間で Kerberos チケットを要求します。</p>
krb5_renewable_lifetime	秒 (s)、分 (m)、時間 (h)、または日 (d) で指定した有効期間の合計で更新可能な Kerberos チケットを要求します。
krb5_renew_interval	チケット更新の必要性をチェックする SSSD の時間を秒単位で指定します。チケットは有効期間の半分が経過すると自動的に更新されます。このオプションが欠如していたり、またはゼロに設定してある場合は、自動チケット更新は無効になります。
krb5_store_password_if_offline	Kerberos 認証プロバイダーがオフラインである場合に、ユーザーパスワードを保存し、そのプロバイダーがオンラインに戻った時にチケット要求にそのキャッシュを使用するかどうかを指定します。デフォルトは <code>false</code> で、パスワードは保存されません。
krb5_kpasswd	パスワード変更サービスが KDC 上で稼働していない場合に使用する、代替の Kerberos <code>kadmin</code> サービスを記載します。

パラメーター	説明
krb5_ccname_template	<p>ユーザーの認証情報キャッシュを保存するディレクトリーを提供します。これはテンプレート化が可能で、以下のトークンがサポートされています。</p> <ul style="list-style-type: none"> ※ %u、ユーザーのログイン名 ※ %U、ユーザーのログイン UID ※ %p、ユーザーのプリンシパル名 ※ %r、レルム名 ※ %h、ユーザーのホームディレクトリー ※ %d、krb5ccache_dir パラメーターの値 ※ %P、SSSD クライアントのプロセス ID ※ %%、文字通りのパーセント記号 (%) ※ XXXXXX、SSSD に一意のファイル名を安全に作成するように指示するテンプレート末尾の文字列 <p>例を示します。</p> <pre>krb5_ccname_template = FILE:%d/krb5cc_%U_XXXXXX</pre>
krb5_ccachedir	<p>認証情報のキャッシュを保存するディレクトリーを指定します。%d と %P を除いて、krb5_ccname_template と同じトークンを使用してテンプレート化できます。%u、%U、%p、または %h が使用される場合は、SSSD は各ユーザー用にプライベートディレクトリーを作成します。それ以外の場合は、パブリックディレクトリーを作成します。</p>
krb5_auth_timeout	<p>オンライン認証、またはパスワード変更要求が中断されるまでの時間を秒単位で提供します。可能な場合は、認証要求はオフラインで継続されます。デフォルトは 15 秒です。</p>
krb5_use_kdcinfo	<p>Kerberos ロケータープラグインが使用する Kerberos 情報ファイルを作成するかどうかを指定します。デフォルトでは、true に設定されています。これが false に設定されていると、ファイルは SSSD で作成されず、Kerberos オプションは手動で krb5.conf ファイルで設定する必要があります。</p>

3.4. SSSD でのローカルシステムユーザーの管理

3.4.1. SSSD ユーティリティのインストール

SSSD キャッシュやユーザーエントリー、グループエントリーを扱う追加ツールは、sssd-tools パッケージに含まれています。このパッケージは必須ではありませんが、ユーザーアカウントの管理に役立ちます。

```
[root@server ~]# yum install sssd-tools
```

3.4.2. SSSD と UID および GID 番号

useradd などのシステムツールや Red Hat Identity Management などのアプリケーション、または他のクライアントツールなどを使ってユーザーが作成されると、このユーザーには自動的にユーザー ID 番号とグループ ID が割り当てられます。

ユーザーがシステムやサービスにログインすると、SSSD はそのユーザー名と関連する UID/GID 番号をキャッシュします。UID 番号はその後、ユーザーを特定する鍵として使用されます。同一のユーザー名で別の UID を持つユーザーがシステムにログインしようとする、SSSD は名前の競合により別の 2 人のユーザーとみなします。

つまり、SSSD は UID 番号の変更を認識しません。別の UID 番号を持つ既存ユーザーではなく、異なる新規ユーザーと解釈します。既存ユーザーが UID 番号を変更すると、そのユーザーは SSSD と関連サービスおよびドメインへのログインができなくなります。これは ID 情報に関して SSSD を使用するクライアントアプリケーションにも影響します。競合のあるユーザーは発見されないか、アプリケーションへアクセスできません。



重要

UID/GID の変更は、SSSD ではサポートされていません。

何らかの理由で UID/GID 番号が変更されているユーザーが再度ログイン可能となるには、そのユーザーの SSSD キャッシュをクリアにする必要があります。例を示します。

```
[root@server ~]# sss_cache -u jsmith
```

SSSD キャッシュの削除については、[「SSSD キャッシュの削除」](#)で説明しています。

3.4.3. ローカルシステムユーザーの作成

ユーザーがログインするのを待ってから追加するよりも、ユーザーを SSSD データベースにシードする方が便利な場合があります。



注記

ユーザーアカウントを手動で追加するには、`sssd-tools` パッケージをインストールする必要があります。

新規システムユーザーを作成する場合、SSSD ローカル ID プロバイダードメイン内にユーザーを作成することができます。これは、新規システムユーザーの作成や SSSD 設定のトラブルシューティング、特定もしくはネスト化されたグループの作成に便利です。

新規ユーザーは、`sss_useradd` コマンドで追加できます。

最も基本的なのは `sss_useradd` コマンドで、これは新規ユーザー名のみを必要とします。

```
[root@server ~]# sss_useradd jsmith
```

他のオプション (`sss_useradd(8)` man ページに記載) は、UID や GID、ホームディレクトリー、ユーザーが所属するグループなどのアカウントの属性の設定に使用できます。

```
[root@server ~]# sss_useradd --UID 501 --home /home/jsmith --groups admin,dev-group jsmith
```

3.4.4. キックスタート中にユーザーを SSSD キャッシュにシードする

**注記**

ユーザーアカウントを手動で追加するには、`sssd-tools` パッケージをインストールする必要があります。

SSSD では、リモートドメインのユーザーは、そのユーザーの ID が ID プロバイダーから取得されるまで、ローカルシステムで利用可能となりません。しかし、ユーザーがログインするまで利用できないネットワークインターフェイスもあります。これは、ユーザー ID がネットワーク上のどこかにある場合は、ネットワークインターフェイスが使えないことになります。その場合、適切なドメインに関連したそのユーザーの ID がある SSSD キャッシュをシードすることで、ユーザーがローカルでログイン可能となり、適切なインターフェイスをアクティブにできます。

これは以下のように `sss_seed` ユーティリティーで実行します。

```
sss_seed --domain EXAMPLE.COM --username testuser --password-file /tmp/sss-pwd.txt
```

このユーティリティーは、少なくともユーザー名とドメイン名、パスワードを特定するオプションを必要とします。

- ※ `--domain` は SSSD 設定からのドメイン名を提供します。このドメイン名は、SSSD 設定に既存のものである必要があります。
- ※ `--username` は、ユーザーアカウントの短い名前です。
- ※ `--password-file` は、シードエントリーの一時的なパスワードを含むファイルのパスおよびファイル名です。ユーザーアカウントが SSSD キャッシュに既にある場合は、このファイルの一時的なパスワードが SSSD キャッシュにある保存済みパスワードを上書きします。

追加のアカウント設定オプションは、`sss_seed(8)` man ページに記載されています。

これはほとんどの場合、キックスタートもしくは自動セットアップの一部として実行されるので、SSSD を有効にし、SSSD ドメインをセットアップし、パスワードファイルを作成する大規模なスクリプトの一部となります。例を示します。

```
function make_sssd {
cat <<- _EOF_
[sssd]
domains = LOCAL
services = nss,pam

[nss]

[pam]

[domain/LOCAL]
id_provider = local
auth_provider = local
access_provider = permit

_EOF_
}

make_sssd >> /etc/sss/sss.conf

authconfig --enablesssd --enablesssdauth --update

function make_pwdfile {
cat <<1 _EOF_
password
```

```
_EOF_
}

make_pwdfile >> /tmp/sss-d-pwd.txt

sss_seed --domain EXAMPLE.COM --username testuser --password-file /tmp/sss-d-pwd.txt
```

3.4.5. SSSD キャッシュの管理

SSSD は、同じタイプのドメインと異なるタイプのドメインを複数定義することができます。SSSD は各ドメイン用に別個のデータベースファイルを維持するので、各ドメインが個別のキャッシュを持つこととなります。これらのキャッシュファイルは `/var/lib/sss/db/` ディレクトリに格納されています。

3.4.5.1. SSSD キャッシュの削除

LDAP 更新はドメイン用の ID プロバイダーになされるため、キャッシュを削除して新しい情報を迅速にリロードする必要があります。

キャッシュ削除ユーティリティの `sss_cache` は、SSSD キャッシュにあるユーザーやドメイン、グループの記録を無効にします。最新の記録を無効にすることで、キャッシュは ID プロバイダーから更新された記録を取得することが強制されるので、変更が迅速に実現されます。



注記

このユーティリティは、`sss-d` パッケージに SSSD と含まれています。

このユーティリティは通常、ドメイン全体のキャッシュを削除し、記録を更新するために使用されます。

例3.13 ドメイン記録の削除

```
[root@server ~]# sss_cache -d LDAP1
```

特定の記録 (ユーザー、グループ、ネットグループ) が更新されたことを管理者が認識している場合、`sss_cache` はその特定のアカウントの記録を削除し、残りのキャッシュはそのまま残しておくことができます。

例3.14 ユーザー記録の削除

```
[root@server ~]# sss_cache -u jsmith
```

表3.13 `sss_cache` オプション

短い引数	長い引数	説明
<code>-d name</code>	<code>--domain name</code>	特定ドメイン内のユーザーやグループ、他のエントリーのキャッシュエントリーのみを無効にします。
<code>-G</code>	<code>--groups</code>	グループ記録すべてを無効にします。 <code>-g</code> も使われた場合は、 <code>-G</code> が優先され、 <code>-g</code> は無視されます。

短い引数	長い引数	説明
-g name	--group name	特定グループのキャッシュエントリーを無効にします。
-N	--netgroups	ネットグループのキャッシュレコードすべてのキャッシュエントリーを無効にします。-n も使われた場合、-N が優先され -n は無視されます。
-n name	--netgroup name	特定ネットグループのキャッシュエントリーを無効にします。
-U	--users	ユーザーレコードすべてのキャッシュエントリーを無効にします。-u も使われた場合、-U が優先され -u は無視されます。
-u name	--user name	特定ユーザーのキャッシュエントリーを無効にします。

3.4.5.2. ドメインキャッシュファイルの削除

すべてのキャッシュファイルにはドメイン向けに名前がつけられています。たとえば、`example ldap` ドメインの場合、キャッシュファイルは `cache_example ldap. ldb` と命名されます。

キャッシュファイルを削除する場合は、注意してください。この操作は重大な影響を及ぼします。

- ※ キャッシュファイルを削除すると、ID 情報とキャッシュされている認証情報の両方のユーザー情報すべてが削除されます。このため、システムがオンラインでドメインのサーバーに対してユーザー名で認証できる状態でない場合は、キャッシュファイルを削除しないでください。認証情報のキャッシュがないと、オフライン認証は失敗します。
- ※ 異なる ID プロバイダーを参照するように設定が変更された場合、SSSD は 変更前のプロバイダーからのキャッシュされたエントリがタイムアウトになるまで、両方のプロバイダーからのユーザーを認識します。

キャッシュを削除するとこれは避けられますが、新規のプロバイダーで異なるドメイン名を使用する方法がより優れたオプションになります。SSSD が再起動すると、新しい名前でも新規キャッシュファイルが作成され、古いファイルは無視されます。

3.5. SSSD のダウングレード

SSSD のバージョンもしくはオペレーティングシステム自体をダウングレードするには、既存の SSSD キャッシュを削除する必要があります。これが削除されないと、SSSD プロセスは停止しますが PID ファイルは残ります。以下の SSSD ログは、キャッシュのバージョンを認識できないので関連ドメインに接続できないことを示しています。

```
(Wed Nov 28 21:25:50 2012) [sssd] [sysdb_domain_init_internal] (0x0010): Unknown DB version [0.14], expected [0.10] for domain AD!
```

するとユーザーは認識されず、ドメインサーバーとホストに認証されなくなります。

SSSD バージョンをダウングレードしたら、以下を実行します。

1. 既存のキャッシュデータベースファイルを削除する。

```
[root@server ~]# rm -rf /var/lib/sss/db/*
```

2. SSSD プロセスを再起動する。

```
[root@server ~]# systemctl restart sssd.service
```

3.6. SSSD と NSCD の使用

SSSD は、NSCD デーモンと併用するには設計されていません。SSSD は直接的に NSCD と競合することはありませんが、両方のサービスを併用すると、特にエントリがキャッシュされる期間に関して予期しない動作が発生する可能性があります。

最も一般的な問題は NFS との競合です。ネットワーク接続の管理に Network Manager を使用していると、ネットワークインターフェイスが立ち上がるのに数分かかる場合があります。この時間内に各種のサービスが起動を試みます。ネットワークが起動して DNS サーバーが利用可能になる前にこれらのサービスが起動した場合、これらのサービスは必要となる順引きまたは逆引きの DNS エントリーの特定に失敗します。これらのサービスは間違ったまたはおそらく空の `resolv.conf` ファイルを読み込むこととなります。このファイルは通常 1 回しか読み取られないため、このファイルへの変更は自動的に適用されません。NSCD サービスが手動で再起動されたと、この状態は NSCD サービスが実行しているマシン上で NFS 失敗の原因になります。

この問題を避けるには、`/etc/nscd.conf` ファイルでホストとサービスのキャッシングを有効にし、`passwd`、`group`、および `netgroup` のエントリーで SSSD キャッシュに依存するようにします。

`/etc/nscd.conf` ファイルを変更します。

```
enable-cache hosts yes
enable-cache passwd no
enable-cache group no
enable-cache netgroup no
enable-cache services no
```

NSCD がホスト要求に応答するようにすると、これらのエントリーは NSCD がキャッシュし、ブートプロセス中に NSCD によって返されます。他のエントリーはすべて SSSD が処理します。

3.7. SSSD のトラブルシューティング

- ✦ [「SSSD ドメイン用にデバッグログを設定する」](#)
- ✦ [「SSSD ログファイルのチェック」](#)
- ✦ [「SSSD 設定に伴う問題」](#)

3.7.1. SSSD ドメイン用にデバッグログを設定する

ドメインでは、それぞれのデバッグログレベルを設定します。ログレベルを上げると、SSSD やドメイン設定での問題についての情報をより多く収集できます。

ログレベルを変更するには、追加のログを作成する `sssd.conf` ファイルで各セクションの `debug_level` パラメータを設定します。例を示します。

```
[domain/LDAP]
cache_credentials = true
debug_level = 9
```

表3.14 デバッグログレベル

レベル	説明
0	致命的な失敗。SSSD の起動を妨げたり、SSSD の実行を停止させるもの。

レベル	説明
1	重大な失敗。SSSD の強制終了にはつながらないものの、メジャーな機能が少なくとも 1 つは適切に動作していないことを示すエラー。
2	深刻な失敗。特定の要求や操作が失敗したことを知らせるエラー。
3	マイナーな失敗。レベル 2 の失敗を引き起こしかねないエラー。
4	設定
5	機能データ
6	操作機能のメッセージを追跡します。
7	内部制御機能のメッセージを追跡します。
8	関心のある可能性がある関数-内部変数のコンテンツ。
9	非常に低いレベルの追跡情報。

SSSD の稼働中にデバッグレベルを変更するには、`sss_debuglevel` ユーティリティを使用します。これは、`sss-tools` パッケージの一部です。このユーティリティの機能方法に関する情報は、`sss_debuglevel man` ページを参照してください。

3.7.2. SSSD ログファイルのチェック

SSSD は、その操作に関する情報をレポートするために、`/var/log/sss/` ディレクトリーにある多くのログファイルを使用します。SSSD は各ドメイン用のログファイルだけでなく、`sss_pam.log` ファイルと `sss_nss.log` ファイルも作成します。

また、`/var/log/secure` ファイルには認証の失敗とその失敗の理由も記録されます。

3.7.3. SSSD 設定に伴う問題

問:

SSSD が起動に失敗します。

答: SSSD はデーモンが起動する前に、すべての要求されたエントリーで設定ファイルが適切にセットアップされることが必要です。

- ※ SSSD はサービス起動前に、少なくとも 1 つのドメインが適切に設定されていることが必要です。このようなドメインがない場合には、SSSD の起動を試みると、ドメインが設定されていないという以下のようなエラーが返されます。

```
# sssd -d4

[sss] [ldb] (3): server_sort:Unable to register control with rootdse!
[sss] [confdb_get_domains] (0): No domains configured, fatal error!
[sss] [get_monitor_config] (0): No domains configured.
```

`/etc/sss/sss.conf` のファイルを編集して、少なくとも 1 つのドメインを作成します。

- ※ また SSSD は起動前に、少なくとも 1 つの利用可能なサービスプロバイダーが必要です。問題がサービスプロバイダーの設定にある場合、サービスが設定されていないことを示す以下のエラーメッセージが表示されます。

```
[sss] [get_monitor_config] (0): No services configured!
```

/etc/sss/sss.conf ファイルを編集して、少なくとも 1 つのサービスプロバイダーを設定します。



重要

SSSD では、/etc/sss/sss.conf ファイル内の単一の *services* エントリー内に、サービスプロバイダーをコンマ区切りのリストで設定する必要があります。サービスが複数のエントリーに記載されている場合、SSSD が認識するのは最後のエントリーのみです。

問:

'id' のあるグループや 'getent group' のあるグループメンバーが見つかりません。

答: sss.conf の [domain/DOMAINNAME] セクションにある ldap_schema の設定が間違っている可能性があります。

SSSD は、RFC 2307 と RFC 2307bis スキーマタイプをサポートします。デフォルトでは、SSSD はより一般的な RFC 2307 スキーマを使用します。

RFC 2307 と RFC 2307bis の違いは、グループメンバーシップが LDAP サーバーに保存される方法の違いです。RFC 2307 サーバーでは、グループメンバーは複数值の *memberuid* 属性として保存され、これにはメンバーであるユーザー名が含まれます。RFC 2307bis サーバーでは、グループメンバーは複数值の *member* または *uniqueMember* 属性として保存され、これにはこのグループのメンバーであるユーザーもしくはグループの DN が含まれます。RFC 2307bis では、ネスト化されたグループの保持も可能になります。

グループルックアップで情報が返されない場合は、以下を実行します。

1. ldap_schema を rfc2307bis に設定します。
2. /var/lib/sss/db/cache_DOMAINNAME.ldb を削除します。
3. SSSD を再起動します。

これで解決しない場合は、以下の行を sss.conf に追加します。

```
ldap_group_name = uniqueMember
```

その後にはキャッシュを削除して、SSSD を再起動します。

問:

LDAP に対する認証が失敗します。

答: 認証を実行するには、SSSD では通信チャンネルの暗号化が必要になります。このため、sss.conf で標準プロトコル (ldap://) による接続が設定されている場合、SSSD は Start TLS で通信チャンネルの暗号化を試みます。sss.conf でセキュアなプロトコル (ldaps://) による接続が設定されている場合は、SSSD は SSL を使用します。

つまり、LDAP サーバーは SSL または TLS で実行するように設定される必要があります。TLS は標準 LDAP ポート (389) で、SSL はセキュアな LDAPS ポート (636) で有効にする必要があります。SSL と TLS のいずれの場合も、LDAP サーバーは有効な証明書信頼で設定されている必要があります。

無効な証明書信頼は、LDAP に対する認証における最も一般的な問題の一つです。クライアントに LDAP サーバー証明書の適切な信頼がないと、接続を確認できず、SSSD はパスワードの送信を拒否します。LDAP プロトコルでは、パスワードがプレーンテキストで LDAP サーバーに送信されることが要求されます。暗号化されていない接続でパスワードをプレーンテキストで送信することは、セキュリティ上の問題となります。

証明書が信頼されないと、`syslog` メッセージが書き込まれ、TLS 暗号化が開始できません。証明書の設定は、LDAP サーバーが SSSD 以外からアクセス可能かどうかをチェックすることでテストできます。以下の例では、TLS 接続で `test.example.com` への匿名バインドをテストします。

```
$ ldapsearch -x -ZZ -h test.example.com -b dc=example,dc=com
```

証明書信頼が適切に設定されていない場合、以下のエラーが出てテストは失敗します。

```
ldap_start_tls: Connect error (-11) additional info: TLS error -8179:Unknown code
___f 13
```

証明書を信頼するには、以下を実行します。

1. 認証機関が LDAP サーバー証明書の署名に使用する公開 CA 証明書のコピーを取得して、ローカルシステムに保存します。
2. ファイルシステム上の CA 証明書に向ける以下の行を `sssd.conf` ファイルに追加します。

```
ldap_tls_cacert = /path/to/cacert
```

3. LDAP サーバーが自己署名証明書を使用している場合は、`sssd.conf` ファイルから `ldap_tls_reqcert` の行を削除します。

このパラメーターは、SSSD が認証局による証明書を信頼するように指示しますが、これは自己署名型の CA 証明書ではセキュリティリスクとなります。

問:

非標準ポートでの LDAP サーバーへの接続が失敗します。

答: enforcing モードで SELinux を実行する場合、クライアントの SELinux ポリシーは非標準ポートで LDAP サーバーに接続するように修正する必要があります。以下に例を挙げます。

```
# semanage port -a -t ldap_port_t -p tcp 1389
```

問:

NSS がユーザー情報提供に失敗します。

答: これは通常、SSSD が NSS サービスに接続できないことを意味します。

※ NSS サービスが稼働していることを確認します。

```
# service sssd status
Redirecting to /bin/systemctl status sssd.service
sssd.service - System Security Services Daemon
Loaded: loaded (/usr/lib/systemd/system/sss.service; enabled)
Active: active (running) since Wed 2015-01-14 10:17:26 CET; 1min 30s ago
Process: 683 ExecStart=/usr/sbin/sss -D -f (code=exited,
status=0/SUCCESS)
Main PID: 745 (sss)
CGroup: /system.slice/sss.service
        └─745 /usr/sbin/sss -D -f
           └─746 /usr/libexec/sss/sss_be --domain default --debug-to-files...
              └─804 /usr/libexec/sss/sss_nss --debug-to-files
```

```
└─805 /usr/libexec/sss/sss_pam --debug-to-files
```

SSSD が **Active: active (running)** 状態で、かつ出力に `sss_nss` が含まれていれば、NSS サービスは稼働しています。

- ※ NSS が稼働している場合は、プロバイダーが `/etc/sss/sss.conf` ファイル内の `[nss]` セクションで正しく設定されていることを確認してください。特に、`filter_users` 属性と `filter_groups` 属性をチェックしてください。
- ※ SSSD が使用するサービスのリスト内に NSS が含まれていることを確認します。
- ※ `/etc/nsswitch.conf` ファイル内の設定を確認します。詳細は、「[SSSD を使用する NSS サービスの設定](#)」を参照してください。

問:

NSS が誤ったユーザー情報を返します。

答: 検索で間違ったユーザー情報が返された場合には、別のドメインで競合するユーザー名がないかチェックしてください。複数のドメインを使用している場合、`/etc/sss/sss.conf` ファイル内の `use_fully_qualified_domains` 属性を `true` に設定します。これで、異なるドメインで同じ名前を持つ異なるユーザーが区別されます。

問:

ローカルの SSSD ユーザー用のパスワード設定で、パスワードの入力が 2 回要求されます。

答: ローカルの SSSD ユーザーのパスワードの変更を試みる際には、パスワードが 2 回要求されることがあります。

```
[root@clientF11 tmp]# passwd user1000
Changing password for user user1000.
New password:
Retype new password:
New Password:
Reenter new Password:
passwd: all authentication tokens updated successfully.
```

この原因は、PAM 設定が間違っているためです。`/etc/pam.d/system-auth` ファイル内の `use_authok` オプションが正しく設定されていることを確認してください。正しい設定例については、「[サービスの設定: PAM](#)」を参照してください。

問:

Active Directory ID プロバイダーは `sss.conf` ファイルで適切に設定されているのに、SSSD は接続に失敗し、GSS-API エラーがでます。

答: SSSD は、ホスト名を使用しないと Active Directory プロバイダーに接続できません。ホスト名が提供されないと、SSSD クライアントはホストへの IP アドレスが解決できず、認証に失敗します。

たとえば以下の設定の場合、

```
[domain/ADEXAMPLE]
debug_level = 0xFFFF
id_provider = ad
ad_server = 172.16.0.1
```

```
ad_domain = example.com
krb5_canonicalize = False
```

SSSD クライアントは以下の GSS-API エラーを返して、認証要求が失敗します。

```
(Fri Jul 27 18:27:44 2012) [sssd[be[ADTEST]]] [sasl_bind_send] (0x0020):
ldap_sasl_bind failed (-2)[Local error]
(Fri Jul 27 18:27:44 2012) [sssd[be[ADTEST]]] [sasl_bind_send] (0x0080): Extended
failure message: [SASL(-1): generic failure: GSSAPI Error: Unspecified GSS
failure. Minor code may provide more information (Cannot determine realm for
numeric host address)]
```

このエラーを避けるには、*ad_server* を Active Directory ホストの名前に設定するか、「DNS サービス検索を使用する」にあるように *_srv_* キーワードを DNS サービスディレクトリーに使用します。

問:

SSSD を中央認証に設定しましたが、Firefox や Adobe などいくつかのアプリケーションが起動しません。

答: 64 ビットシステム上でも、32 ビットのアプリケーションはパスワードや ID キャッシュへのアクセスに 32 ビットバージョンの SSSD を必要とします。32 ビットバージョンの SSSD が利用できない場合でも、システムは SSSD キャッシュを使うように設定されており、したがって 32 ビットのアプリケーションは起動に失敗します。

たとえば、Firefox はパーミッション拒否のエラーで失敗します。

```
Failed to contact configuration server. See http://www.gnome.org/projects/gconf/
for information. (Details - 1: IOR file '/tmp/gconfd-somebody/lock/ior'
not opened successfully, no gconfd located: Permission denied 2: IOR
file '/tmp/gconfd-somebody/lock/ior' not opened successfully, no gconfd
located: Permission denied)
```

Adobe Reader の場合、以下のエラーでは現行のシステムユーザーが認識されていないことを示しています。

```
[jsmith@server ~]$ acroread
(acroread:12739): GLib-WARNING **: getpwuid_r(): failed due to unknown
user id (366)
```

他のアプリケーションでも、同様のユーザーもしくはパーミッションエラーが表示される場合があります。

問:

SSSD は、削除した自動マウントの場所を示しています。

答: 自動マウントの場所の SSSD キャッシュは、その場所が後で変更されたり削除されたりしても、消えずに残ります。SSSD の *autofs* 情報を更新するには、以下を実行します。

1. 「SSSD キャッシュの削除」の説明にあるように、*autofs* キャッシュを削除します。
2. 「SSSD の起動と停止」にあるように、SSSD を再起動します。

[2] ほとんどのサービスでは、そのサービスで特定のサーバーが設定されていなければ、ID プロバイダーサーバーをデフォルトとしています。

第4章 `realmd` を使った ID ドメインへの接続

`realmd` システムは、ID ドメインを発見して参加する明確かつ簡単な方法を提供します。これはドメイン自体には接続しませんが、SSSD や Winbind といった基礎となる Linux システムサービスがドメインに接続するよう設定します。

Windows Integration Guide では、`realmd` を使用して Microsoft Active Directory (AD) ドメインに接続する方法を説明しています。`realmd` を使用して AD 以外の ID ドメインに接続する方法には、同様の手順が適用されます。対応する [Windows Integration Guide](#) の章を参照してください。

パート III. セキュアなアプリケーション

第5章 PAM (プラグ可能な認証モジュール) の使用

プラグ可能な認証モジュール (PAM) とは、認証と承認のための一般的なフレームワークです。Red Hat Enterprise Linux のほとんどのシステムアプリケーションは、認証および承認で基礎となる PAM 設定に依存しています。

5.1. PAM について

PAM (プラグ可能な認証モジュール) は、集中型認証メカニズムを提供します。システムアプリケーションは、これを使って認証を中央で設定されたフレームワークに中継できます。

PAM には異なるタイプの認証ソース (Kerberos や SSSD、NIS、ローカルファイルシステム) 用のモジュールがあるので、プラグ可能になっています。異なる認証ソースに優先順位を付けることができます。

PAM はモジュラー型アーキテクチャーであることから、管理者はシステム用の認証ポリシーを柔軟に設定することができます。PAM は以下のような理由で開発者および管理者にとって便利なシステムとなっています。

- PAM は多様なアプリケーションで使用できる共通の認証スキームを提供します。
- PAM は、システム管理者に認証における制御と多大な柔軟性を提供します。
- PAM は、完全に文書化された単一ライブラリーを提供します。開発者は、独自の認証スキームを作成することなくプログラムの作成ができます。

5.1.1. 他の PAM リソース

PAM には、PAM の使用および PAM を他のアプリケーションと統合拡張するためのモジュール作成に関する詳細かつ大規模なドキュメンテーションがあります。PAM と使用するほとんどのメジャーなモジュールおよび設定ファイルには、独自の man ページがあります。さらに、`/usr/share/doc/pam-version#` ディレクトリーには、『System Administrators' Guide』、『Module Writers' Manual』、『Application Developers' Manual』と、PAM の標準である DCE-RFC 86.0 が含まれています。

PAM 用のライブラリーは、<http://www.linux-pam.org> にあります。これは Linux-PAM プロジェクトの主要なディストリビューションウェブサイト、様々な PAM モジュールに関する情報やよくある質問、追加の PAM ドキュメンテーションがあります。

5.1.2. PAM モジュールのカスタマイズ

新規の PAM モジュールは、PAM 対応アプリケーションでいつでも作成、追加ができます。PAM 対応プログラムは、新規モジュールとそれが定義するメソッドを再編集したり修正することなく、即座に使用することができます。このため開発者およびシステム管理者は、異なるプログラム用の認証メソッドを再編集することなく組み合わせたりテストしたりすることができます。

モジュール作成に関するドキュメンテーションは、`/usr/share/doc/pam-devel-version#` に格納されています。

5.2. PAM 設定ファイルについて

PAM 対応アプリケーションまたは サービスはそれぞれ `/etc/pam.d/` ディレクトリーにファイルがあり、これにはファイルがアクセスを制御するサービスと同じ名前が付けられています。たとえば、`login` プログラムはサービス名を `login` と定義し、`/etc/pam.d/login` という名前の PAM 設定ファイルをインストールします。



警告

PAM の設定には、PAM 設定ファイルを手動で編集するのではなく、`authconfig` ツールを使用することが強く推奨されます。

5.2.1. PAM 設定ファイルの書式

各 PAM 設定ファイルには、モジュール (認証設定エリア) を定義するディレクティブとその制御もしくは属性が含まれています。

ディレクティブの構文はすべて簡単なもので、モジュールの目的 (インターフェイス) とモジュールの設定を特定します。

```
module_interface control_flag module_name module_arguments
```

PAM 設定ファイルでは、モジュールインターフェイスは以下のように最初のフィールドで定義されます。

```
auth required pam_unix.so
```

PAM インターフェイスとは、本質的にはその特定のモジュールが実行可能な認証アクションのタイプのことです。PAM モジュールインターフェイスでは 4 つのタイプが利用可能で、それぞれが認証および承認プロセスの別の要素に対応しています。

- ※ **auth** — このモジュールインターフェイスは、ユーザーを認証します。たとえば、パスワードの有効性を要求したり検証したりします。このインターフェイスがあるモジュールは、グループメンバーシップといった認証情報も設定できます。
- ※ **account** — このモジュールインターフェイスは、アクセスが許可されたことを確認します。たとえば、ユーザーアカウントの期限が切れたか、またはユーザーが 1 日の特定の時間にログインを許可されるかどうかをチェックします。
- ※ **password** — このモジュールインターフェイスは、ユーザーのパスワード変更に使用されます。
- ※ **session** — このモジュールインターフェイスは、ユーザーセッションを設定、管理します。このインターフェイスのあるモジュールは、ユーザーのホームディレクトリーをマウントしたり、ユーザーのメールボックスを利用可能にするなど、アクセスの許可を必要とする追加タスクも実行できます。

個別のモジュールは、いずれかのインターフェイス、またはすべてのインターフェイスを提供できます。たとえば、`pam_unix.so` は 4 つすべてのモジュールインターフェイスを提供します。

`pam_unix.so` といったモジュール名は、指定されたモジュールインターフェイスのライブラリー名を PAM に提供します。ディレクトリー名は省略されています。アプリケーションが適切なバージョンの `libpam` にリンクされており、これがモジュールの適切なバージョンを見つけることができるためです。

PAM モジュールはすべて、呼び出されると成功か失敗の結果を生成します。**制御フラグ** が PAM に結果をどうするかについて指示します。モジュールは特定の順番で記載することができ (**スタック**)、ユーザーのサービスへの認証全体において特定モジュールの成功もしくは失敗の重要性を制御フラグが判断します。

フラグには簡単なものがいくつかあり ^[3]、これらの設定にはキーワードのみを使用します。

- ※ **required** — 認証を継続するには、モジュール結果が成功する必要があります。この時点でテストが失敗すると、該当インターフェイスを参照するすべてのモジュールテストの結果が完了するまでユーザーには通知されません。

- ※ **requisite** — 認証を継続するには、モジュール結果が成功する必要があります。ただし、この時点でテストが失敗するとユーザーに直ちに通知され、そのメッセージには最初に失敗した **required** または **requisite** モジュールテストが反映されます。
- ※ **sufficient** — モジュール結果は失敗した場合でも無視されます。ただし、**sufficient** のフラグの付いたモジュールが成功して、かつ **required** のフラグが付いたモジュールがそれまでに失敗していなければ、それ以上の結果は要求されず、ユーザーはサービスに認証されます。
- ※ **optional** — モジュール結果は無視されます。**optional** のフラグの付いたモジュールは、他のモジュールが該当インターフェイスを参照しない場合にのみ、認証成功に必要となります。
- ※ **include** — 他の制御とは違い、これはモジュール結果の処理には関係しません。このフラグは、特定パラメーターに合致する設定ファイル内のすべての行を取得し、それらを引数としてモジュールに追加します。

モジュールインターフェイスのディレクティブは、重ねて配置することで **スタック化** が可能なので、複数のモジュールをまとめて 1 つの目的に使用することができます。



注記

モジュールの制御フラグが **sufficient** または **requisite** の値を使用している場合、モジュールの記載順序は認証プロセスで重要になります。

スタック化により管理者は、ユーザー認証を許可する前提条件を設定することが容易になります。たとえば、**setup** ユーティリティは通常、PAM 設定ファイルにあるように、複数のスタック化されたモジュールを使用します。

```
[root@MyServer ~]# cat /etc/pam.d/setup
```

```
auth      sufficient pam_rootok.so
auth      include system-auth
account   required pam_permit.so
session   required pam_permit.so
```

- ※ **auth sufficient pam_rootok.so** — この行は **pam_rootok.so** モジュールを使用して、現行ユーザーの UID が 0 であることを確認して、ユーザーが root かどうかをチェックします。テストが成功すると、他のモジュールは参照されず、コマンドが実行されます。テストが失敗すると、次のモジュールに移ります。
- ※ **auth include system-auth** — この行には **/etc/pam.d/system-auth** モジュールのコンテンツが含まれており、認証のためにこのコンテンツを処理します。
- ※ **account required pam_permit.so** — この行は **pam_permit.so** モジュールを使って、root ユーザーもしくはコンソールにログインしているユーザーがシステム再起動をできるようにします。
- ※ **session required pam_permit.so** — この行は、セッション設定に関するものです。**pam_permit.so** を使って **setup** ユーティリティが失敗しないようにします。

PAM はいくつかのモジュール向けの認証中に **引数** を使って情報をプラグ可能なモジュールに渡します。

たとえば、**pam_pwquality.so** モジュールはパスワードの強度をチェックし、複数の引数を取ることができます。以下の例では、**enforce_for_root** で root ユーザーのパスワードでも強度チェックをパスする必要があることを指定し、**retry** でユーザーが強固なパスワードを 3 回入力できることを定義しています。

```
password requisite pam_pwquality.so enforce_for_root retry=3
```

無効な引数は通常無視され、PAM モジュールの成功や失敗には影響しません。ただし、モジュールのなかには無効な引数で失敗するものもあります。ほとんどのモジュールはエラーを `journald` サービスに報告します。`journald` の使用方法と関連の `journalctl` ツールに関する情報は、[システム管理者のガイド](#) を参照してください。

注記

`journald` サービスは Red Hat Enterprise Linux 7.1 で導入されました。Red Hat Enterprise Linux の以前のバージョンでは、ほとんどのモジュールはエラーを `/var/log/secure` ファイルに報告します。

5.2.2. 注釈付きの PAM 設定例

[例5.1「シンプルな PAM 設定」](#) は、PAM アプリケーション設定ファイルのサンプルです。

例5.1 シンプルな PAM 設定

```
#%PAM-1.0
auth required pam_securetty.so
auth required pam_unix.so nullok
auth required pam_nologin.so
account required pam_unix.so
password required pam_pwquality.so retry=3
password required pam_unix.so shadow nullok use_authok
session required pam_unix.so
```

- 最初の行は、行頭のハッシュ記号 (#) が示すように、コメントになります。
- 2 行目から 4 行目は、ログイン認証用に 3 つのモジュールをスタックしています。

auth required pam_securetty.so — このモジュールは、もしユーザーが root としてログインしようとしている場合、ユーザーがログインしている tty が `/etc/securetty` ファイルが存在する場合に、そのファイルに記載されていることを確認します。

tty がファイルに記載されていない場合は、root としてログインしようとする、**Login incorrect** メッセージとともに失敗します。

auth required pam_unix.so nullok — このモジュールはユーザーにパスワードを要求し、`/etc/passwd` と、存在する場合は `/etc/shadow` に保存されている情報を使用して、パスワードを確認します。

引数 `nullok` は `pam_unix.so` モジュールに空白のパスワードを許可するよう指示します。

- **auth required pam_nologin.so** — これは認証の最終ステップです。`/etc/nologin` ファイルが存在するかどうかを確認します。このファイルが存在してユーザーが root でない場合は、認証に失敗します。

注記

この例では、最初の **auth** モジュールが失敗しても、3 つすべての **auth** モジュールが確認されます。これにより、どの段階で認証に失敗したかをユーザーに知らせずに済みます。この情報が攻撃者の手に渡ると、システムの攻撃方法がより簡単に推測されてしまいます。

- **account required pam_unix.so** — このモジュールは、必要なアカウント確認を実行します。たとえば、

シャドウパスワードが有効になっていると、`pam_unix.so` モジュールのアカウントインターフェイスはアカウントの有効期間が切れているかどうか、またはユーザーが許可されている猶予期間内にパスワードを変更していないかを確認します。

- ※ `password required pam_pwquality.so retry=3` — パスワードの有効期間が切れている場合は、`pam_pwquality.so` モジュールのパスワードコンポーネントが新たなパスワードを要求します。そして、新規に作成されたパスワードが辞書ベースのパスワード解読プログラムで容易に判断できるかどうかをテストします。

引数 `retry=3` は、テストに 1 回失敗しても、ユーザーは強固なパスワードを作成する機会があと 2 回あることを示しています。

- ※ `password required pam_unix.so shadow nullok use_authtok` — この行は、`pam_unix.so` モジュールの `password` インターフェイスを使って、プログラムがユーザーのパスワードを変更するかどうかを指定します。
 - 引数 `shadow` は、ユーザーのパスワード更新の際にシャドウパスワードを作成するようモジュールに指示します。
 - 引数 `nullok` は、ユーザーが空白のパスワードから変更できるようにし、それ以外の場合は `null` パスワードをアカウントロックとして扱うようモジュールに指示します。
 - この行の最後の引数 `use_authtok` は、PAM モジュールのスタック化の際における順序の重要性の例を提供します。この引数は、ユーザーに新規パスワードを要求しないようモジュールに指示します。代わりに、以前のパスワードモジュールが記録したパスワードを受け入れます。これにより、新規パスワードはすべて、受け入れ前にパスワードの安全テスト `pam_pwquality.so` をパスする必要があります。
- ※ `session required pam_unix.so` — 最後の行は、`pam_unix.so` モジュールのセッション引数にセッションを管理するよう指示します。このモジュールはユーザー名とサービスタイプを各セッションの最初と最後に `/var/log/secure` に記録します。このモジュールは他のセッションモジュールとスタック化した追加機能を補うことができます。

5.3. PAM と管理認証情報のキャッシング

GNOME の `control-center` など、Red Hat Enterprise Linux 内の多くのグラフィカル管理ツールは、システム特権を持つユーザーに最大 5 分間の `pam_timestamp.so` モジュール使用を提供します。`pam_timestamp.so` が実行中の端末をユーザーが離れると、このコンソールに物理的にアクセス可能な人物がマシンを操作できるようになってしまうため、このメカニズムの作動方法を理解することは重要になります。

PAM タイムスタンプスキームでは、グラフィカルの管理アプリケーションは起動時に `root` パスワードをユーザーに求めます。ユーザーが認証されると、`pam_timestamp.so` モジュールはタイムスタンプファイルを作成します。デフォルトでは、これは `/var/run/sudo/` ディレクトリーに作成されます。既にタイムスタンプファイルがある場合は、グラフィカル管理プログラムはパスワードを要求しません。代わりに `pam_timestamp.so` モジュールはタイムスタンプファイルをリフレッシュして、ユーザーに無条件の管理者アクセスをさらに 5 分間確保します。

タイムスタンプファイルの実際の状態は、`/var/run/sudo/user` ディレクトリーで確認できます。デスクトップの場合は、関連ファイルは `unknown:root` になります。これが存在してそのタイムスタンプが 5 分未満の場合、認証情報は有効です。

タイムスタンプファイルが存在すると、パネルの通知スペースに認証アイコンが表示されます。



図5.1 認証アイコン

5.3.1. タイムスタンプファイルを削除する

PAM タイムスタンプがアクティブな状態のコンソールは、放置する前にタイムスタンプファイルを破棄することが推奨されます。グラフィカル環境でこれを実行するには、パネルの認証アイコンをクリックします。これでダイアログボックスが開きます。**Forget Authorization** ボタンをクリックしてアクティブなタイムスタンプファイルを破棄します。

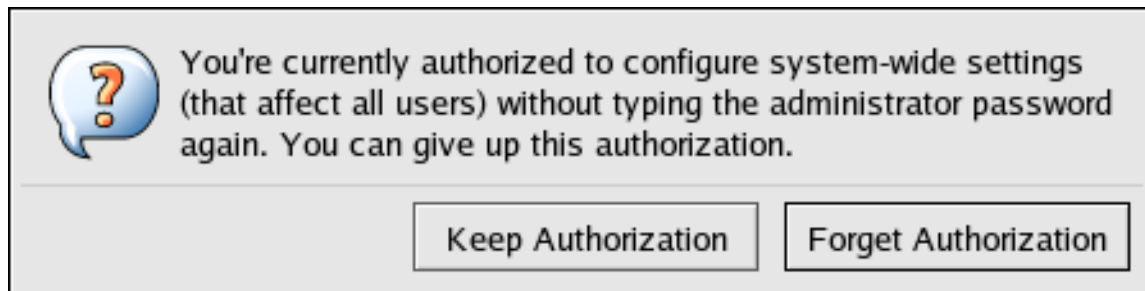


図5.2 認証ダイアログを閉じる

PAM タイムスタンプファイルには、以下の重要な特徴があります。

- ※ `ssh` を使用してシステムにリモートでログインしている場合は、`/sbin/pam_timestamp_check -k root` コマンドを使用してタイムスタンプファイルを破棄します。
- ※ 権限のあるアプリケーションが起動されたものと同じ端末ウィンドウから `/sbin/pam_timestamp_check -k root` コマンドを実行します。
- ※ `pam_timestamp.so` モジュールを最初に起動したログイン済みユーザーは、`/sbin/pam_timestamp_check -k` コマンドを実行するユーザーである必要があります。`root` でこのコマンドを実行しないでください。
- ※ デスクトップでアイコン上の **Forget Authorization** アクションを使用せずに認証情報を破棄するには、`/sbin/pam_timestamp_chec` コマンドを使用します。

```
/sbin/pam_timestamp_check -k root </dev/null >/dev/null 2>/dev/null
```

他の方法は、コマンドが実行される `pty` から認証情報を削除するだけです。

`pam_timestamp_check` を使用してタイムスタンプファイルを破棄する詳細情報については、`pam_timestamp_check man` ページを参照してください。

5.3.2. 一般的な `pam_timestamp` ディレクティブ

`pam_timestamp.so` モジュールはいくつかのディレクティブを受け付けます。最も一般的なものは以下の 2 つです。

- ※ `timestamp_timeout` — タイムスタンプの有効期間を秒単位で指定します。デフォルト値は 300 (5 分間) です。
- ※ `timestampdir` — タイムスタンプファイルの保存先となるディレクトリーを指定します。デフォルト値は `/var/run/sudo/` です。

[3] 設定可能な制御フラグには複雑なものもあります。これらは、`属性=値` のペアで設定します。属性の完全な一覧は、`pam.d man` ページで確認できます。

第6章 Kerberos の使用

ネットワーク内でのシステムのセキュリティと整合性の維持は非常に重要であり、これはネットワークインフラストラクチャー内のすべてのユーザー、アプリケーション、サービスおよびサーバーに及びます。この維持には、ネットワーク上で実行中のすべてのサービスやそれらがどのように使用されているかを理解していることが必要です。このセキュリティ維持の中心的なタスクは、アプリケーションやサービスへのアクセスの維持と、そのアクセスの実施になります。

Kerberos は、ユーザーとマシンの両方がネットワークに対して自らを識別し、管理者が設定した領域とサービスへの定義済みかつ制限されたアクセスをユーザーとマシンが受けられるようにするメカニズムを提供します。Kerberos はエンティティの ID を検証してそれらを認証するほか、この認証情報データを保護することで、外部の人間によるこのデータへのアクセス、使用または改ざんを防ぎます。

6.1. Kerberos について

Kerberos は対照鍵暗号 [4] を用いてネットワークサービスに対してユーザーを認証します。つまり、パスワードがネットワーク経由で送信されることはありません。

そのため、ユーザーが Kerberos を使用してネットワークサービスに対して認証を行う際に、ネットワークトラフィックを監視してパスワードの収集を図っている不正なユーザーを効果的に阻止することができます。

6.1.1. Kerberos の基本的なしくみ

従来のほとんどのネットワークサービスではパスワードベースの認証スキームを使用しており、その場合はユーザーが特定のネットワークサーバーにアクセスするためにパスワードを提供します。ただし、多くのサービスにおける認証情報は暗号化されずに送信されています。このようなスキームをセキュアにするには、ネットワークを外部からアクセス不可とし、ネットワーク上のすべてのコンピューターとユーザーを信頼し、信頼可能とする必要があります。

シンプルなパスワードベースの認証を使う際には、インターネットに接続されているネットワークが安全であるとは想定できません。ネットワークにアクセスする攻撃者は誰でもパケットアナライザーもしくはパケットスニフアーを使用してユーザー名とパスワードを傍受し、ユーザーアカウントの安全性を脅かすことができるので、セキュリティインフラストラクチャー全体の整合性が脅かされます。

Kerberos はネットワーク経由で暗号化されていないパスワード送信をなくし、攻撃者がネットワークを傍受する潜在的脅威を取り除きます。

シンプルなパスワード認証で個別のユーザーが個別のネットワークサービスに対して認証を行うのではなく、Kerberos は対照暗号と信頼できるサードパーティ(キー配布センター KDC) を用いてユーザーをネットワークサービスのスイートに対して認証します。その KDC とセカンダリー KDC が管理するコンピューターが **レルム** を構成します。

ユーザーが KDC に対して認証を行うと、KDC はそのセッションに特定した認証情報のセット (**チケット**) をユーザーのマシンに送り返します。Kerberos 対応のサービスでは、ユーザーがパスワードを使用して認証する必要はなく、サービスすべてがユーザーのマシン上でこのチケットを探します。

[図6.1「Kerberos 認証のステップ」](#)にあるように、各ユーザーは一意的 ID で KDC に識別されます。この ID は **プリンシパル** と呼ばれます。Kerberos 対応ネットワーク上でユーザーがワークステーションにログインすると、このユーザーのプリンシパルが認証サーバーから **ticket-granting ticket** (または TGT) のリクエストの一部として KDC に送信されます。このリクエストはログインプログラムによって送信されてユーザーに対して透過性を持つようにするか、ユーザーがログイン後に手動で **krinit** プログラムを用いて送信することができます。

すると KDC はデータベース内でプリンシパルを確認します。プリンシパルが見つかったら、KDC は TGT を作成し、ユーザーの鍵を使ってこれを暗号化し、TGT をそのユーザーに送信します。

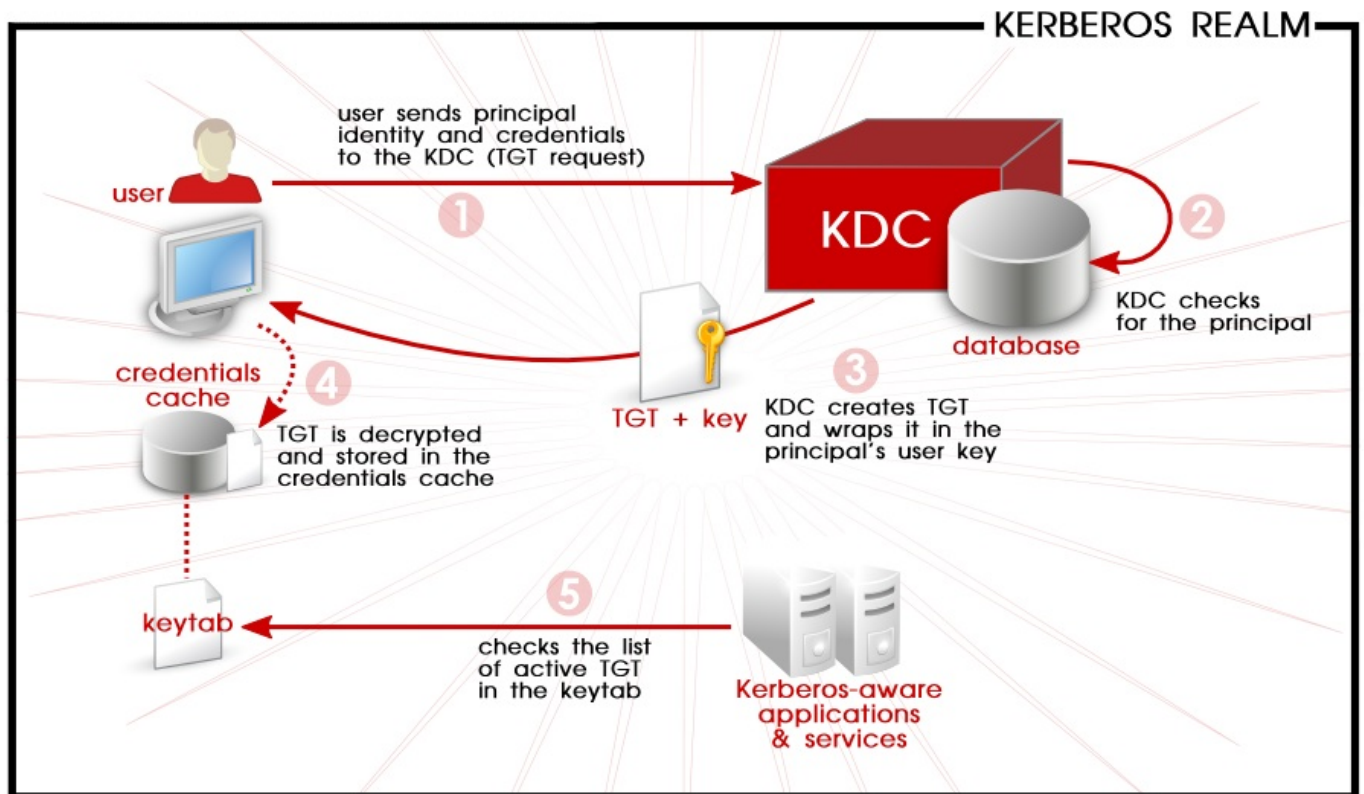


図6.1 Kerberos 認証のステップ

次にクライアント上のログインもしくは `kinit` プログラムがユーザーの鍵を使って TGT を暗号解除します。ユーザーの鍵は、ログインもしくは `kinit` プログラムがユーザーのパスワードから計算します。ユーザーの鍵はクライアントマシン上でのみ使用され、ネットワーク経由では送信されません。KDC が送信したチケット (または認証情報) はローカルストアである **認証情報キャッシュ (ccache)** に保存されます。Kerberos 対応サービスはこのキャッシュをチェックすることができます。Red Hat Enterprise Linux 7 では、以下のタイプの認証情報キャッシュに対応しています。

- ✧ KEYRING。永続性のある KEYRING ccache タイプが Red Hat Enterprise Linux 7 のデフォルトになります。
- ✧ FILE
- ✧ DIR
- ✧ MEMORY

認証後は、サーバーは `kinit` を確認するのではなく、認識されたプリンシパルとそれらの鍵の暗号化されていないリストをチェックできます。このリストは **keytab** に保持されます。

TGT は特定期間の後に (通常は 10 から 24 時間) 有効期限が切れるように設定され、クライアントマシンの認証情報キャッシュに保存されます。TGT に期限が設定されているのは、攻撃者が TGT を使用できる期間を短時間にするためです。TGT の発行後は、TGT の有効期限が切れるまで、またはユーザーがログアウトして再度ログインするまで、ユーザーはパスワードを再入力する必要がありません。

ユーザーがネットワークサービスにアクセスする必要がある場合は、クライアントのソフトウェアは TGT を使って ticket-granting サーバー (TGS) から該当サービス用の新規チケットを要求します。サービスチケットは該当サービスに対するユーザーの認証を透過的に行うために使用されます。

6.1.2. ドメインからレルムへのマッピング

クライアントが特定サーバー上で実行中のサービスにアクセスしようとする際は、サービス名 (**ホスト**) とサーバー名 (**foo.example.com**) は分かっていますが、ネットワーク上には複数のレルムが導入されることが可能なので、クライアントはサービスが存在する Kerberos レルムの名前を推測する必要があります。

デフォルトでは、レルム名はサーバーのドメイン名をすべて大文字にしたものになります。

```
foo.example.org → EXAMPLE.ORG
foo.example.com → EXAMPLE.COM
foo.hq.example.com → HQ.EXAMPLE.COM
```

設定によってはこれで十分な場合もありますが、派生したレルム名が存在しないレルムの名前になることもあります。そのような場合には、サーバーの DNS ドメイン名からレルム名へのマッピングをクライアントシステムの `/etc/krb5.conf` ファイルの `domain_realm` セクションで指定する必要があります。例を示します。

```
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

上記の設定では、2 つのマッピングを指定しています。最初のマッピングでは、`example.com` DNS ドメインのシステムはすべて **EXAMPLE.COM** レルムに所属すると指定しています。2 つ目では、`example.com` という名前そのものも同じレルムに所属することを指定しています。ドメインと特定ホストの違いは、最初のピリオド記号の有無で決まります。このマッピングは `"_kerberos TXT"` レコードを使って DNS に直接保存することもできます。例を示します。

```
$ORIGIN example.com
_kerberos TXT "EXAMPLE.COM"
```

6.1.3. 環境要件

Kerberos はマシン名の解決に依存していることから、作業中のドメインネームサービス (DNS) を必要とします。ネットワーク上の DNS エントリとホストは両方とも適切に設定する必要があります。これは `/usr/share/doc/krb5-server-version-number` にある Kerberos ドキュメンテーションで説明されています。

Kerberos 認証を許可するアプリケーションは、時間の同期を必要とします。ネットワーク上のマシン間のクロック同期は、`ntpd` などのサービスを使って設定できます。`ntpd` サービスに関する情報は、`/usr/share/doc/ntp-version-number/html/index.html` にあるドキュメンテーションか `ntpd(8) man` ページを参照してください。



注記

Red Hat Enterprise Linux 7 上で実行中の Kerberos クライアントは KDC との自動時間調整をサポートしており、厳密な時間要件はありません。これにより、Red Hat Enterprise Linux 7 で IdM クライアントを導入する際には時間の差異に対する耐性が高くなります。

6.1.4. Kerberos 導入における考慮点

Kerberos は一般的かつ重大なセキュリティの脅威を排除しますが、以下のような理由から実装は容易ではありません。

- ※ Kerberos は、各ユーザーは信頼されているものの、信頼できないネットワーク上で信頼できないホストを使用していることを想定しています。Kerberos の第一の目的は、暗号化されていないパスワードをネットワーク経由で送信されないようにすることです。ただし、認証に使用されるチケットを発行するあるホスト (KDC) に適当なユーザー以外の誰かがアクセスすると、Kerberos 認証システム全体が危険にさらされます。
- ※ アプリケーションが Kerberos を使用するには、そのソースを修正して Kerberos ライブラリーに適切なコールをするようにする必要があります。このような修正を受けたアプリケーションは、*Kerberos* 対応もしくは

kerberized とみなされます。アプリケーションのなかには、そのサイズやデザインが理由でこれが問題になるものもあります。また、互換性のない他のアプリケーションでは、サーバーとクライアントが通信する方法を変更する必要があります。これはかなり大幅なプログラミングになる可能性があります。多くの場合、デフォルトで Kerberos 対応となっていないクローズソースのアプリケーションが最も問題のあるものとなります。

- Kerberos は妥協のできないソリューションです。Kerberos でネットワークの安全を図るには、暗号化されていないパスワードを送信するすべてのクライアント/サーバーアプリケーションのバージョンで Kerberos 対応のものを使用するか、そのようなクライアント/サーバーアプリケーションをまったく使用しないかのどちらかにする必要があります。
- `/etc/passwd` や `/etc/shadow` といった標準 UNIX パスワードデータベースから Kerberos パスワードデータベースへのユーザーパスワードの移行は、面倒な作業になります。このタスクを実行する自動メカニズムはありません。移行方法は、Kerberos を導入する方法によって大幅に異なります。Identity Management 機能の使用が推奨されるのは、このためです。これには移行のための特別のツールとメソッドが備わっています。



警告

ネットワーク上のユーザーが Kerberos 非対応サービスに対してプレーンテキストでパスワードを送信すると、Kerberos システムは危険にさらされます。(telnet および FTP を含む) Kerberos 非対応サービスを使用しないことが強く推奨されます。SSH や SSL でセキュアとなったサービスのような他の暗号化プロトコルは暗号化されていないサービスよりは優れているものの、それでも理想的なものではありません。

6.1.5. Kerberos に関するその他のリソース

Kerberos の導入には柔軟性があることから、その実装は複雑なサービスとすることが可能です。[表6.1「外部の Kerberos 資料」](#)と [表6.2「重要な Kerberos Man ページ」](#)では、Kerberos の使用に関する詳細情報で最ももしくは最も有益なソースを一覧表示しています。

表6.1 外部の Kerberos 資料

資料	場所
Kerberos V5 Installation Guide (PostScript と HTML の両方)	<code>/usr/share/doc/krb5-server-version-number</code>
Kerberos V5 System Administrator's Guide (PostScript と HTML の両方)	<code>/usr/share/doc/krb5-server-version-number</code>
Kerberos V5 UNIX User's Guide (PostScript と HTML の両方)	<code>/usr/share/doc/krb5-workstation-version-number</code>
"Kerberos: The Network Authentication Protocol" (MIT のウェブページ)	http://web.mit.edu/kerberos/www/
The Kerberos Frequently Asked Questions (FAQ)	http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html
『Kerberos: An Authentication Service for Open Network Systems』Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schille 著。Kerberos に関するオリジナルの論文。PostScript 形式	ftp://athena-dist.mit.edu/pub/kerberos/doc/usenix.PS
『Designing an Authentication System: a Dialogue in Four Scenes』、Bill Bryant 著 1988。Theodore Ts'o による修正 1997。このドキュメントは、2 人の開発者による Kerberos スタイルの認証システムについての考察です。2 人の議論は会話形式となっており、Kerberos をまったく知らない読者にとっても分かりやすいものになっています。	http://web.mit.edu/kerberos/www/dialogue.html

資料	場所
ネットワークを kerberos 化する方法についての記事です。	http://www.ornl.gov/~jar/HowToKerb.html
『Kerberos Network Design Manual』は、Kerberos システムについての包括的な概要です。	http://www.networkcomputing.com/netdesign/kerb1.html

man ページのファイルは、`man command_name` を実行すると開きます。

表6.2 重要な Kerberos Man ページ

Man ページ	説明
クライアントアプリケーション	
kerberos	Kerberos システムへの導入では、認証情報がどのように機能するかが説明され、Kerberos チケットの取得および破棄に関する推奨事項が提供されます。man ページの下部では、関連する man ページが多く参照されています。
kinit	このコマンドを使用して ticket-granting ticket を取得、キャッシュする方法が説明されています。
kdestroy	このコマンドを使用して Kerberos 認証情報を破棄する方法が説明されています。
klist	このコマンドを使用して、キャッシュされた Kerberos 認証情報を一覧表示する方法が説明されています。
管理アプリケーション	
kadmin	このコマンドを使用して Kerberos V5 データベースを管理する方法が説明されています。
kdb5_util	このコマンドを使用して Kerberos V5 データベース上で低レベルの管理機能を作成、実行する方法が説明されています。
サーバーアプリケーション	
krb5kdc	Kerberos V5 KDC に利用可能なコマンドラインオプションが説明されています。
kadmind	Kerberos V5 管理サーバー に利用可能なコマンドラインオプションが説明されています。
設定ファイル	
krb5.conf	Kerberos V5 ライブラリー用の設定ファイル内で利用可能な形式とオプションを説明しています。
kdc.conf	Kerberos V5 AS および KDC 用の設定ファイル内で利用可能な形式とオプションを説明しています。

6.2. Kerberos KDC の設定

マスター KDC を最初にインストールして設定した後に、必要なセカンダリーサーバーをインストールします。



重要

Kerberos KDC を手動で設定することは推奨されません。Red Hat Enterprise Linux 環境に Kerberos を導入する際は、Identity Management 機能を使用することが推奨されます。

6.2.1. マスター KDC サーバーの設定



重要

KDC システムは専用マシンにしてください。このマシンは非常に安全である必要があります。可能な場合は、このマシン上で KDC 以外のサービスを実行しないでください。

1. KDC に必要なパッケージをインストールします。

```
[root@server ~]# yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
```

2. `/etc/krb5.conf` と `/var/kerberos/krb5kdc/kdc.conf` の設定ファイルをレルム名とドメインからレルムへのマッピングを反映させるように編集します。例を示します。

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
allow_weak_crypto = true

[realms]
EXAMPLE.COM = {
    kdc = kdc.example.com:88
    admin_server = kdc.example.com
    default_domain = example.com
}

[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

シンプルなレルムは、`EXAMPLE.COM` と `example.com` を適切なドメイン名で置き換え (大文字と小文字を正確な形式になるよう確認してください)、KDC を `kerberos.example.com` から Kerberos サーバーに変更することで構成できます。慣例では、レルム名はすべて大文字、DNS ホスト名およびドメイン名はすべて小文字になります。これら設定ファイルの man ページには、ファイル形式に関する詳細が記載されています。

3. `kdb5_util` ユーティリティを使用してデータベースを作成します。

```
[root@server ~]# kdb5_util create -s
```

`create` コマンドは、Kerberos レルム用の鍵を保存するデータベースを作成します。`-s` 引数は、マスター鍵を保存する `stash` ファイルを作成します。鍵を読み取る `stash` ファイルがない場合、Kerberos サーバー (`krb5kdc`) は起動時に毎回、ユーザーに対してマスターサーバーのパスワード (これを使って鍵を再生成することが可能) を要求します。

4. `/var/kerberos/krb5kdc/kadm5.acl` ファイルを編集します。`kadmind` はこのファイルを使って Kerberos データベースへの管理アクセスがあるプリンシパルとそのアクセスレベルを判断します。ほとんどの組織では、以下の一行で対応できます。

```
*/admin@EXAMPLE.COM *
```

ほとんどのユーザーはデータベース内で単一プリンシパル (`joe@EXAMPLE.COM` のような `NULL` または空白のインスタンス) により表示されます。この設定では、`admin` のインスタンスがある 2 番目のプリンシパル (たとえば `joe/admin@EXAMPLE.COM`) を持つユーザーは、レルムの Kerberos データベースに対して完全な管理制御を行使することができます。

`kadmind` がサーバー上で開始されると、ユーザーはレルム内のクライアントもしくはサーバー上で `kadmind` を実行することでこのサービスにアクセスできます。ただし、データベースの編集が可能なのは (自身のパスワード変更を除く) `kadm5.acl` ファイルに名前が記載されているユーザーのみになります。



注記

`kadmind` ユーティリティーはネットワーク経由で `kadmind` サーバーと通信し、Kerberos を使って認証を処理します。このため、ネットワーク経由でサーバーに接続してこの処理を行う前に、最初のプリンシパルが存在している必要があります。最初のプリンシパルは、`kadmind.local` コマンドで作成します。このコマンドは、KDC と同じホスト上で使用するよう特別に設計されており、認証に Kerberos を使用しません。

5. KDC ターミナルで `kadmind.local` を使用して最初のプリンシパルを作成します。

```
[root@server ~]# kadmind.local -q "addprinc username/admin"
```

6. 以下のコマンドを使用して Kerberos を起動します。

```
[root@server ~]# systemctl start krb5kdc.service
[root@server ~]# systemctl start kadmind.service
```

7. `kadmind` 内で `addprinc` コマンドを使用してユーザーにプリンシパルを追加します。`kadmind` および `kadmind.local` は、KDC のコマンドラインインターフェイスになります。このため、`addprinc` のようなコマンドの多くは、`kadmind` プログラムの起動後に利用可能になります。詳細は、`kadmind man` ページを参照してください。
8. KDC がチケットを発行していることを確認します。まず、`kinit` を実行してチケットを取得し、認証情報キャッシュファイルに保存します。次に `klist` を使ってキャッシュ内の認証情報一覧を表示し、`kdestroy` を使ってキャッシュとそこに含まれる認証情報を破棄します。



注記

デフォルトでは、`kinit` は同一のシステムログインユーザー名 (Kerberos サーバーではなく) を使って認証を試みます。このユーザー名が Kerberos データベース内のプリンシパルに対応しない場合は、`kinit` はエラーメッセージを発行します。この場合は、コマンドラインで `kinit` の引数として適切なプリンシパルの名前を提供してください。

```
kinit principal
```

6.2.2. セカンダリー KDC の設定

あるレルムに複数の KDC がある場合、1 つの KDC (マスター KDC) が書き込み可能なレルムデータベースのコピーを保持し、`kadmin` を実行します。マスター KDC はレルムの管理サーバーでもあります。追加のセカンダリー KDC はデータベースの読み取り専用コピーを保持して、`kpropd` を実行します。

マスターおよびスレーブを伝達するステップでは、マスター KDC がデータベースを一時ダンプファイルにダンプして、そのファイルを各スレーブに送信する必要があります。このファイルは、そのダンプファイルのコンテンツでこれ以前に受信したデータベースの読み取り専用コピーを上書きします。

セカンダリー KDC を設定するには、以下の手順にしたがいます。

1. KDC に必要なパッケージをインストールします。

```
[root@slavekdc ~]# yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
```

2. マスター KDC の `krb5.conf` と `kdc.conf` のファイルをセカンダリー KDC にコピーします。

3. マスター KDC で root シェルから `kadmin.local` を起動します。

- a. `kadmin.local add_principal` コマンドを使ってマスター KDC の `host` サービス用の新規エントリーを作成します。

```
[root@slavekdc ~]# kadmin.local -r EXAMPLE.COM
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin: add_principal -randkey host/masterkdc.example.com
Principal "host/masterkdc.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/masterkdc.example.com
Entry for principal host/masterkdc.example.com with kvno 3, encryption
type Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/krb5.keytab.
Entry for principal host/masterkdc.example.com with kvno 3, encryption
type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/masterkdc.example.com with kvno 3, encryption
type DES with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/masterkdc.example.com with kvno 3, encryption
type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: quit
```

- b. `kadmin.local ktadd` コマンドを使ってサービス用にランダムな鍵を設定し、その鍵をマスターのデフォルト keytab ファイルに保存します。



注記

この鍵は `kprop` コマンドでセカンダリーサーバーへの認証に使用します。インストールされているセカンダリー KDC サーバーの数にかかわらず、この設定が必要なのは 1 回のみです。

4. セカンダリー KDC で root シェルから `kadmin` を起動します。

- a. `kadmin.local add_principal` コマンドを使ってセカンダリー KDC の `host` サービス用の新規エントリーを作成します。

```
[root@slavekdc ~]# kadmin -p jsmith/admin@EXAMPLE.COM -r EXAMPLE.COM
Authenticating as principal jsmith/admin@EXAMPLE.COM with password.
Password for jsmith/admin@EXAMPLE.COM:
kadmin: add_principal -randkey host/slavekdc.example.com
Principal "host/slavekdc.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/slavekdc.example.com@EXAMPLE.COM
Entry for principal host/slavekdc.example.com with kvno 3, encryption
type Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/krb5.keytab.
Entry for principal host/slavekdc.example.com with kvno 3, encryption
type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/slavekdc.example.com with kvno 3, encryption
type DES with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/slavekdc.example.com with kvno 3, encryption
type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: quit
```

- b. **kadmin.local ktadd** コマンドを使ってサービス用にランダム鍵を設定し、その鍵をセカンダリサーバーのデフォルト keytab ファイルに保存します。この鍵は、クライアントの認証時に **kpropd** で使用します。
5. セカンダリ KDC はサービス鍵を使って、接続するクライアントを認証することができます。もちろん、すべてのクライアントが新規レルムデータベースで **kprop** サービスの提供を許可されるべきではありません。アクセスを制限するために、セカンダリ KDC 上の **kprop** サービスは、そのプリンシパル名が **/var/kerberos/krb5kdc/kpropd.acl** に記載されているクライアントからの更新しか受け付けません。

マスター KDC のホストサービス名をこのファイルに追加します。

```
[root@slavekdc ~]# echo host/masterkdc.example.com@EXAMPLE.COM >
/var/kerberos/krb5kdc/kpropd.acl
```

6. セカンダリ KDC がデータベースのコピーを取得すると、その暗号化に使用されたマスター鍵も必要になります。KDC データベースのマスター鍵がマスター KDC 上の **stash** ファイル (通常、**/var/kerberos/krb5kdc/.k5.REALM**) に保存されている場合は、これを安全な方法でセカンダリ KDC にコピーするか、**kdb5_util create -s** を実行してセカンダリ KDC 上にダミーデータベースおよび同一の **stash** ファイルを作成し、同じパスワードを提供します。ダミーデータベースは、データベースが最初に正常に伝達される際に上書きされます。
7. セカンダリ KDC のファイアウォールで、マスター KDC がポート 754 上の TCP (**krb5_prop**) を使用して接続し、**kprop** サービスを開始できるように許可されていることを確認します。
8. **kadmin** サービスが無効になっていることを再確認します。
9. マスター KDC 上のレルムデータベースを、**kprop** コマンドが読み取るデフォルトのデータファイル (**/var/kerberos/krb5kdc/slave_datatrans**) にダンプして、手動でのデータベース伝達テストを実行します。

```
[root@masterkdc ~]# kdb5_util dump /var/kerberos/krb5kdc/slave_datatrans
```

10. **kprop** コマンドを使用して、そのコンテンツをセカンダリ KDC に送信します。

```
[root@slavekdc ~]# kprop slavekdc.example.com
```

11. **kinit** を使用して、クライアントシステムが KDC から正常に初回認証情報を取得できることを確認します。

クライアントの `/etc/krb5.conf` は、KDC 一覧内にセカンダリー KDC のみを記載しているはずですが。

```
[realms]
EXAMPLE.COM = {
  kdc = slavekdc.example.com:88
  admin_server = kdc.example.com
  default_domain = example.com
}
```

12. レルムデータベースをダンプし、`kprop` コマンドの実行によりデータベースを各セカンダリー KDC に送信するスクリプトを作成します。このスクリプトを定期的に行うように `cron` サービスを設定します。

6.3. Kerberos クライアントの設定

Kerberos 5 クライアントの設定に必要なのは、クライアントパッケージをインストールし、各クライアントに有効な `krb5.conf` 設定ファイルを提供することです。`ssh` および `slogin` がクライアントシステムへのリモートでのログイン方法として推奨されますが、`rsh` および `rlogin` の Kerberos 化バージョンも追加の設定変更で利用可能になります。

1. `krb5-libs` および `krb5-workstation` のパッケージをすべてのクライアントマシンにインストールします。

```
[root@server ~]# yum install krb5-workstation krb5-libs krb5-auth-dialog
```

2. 各クライアントに有効な `/etc/krb5.conf` ファイルを提供します (これは通常、KDC が使用する `krb5.conf` ファイルと同じものです)。例を示します。

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
allow_weak_crypto = true

[realms]
EXAMPLE.COM = {
  kdc = kdc.example.com:88
  admin_server = kdc.example.com
  default_domain = example.com
}

[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

3. Kerberos 化した `rsh` と `rlogin` のサービスを使用するために、`rsh` パッケージをインストールします。
4. ワークステーションで Kerberos を使って、`ssh`、`rsh`、または `rlogin` を使用して接続するユーザーを認証

する前に、Kerberos データベースに独自のホストプリンシパルが必要になります。`sshd`、`kshd`、および `klogind` の各サーバープログラムは、すべてホストサービスのプリンシパルの鍵へのアクセスが必要になります。

- a. `kadmin` を使ってワークステーション用のホストプリンシパルを KDC 上に追加します。このケースでのインスタンスは、ワークステーションのホスト名になります。`kadmin` の `addprinc` コマンドに `-randkey` オプションを使用してプリンシパルを作成し、それをランダムな鍵に割り当てます。

```
addprinc -randkey host/server.example.com
```

- b. ワークステーション用の鍵は、`kadmin` をワークステーション上で実行して、`ktadd` コマンドを使用すると抽出できます。

```
ktadd -k /etc/krb5.keytab host/server.example.com
```

5. Kerberos 化したネットワークサービスを使用するには、`krb5-server` パッケージをインストールしてサービスを起動します。Kerberos 化したサービスは [表6.3「一般的な Kerberos 化したサービス」](#) に一覧表示されています。

表6.3 一般的な Kerberos 化したサービス

サービス名	使用方法
ssh	クライアントおよびサーバー両方の設定で <code>GSSAPIAuthentication</code> が有効になっている場合、OpenSSH は GSS-API を使ってサーバーへのユーザー認証を行います。クライアントの <code>GSSAPIDelegateCredentials</code> も有効になっている場合は、ユーザーの認証情報がリモートシステムでも利用可能になります。
rsh および rlogin	<code>klogin</code> 、 <code>eklogin</code> 、および <code>kshell</code> を有効にします。
Telnet	<code>krb5-telnet</code> を有効にします。
FTP	<code>ftp</code> の <code>root</code> でプリンシパル用の鍵を作成、抽出します。インスタンスを FTP サーバーの完全修飾ホスト名に設定してから、 <code>gssftp</code> を有効にします。
IMAP	<code>cyrus-sasl-gssapi</code> パッケージもインストールされていれば、 <code>cyrus-imap</code> パッケージは Kerberos 5 を使用します。 <code>cyrus-sasl-gssapi</code> パッケージには Cyrus SASL プラグインが含まれており、これは GSS-API 認証に対応しています。Cyrus IMAP は、 <code>cyrus</code> ユーザーが <code>/etc/krb5.keytab</code> で適切な鍵を見つけることができ、プリンシパルの <code>root</code> が <code>imap</code> (<code>kadmin</code> で作成) に設定されていれば、Kerberos と正常に機能します。 <code>cyrus-imap</code> の代わりは <code>dovecot</code> パッケージ内にあり、これは Red Hat Enterprise Linux に含まれています。このパッケージには IMAP サーバーは含まれていますが、現在のところ GSS-API と Kerberos には対応していません。

6.4. スマートカード用の Kerberos クライアントの設定

スマートカードは Kerberos との使用が可能です。スマートカード上で X.509 (SSL) ユーザー証明書を認識するための追加設定が必要になります。

1. 他のクライアントパッケージと共に、必要となる PKI/OpenSSL パッケージをインストールします。

```
[root@server ~]# yum install krb5-pkinit-openssl
[root@server ~]# yum install krb5-workstation krb5-libs krb5-auth-dialog
```

2. `/etc/krb5.conf` 設定ファイルを編集し、公開鍵インフラストラクチャー (PKI) 用のパラメーターを設定ファイルの `[realms]` セクションに追加します。`pkinit_anchors` パラメーターは CA 証明書バンドルファイルの場所を設定します。

```
[realms]
EXAMPLE.COM = {
    kdc = kdc.example.com:88
    admin_server = kdc.example.com
    default_domain = example.com
    ...
    pkinit_anchors = FILE:/usr/local/example.com.crt
}
```

3. スマートカード認証 (`/etc/pam.d/smartcard-auth`) およびシステム認証 (`/etc/pam.d/system-auth`) 用の PAM 設定に PKI モジュール情報を追加します。両ファイルに追加する行は、以下のようになります。

```
auth optional pam_krb5.so use_first_pass no_subsequent_prompt
preauth_options=X509_user_identity=PKCS11:/usr/lib64/pkcs11/libcoolkeypk11.so
```

6.5. レルム間 Kerberos 信頼の設定

Kerberos v5 レルムは、接続されているすべてのマスターおよびスレーブ上の Kerberos データベースで定義されている Kerberos プリンシパルのセットです。異なるレルム間のプリンシパルが相互に通信できるようにするには、レルム間 Kerberos 信頼を設定する必要があります。

混在環境の場合と同様に、多くの Linux 環境でシングルサインオンやアプリケーション認証、ユーザー管理用に Kerberos レルムが導入されています。この場合、Kerberos は異なるドメインや (Windows-Linux などの) 混在環境における共通の統合パスとなり得ます。Linux 環境で Identity Management のようなより構成化されたドメイン設定が使用されていない場合は、特にこれが該当します。

6.5.1. 信頼関係

信頼 とは、あるレルム内のユーザーが別のドメイン内のリソースにアクセスする際にこの別のレルムに所属しているかのように信頼されていることを指します。これは、両方のドメインで共通して保持される単一のプリンシパル用に共有鍵を作成することで実現します。

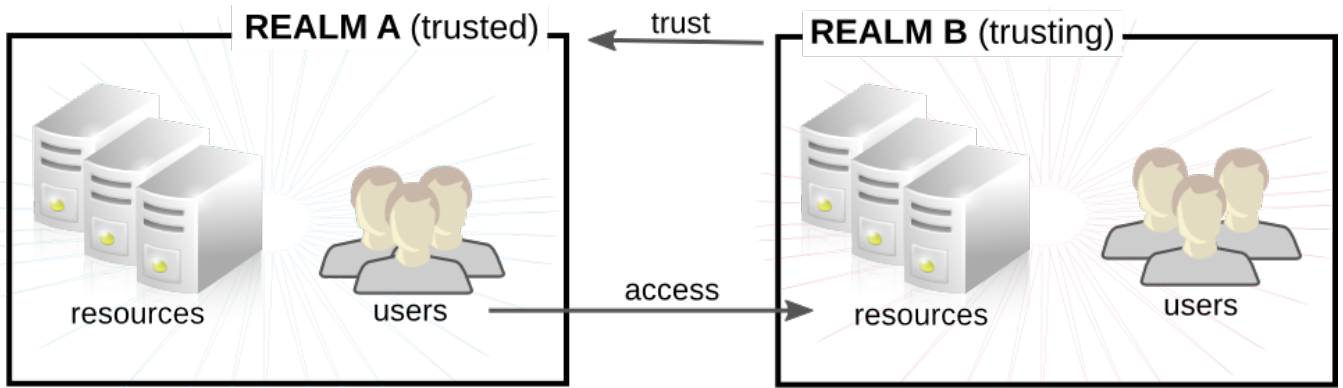


図6.2 基本的な信頼

図6.2「基本的な信頼」では、共有プリンシパルはドメイン B (`krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM`) に所属します。このプリンシパルがドメイン A にも追加されると、ドメイン A のクライアントはドメイン B 内のリソースにアクセスできるようになります。設定済みのプリンシパルは両方のレルムに存在することになります。この共有プリンシパルには以下の 3 つの特徴があります。

- ▶ 両方のレルムに存在します。
- ▶ 鍵が作成されると、両方のレルムで同じパスワードが使えます。
- ▶ 鍵には同一の鍵バージョン番号 (kvno) があります。

デフォルトでは、レルム間信頼は一方方向です。この信頼は自動で往復しないので、`B.EXAMPLE.COM` レルムは `A.EXAMPLE.COM` レルム内のサービスに対して認証するよう信頼されています。反対方向の信頼を確立するには、両方のレルムで `krbtgt/A.EXAMPLE.COM@B.EXAMPLE.COM` サービスの鍵を共有する必要があります。これは、上記の例とは反対のマッピングを持つエントリーになります。

レルムには複数の信頼関係を確立することが可能です。信頼関係は、レルムが他のレルムを信頼することと、他のレルムが当該レルムを信頼することの両方を定義できます。Kerberos の信頼を使うと、信頼は連鎖して流れることが可能になります。たとえば、レルム A がレルム B を信頼し、レルム B がレルム C を信頼していれば、暗示的にレルム A はレルム C を信頼します。信頼がレルムに沿って流れることになります。これが **推移的** 信頼です。

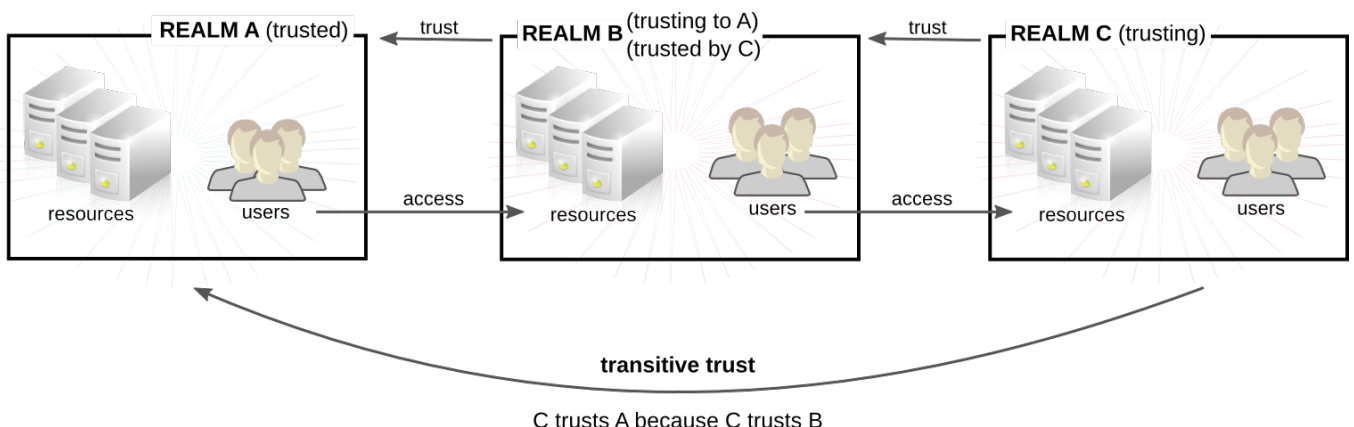


図6.3 推移的信頼

推移的信頼の方向は、**trust flow** (トラストフロー) と呼ばれます。トラストフローは、まずサービスが所属するレルムを認識し、次にそのサービスにアクセスするためにクライアントが連絡する必要のあるレルムを特定することで定義されます。

Kerberos プリンシパル名は、`service/hostname@REALM` という形式になります。`service` は通常、LDAP

や IMAP、HTTP、host といったプロトコルになります。**hostname** はホストシステムの完全修飾ドメイン名になります。**REALM** は所属する Kerberos レalm です。Kerberos クライアントは通常、ホスト名もしくは DNS ドメイン名を Kerberos レalm マッピングに使用します。このマッピングは明示的または暗示的とすることができます。明示的マッピングでは、`/etc/krb5.conf` ファイルの `domain_realm` セクションを使用します。暗示的マッピングでは、ドメイン名が大文字に変換されます。変換された名前が 検索する Kerberos レalm であるとみなされます。

信頼を移動させる際、Kerberos は各レalm が root ドメインとサブドメインからなる階層 DNS ドメインのように構成されていることを想定しています。つまり、信頼は共有 root まで移動することになります。ホップ と呼ばれる各ステップには共有鍵があります。[図6.4「同ドメイン内の信頼」](#)では、A が鍵を EXAMPLE.COM と共有しており、EXAMPLE.COM が鍵を B と共有しています。

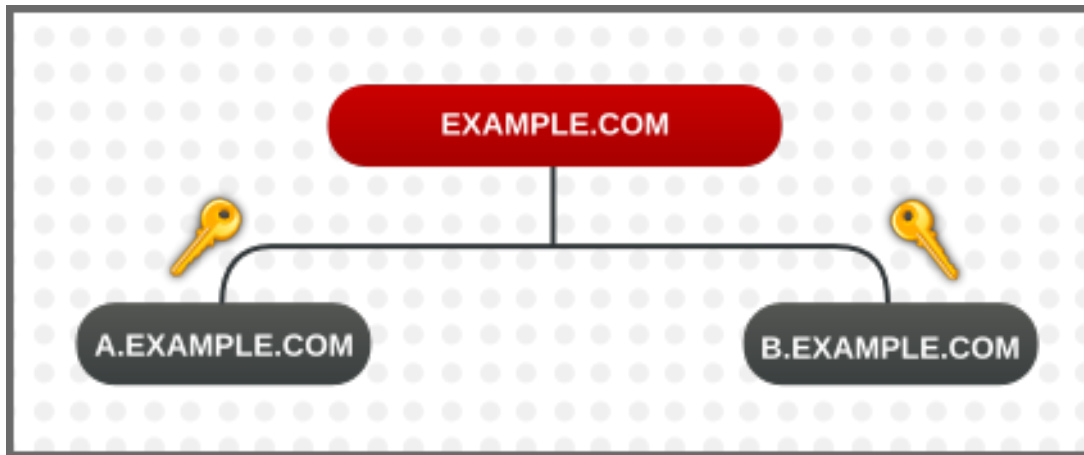


図6.4 同ドメイン内の信頼

クライアントはレalm 名を DNS 名として扱い、root 名に達するまで自身のレalm 名の要素を取り除くことで信頼パスを判断します。そして、サービスのレalm に到達するまで名前を先頭につけ始めます。

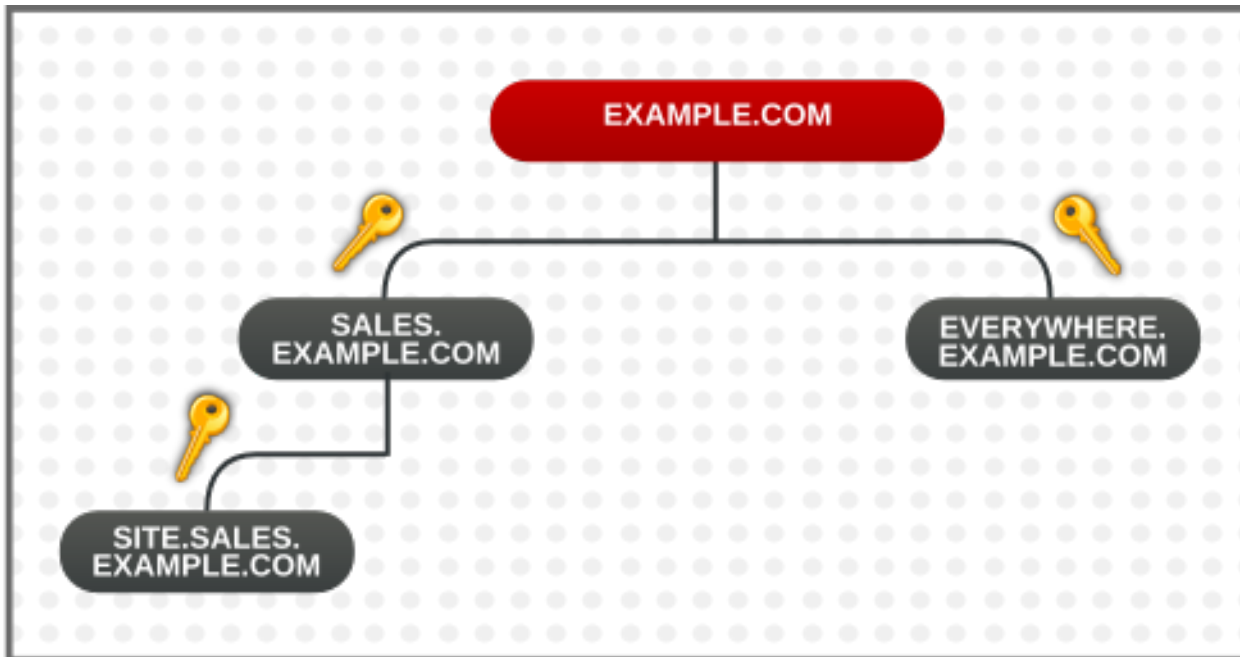


図6.5 同ドメイン内の子/親の信頼

信頼の推移的な性質は以下ようになります。SITE.SALES.EXAMPLE.COM には SALES.EXAMPLE.COM との共有鍵が 1 つだけあります。しかし、小さい信頼が連続することで、SITE.SALES.EXAMPLE.COM から EVERYWHERE.EXAMPLE.COM に信頼が移動するという大きなトラストフローが可能になります。

このトラストフローは、共有鍵をドメインレベルで作成することで、完全に異なるドメイン間での行き来が可能になります。この場合、サイト間で共有される共通の接尾辞はありません。

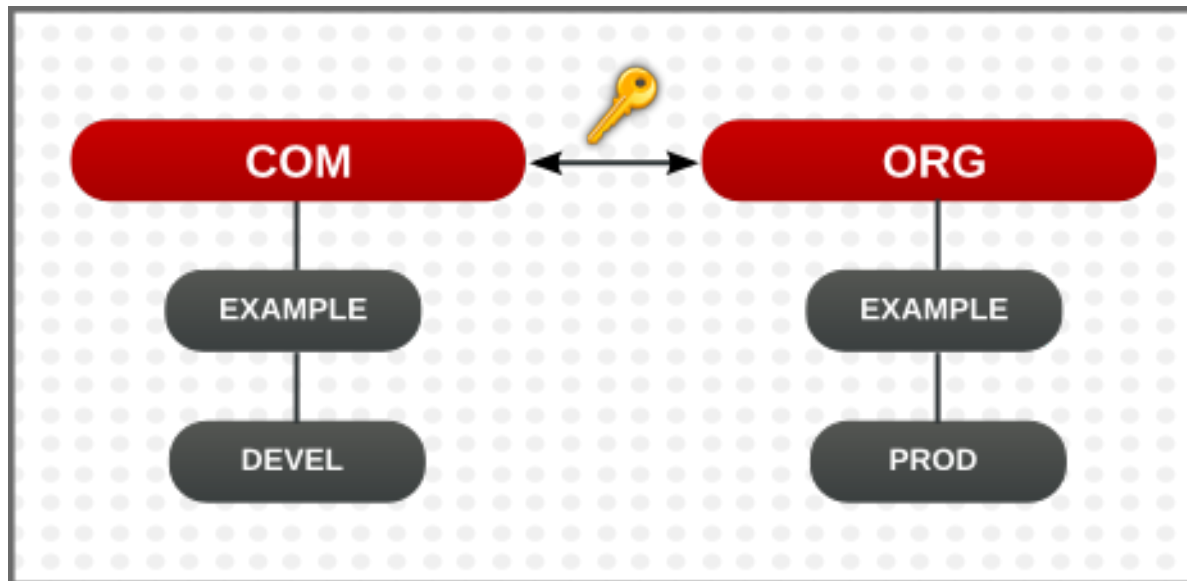


図6.6 異なるドメインでの信頼

また、フローを明示的に定義することで、ホップ数を減らして非常に複雑な信頼を提示することも可能です。/etc/krb5.conf ファイルの `capaths` セクションで異なるレルム間のトラストフローを定義します。

`capaths` セクションの書式は比較的簡単なものです。クライアントがプリンシパルを保持している各レルムのメインエントリーがあり、次に各レルムセクション内にクライアントの認証情報の取得先となる中間レルムの一覧があります。

以下の例では、**A.EXAMPLE.COM** のレルムがあり、A から D までのホップがあります。レルム A のクライアントはまず認証情報をレルム B から取得する必要があります (. は、中間ホップなしで認証情報を直接取得できることを示しています。これができないと、階層を通して認証情報の取得を試みることになります)。次に、B の認証情報を使って C から認証情報を取得、C の認証情報を使って D の認証情報をレルム D から取得する必要があります。

```
[capaths]
A.EXAMPLE.COM = {
B.EXAMPLE.COM = .
C.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = C.EXAMPLE.COM
}
```

6.5.2. レルム信頼の設定

以下の例では、Kerberos レルムは **A.EXAMPLE.COM** および **B.EXAMPLE.COM** とします。

`kadmind` を使って A レルム内に B レルム用に共通プリンシパルのエントリーを作成します。

```
[root@server ~]# kadmind -r A.EXAMPLE.COM
kadmind: add_principal krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM
Enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":
Re-enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":
```

```
Principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM" created.  
quit
```

上記では、A レルムが B プリンシパルを信頼していることを示しています。



重要

レルム間プリンシパルのパスワードは、非常に強固なものにすることが推奨されます。1日に何度も入力を求められる他の多くのパスワードとは異なり、システムはレルム間プリンシパルのパスワードを頻繁に要求しません。このため、このパスワードを簡単に記憶できるものにする必要はありません。

双方向の信頼を作成するには、反対方向に移動するプリンシパルを作成します。`kadmin` を使って B レルム内に A レルム用のプリンシパルを作成します。

```
[root@server ~]# kadmin -r B.EXAMPLE.COM  
kadmin: add_principal krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM  
Enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":  
Re-enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":  
Principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM" created.  
quit
```

`get_principal` コマンドを使用して、鍵バージョン番号 (`kvno` の値) と暗号化タイプが両方のエントリーで一致するか確認します。



重要

よくある間違った例では、管理者が `add_principal` コマンドを `-randkey` オプションと使用して、パスワードではなくランダムな鍵を割り当て、最初のレルムのデータベースから新規エントリーをダンプし、これを2番目にインポートしてしまいます。これは、レルムデータベースのマスター鍵が同じものでなければ、機能しません。データベースダンプに含まれている鍵自体が、マスター鍵を使って暗号化されているためです。

[4] クライアントとサーバーの両方が共通の鍵を共有し、これを用いてネットワーク通信を暗号化、暗号解除するシステム。

第7章 certmonger を使った作業

マシンの認証管理には、マシン証明書の管理を伴います。**certmonger** サービスはアプリケーションの証明書ライフサイクルを管理し、正常に設定されていれば、認証局 (CA) とともに証明書を更新、取り消すことができます。

certmonger デーモンとそのコマンドラインクライアントを使うと、公開鍵と秘密鍵のペア生成や証明書リクエストの作成、CA に対する署名のリクエスト提出といった処理を簡素化することができます。証明書管理の一部として、**certmonger** デーモンは証明書の有効期限を監視し、期限が切れそうになった証明書を更新することができます。**certmonger** が監視する証明書は、設定可能なディレクトリー内に保存されるファイルで追跡されます。デフォルトの場所は、`/var/lib/certmonger/requests` になります。

7.1. certmonger および認証局

certmonger はデフォルトで、以下の 3 種類の証明書を自動的に取得することができます。これらは、証明書が用いるソースによって異なります。

- ※ 自己署名証明書

自己署名証明書は証明書自体の鍵を使って署名されるので、この生成には CA は関わりません。自己署名証明書を確認するソフトウェアには、その確認で証明書を直接信頼するよう指示する必要があります。

自己署名証明書を取得するには、**selfsigned-getcert** コマンドを実行します。

- ※ Red Hat Enterprise Linux IdM の一部としての Dogtag Certificate System CA からの証明書

IdM サーバーを使用して証明書を取得するには、**ipa-getcert** コマンドを実行します。

- ※ システム上にあるローカルの CA が署名する証明書

ローカル署名の証明書を確認するソフトウェアには、その確認でこのローカル署名者からの証明書を信頼するよう指示する必要があります。

ローカル署名の証明書を取得するには、**local-getcert** コマンドを実行します。

他の CA も **certmonger** を使用して証明書を管理できますが、特別な **CA ヘルパー** を作成して、**certmonger** にサポートを追加する必要があります。CA ヘルパーの作成に関する詳細情報は、**certmonger** プロジェクトのドキュメンテーションを <https://git.fedorahosted.org/cgit/certmonger.git/tree/doc/submit.txt> で参照してください。

7.2. certmonger を使った証明書のリクエスト

certmonger で証明書をリクエストするには、**getcert request** ユーティリティを使用します。

証明書と鍵は **.pem** 拡張子を付けてプレーンテキストファイル内でローカルに保存されるか NSS データベースで保存され、証明書のニックネームで識別されます。証明書をリクエストする際には、リクエストで証明書の保存場所とニックネームを特定します。例を示します。

```
[root@server ~]# selfsign-getcert request -d /etc/pki/nssdb -n Server-Cert
```

`/etc/pki/nssdb` ファイルはグローバルの NSS データベースで、**Server-Cert** はこの証明書のニックネームになります。証明書のニックネームはこのデータベース内で一意のものである必要があります。

証明書を生成するこのコマンドで使用可能なオプションは、他の設定に加えて、リクエストしている証明書の種類や最終的な証明書で希望する設定によって異なります。

- ※ **-r** は、鍵のペアが既に存在している場合で、証明書の有効期限が切れそうになると、自動的に証明書を更新します。このオプションはデフォルトで使用されます。
- ※ **-f** は、証明書を特定ファイルに保存します。
- ※ **-k** は、鍵を特定ファイルに保存するか、鍵のファイルが存在する場合は、ファイル内の鍵を使用します。
- ※ **-K** は、証明書を使用するサービスの Kerberos プリンシパル名を提供します。IdM サーバーからの証明書をリクエストしている場合は **-K** は必須となり、自己署名またはローカル署名の証明書をリクエストしている場合は任意となります。
- ※ **-N** は、サブジェクト名を提供します。
- ※ **-D** は、`subjectAltName` 値として DNS ドメイン名が証明書に含まれるようリクエストします。
- ※ **-U** は、拡張鍵使用フラグを設定します。
- ※ **-A** は、`subjectAltName` 値として IP アドレスが証明書に含まれるようリクエストします。
- ※ **-I** は、リクエストの ID を設定します。`certmonger` はこのニックネームを使ってストレージの場所の組み合わせを参照し、オプションをリクエストします。ニックネームは、`getcrt list` コマンドの出力にも表示されます。このオプションを指定しないと、`certmonger` は自動生成のニックネームを割り当てます。

IdM にあるような実際の CA は、CA 自体のポリシーにしたがって、署名リクエストで指定した **-K**、**-N**、**-D**、**-U**、**-A** などのオプションを無視することができます。たとえば、IdM では **-K** と **-N** がローカルホスト名と一致している必要があります。一方で、`selfsign-getcert` および `local-getcert` コマンドを使用して生成した証明書は、これらのコマンドがポリシーを強制しないので、指定したオプションに一致したものになります。

例7.1 サービスにおける `certmonger` の使用

```
[root@server ~]# selfsign-getcert request -f /etc/httpd/conf/ssl.crt/server.crt -k
/etc/httpd/conf/ssl.key/server.key -N CN=`hostname --fqdn` -D `hostname` -U id-kp-
serverAuth
```

7.3. NSS データベースでの証明書の保存

デフォルトでは、`certmonger` は `.pem` ファイルを使用して鍵と証明書を保存します。NSS データベースに鍵と証明書を保存するには、証明書をリクエストするコマンドで **-d** と **-n** を指定します。

- ※ **-d** は、セキュリティデータベースの場所を設定します。
- ※ **-n** は、NSS データベースで使用される証明書のニックネームを指定します。



注記

-d および **-n** のオプションは、`.pem` ファイル に提供する **-f** および **-k** オプションの代わりに使用します。

例を示します。

```
[root@server ~]# selfsign-getcert request -d /export/alias -n ServerCert ...
```

`ipa-getcert` および `local-getcert` を使って証明書リクエストすると、さらに 2 つのオプションが指定できます。

- ※ **-F** では、CA の証明書が保存されているファイルを指定します。
- ※ **-a** では、CA の証明書が保存されている NSS データベースの場所を指定します。



注記

selfsign-getcert を使って証明書をリクエストする場合は、自己署名証明書の生成に CA が関与しないので、**-F** および **-a** オプションを指定する必要はありません。

local-getcert で **-F** または **-a** オプションのいずれか、もしくはこれら両方を指定すると、ローカル署名者が発行した証明書を確認するために必要な CA 証明書のコピーを取得することができます。例を示します。

```
[root@server ~]# local-getcert request -F /etc/httpd/conf/ssl.crt/ca.crt -n
ServerCert -f /etc/httpd/conf/ssl.crt/server.crt -k
/etc/httpd/conf/ssl.key/server.key
```

7.4. certmonger を使った証明書の追跡

certmonger は、証明書の有効期限を監視し、期限終了時に証明書を更新することができます。証明書をこの方法で追跡するには、**getcert start-tracking** コマンドを実行します。



注記

getcert request の実行後に **getcert start-tracking** を実行することは、必須ではありません。**getcert request** コマンドはデフォルトで、自動的にリクエストした証明書を追跡、更新します。**getcert start-tracking** コマンドは、別のプロセスで鍵と証明書を既に取得しており、このため手動で **certmonger** に追跡を開始するよう指示する必要がある場合に使用します。

getcert start-tracking コマンドは以下のオプションを取ります。

- ※ **-r** は、鍵のペアが既に存在している場合で、証明書の有効期限が切れそうになると、自動的に証明書を更新します。このオプションはデフォルトで使用されます。
- ※ **-I** は、追跡リクエストの ID を設定します。**certmonger** はこのニックネームを使ってストレージの場所とリクエストオプションの組み合わせを参照します。ニックネームは、**getcert list** コマンドの出力にも表示されます。このオプションを指定しないと、**certmonger** は自動生成のニックネームを割り当てます。

```
[root@server ~]# getcert start-tracking -I cert1-tracker -d /export/alias -n
ServerCert
```

証明書の追跡をキャンセルするには、**stop-tracking** コマンドを実行します。

第8章 アプリケーションのシングルサインオン用の設定

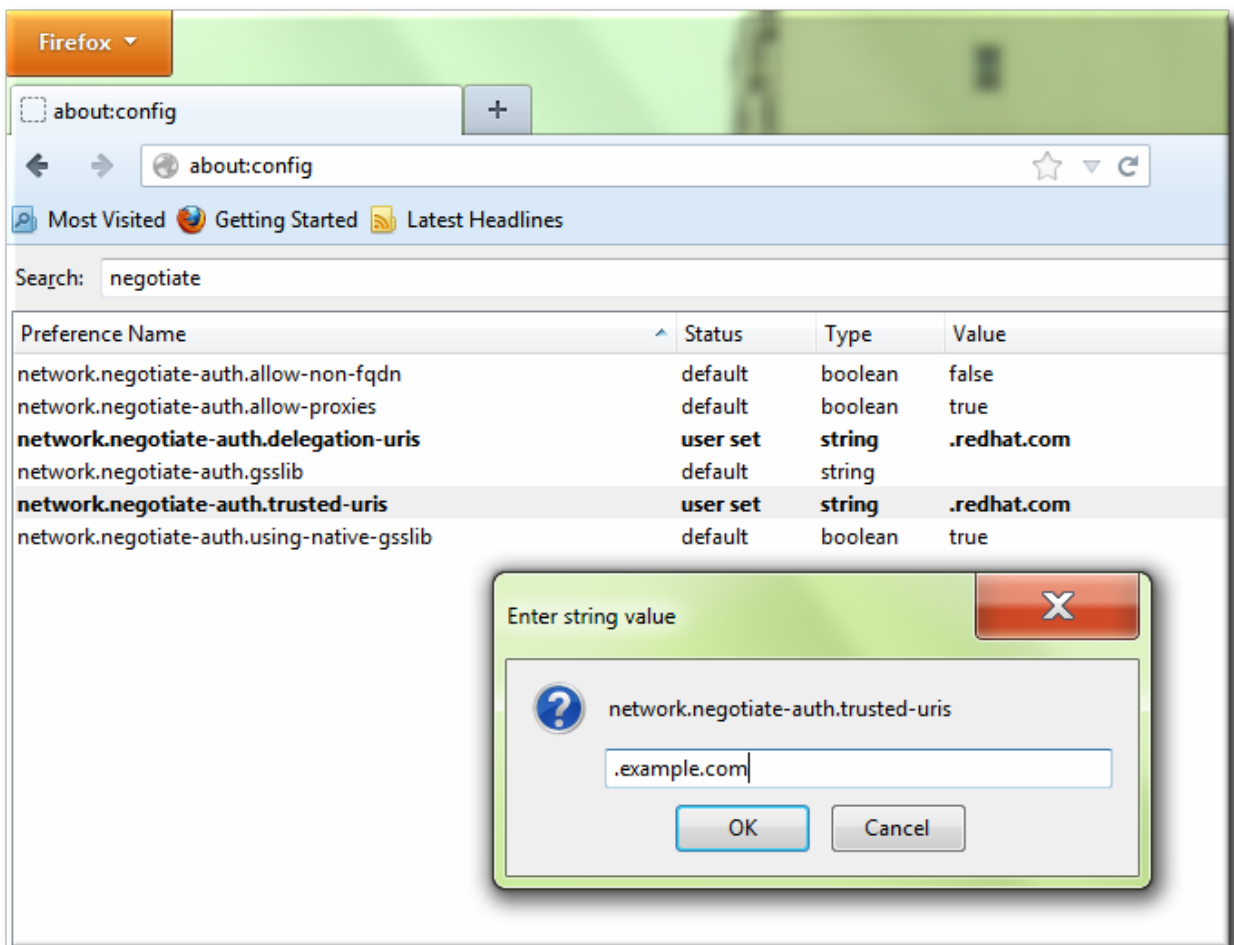
ブラウザやメールクライアントなどの一般的なアプリケーションのなかには、Kerberos チケットや SSL 証明書、トークンなどをユーザー認証の手段として使用するよう設定可能なものもあります。

これらアプリケーションの設定手順は、それぞれ異なります。本章での例 (Mozilla Thunderbird と Mozilla Firefox) では、ユーザーアプリケーションが Kerberos や他の認証情報を使用する設定方法についての考え方を提供します。

8.1. Firefox でシングルサインオンに Kerberos を使用するように設定する

Firefox は Kerberos を使って、イントラネットのサイトや他の保護されたウェブサイトシングルサインオンを使用することができます。Firefox が Kerberos を使用するには、まず Kerberos 認証情報を適切な KDC に送信するよう設定する必要があります。

1. Firefox のアドレスバーに `about:config` と入力して、現在の設定オプションを表示します。
2. **検索** フィールドに `negotiate` と入力して、オプション一覧を絞り込みます。
3. `network.negotiate-auth.trusted-uris` のエントリーをダブルクリックします。
4. 認証対象とするドメイン名を入力します。



5. 次に `network.negotiate-auth.delegation-uris` エントリーに `network.negotiate-auth.trusted-uris` と同じドメインを使用して設定します。

注記

Firefox は Kerberos 認証情報を渡すように設定した後でも、有効な Kerberos チケットを必要とします。Kerberos チケットの生成には `kinit` コマンドを使用し、KDC 上のユーザーにユーザーパスワードを提供します。

```
[jsmith@host ~] $ kinit
Password for jsmith@EXAMPLE.COM:
```

Kerberos 認証が機能しない場合は、認証プロセスにおける詳細ロギングをオンにします。

1. Firefox のすべてのインスタンスを閉じます。
2. コマンドプロンプトで、`NSPR_LOG_*` 変数の値をエクスポートします。

```
export NSPR_LOG_MODULES=negotiateauth:5
export NSPR_LOG_FILE=/tmp/moz.log
```

3. そのシェルから Firefox を再起動し、Kerberos 認証に失敗していたウェブサイトを開きます。
4. `/tmp/moz.log` ファイルにあるエラーメッセージで `nsNegotiateAuth` があるものを確認します。

Kerberos 認証では、以下のようなエラーが一般的です。

- ※ 1 つ目の例では、認証情報が見つからなかったと表示されています。

```
-1208550944[90039d0]: entering nsNegotiateAuth::GetNextToken()
-1208550944[90039d0]: gss_init_sec_context() failed: Miscellaneous failure
No credentials cache found
```

つまり、Kerberos チケットがなかったことを意味します (有効期限が切れたか、生成されていなかったため)。これを解決するには、`kinit` を実行して Kerberos チケットを生成し、ウェブサイトを再度開きます。

- ※ 2 つ目の可能性は、ブラウザが KDC に連絡できず、`Server not found in Kerberos database` のエラーメッセージが出る場合です。

```
-1208994096[8d683d8]: entering nsAuthGSSAPI::GetNextToken()
-1208994096[8d683d8]: gss_init_sec_context() failed: Miscellaneous failure
Server not found in Kerberos database
```

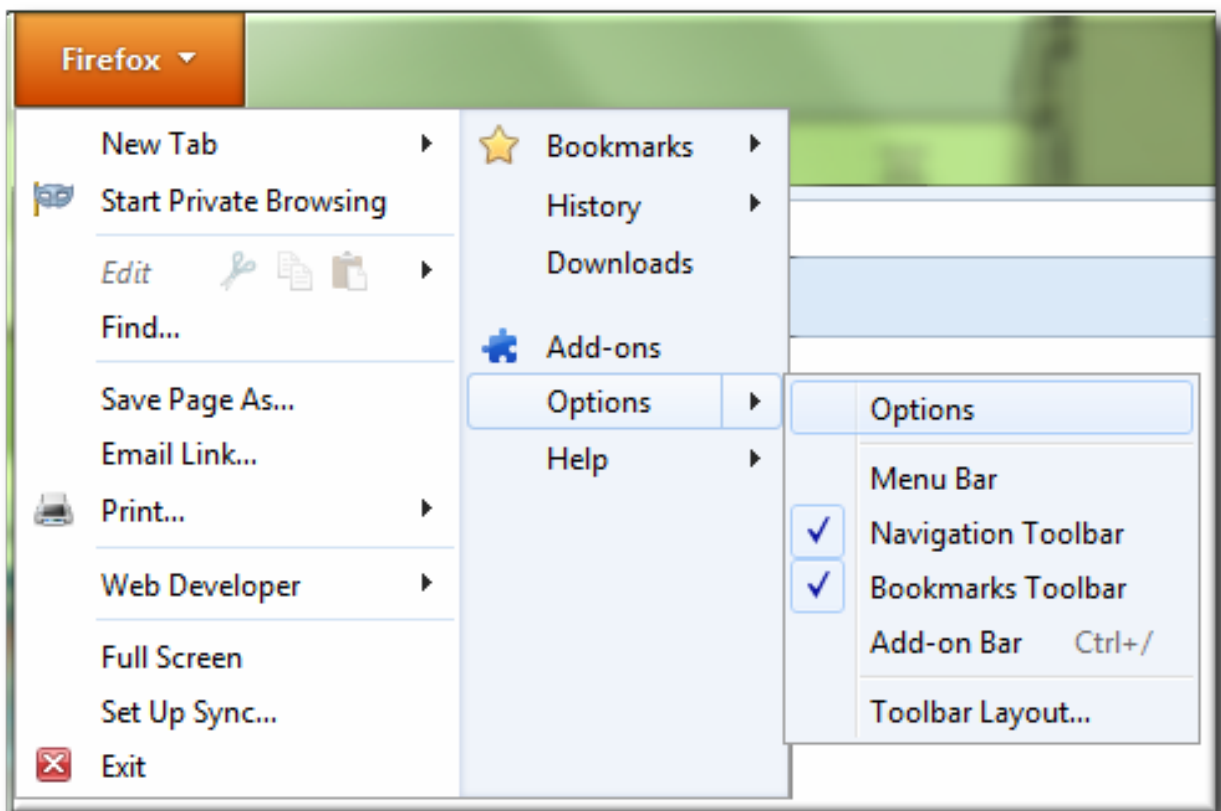
これは通常 Kerberos の設定問題です。`/etc/krb5.conf` ファイルの `[domain_realm]` セクションで、正しいエントリでドメインを指定する必要があります。例を示します。

```
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

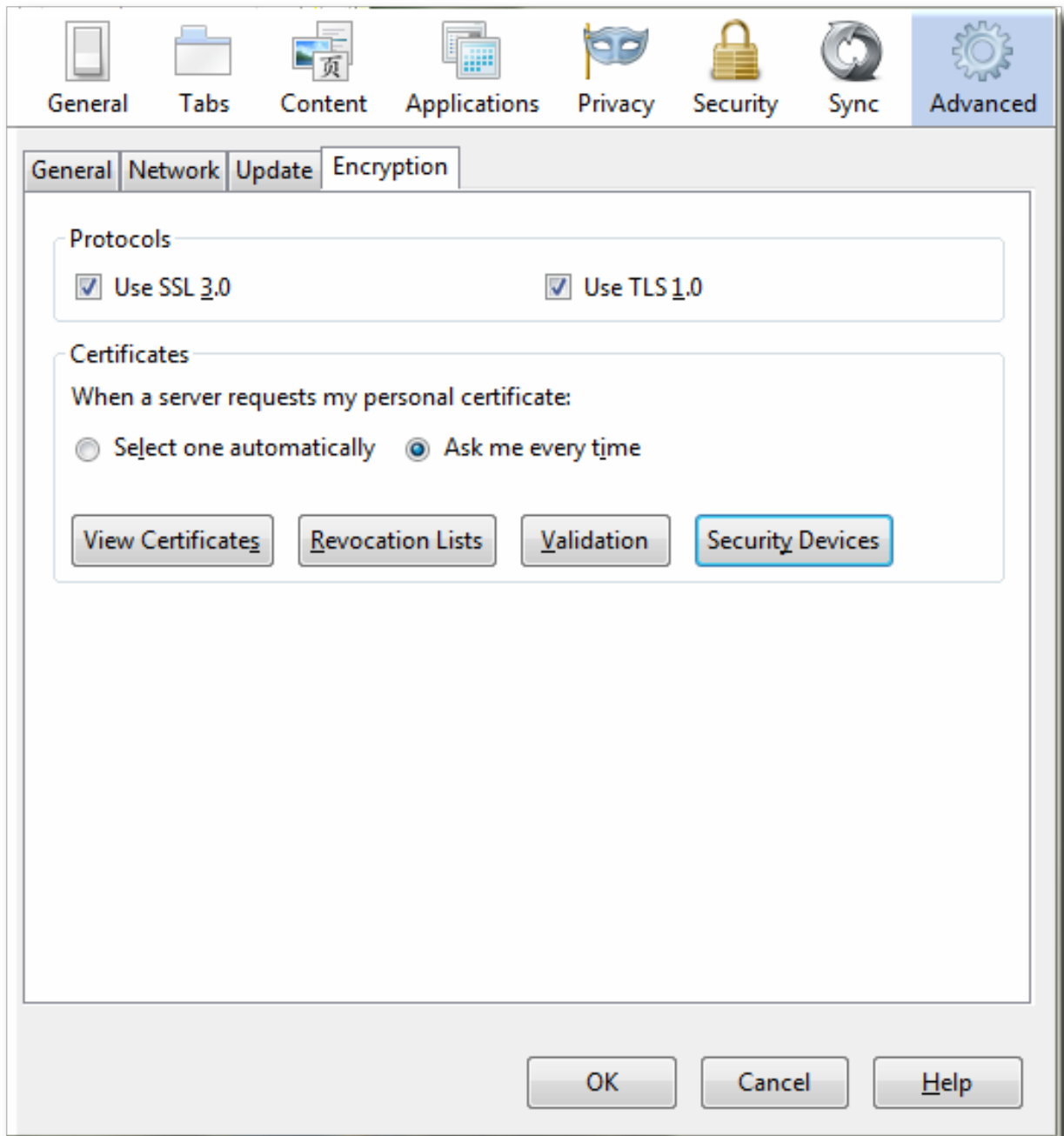
- ※ ログにエラーがない場合は、HTTP プロキシサーバーが Kerberos 認証に必要な HTTP ヘッダーを削除している可能性があります。HTTPS を使用してサイトに接続することで、リクエストを修正することなく渡すことができます。

8.2. トークン用の SSL をサポートするブラウザの設定

1. Mozilla Firefox で **Firefox** メニューを開きます。
2. **オプション** を選択し、さらに **オプション** をクリックします。

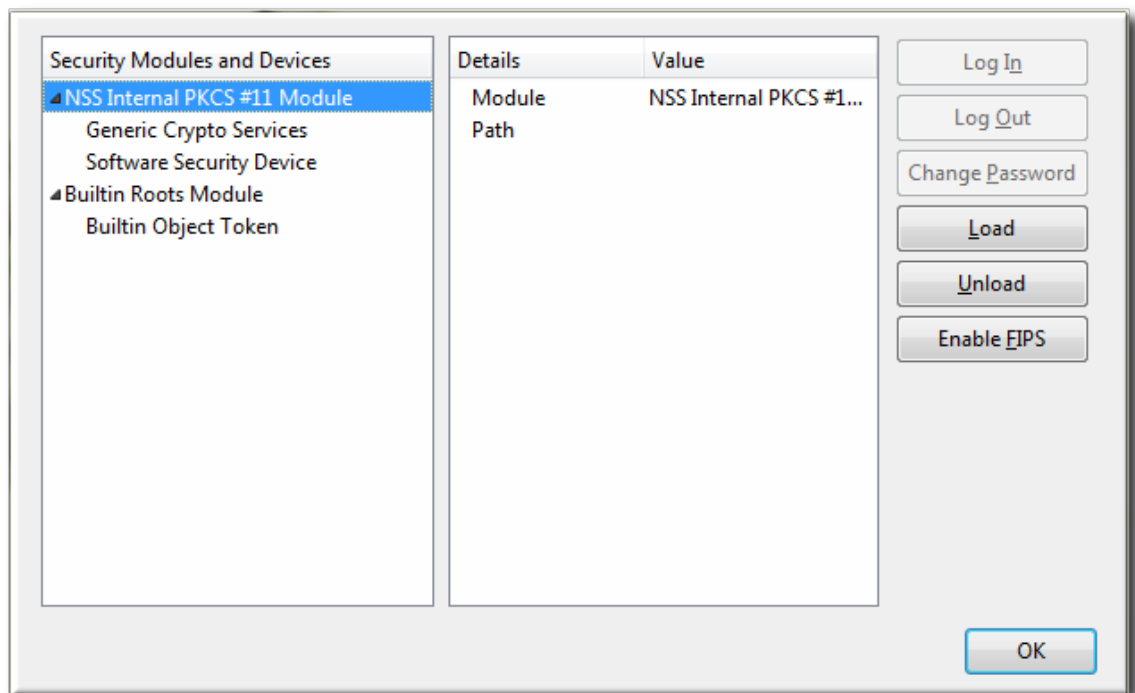


3. **詳細** セクションを開きます。
4. **証明書** タブを開きます。



5. PKCS #11 ドライバーを追加します。

- a. **セキュリティデバイス** をクリックして **デバイスマネージャ** ウィンドウを開き、**追加** ボタンをクリックします。
- b. **token key pk11 driver** といったモジュール名を入力します。



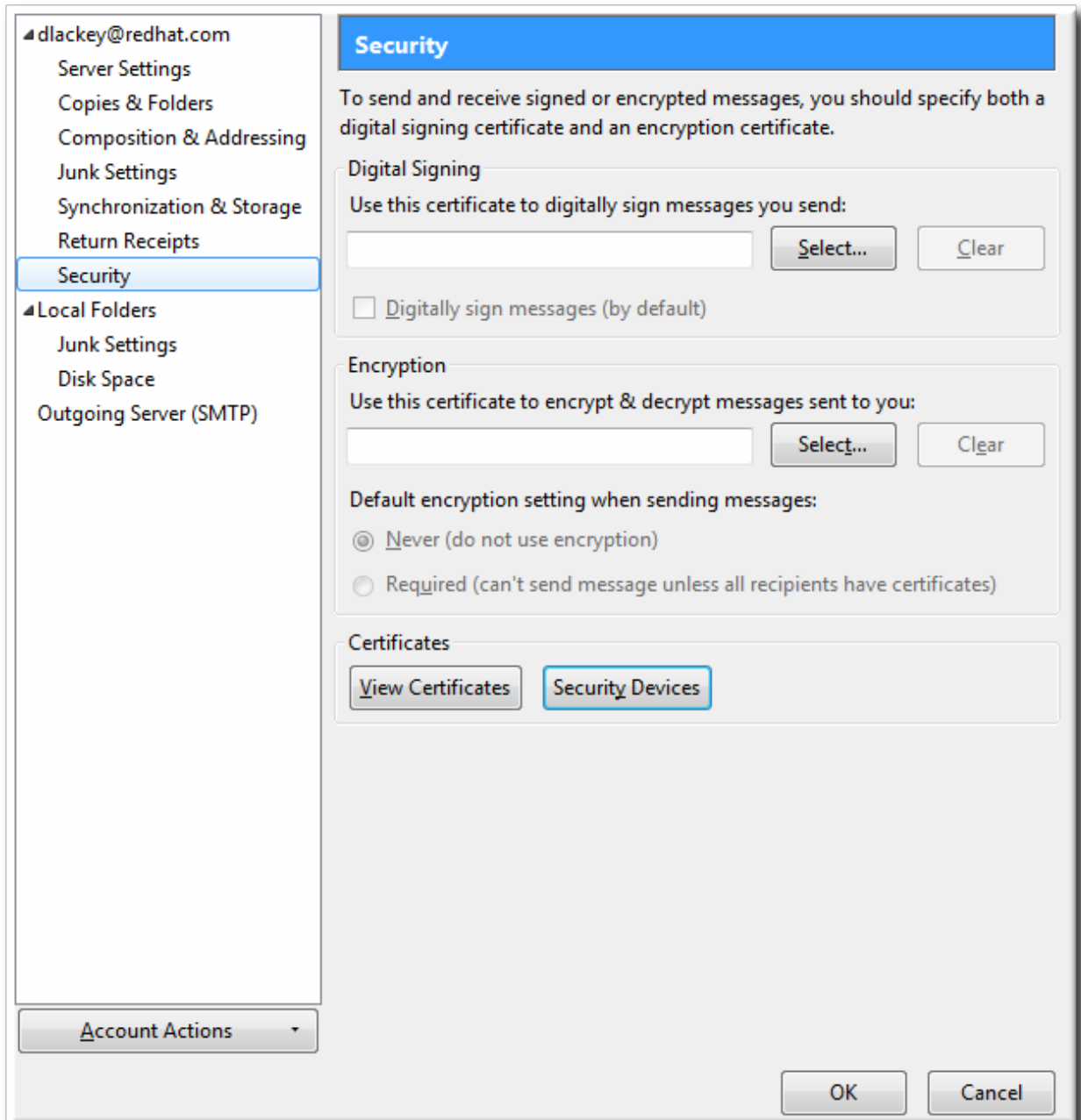
- c. **Browse** をクリックして PKCS #11 ドライバーを見つけて、**OK** をクリックします。これらのアプリケーションが使用する PKCS #11 モジュールは、デフォルトで `/usr/lib/libcoolkeypk11.so` に格納されます。
6. CA がまだ信頼されていない場合、CA 証明書をダウンロードしてインポートします。
- a. 証明書チェーンを保存する適切なディレクトリーを選択して、**OK** をクリックします。
 - b. **Firefox** → **Options** → **Options** をクリックして、**Advanced** 詳細タブを選択します。
 - c. **証明書を表示** ボタンをクリックします。
 - d. **認証局証明書** をクリックして、CA 証明書をインポートします。
7. 証明書の信頼関係を設定します。
- a. **Firefox** → **Options** → **Options** をクリックして、**Advanced** 詳細タブを選択します。
 - b. **証明書を表示** ボタンをクリックします。
 - c. **信頼性を設定** をクリックして、ウェブサイトの信頼を設定します。

証明書は SSL に使用することができます。

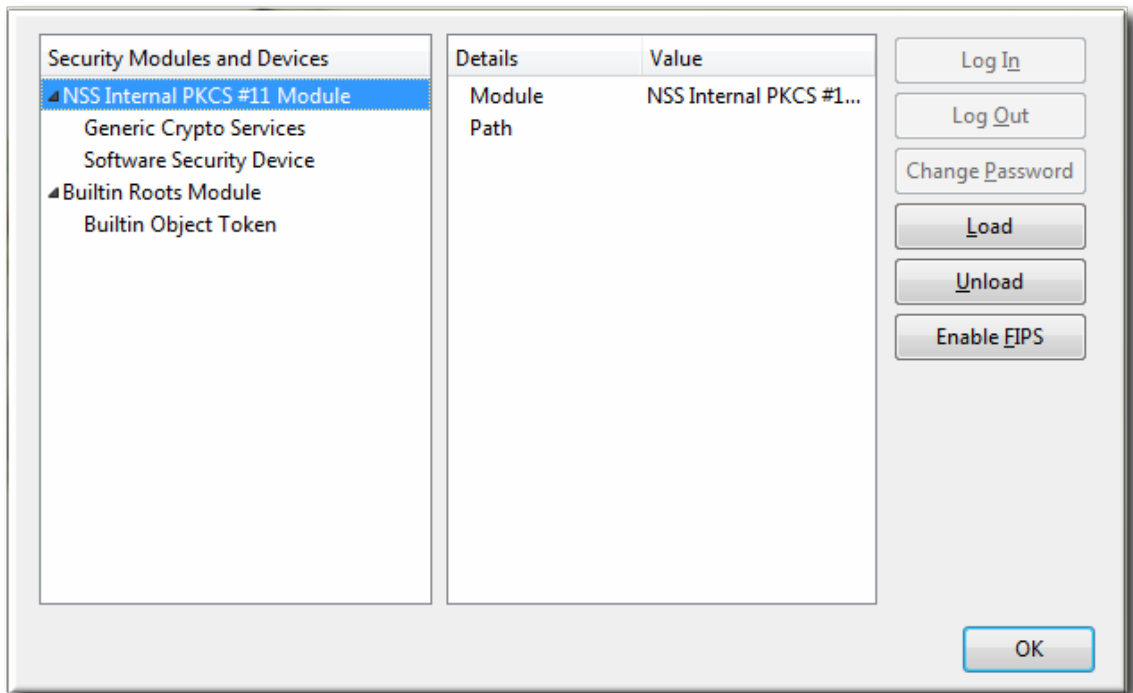
8.3. 証明書をメールクライアントのトークンに使用する

以下の例は、Mozilla Thunderbird を使用したものです。ここでは、メールクライアントが認証にトークンを使用する一般的な設定の手順を紹介しています。

1. Thunderbird を開きます。
2. (Inbox ではなく) トップでメインアカウントを選択します。
3. **編集** メニューを開き、**プロパティ** を選択します。
4. **アカウント設定** のエリアで **セキュリティ** を選択します。



5. PKCS #11 ドライバーを追加します。
 - a. セキュリティデバイス をクリックして、デバイスマネージャ ウィンドウを開きます。
 - b. 追加 ボタンをクリックします。



- c. **token key pk11 driver** といったモジュール名を入力します。
 - d. **Browse** をクリックして Enterprise Security Client PKCS #11 ドライバーを見つけて、**OK** をクリックします。これらのアプリケーションが使用する PKCS #11 モジュールは、デフォルトで `/usr/lib/libcoolkeypk11.so` に格納されます。
6. CA がまだ信頼されていない場合、CA 証明書をダウンロードしてインポートします。
 - a. 証明書チェーンを保存する適切なディレクトリーを選択して、**OK** をクリックします。
 - b. Mozilla Thunderbird の **編集** メニューを開き、**設定** を選択し、**詳細** をクリックします。
 - c. **証明書** タブを開き、**証明書を表示** ボタンをクリックします。
 - d. **認証局証明書** タブをクリックして、CA 証明書をインポートします。
 7. 証明書の信頼関係を設定します。
 - a. Mozilla Thunderbird の **編集** メニューを開き、**設定** を選択し、**詳細** をクリックします。
 - b. **証明書** タブを開き、**証明書を表示** ボタンをクリックします。
 - c. **認証局証明書** タブで CA を選択し、**信頼性を設定** ボタンをクリックします。
 - d. ウェブサイトおよびメニューユーザー識別における信頼性を設定します。
 - e. **セキュリティ** パネルの **デジタル署名** セクションで、**選択** ボタンをクリックしてメッセージの署名に使用する証明書を選択します。
 8. **セキュリティ** パネルの **暗号化** セクションで、**選択** ボタンをクリックしてメッセージの暗号化および暗号解除に使用する証明書を選択します。

付録A 改訂履歴

改訂 7.0-6.2 翻訳完成	Tue Dec 1 2015	Kenzo Moriguchi
改訂 7.0-6.1 翻訳ファイルをXMLソースバージョン7.0-6 と同期	Tue Dec 1 2015	Kenzo Moriguchi
改訂 7.0-6 7.1 GA リリース用バージョン。	Wed Feb 25 2015	Tomáš Čapek
改訂 7.0-4 スプラッシュページでの分類順序を更新して再構築。	Fri Dec 05 2014	Tomáš Čapek
改訂 7.0-1 RHEL 7.0 用初期ドラフト。	July 16, 2014	Ella Deon Ballard