Red Hat Enterprise Linux 9

Configuring authentication and authorization in RHEL

Using SSSD, authselect, and sssctl to configure authentication and authorization

Last Updated: 2024-10-03

Red Hat Enterprise Linux 9 Configuring authentication and authorization in RHEL

Using SSSD, authselect, and sssctl to configure authentication and authorization

Legal Notice

Abstract

You can configure Red Hat Enterprise Linux (RHEL) to authenticate and authorize users to services, such as Red Hat Identity Management (IdM), Active Directory (AD), and LDAP directories. For that, RHEL uses the System Security Services Daemon (SSSD) to communicate to these services. Utilities, such as authselect and sssctl support you in configuring SSSD, Pluggable Authentication Modules (PAM) and the Name Service Switch (NSS).

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. INTRODUCTION TO SYSTEM AUTHENTICATION

One of the cornerstones of establishing a secure network environment is ensuring that access is restricted to authorized users. When access is allowed, users can authenticate to the system, verifying their identities.

On any Red Hat Enterprise Linux system, various services are available to create and manage user identities. These can include local system files, services that connect to larger identity domains like Kerberos or Samba, or tools to create those domains.

1.1. CONFIRMING USER IDENTITIES

Authentication is the process of confirming an identity. For network interactions, authentication involves the identification of one party by another party. There are many ways to use authentication over networks, such as simple passwords, certificates, passwordless methods, one-time password (OTP) tokens, or biometric scans.

Authorization defines what the authenticated party is allowed to do or access.

Authentication requires that a user presents some kind of credential to verify his identity. The kind of credential that is required is defined by the authentication mechanism being used. There are several kinds of authentication for local users on a system:

Password-based authentication

Almost all software permits the user to authenticate by providing a recognized username and password. This is also called simple authentication.

Certificate-based authentication

Client authentication based on certificates is part of the Secure Sockets Layer (SSL) protocol. The client digitally signs a randomly generated piece of data and sends both the certificate and the signed data across the network. The server validates the signature and confirms the validity of the certificate.

Kerberos authentication

Kerberos establishes a system of short-lived credentials, called ticket-granting tickets (TGTs). The user presents credentials, that is, user name and password, that identify the user and indicate to the system that the user can be issued a ticket. TGT can then be repeatedly used to request access tickets to other services, like websites and email. Authentication using Kerberos allows the user to undergo only a single authentication process in this way.

Smart card-based authentication

This is a variant of certificate-based authentication. The smart card (or token) stores user certificates; when a user inserts the token into a system, the system reads the certificates and grant access. Single sign-on using smart cards goes through three steps:

1. A user inserts a smart card into the card reader. Pluggable authentication modules (PAMs) on Red Hat Enterprise Linux detect the inserted smart card.

2. The system maps the certificate to the user entry and then compares the presented certificates on the smart card, which are encrypted with a private key as explained under the certificate-based authentication, to the certificates stored in the user entry.

3. If the certificate is successfully validated against the key distribution center (KDC), then the user is allowed to log in.

Smart card-based authentication builds on the simple authentication layer established by Kerberos by adding certificates as additional identification mechanisms as well as by adding physical access requirements. For more information see Managing smart card authentication.

One-time password authentication

One-time passwords bring an additional step to your authentication security. The authentication uses your password in combination with an automatically generated one time password. For more information see One time password (OTP) authentication in Identity Management .

Passkey authentication

A passkey is a FIDO2 authentication device that is supported by the libfido2 library, such as Yubikey 5 and Nitrokey. It allows passwordless and multi-factor authentication. If your system is enrolled and connected to an IdM environment, this authentication method issues a Kerberos ticket automatically, which enables single sign-on (SSO) for an Identity Management (IdM) user. For more information see Enabling passkey authentication in IdM environment .

External identity providers

You can associate users with external identity providers (IdP) that support the OAuth 2 device authorization flow. When these users authenticate with the SSSD version available in RHEL 9.1 or later, they receive RHEL Identity Management (IdM) single sign-on capabilities with Kerberos tickets after performing authentication and authorization at the external IdP. For more information see Using external identity providers to authenticate to IdM .

1.2. PLANNING SINGLE SIGN-ON

Without a central identity store and every application maintaining its own set of users and credentials, a user has to enter a password for every single service or application they open.

By configuring single sign-on, administrators create a single password store so that users can log in once, by using a single password, and be authenticated to all network resources.

Red Hat Enterprise Linux supports single sign-on for several resources, including logging into workstations, unlocking screen savers, and accessing secured web pages using Mozilla Firefox. With other available system services such as Privileged Access Management (PAM), Name Service Switch (NSS), and Kerberos, other system applications can be configured to use those identity sources.

Single sign-on is both a convenience to users and another layer of security for the server and the network. Single sign-on hinges on secure and effective authentication. Red Hat Enterprise Linux provides two authentication mechanisms which can be used to enable single sign-on:

- Kerberos-based authentication, through both Kerberos realms and Active Directory domains
- Smart card-based authentication

Both of these methods create a centralized identity store (either through a Kerberos realm or a certificate authority in a public key infrastructure), and the local system services then use those identity domains rather than maintaining multiple local stores.

1.3. SERVICES AVAILABLE FOR LOCAL USER AUTHENTICATION

All Red Hat Enterprise Linux systems have some services already available to configure authentication for local users on local systems. These include:

Authentication setup

- The Authentication Configuration tool **authselect** sets up different identity back ends and means of authentication (such as passwords, fingerprints, or smart cards) for the system.

Identity back end setup

- The Security System Services Daemon (SSSD) sets up multiple identity providers (primarily LDAP-based directories such as Microsoft Active Directory or Red Hat Enterprise Linux IdM) which can then be used by both the local system and applications for users. Passwords and tickets are cached, allowing both offline authentication and single sign-on by reusing credentials.

- The **realmd** service is a command-line utility that allows you to configure an authentication back end, which is SSSD for IdM. The **realmd** service detects available IdM domains based on the DNS records, configures SSSD, and then joins the system as an account to a domain.

- Name Service Switch (NSS) is a mechanism for low-level system calls that return information about users, groups, or hosts. NSS determines what source, that is, which modules, should be used to obtain the required information. For example, user information can be located in traditioal UNIX files, such as the **/etc/passwd** file, or in LDAP-based directories, while host addresses can be read from files, such as the **/etc/hosts** file, or the DNS records; NSS locates where the information is stored.

Authentication Mechanisms

- Pluggable Authentication Modules (PAM) provide a system to set up authentication policies. An application using PAM for authentication loads different modules that control different aspects of authentication; which PAM module an application uses is based on how the application is configured. The available PAM modules include Kerberos, Winbind, SSSD, or local UNIX file-based authentication.

Other services and applications are also available, but these are common ones.

CHAPTER 2. CONFIGURING USER AUTHENTICATION USING AUTHSELECT

authselect is a utility that allows you to configure system identity and authentication sources by selecting a specific profile. Profile is a set of files that describes how the resulting Pluggable Authentication Modules (PAM) and Network Security Services (NSS) configuration will look like. You can choose the default profile set or create a custom profile.

2.1. WHAT IS AUTHSELECT USED FOR

You can use the **authselect** utility to configure user authentication on a Red Hat Enterprise Linux 9 host.

You can configure identity information and authentication sources and providers by selecting one of the ready-made profiles:

- The default **sssd** profile enables the System Security Services Daemon (SSSD) for systems that use LDAP authentication.
- The **winbind** profile enables the Winbind utility for systems directly integrated with Microsoft Active Directory.
- The **minimal** profile serves only local users and groups directly from system files, which allows administrators to remove network authentication services that are no longer needed.

After selecting an **authselect** profile for a given host, the profile is applied to every user logging into the host.

Red Hat recommends using **authselect** in semi-centralized identity management environments, for example if your organization utilizes LDAP or Winbind databases to authenticate users to use services in your domain.

WARNING

You do not need to use **authselect** if:

- Your host is part of Red Hat Enterprise Linux Identity Management (IdM). Joining your host to an IdM domain with the **ipa-client-install** command automatically configures SSSD authentication on your host.
- Your host is part of Active Directory via SSSD. Calling the **realm join** command to join your host to an Active Directory domain automatically configures SSSD authentication on your host.

Red Hat recommends against changing the **authselect** profiles configured by **ipa-client-install** or **realm join**. If you need to modify them, display the current settings before making any modifications, so you can revert back to them if necessary:

```
$ authselect current
Profile ID: sssd
Enabled features:
- with-sudo
- with-mkhomedir
- with-smartcard
```

2.1.1. Files and directories authselect modifies

The **authconfig** utility, used in previous Red Hat Enterprise Linux versions, created and modified many different configuration files, making troubleshooting more difficult. **Authselect** simplifies testing and troubleshooting because it only modifies the following files and directories:

/etc/nsswitch.conf	The GNU C Library and other applications use this Name Service Switch (NSS) configuration file to determine the sources from which to obtain name-service information in a range of categories, and in what order. Each category of information is identified by a database name.

/etc/pam.d/* files	Linux-PAM (Pluggable Authentication Modules) is a system of modules that handle the authentication tasks of applications (services) on the system. The nature of the authentication is dynamically configurable: the system administrator can choose how individual service-providing applications will authenticate users. The configuration files in the **/etc/pam.d/** directory list the PAMs that will perform authentication tasks required by a service, and the appropriate behavior of the PAM-API in the event that individual PAMs fail. Among other things, these files contain information about: • User password lockout conditions • The ability to authenticate with a smart card • The ability to authenticate with a fingerprint reader
/etc/dconf/db/distro.d/* files	This directory holds configuration profiles for the **dconf** utility, which you can use to manage settings for the GNOME Desktop Graphical User Interface (GUI).

2.1.2. Data providers in /etc/nsswitch.conf

The default **sssd** profile establishes SSSD as a source of information by creating **sss** entries in **/etc/nsswitch.conf**:

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

This means that the system first looks to SSSD if information concerning one of those items is requested:

- **passwd** for user information
- **group** for user group information
- **netgroup** for NIS **netgroup** information
- **automount** for NFS automount information
- **services** for information regarding services

Only if the requested information is not found in the **sssd** cache and on the server providing authentication, or if **sssd** is not running, the system looks at the local files, that is **/etc/***.

For example, if information is requested about a user ID, the user ID is first searched in the **sssd** cache. If it is not found there, the **/etc/passwd** file is consulted. Analogically, if a user's group affiliation is requested, it is first searched in the **sssd** cache and only if not found there, the **/etc/group** file is consulted.

In practice, the local **files** database is not normally consulted. The most important exception is the case of the **root** user, which is never handled by **sssd** but by **files**.

2.2. CHOOSING AN AUTHSELECT PROFILE

As a system administrator, you can select a profile for the **authselect** utility for a specific host. The profile will be applied to every user logging into the host.

Prerequisites

- You need **root** credentials to run **authselect** commands

Procedure

- Select the **authselect** profile that is appropriate for your authentication provider. For example, for logging into the network of a company that uses LDAP, choose **sssd**.

```
# authselect select sssd
```

  - Optional: You can modify the default profile settings by adding the following options to the **authselect select sssd** or **authselect select winbind** command, for example:
    - **with-faillock**
    - **with-smartcard**
    - **with-fingerprint**

  To see the full list of available options, see Converting your scripts from authconfig to authselect or the **authselect-migration(7)** man page on your system.

NOTE

Make sure that the configuration files that are relevant for your profile are configured properly before finishing the **authselect select** procedure. For example, if the **sssd** daemon is not configured correctly and active, running **authselect select** results in only local users being able to authenticate, using **pam_unix**.

Verification

1. Verify **sss** entries for SSSD are present in **/etc/nsswitch.conf**:

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

2. Review the contents of the **/etc/pam.d/system-auth** file for **pam_sss.so** entries:

```
# Generated by authselect on Tue Sep 11 22:59:06 2018
# Do not modify this file manually.
```

```
auth        required      pam_env.so
auth        required      pam_faildelay.so delay=2000000
auth        [default=1 ignore=ignore success=ok]    pam_succeed_if.so uid >= 1000 quiet
auth        [default=1 ignore=ignore success=ok]    pam_localuser.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        sufficient    pam_sss.so forward_pass
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
...
```

Additional Resources

- What is authselect used for
- Modifying a ready-made authselect profile
- Creating and deploying your own authselect profile

2.3. MODIFYING A READY-MADE AUTHSELECT PROFILE

As a system administrator, you can modify one of the default profiles to suit your needs.

You can modify any of the items in the **/etc/authselect/user-nsswitch.conf** file with the exception of:

- **passwd**
- **group**
- **netgroup**
- **automount**
- **services**

Running **authselect select profile_name** afterwards will result in transferring permissible changes from **/etc/authselect/user-nsswitch.conf** to the **/etc/nsswitch.conf** file. Unacceptable changes are overwritten by the default profile configuration.

IMPORTANT

Do not modify the **/etc/nsswitch.conf** file directly.

Procedure

1. Select an **authselect** profile, for example:

```
# authselect select sssd
```

2. Edit the **/etc/authselect/user-nsswitch.conf** file with your desired changes.
3. Apply the changes from the **/etc/authselect/user-nsswitch.conf** file:

```
# authselect apply-changes
```

Verification

- Review the **/etc/nsswitch.conf** file to verify that the changes from **/etc/authselect/user-nsswitch.conf** have been propagated there.

Additional Resources

- What is authselect used for

2.4. CREATING AND DEPLOYING YOUR OWN AUTHSELECT PROFILE

As a system administrator, you can create and deploy a custom profile by making a customized copy of one of the default profiles.

This is particularly useful if Modifying a ready-made authselect profile is not enough for your needs. When you deploy a custom profile, the profile is applied to every user logging into the given host.

Procedure

1. Create your custom profile by using the **authselect create-profile** command. For example, to create a custom profile called **user-profile** based on the ready-made **sssd** profile but one in which you can configure the items in the **/etc/nsswitch.conf** file yourself:

```
# authselect create-profile user-profile -b sssd --symlink-meta --symlink-pam
New profile was created at /etc/authselect/custom/user-profile
```

WARNING

If you are planning to modify **/etc/authselect/custom/user-profile/{password-auth,system-auth,fingerprint-auth,smartcard-auth,postlogin}**, then enter the command above without the **--symlink-pam** option. This is to ensure that the modification persists during the upgrade of **authselect-libs**.

Including the **--symlink-pam** option in the command means that PAM templates will be symbolic links to the origin profile files instead of their copy; including the **--symlink-meta** option means that meta files, such as README and REQUIREMENTS will be symbolic links to the origin profile files instead of their copy. This ensures that all future updates to the PAM templates and meta files in the original profile will be reflected in your custom profile, too.

The command creates a copy of the **/etc/nsswitch.conf** file in the **/etc/authselect/custom/user-profile/** directory.

2. Configure the **/etc/authselect/custom/user-profile/nsswitch.conf** file.

3. Select the custom profile by running the **authselect select** command, and adding **custom/*name_of_the_profile*** as a parameter. For example, to select the **user-profile** profile:

```
# authselect select custom/user-profile
```

Selecting the **user-profile** profile for your machine means that if the **sssd** profile is subsequently updated by Red Hat, you will benefit from all the updates with the exception of updates made to the **/etc/nsswitch.conf** file.

Example 2.1. Creating a profile

The following procedure shows how to create a profile based on the **sssd** profile which only consults the local static table lookup for hostnames in the **/etc/hosts** file, not in the **dns** or **myhostname** databases.

1. Edit the **/etc/nsswitch.conf** file by editing the following line:

```
hosts:      files
```

2. Create a custom profile based on **sssd** that excludes changes to **/etc/nsswitch.conf**:

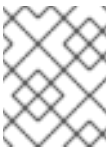
```
# authselect create-profile user-profile -b sssd --symlink-meta --symlink-pam
```

3. Select the profile:

```
# authselect select custom/user-profile
```

4. Optional: Check that selecting the custom profile has
   - created the **/etc/pam.d/system-auth** file according to the chosen **sssd** profile
   - left the configuration in the **/etc/nsswitch.conf** unchanged:

```
hosts:      files
```



NOTE

Running **authselect select sssd** would, in contrast, result in **hosts: files dns myhostname**

Additional Resources

- What is authselect used for

2.5. CONVERTING YOUR SCRIPTS FROM AUTHCONFIG TO AUTHSELECT

If you use **ipa-client-install** or **realm join** to join a domain, you can safely remove any **authconfig** call in your scripts. If this is not possible, replace each **authconfig** call with its equivalent **authselect** call. In doing that, select the correct profile and the appropriate options. In addition, edit the necessary configuration files:

- **/etc/krb5.conf**
- **/etc/sssd/sssd.conf** (for the **sssd** profile) or **/etc/samba/smb.conf** (for the **winbind** profile)

Relation of authconfig options to authselect profiles and Authselect profile option equivalents of authconfig options show the **authselect** equivalents of **authconfig** options.

Table 2.1. Relation of authconfig options to authselect profiles

Authconfig options	Authselect profile
--enableldap --enableldapauth	**sssd**
--enablesssd --enablesssdauth	**sssd**
--enablekrb5	**sssd**
--enablewinbind --enablewinbindauth	**winbind**

Table 2.2. Authselect profile option equivalents of authconfig options

Authconfig option	Authselect profile feature
--enablesmartcard	**with-smartcard**
--enablefingerprint	**with-fingerprint**
--enableecryptfs	**with-ecryptfs**
--enablemkhomedir	**with-mkhomedir**
--enablefaillock	**with-faillock**
--enablepamaccess	**with-pamaccess**
--enablewinbindkrb5	**with-krb5**

Examples of authselect command equivalents to authconfig commands shows example transformations of Kickstart calls to **authconfig** into Kickstart calls to **authselect**.

Table 2.3. Examples of authselect command equivalents to authconfig commands

authconfig command	authselect equivalent
authconfig --enableldap --enableldapauth --enablefaillock --updateall	**authselect select sssd with-faillock**
authconfig --enablesssd --enablesssdauth --enablesmartcard --smartcardmodule=sssd --updateall	**authselect select sssd with-smartcard**

authconfig command	authselect equivalent
authconfig --enableecryptfs --enablepamaccess --updateall	**authselect select sssd with-ecryptfs with-pamaccess**
authconfig --enablewinbind --enablewinbindauth --winbindjoin=Administrator --updateall	**realm join -U Administrator --client-software=winbind WINBINDDOMAIN**

2.6. ADDITIONAL RESOURCES

- What is pam_faillock and how to use it in Red Hat Enterprise Linux 8 & 9?
- Set Password Policy/Complexity in Red Hat Enterprise Linux 8

CHAPTER 3. UNDERSTANDING SSSD AND ITS BENEFITS

The System Security Services Daemon (SSSD) is a system service to access remote directories and authentication mechanisms. The following chapters outline how SSSD works, what are the benefits of using it, how the configuration files are processed, as well as what identity and authentication providers you can configure.

3.1. HOW SSSD WORKS

The System Security Services Daemon (SSSD) is a system service that allows you to access remote directories and authentication mechanisms. You can connect a local system, an SSSD *client*, to an external back-end system, a *provider*.

For example:

- An LDAP directory
- An Identity Management (IdM) domain
- An Active Directory (AD) domain
- A Kerberos realm

SSSD works in two stages:

1. It connects the client to a remote provider to retrieve identity and authentication information.
2. It uses the obtained authentication information to create a local cache of users and credentials on the client.

Users on the local system are then able to authenticate using the user accounts stored in the remote provider.

SSSD does not create user accounts on the local system. However, SSSD can be configured to create home directories for IdM users. Once created, an IdM user home directory and its contents on the client are not deleted when the user logs out.

Figure 3.1. How SSSD works

SSSD can also provide caches for several system services, such as Name Service Switch (NSS) or Pluggable Authentication Modules (PAM).

NOTE

Only use the SSSD service for caching user information. Running both Name Service Caching Daemon (NSCD) and SSSD for caching on the same system might lead to performance issues and conflicts.

3.2. BENEFITS OF USING SSSD

Using the System Security Services Daemon (SSSD) provides multiple benefits regarding user identity retrieval and user authentication.

Offline authentication

SSSD optionally keeps a cache of user identities and credentials retrieved from remote providers. In this setup, a user - provided they have already authenticated once against the remote provider at the start of the session - can successfully authenticate to resources even if the remote provider or the client are offline.

A single user account: improved consistency of the authentication process

With SSSD, it is not necessary to maintain both a central account and a local user account for offline authentication. The conditions are:

- In a particular session, the user must have logged in at least once: the client must be connected to the remote provider when the user logs in for the first time.
- Caching must be enabled in SSSD.
  Without SSSD, remote users often have multiple user accounts. For example, to connect to a virtual private network (VPN), remote users have one account for the local system and another account for the VPN system. In this scenario, you must first authenticate on the private network to fetch the user from the remote server and cache the user credentials locally.

  With SSSD, thanks to caching and offline authentication, remote users can connect to network resources simply by authenticating to their local machine. SSSD then maintains their network credentials.

Reduced load on identity and authentication providers

When requesting information, the clients first check the local SSSD cache. SSSD contacts the remote providers only if the information is not available in the cache.

3.3. MULTIPLE SSSD CONFIGURATION FILES ON A PER-CLIENT BASIS

The default configuration file for SSSD is **/etc/sssd/sssd.conf**. Apart from this file, SSSD can read its configuration from all ***.conf** files in the **/etc/sssd/conf.d/** directory.

This combination allows you to use the default **/etc/sssd/sssd.conf** file on all clients and add additional settings in further configuration files to extend the functionality individually on a per-client basis.

How SSSD processes the configuration files

SSSD reads the configuration files in this order:

1. The primary **/etc/sssd/sssd.conf** file
2. Other ***.conf** files in **/etc/sssd/conf.d/**, in alphabetical order

If the same parameter appears in multiple configuration files, SSSD uses the last read parameter.

NOTE

SSSD does not read hidden files (files starting with **.**) in the **conf.d** directory.

3.4. IDENTITY AND AUTHENTICATION PROVIDERS FOR SSSD

You can connect an SSSD client to the external identity and authentication providers, for example an LDAP directory, an Identity Management (IdM), Active Directory (AD) domain, or a Kerberos realm. The SSSD client then get access to identity and authentication remote services using the SSSD provider. You can configure SSSD to use different identity and authentication providers or a combination of them.

Identity and Authentication Providers as SSSD domains

Identity and authentication providers are configured as *domains* in the SSSD configuration file, **/etc/sssd/sssd.conf**. The providers are listed in the **[domain/*name of the domain*]** or **[domain/default]** section of the file.

A single domain can be configured as one of the following providers:

- An *identity provider*, which supplies user information such as UID and GID.
  - Specify a domain as the *identity provider* by using the **id_provider** option in the **[domain/*name of the domain*]** section of the **/etc/sssd/sssd.conf** file.
- An *authentication provider*, which handles authentication requests.
  - Specify a domain as the *authentication provider* by using the **auth_provider** option in the **[domain/*name of the domain*]** section of **/etc/sssd/sssd.conf**.
- An *access control provider*, which handles authorization requests.
  - Specify a domain as the *access control provider* using the **access_provider** option in the **[domain/*name of the domain*]** section of **/etc/sssd/sssd.conf**. By default, the option is set to **permit**, which always allows all access. See the **sssd.conf**(5) man page for details.
- A combination of these providers, for example if all the corresponding operations are performed within a single server.
  - In this case, the **id_provider**, **auth_provider**, and **access_provider** options are all listed in the same **[domain/*name of the domain*]** or **[domain/default]** section of **/etc/sssd/sssd.conf**.

NOTE

You can configure multiple domains for SSSD. You must configure at least one domain, otherwise SSSD will not start.

Proxy Providers

A proxy provider works as an intermediary relay between SSSD and resources that SSSD would otherwise not be able to use. When using a proxy provider, SSSD connects to the proxy service, and the proxy loads the specified libraries.

You can configure SSSD to use a proxy provider to enable:

- Alternative authentication methods, such as a fingerprint scanner
- Legacy systems, such as NIS
- A local system account defined in the **/etc/passwd** file as an identity provider and a remote authentication provider, for example Kerberos
- Authentication of local users using smart cards

Available Combinations of Identity and Authentication Providers

You can configure SSSD to use the following combinations of identity and authentication providers.

Table 3.1. Available Combinations of Identity and Authentication Providers

Identity Provider	Authentication Provider
Identity Management [a]	Identity Management
Active Directory	Active Directory
LDAP	LDAP
LDAP	Kerberos
Proxy	Proxy
Proxy	LDAP
Proxy	Kerberos

[a] An extension of the LDAP provider type.

Additional resources

- Configuring user authentication using authselect
- Querying domain information using SSSD [1]
- Reporting on user access on hosts using SSSD

[1] To list and verify the status of the domains using the **sssctl** utility, your host should be enrolled in Identity Management (IdM) that is in a trust agreement with an Active Directory (AD) forest.

CHAPTER 4. CONFIGURING SSSD TO USE LDAP AND REQUIRE TLS AUTHENTICATION

The System Security Services Daemon (SSSD) is a daemon that manages identity data retrieval and authentication on a Red Hat Enterprise Linux host. A system administrator can configure the host to use a standalone LDAP server as the user account database. The administrator can also specify the requirement that the connection with the LDAP server must be encrypted with a TLS certificate.

NOTE

The SSSD configuration option to enforce TLS, **ldap_id_use_start_tls**, defaults to **false**. When using **ldap://** without TLS for identity lookups, it can pose a risk for an attack vector, namely a man-in-the-middle (MITM) attack which could allow you to impersonate a user by altering, for example, the UID or GID of an object returned in an LDAP search.

Ensure that your setup operates in a trusted environment and decide if it is safe to use unencrypted communication for **id_provider = ldap**. Note **id_provider = ad** and **id_provider = ipa** are not affected as they use encrypted connections protected by SASL and GSSAPI.

If it is not safe to use unencrypted communication, you should enforce TLS by setting the **ldap_id_use_start_tls** option to **true** in the **/etc/sssd/sssd.conf** file.

4.1. AN OPENLDAP CLIENT USING SSSD TO RETRIEVE DATA FROM LDAP IN AN ENCRYPTED WAY

The authentication method of the LDAP objects can be either a Kerberos password or an LDAP password. Note that the questions of authentication and authorization of the LDAP objects are not addressed here.

IMPORTANT

Configuring SSSD with LDAP is a complex procedure requiring a high level of expertise in SSSD and LDAP. Consider using an integrated and automated solution such as Active Directory or Red Hat Identity Management (IdM) instead. For details about IdM, see Planning Identity Management.

Identity :leveloffset: +1

CHAPTER 5. CONFIGURING SSSD TO USE LDAP AND REQUIRE TLS AUTHENTICATION

Complete this procedure to configure your Red Hat Enterprise Linux (RHEL) system as an OpenLDAP client.

Use the following client configuration:

- The RHEL system authenticates users stored in an OpenLDAP user account database.
- The RHEL system uses the System Security Services Daemon (SSSD) service to retrieve user data.
- The RHEL system communicates with the OpenLDAP server over a TLS-encrypted connection.

NOTE

You can alternatively use this procedure to configure your RHEL system as a client of a Red Hat Directory Server.

Prerequisites

- The OpenLDAP server is installed and configured with user information.
- You have root permissions on the host you are configuring as the LDAP client.
- On the host you are configuring as the LDAP client, the **/etc/sssd/sssd.conf** file has been created and configured to specify **ldap** as the **autofs_provider** and the **id_provider**.
- You have a PEM-formatted copy of the root CA signing certificate chain from the Certificate Authority that issued the OpenLDAP server certificate, stored in a local file named **core-dirsrv.ca.pem**.

Procedure

1. Install the requisite packages:

```
# dnf -y install openldap-clients sssd sssd-ldap oddjob-mkhomedir
```

2. Switch the authentication provider to **sssd**:

```
# authselect select sssd with-mkhomedir
```

3. Copy the **core-dirsrv.ca.pem** file containing the root CA signing certificate chain from the Certificate Authority that issued the OpenLDAP server's SSL/TLS certificate into the **/etc/openldap/certs** folder.

```
# cp core-dirsrv.ca.pem /etc/openldap/certs
```

4. Add the URL and suffix of your LDAP server to the **/etc/openldap/ldap.conf** file:

```
URI ldap://ldap-server.example.com/
BASE dc=example,dc=com
```

5. In the **/etc/openldap/ldap.conf** file, add a line pointing the **TLS_CACERT** parameter to **/etc/openldap/certs/core-dirsrv.ca.pem**:

```
# When no CA certificates are specified the Shared System Certificates
# are in use. In order to have these available along with the ones specified
# by TLS_CACERTDIR one has to include them explicitly:
TLS_CACERT /etc/openldap/certs/core-dirsrv.ca.pem
```

6. In the **/etc/sssd/sssd.conf** file, add your environment values to the **ldap_uri** and **ldap_search_base** parameters and set the **ldap_id_use_start_tls** to **True**:

```
[domain/default]
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://ldap-server.example.com/
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = True
cache_credentials = True
ldap_tls_cacertdir = /etc/openldap/certs
ldap_tls_reqcert = allow

[sssd]
services = nss, pam, autofs
domains = default

[nss]
homedir_substring = /home
…
```

7. In **/etc/sssd/sssd.conf**, specify the TLS authentication requirement by modifying the **ldap_tls_cacert** and **ldap_tls_reqcert** values in the **[domain]** section:

```
…
cache_credentials = True
ldap_tls_cacert = /etc/openldap/certs/core-dirsrv.ca.pem
ldap_tls_reqcert = hard
…
```

8. Change the permissions on the **/etc/sssd/sssd.conf** file:

```
# chmod 600 /etc/sssd/sssd.conf
```

9. Restart and enable the SSSD service and the **oddjobd** daemon:

```
# systemctl restart sssd oddjobd
# systemctl enable sssd oddjobd
```

10. Optional: If your LDAP server uses the deprecated TLS 1.0 or TLS 1.1 protocols, switch the system-wide cryptographic policy on the client system to the LEGACY level to allow RHEL to communicate using these protocols:

```
# update-crypto-policies --set LEGACY
```

For more details, see the Strong crypto defaults in RHEL 8 and deprecation of weak crypto algorithms Knowledgebase article on the Red Hat Customer Portal and the **update-crypto-policies(8)** man page on your system.

Verification

- Verify you can retrieve user data from your LDAP server by using the **id** command and specifying an LDAP user:

```
# id ldap_user
uid=17388(ldap_user) gid=45367(sysadmins)
groups=45367(sysadmins),25395(engineers),10(wheel),1202200000(admins)
```

The system administrator can now query users from LDAP using the **id** command. The command returns a correct user ID and group membership.

CHAPTER 6. ADDITIONAL CONFIGURATION FOR IDENTITY AND AUTHENTICATION PROVIDERS

The System Security Services Daemon (SSSD) is a system service to access remote directories and authentication mechanisms. The main configuration file for SSSD is **/etc/sssd/sssd.conf**. The following chapters outline how you can configure SSSD services and domains by modifying the **/etc/sssd/sssd.conf** file to:

- Adjust how SSSD interprets and prints full user names to enable offline authentication.
- Configure DNS Service Discovery, simple Access Provider Rules, and SSSD to apply an LDAP Access Filter.

6.1. ADJUSTING HOW SSSD INTERPRETS FULL USER NAMES

SSSD parses full user name strings into the user name and domain components. By default, SSSD interprets full user names in the format **user_name@domain_name** based on the following regular expression in Python syntax:

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

NOTE

For Identity Management and Active Directory providers, the default user name format is **user_name@domain_name** or **NetBIOS_name\user_name**.

You can adjust how SSSD interprets full user names by adding the **re_expression** option to the **/etc/sssd/sssd.conf** file and defining a custom regular expression.

- To define the regular expression globally, add the regular expression to the **[sssd]** section of the **sssd.conf** file as shown in the Defining regular expressions globally example.
- To define the regular expression for a particular domain, add the regular expression to the corresponding domain section (for example, **[domain/LDAP]**) of the **sssd.conf** file as shown in the Defining regular expressions a particular domain example.

Prerequisites

- **root** access

Procedure

1. Open the **/etc/sssd/sssd.conf** file.
2. Use the **re_expression** option to define a custom regular expression.

Example 6.1. Defining regular expressions globally

To define the regular expressions globally for all domains, add **re_expression** to the **[sssd]** section of the **sssd.conf** file.

You can use the following global expression to define the username in the format of **domain\\username** or **domain@username**:

```
[sssd]
[... file truncated ...]
re_expression = (?P<domain>[^\\]*?)\\?(?P<name>[^\\]+$)
```

Example 6.2. Defining regular expressions a particular domain

To define the regular expressions individually for a particular domain, add **re_expression** to the corresponding domain section of the **sssd.conf** file.

You can use the following global expression to define the username in the format of **domain\\username** or **domain@username** for the LDAP domain:

```
[domain/LDAP]
[... file truncated ...]
re_expression = (?P<domain>[^\\]*?)\\?(?P<name>[^\\]+$)
```

For more details, see the descriptions for **re_expression** in the **SPECIAL SECTIONS** and **DOMAIN SECTIONS** parts of the **sssd.conf(5)** man page on your system.

6.2. ADJUSTING HOW SSSD PRINTS FULL USER NAMES

If the **use_fully_qualified_names** option is enabled in the **/etc/sssd/sssd.conf** file, SSSD prints full user names in the format **name@domain** based on the following expansion by default:

```
%1$s@%2$s
```

NOTE

If **use_fully_qualified_names** is not set or is explicitly set to **false** for trusted domains, it only prints the user name without the domain component.

You can adjust the format in which SSSD prints full user names by adding the **full_name_format** option to the **/etc/sssd/sssd.conf** file and defining a custom expansion.

Prerequisites

- **root** access

Procedure

1. As **root**, open the **/etc/sssd/sssd.conf** file.
2. To define the expansion globally for all domains, add **full_name_format** to the **[sssd]** section of **sssd.conf**.

   ```
   [sssd]
   [... file truncated ...]
   full_name_format = %1$s@%2$s
   ```

   In this case the user name is displayed as **user@domain.test**.

3. To define the user name printing format for a particular domain, add **full_name_format** to the corresponding domain section of **sssd.conf**.

   - To configure the expansion for the Active Directory (AD) domain using **%2$s\%1$s**:

     ```
     [domain/ad.domain]
     [... file truncated ...]
     full_name_format = %2$s\%1$s
     ```

     In this case the user name is displayed as **ad.domain\user**.

   - To configure the expansion for the Active Directory (AD) domain using **%3$s\%1$s**:

     ```
     [domain/ad.domain]
     [... file truncated ...]
     full_name_format = %3$s\%1$s
     ```

     In this case the user name is displayed as **AD\user** if the flat domain name of the Active Directory domain is set to **AD**.

For more details, see the descriptions for **full_name_format** in the **SPECIAL SECTIONS** and **DOMAIN SECTIONS** parts of the **sssd.conf(5)** man page on your system.

NOTE

SSSD can strip the domain component of the name in some name configurations, which can cause authentication errors. If you set **full_name_format** to a non-standard value, you will get a warning prompting you to change it to a standard format.

6.3. ENABLING OFFLINE AUTHENTICATION

SSSD does not cache user credentials by default. When processing authentication requests, SSSD always contacts the identity provider. If the provider is unavailable, user authentication fails.

To ensure that users can authenticate even when the identity provider is unavailable, you can enable credential caching by setting **cache_credentials** to **true** in the **/etc/sssd/sssd.conf** file. Cached credentials refer to passwords and the first authentication factor if two-factor authentication is used. Note that for passkey and smart card authentication, you do not need to set **cache_credentials** to true or set any additional configuration; they are expected to work offline as long as a successful online authentication is recorded in the cache.

IMPORTANT

SSSD never caches passwords in plain text. It stores only a hash of the password.

While credentials are stored as a salted SHA-512 hash, this potentially poses a security risk in case an attacker manages to access the cache file and break a password using a brute force attack. Accessing a cache file requires privileged access, which is the default on RHEL.

Prerequisites

- **root** access

Procedure

Procedure

1. Open the **/etc/sssd/sssd.conf** file.
2. In a domain section, add the **cache_credentials = true** setting:

```
[domain/your-domain-name]
cache_credentials = true
```

3. *Optional, but recommended*: Configure a time limit for how long SSSD allows offline authentication if the identity provider is unavailable:
   a. Configure the PAM service to work with SSSD.
   See Configuring user authentication using authselect for more details.
   b. Use the **offline_credentials_expiration** option to specify the time limit.
   Note that the limit is set in days.

   For example, to specify that users are able to authenticate offline for 3 days since the last successful login, use:

```
[pam]
offline_credentials_expiration = 3
```

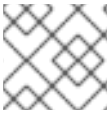
Additional resources

- **sssd.conf(5)** man page on your system

6.4. CONFIGURING DNS SERVICE DISCOVERY

DNS service discovery enables applications to check the SRV records in a given domain for certain services of a certain type, and then returns any servers that match the required type. If the identity or authentication server is not explicitly defined in the **/etc/sssd/sssd.conf** file, SSSD can discover the server dynamically using DNS service discovery.

For example, if **sssd.conf** includes the **id_provider = ldap** setting, but the **ldap_uri** option does not specify any host name or IP address, SSSD uses DNS service discovery to discover the server dynamically.



NOTE

SSSD cannot dynamically discover backup servers, only the primary server.

Prerequisites

- **root** access

Procedure

1. Open the **/etc/sssd/sssd.conf** file.
2. Set the primary server value to **_srv_**.
   For an LDAP provider, the primary server is set using the **ldap_uri** option:

```
[domain/your-domain-name]
id_provider = ldap
ldap_uri = _srv_
```

3. Enable service discovery in the password change provider by setting a service type:

```
[domain/your-domain-name]
id_provider = ldap
ldap_uri = _srv_

chpass_provider = ldap
ldap_chpass_dns_service_name = ldap
```

4. Optional: By default, the service discovery uses the domain portion of the system host name as the domain name. To use a different DNS domain, specify the domain name by using the **dns_discovery_domain** option.

5. Optional: By default, the service discovery scans for the LDAP service type. To use a different service type, specify the type by using the **ldap_dns_service_name** option.

6. Optional: By default, SSSD attempts to look up an IPv4 address. If the attempt fails, SSSD attempts to look up an IPv6 address. To customize this behavior, use the **lookup_family_order** option.

7. For every service with which you want to use service discovery, add a DNS record to the DNS server:

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

Additional resources

- RFC 2782 on DNS service discovery
- **sssd.conf(5)** man page on your system

6.5. CONFIGURING SIMPLE ACCESS PROVIDER RULES

The **simple** access provider allows or denies access based on a list of user names or groups. It enables you to restrict access to specific machines.

For example, you can use the **simple** access provider to restrict access to a specific user or group. Other users or groups will not be allowed to log in even if they authenticate successfully against the configured authentication provider.

Prerequisites

- **root** access

Procedure

1. Open the **/etc/sssd/sssd.conf** file.

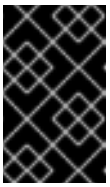
2. Set the **access_provider** option to **simple**:

```
[domain/your-domain-name]
access_provider = simple
```

3. Define the access control rules for users.

   a. To allow access to users, use the **simple_allow_users** option.

   b. To deny access to users, use the **simple_deny_users** option.



IMPORTANT

If you deny access to specific users, you automatically allow access to everyone else. Allowing access to specific users is considered safer than denying.

4. Define the access control rules for groups. Choose one of the following:

   a. To allow access to groups, use the **simple_allow_groups** option.

   b. To deny access to groups, use the **simple_deny_groups** option.

IMPORTANT

If you deny access to specific groups, you automatically allow access to everyone else. Allowing access to specific groups is considered safer than denying.

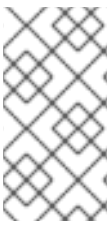
Example 6.3. Allowing access to specific users and groups

The following example allows access to user1, user2, and members of group1, while denying access to all other users:

```
[domain/your-domain-name]
access_provider = simple
simple_allow_users = user1, user2
simple_allow_groups = group1
```

IMPORTANT

Keeping the deny list empty can lead to allowing access to everyone.



NOTE

If you are adding a trusted AD user to the **simple_allow_users** list, ensure that you use the fully qualified domain name (FQDN) format, for example, aduser@ad.example.com. As short names in different domains can be the same, this prevents issues with the access control configuration.

Additional resources

- **sssd-simple** man page on your system

6.6. CONFIGURING SSSD TO APPLY AN LDAP ACCESS FILTER

When the **access_provider** option is set in **/etc/sssd/sssd.conf**, SSSD uses the specified access provider to evaluate which users are granted access to the system. If the access provider you are using is an extension of the LDAP provider type, you can also specify an LDAP access control filter that a user must match to be allowed access to the system.

For example, when using the Active Directory (AD) server as the access provider, you can restrict access to the Linux system only to specified AD users. All other users that do not match the specified filter have access denied.

NOTE

The access filter is applied on the LDAP user entry only. Therefore, using this type of access control on nested groups might not work. To apply access control on nested groups, see Configuring **simple** Access Provider Rules .

IMPORTANT

When using offline caching, SSSD checks if the user's most recent online login attempt was successful. Users who logged in successfully during the most recent online login will still be able to log in offline, even if they do not match the access filter.

Prerequisites

- **root** access

Procedure

1. Open the **/etc/sssd/sssd.conf** file.
2. In the **[domain]** section, specify the access control filter.
   - For an LDAP, use the **ldap_access_filter** option.
   - For an AD, use the **ad_access_filter** option. Additionally, you must disable the GPO-based access control by setting the **ad_gpo_access_control** option to **disabled**.

     Example 6.4. Allowing access to specific AD users

     For example, to allow access only to AD users who belong to the **admins** user group and have a **unixHomeDirectory** attribute set, use:

     ```
     [domain/your-AD-domain-name]
     access provider = ad
     [... file truncated ...]
     ad_access_filter = (&(memberOf=cn=admins,ou=groups,dc=example,dc=com)
     (unixHomeDirectory=*))
     ad_gpo_access_control = disabled
     ```

SSSD can also check results by the **authorizedService** or **host** attribute in an entry. In fact, all options MDASH LDAP filter, **authorizedService**, and **host** MDASH can be evaluated, depending on the user entry and the configuration. The **ldap_access_order** parameter lists all access control methods to use, ordered as how they should be evaluated.

```
[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
```

Additional resources

- **sssd-ldap(5)** and **sssd-ad(5)** man pages on your system

CHAPTER 7. SSSD CLIENT-SIDE VIEW

SSSD provides the **sss_override** utility, which allows you to create a local view that displays values for POSIX user or group attributes that are specific to your local machine. You can configure overrides for all **id_provider** values, except **ipa**.

If you are using the **ipa** provider, define ID views centrally in IPA. For more information, see Using an ID view to override a user attribute value on an IdM client.

For information about a potential negative impact on the SSSD performance, see Potential negative impact of ID views on SSSD performance.

7.1. OVERRIDING THE LDAP USERNAME ATTRIBUTE

As an administrator, you can configure an existing host to use accounts from LDAP. However, the values for a user (name, UID, GID, home directory, shell) in LDAP are different from the values on the local system. You can override the LDAP **username** attribute by defining a secondary **username** with the following procedure.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

1. Display the current information for the user:

```
# id username
```

   Replace *username* with the name of the user.

2. Add the secondary **username**:

```
# sss_override user-add username -n secondary-username
```

   Replace *username* with the name of the user and replace *secondary-username* with the new **username**.

3. After creating the first override using the **sss_override user-add** command, restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

Verification

- Verify that the new **username** is added:

```
# id secondary-username
```

- *Optional*. Display the overrides for the user:

```
# sss_override user-show user-name
user@ldap.example.com:secondary-username::::::
```

Example 7.1. Defining a secondary username

To add a secondary **username** *sarah* for the user *sjones*:

1. Display the current information for the user *sjones*:

```
# id sjones
uid=1001(sjones) gid=6003 groups=6003,10(wheel)
```

2. Add the secondary **username**:

```
# sss_override user-add sjones -n sarah
```

3. Verify that the new **username** has been added and overrides for the user display correctly:

```
# id sarah
uid=1001(sjones) gid=6003(sjones) groups=6003(sjones),10(wheel)

# sss_override user-show sjones
user@ldap.example.com:sarah::::::
```

Additional resources

- **sss_override** man page on your system

7.2. OVERRIDING THE LDAP UID ATTRIBUTE

As an administrator, you can configure an existing host to use accounts from LDAP. However, the values for a user (name, UID, GID, home directory, shell) in LDAP are different from the values on the local system. You can override the LDAP UID attribute by defining a different UID with the following procedure.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

1. Display the current UID of the user:

```
# id -u user-name
```

Replace *user-name* with the name of the user.

2. Override the UID of the user's account:

```
# sss_override user-add user-name -u new-UID
```

Replace *user-name* with the name of the user and replace *new-UID* with the new UID number.

3. Expire the in-memory cache:

```
# sss_cache --users
```

4. After creating the first override using the **sss_override user-add** command, restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

Verification

- Verify that the new UID has been applied:

```
# id -u user-name
```

- *Optional*. Display the overrides for the user:

```
# sss_override user-show user-name
user@ldap.example.com::new-UID:::::
```

Example 7.2. Overriding the UID of the user

To override the UID of the user *sarah* with UID *6666*:

1. Display the current UID of the user *sarah*:

```
# id -u sarah
1001
```

2. Override the UID of the user *sarah*'s account with UID *6666*:

```
# sss_override user-add sarah -u 6666
```

3. Manually expire the in-memory cache:

```
# sss_cache --users
```

4. Restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

5. Verify that the new UID is applied and overrides for the user display correctly:

```
# id sarah
6666

# sss_override user-show sarah
user@ldap.example.com::6666:::::
```

```
```

Additional resources

- **sss_override** man page on your system

7.3. OVERRIDING THE LDAP GID ATTRIBUTE

As an administrator, you can configure an existing host to use accounts from LDAP. However, the values for a user (name, UID, GID, home directory, shell) in LDAP are different from the values on the local system. You can override the LDAP GID attribute by defining a different GID with the following procedure.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

1. Display the current GID of the user:

```
# id -g user-name
```

   Replace *user-name* with the name of the user.

2. Override the GID of the user's account:

```
# sss_override user-add user-name -g new-GID
```

   Replace *user-name* with the name of the user and replace *new-GID* with the new GID number.

3. Expire the in-memory cache:

```
# sss_cache --users
```

4. After creating the first override using the **sss_override user-add** command, restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

Verification

- Verify that the new GID is applied:

```
# id -g user-name
```

- *Optional*. Display the overrides for the user:

```
# sss_override user-show user-name
user@ldap.example.com:::6666::::
```

Example 7.3. Overriding the GID of the user

To override the GID of the user *sarah* with GID *6666*:

1. Display the current GID of the user *sarah*:

```
# id -g sarah
6003
```

2. Override the GID of the user *sarah*'s account with GID *6666*:

```
# sss_override user-add sarah -g 6666
```

3. Manually expire the in-memory cache:

```
# sss_cache --users
```

4. If this is your first override, restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

5. Verify that the new GID is applied and overrides for the user display correctly:

```
# id -g sarah
6666

# sss_override user-show sarah
user@ldap.example.com::6666:::::
```

Additional resources

- **sss_override** man page on your system

7.4. OVERRIDING THE LDAP HOME DIRECTORY ATTRIBUTE

As an administrator, you can configure an existing host to use accounts from LDAP. However, the values for a user (name, UID, GID, home directory, shell) in LDAP are different from the values on the local system. You can override the LDAP home directory attribute by defining a different home directory with the following procedure.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

1. Display the current home directory of the user:

```
# getent passwd user-name
user-name:x:XXXX:XXXX::/home/home-directory:/bin/bash
```

Replace *user-name* with the name of the user.

2. Override the home directory of the user:

```
# sss_override user-add user-name -h new-home-directory
```

Replace *user-name* with the name of the user and replace *new-home-directory* with the new home directory.

3. Restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

Verification

- Verify that the new home directory is defined:

```
# getent passwd user-name
user-name:x:XXXX:XXXX::/home/new-home-directory:/bin/bash
```

- *Optional*. Display the overrides for the user:

```
# sss_override user-show user-name
user@ldap.example.com:::::::new-home-directory::
```

Example 7.4. Overriding the home directory of the user

To override the home directory of the user *sarah* with *admin*:

1. Display the current home directory of the user *sarah*:

```
# getent passwd sarah
sarah:x:1001:6003::sarah:/bin/bash
```

2. Override the home directory of the user *sarah* with new home directory *admin*:

```
# sss_override user-add sarah -h admin
```

3. Restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

4. Verify that the new home directory is defined and overrides for the user display correctly:

```
# getent passwd sarah
sarah:x:1001:6003::admin:/bin/bash

# sss_override user-show user-name
user@ldap.example.com:::::::admin::
```

Additional resources

- **sss_override** man page on your system

7.5. OVERRIDING THE LDAP SHELL ATTRIBUTE

As an administrator, you can configure an existing host to use accounts from LDAP. However, the values for a user (name, UID, GID, home directory, shell) in LDAP are different from the values on the local system. You can override the LDAP shell attribute by defining a different shell with the following procedure.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

1. Display the current shell of the user:

```
# getent passwd user-name
user-name:x:XXXX:XXXX::/home/home-directory:/bin/bash
```

   Replace *user-name* with the name of the user.

2. Override the shell of the user:

```
# sss_override user-add user-name -s new-shell
```

   Replace *user-name* with the name of the user and replace *new-shell* with the new shell.

3. Restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

Verification

- Verify that the new shell is defined:

```
# getent passwd user-name
user-name:x:XXXX:XXXX::/home/home-directory:new-shell
```

- *Optional*. Display the overrides for the user:

```
# sss_override user-show user-name
user@ldap.example.com::::::new-shell:
```

Example 7.5. Overriding the shell of the user

To change the shell of the user *sarah* from **/bin/bash** to **sbin/nologin**:

1. Display the current shell of the user *sarah*:

```
# getent passwd sarah
sarah:x:1001:6003::sarah:/bin/bash
```

2. Override the shell of the user *sarah* with new **/sbin/nologin** shell:

```
# sss_override user-add sarah -s /sbin/nologin
```

3. Restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

4. Verify that the new shell is defined and overrides for the user display correctly:

```
# getent passwd sarah
sarah:x:1001:6003::sarah:/sbin/nologin

# sss_override user-show user-name
user@ldap.example.com:::::::/sbin/nologin:
```

Additional resources

- **sss_override** man page on your system

7.6. LISTING OVERRIDES ON A HOST

As an administrator, you can list all user and group overrides on a host to verify that the correct attributes have been overridden.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

- List all user overrides:

```
# sss_override user-find
user1@ldap.example.com::8000::::/bin/zsh:
user2@ldap.example.com::8001::::/bin/bash:
...
```

- List all group overrides:

```
# sss_override group-find
group1@ldap.example.com::7000
group2@ldap.example.com::7001
...
```

7.7. REMOVING A LOCAL OVERRIDE

If you want to remove local override that is defined in the global LDAP directory, use the following procedure.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

- To remove the override for a user account, use:

```
# sss_override user-del user-name
```

  Replace *user-name* with the name of the user. The changes take effect immediately.

- To remove an override for a group, use:

```
# sss_override group-del group-name
```

- After removing the first override using the **sss_override user-del** or **sss_override group-del** command, restart SSSD for the changes to take effect:

```
# systemctl restart sssd
```

When you remove overrides for a user or group, all overrides for this object are removed.

7.8. EXPORTING AND IMPORTING LOCAL VIEW

Your local overrides are stored in the local SSSD cache. You can export user and group overrides from this cache to a file to create a backup. This ensures that even if the cache is cleared, you can restore the configurations later.

Prerequisites

- **root** access
- Installed **sssd-tools**

Procedure

- To back up user and group view, use:

```
# sss_override user-export /var/lib/sss/backup/sssd_user_overrides.bak
# sss_override group-export /var/lib/sss/backup/sssd_group_overrides.bak
```

- To restore user and group view, use:

```
# sss_override user-import /var/lib/sss/backup/sssd_user_overrides.bak
# sss_override group-import /var/lib/sss/backup/sssd_group_overrides.bak
```

CHAPTER 8. CONFIGURING A RHEL HOST TO USE AD AS AN AUTHENTICATION PROVIDER

As a system administrator, you can use Active Directory (AD) as the authentication provider for a Red Hat Enterprise Linux (RHEL) host without joining the host to AD.

This can be done if, for example:

- You do not want to grant AD administrators the control over enabling and disabling the host.
- The host, which can be a corporate PC, is only meant to be used by one user in your company.

IMPORTANT

Implement this procedure only in the rare cases where this approach is preferred.

Consider fully joining the system to AD or Red Hat Identity Management (IdM) instead. Joining the RHEL host to a domain makes the setup easier to manage. If you are concerned about client access licences related to joining clients into AD directly, consider leveraging an IdM server that is in a trust agreement with AD. For more information about an IdM-AD trust, see Planning a cross-forest trust between IdM and AD and Installing a trust between IdM and AD.

This procedure enables the user named **AD_user** to log in to the **rhel_host** system using the password set in the Active Directory (AD) user database in the **example.com** domain. In this example, the **EXAMPLE.COM** Kerberos realm corresponds to the **example.com** domain.

Prerequisites

- You have root access to **rhel_host**.
- The **AD_user** user account exists in the **example.com** domain.
- The Kerberos realm is **EXAMPLE.COM**.
- **rhel_host** has not been joined to AD using the **realm join** command.
- You have installed the **sssd-proxy** package.

```
$ dnf install sssd-proxy
```

Procedure

1. Create the **AD_user** user account locally without assigning a password to it:

```
# useradd AD_user
```

2. Open the **/etc/nsswitch.conf** file for editing, and make sure that it contains the following lines:

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

3. Open the **/etc/krb5.conf** file for editing, and make sure that it contains the following sections and items:

```
# To opt out of the system crypto-policies configuration of krb5, remove the
# symlink at /etc/krb5.conf.d/crypto-policies which will not be recreated.
includedir /etc/krb5.conf.d/

[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
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = /etc/pki/tls/certs/ca-bundle.crt
    spake_preauth_groups = edwards25519
    default_realm = EXAMPLE.COM
    default_ccache_name = KEYRING:persistent:%{uid}

[realms]
 EXAMPLE.COM = {
     kdc = ad.example.com
     admin_server = ad.example.com
 }

[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

4. Create the **/etc/sssd/sssd.conf** file and insert the following sections and lines into it:

```
[sssd]
    services = nss, pam
    domains = EXAMPLE.COM

[domain/EXAMPLE.COM]
    id_provider = proxy
    proxy_lib_name = files
    auth_provider = krb5
    krb5_realm = EXAMPLE.COM
    krb5_server = ad.example.com
```

5. Change the permissions on the **/etc/sssd/sssd.conf** file:

```
# chmod 600 /etc/sssd/sssd.conf
```

6. Start the Security System Services Daemon (SSSD):

```
# systemctl start sssd
```

7. Enable SSSD:

```
# systemctl enable sssd
```

8. Open the **/etc/pam.d/system-auth** file, and modify it so that it contains the following sections and lines:

```
# Generated by authselect on Wed May  8 08:55:04 2019
# Do not modify this file manually.

auth        required                                    pam_env.so
auth        required                                    pam_faildelay.so delay=2000000
auth        [default=1 ignore=ignore success=ok]        pam_succeed_if.so uid >= 1000 quiet
auth        [default=1 ignore=ignore success=ok]        pam_localuser.so
auth        sufficient                                  pam_unix.so nullok try_first_pass
auth        requisite                                   pam_succeed_if.so uid >= 1000 quiet_success
auth        sufficient                                  pam_sss.so forward_pass
auth        required                                    pam_deny.so

account     required                                    pam_unix.so
account     sufficient                                  pam_localuser.so
account     sufficient                                  pam_succeed_if.so uid < 1000 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 local_users_only
password    sufficient                                  pam_unix.so sha512 shadow nullok try_first_pass
use_authtok
password    sufficient                                  pam_sss.so use_authtok
password    required                                    pam_deny.so

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

9. Copy the contents of the **/etc/pam.d/system-auth** file into the **/etc/pam.d/password-auth** file. Enter **yes** to confirm the overwriting of the current contents of the file:

```
# cp /etc/pam.d/system-auth /etc/pam.d/password-auth
cp: overwrite '/etc/pam.d/password-auth'? yes
```

Verification

1. Request a Kerberos ticket-granting ticket (TGT) for **AD_user**. Enter the password of **AD_user** as requested:

```
# kinit AD_user
Password for AD_user@EXAMPLE.COM:
```

2. Display the obtained TGT:

```
# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: AD_user@EXAMPLE.COM

Valid starting     Expires            Service principal
11/02/20 04:16:38  11/02/20 14:16:38  krbtgt/EXAMPLE.COM@EXAMPLE.COM
 renew until 18/02/20 04:16:34
```

AD_user has successfully logged in to **rhel_host** using the credentials from the **EXAMPLE.COM** Kerberos domain.

CHAPTER 9. REPORTING ON USER ACCESS ON HOSTS USING SSSD

The Security System Services Daemon (SSSD) tracks which users can or cannot access clients. This chapter describes creating access control reports and displaying user data using the **sssctl** tool.

Prerequisites

- SSSD packages are installed in your network environment

9.1. THE SSSCTL COMMAND

sssctl is a command-line tool that provides a unified way to obtain information about the Security System Services Daemon (SSSD) status.

You can use the **sssctl** utility to gather information about:

- Domain state
- Client user authentication
- User access on clients of a particular domain
- Information about cached content

With the **sssctl** tool, you can:

- Manage the SSSD cache
- Manage logs
- Check configuration files

NOTE

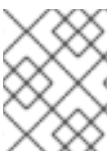
The **sssctl** tool replaces **sss_cache** and **sss_debuglevel** tools.

Additional resources

- **sssctl --help**

9.2. GENERATING ACCESS CONTROL REPORTS USING SSSCTL

You can list the access control rules applied to the machine on which you are running the report because SSSD controls which users can log in to the client.



NOTE

The access report is not accurate because the tool does not track users locked out by the Key Distribution Center (KDC).

Prerequisites

- You must be logged in with administrator privileges
- The **sssctl** tool is available on RHEL 7, RHEL 8, and RHEL 9 systems.

Procedure

- To generate a report for the **idm.example.com** domain, enter:

```
[root@client1 ~]# sssctl access-report idm.example.com
1 rule cached

Rule name: example.user
 Member users: example.user
 Member services: sshd
```

9.3. DISPLAYING USER AUTHORIZATION DETAILS USING SSSCTL

The **sssctl user-checks** command helps debug problems in applications that use the System Security Services Daemon (SSSD) for user lookup, authentication, and authorization.

The **sssctl user-checks [USER_NAME]** command displays user data available through Name Service Switch (NSS) and the InfoPipe responder for the D-Bus interface. The displayed data shows whether the user is authorized to log in using the **system-auth** Pluggable Authentication Module (PAM) service.

The command has two options:

- **-a** for a PAM action
- **-s** for a PAM service

If you do not define **-a** and **-s** options, the **sssctl** tool uses default options: **-a acct -s system-auth**.

Prerequisites

- You must be logged in with administrator privileges
- The **sssctl** tool is available on RHEL 7, RHEL 8, and RHEL 9 systems.

Procedure

- To display user data for a particular user, enter:

```
[root@client1 ~]# sssctl user-checks -a acct -s sshd example.user
user: example.user
action: acct
service: sshd
....
```

Additional resources

- **sssctl user-checks --help**

CHAPTER 10. QUERYING DOMAIN INFORMATION USING SSSD

Security System Services Daemon (SSSD) can list domains in Identity Management (IdM) as well as the domains in Active Directory that is connected to IdM by a cross-forest trust.

10.1. LISTING DOMAINS USING SSSCTL

You can use the **sssctl domain-list** command to debug problems with the domain topology.

NOTE

The status might not be available immediately. If the domain is not visible, repeat the command.

Prerequisites

- You must be logged in with administrator privileges
- The **sssctl** tool is available on RHEL 7, RHEL 8, and RHEL 9 systems.

Procedure

1. To display help for the sssctl command, enter:

```
[root@client1 ~]# sssctl --help
....
```

2. To display a list of available domains, enter:

```
[root@client1 ~]# sssctl domain-list
implicit_files
idm.example.com
ad.example.com
sub1.ad.example.com
```

The list includes domains in the cross-forest trust between Active Directory and Identity Management.

10.2. VERIFYING THE DOMAIN STATUS USING SSSCTL

You can use the **sssctl domain-status** command to debug problems with the domain topology.

NOTE

The status might not be available immediately. If the domain is not visible, repeat the command.

Prerequisites

- You must be logged in with administrator privileges
- The **sssctl** tool is available on RHEL 7, RHEL 8, and RHEL 9 systems.

Procedure

1. To display help for the sssctl command, enter:

```
[root@client1 ~]# sssctl --help
```

2. To display user data for a particular domain, enter:

```
[root@client1 ~]# sssctl domain-status idm.example.com
Online status: Online

Active servers:
IPA: server.idm.example.com

Discovered IPA servers:
- server.idm.example.com
```

The domain **idm.example.com** is online and visible from the client where you applied the command.

If the domain is not available, the result is:

```
[root@client1 ~]# sssctl domain-status ad.example.com
Unable to get online status
```

CHAPTER 11. RESTRICTING DOMAINS FOR PAM SERVICES USING SSSD

Pluggable authentication modules (PAMs) are a common framework for authentication and authorization. Most system applications in Red Hat Enterprise Linux depend on underlying PAM configuration for authentication and authorization.

System Security Services Daemon (SSSD) enables you to restrict which domains PAM services can access. SSSD evaluates authentication requests from PAM services based on the user that runs the particular PAM service. This means, if the PAM service user can access an SSSD domain then the PAM service also can access that domain.

11.1. ABOUT PAM

Pluggable Authentication Modules (PAMs) provide a centralized authentication mechanism, which a system application can use to relay authentication to a centrally configured framework.

PAM is pluggable because a PAM module exists for different types of authentication sources, such as Kerberos, SSSD, NIS, or the local file system. You can prioritize different authentication sources.

This modular architecture offers administrators a great deal of flexibility in setting authentication policies for the system. PAM is a useful system for developers and administrators for several reasons:

- PAM provides a common authentication scheme, which can be used with a wide variety of applications.
- PAM provides significant flexibility and control over authentication for system administrators.
- PAM provides a single, fully-documented library, which allows developers to write programs without having to create their own authentication schemes.

11.2. DOMAIN-ACCESS RESTRICTION OPTIONS

The following options are available to restrict access to selected domains:

pam_trusted_users in /etc/sssd/sssd.conf

This option accepts a list of numerical UIDs or user names representing the PAM services that SSSD trusts. The default setting is **all**, which means all service users are trusted and can access any domain.

pam_public_domains in /etc/sssd/sssd.conf

This option accepts a list of public SSSD domains. Public domains are domains accessible even for untrusted PAM service users. The option also accepts the **all** and **none** values. The default value is **none**, which means no domains are public and untrusted service users cannot access any domain.

domains for PAM configuration files

This option specifies a list of domains against which a PAM service can authenticate. If you use **domains** without specifying any domain, the PAM service will not be able to authenticate against any domain, for example:

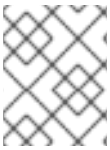
```
auth     required   pam_sss.so domains=
```

If the PAM configuration file uses **domains**, the PAM service is able to authenticate against all domains when that service is running under a trusted user.

The **domains** option in the **/etc/sssd/sssd.conf** SSSD configuration file also specifies a list of domains to which SSSD attempts to authenticate. Note that the **domains** option in a PAM configuration file cannot extend the list of domains in **sssd.conf**, it can only restrict the **sssd.conf** list of domains by specifying a shorter list. Therefore, if a domain is specified in the PAM file but not in **sssd.conf**, the PAM service cannot authenticate against the domain.

The default settings **pam_trusted_users = all** and **pam_public_domains = none** specify that all PAM service users are trusted and can access any domain. Using the **domains** option for PAM configuration files restricts the access to the domains.

Specifying a domain using **domains** in the PAM configuration file while **sssd.conf** contains **pam_public_domains** also requires to specify the domain in **pam_public_domains**. The **pam_public_domains** option without including the required domain leads the PAM service to unsuccessful authentication against the domain in case this service is running under an untrusted user.



NOTE

Domain restrictions defined in a PAM configuration file apply to authentication actions only, not to user lookups.

Additional resources

- For more details on the **pam_trusted_users** and **pam_public_domains** options, see the **sssd.conf(5)** man page on your system.
- For more details on the **domains** option used in PAM configuration files, see the **pam_sss(8)** man page on your system.

11.3. RESTRICTING DOMAINS FOR A PAM SERVICE

This procedure shows how to restrict a PAM service authentication against the domains.

Prerequisites

- SSSD installed and running.

Procedure

1. Configure SSSD to access the required domain or domains. Define the domains against which SSSD can authenticate in the **domains** option in the **/etc/sssd/sssd.conf** file:

```
[sssd]
domains = domain1, domain2, domain3
```

2. Specify the domain or domains to which a PAM service can authenticate by setting the **domains** option in the PAM configuration file. For example:

```
auth        sufficient    pam_sss.so forward_pass domains=domain1
account     [default=bad success=ok user_unknown=ignore] pam_sss.so
password    sufficient    pam_sss.so use_authtok
```

In this example, you allow the PAM service to authenticate against **domain1** only.

Verification

Verification

- Authenticate against **domain1**. It must be successful.

11.4. ABOUT PAM CONFIGURATION FILES

PAM configuration files specify the authentication methods and policies for services in Red Hat Enterprise Linux. Each PAM aware application or service has a corresponding file in the **/etc/pam.d/** directory. The file is named after the service it controls access to. For example, the **login** program has a corresponding PAM configuration file named **/etc/pam.d/login**.

WARNING

Manual editing of PAM configuration files might lead to authentication and access issues. Configure PAMs using the **authselect** tool.

PAM configuration file format

Each PAM configuration file consists of directives that define settings for a specific module. PAM uses arguments to pass information to a pluggable module during authentication for some modules.

```
module_type control_flag module_name module_arguments
```

For example:

```
auth required pam_unix.so
```

PAM module types

A PAM module type specifies the type of authentication task that a module performs. The module can perform the following tasks:

- *account management*
- *authentication management*
- *password management*
- *session management*

An individual module can provide any or all module types. For example, **pam_unix.so** provides all four module types.

The module name, such as **pam_unix.so**, provides PAM with the name of the library containing the specified module type. The directory name is omitted because the application is linked to the appropriate version of **libpam**, which can locate the correct version of the module.

Module type directives can be stacked, or placed upon one another, so that multiple modules are used together for one purpose. The order of the modules is important, along with the control flags, it determines how significant the success or failure of a particular module is to the overall goal of authenticating the user to the service.

You can stack PAM modules to enforce specific conditions that must be met before a user is allowed to authenticate.

PAM control flags
When a PAM module performs its function, it returns a success or failure result. Control flags instruct PAM on how to handle this result.

Simple flags use a keyword, more complex syntax follows **[value1=action1 value2=action2 ...]** format.

NOTE

When a module's control flag uses the **sufficient** or **requisite** value, the order in which the modules are listed is important to the authentication process.

For a detailed description of PAM control flags, including a list of options, see the **pam.conf(5)** man page.

Additional resources

- **pam.conf(5)** man page
- **pam(8)** man page

11.5. PAM CONFIGURATION EXAMPLE

See an example of PAM configuration file with the detailed description.

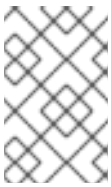
Annotated PAM configuration example

```
#%PAM-1.0
auth  required pam_securetty.so 1
auth  required pam_unix.so nullok 2
auth  required pam_nologin.so 3
account  required pam_unix.so 4
password required pam_pwquality.so retry=3 5
password required pam_unix.so shadow nullok use_authtok 6
session  required pam_unix.so 7
```

1. This line ensures that the root login is allowed only from a terminal which is listed in the **/etc/securetty** file, if this file exists. If the terminal is not listed in the file, the login as root fails with a **Login incorrect** message.

2. Prompts the user for a password and checks it against the information stored in **/etc/passwd** and, if it exists, **/etc/shadow**. The **nullok** argument allows a blank password.

3. Checks if **/etc/nologin** file exists. If it exists, and the user is not root, authentication fails.



NOTE

In this example, all three **auth** modules are checked, even if the first **auth** module fails. This is a good security approach that prevents potential attacker from knowing at what stage their authentication failed.

4. Verifies the user's account. For example, if shadow passwords have been enabled, the account type of the **pam_unix.so** module checks for expired accounts or if the user needs to change the

5. Prompts for a new password if the current one has expired or when the user manually requests a password change. Then it checks the strength of the newly created password to ensure it meets quality requirements and is not easily determined by a dictionary-based password cracking program. The argument **retry=3** specifies that user has three attempts to create a strong password.

6. The **pam_unix.so** module manages password changes. The **shadow** argument instructs the module to create shadow passwords when updating a user's password. The **nullok** argument allows the user to change their password from a blank password, otherwise a null password is treated as an account lock. The **use_authtok** argument accepts any password that was entered previously without prompting the user again. In this way, all new passwords must pass the **pam_pwquality.so** check for secure passwords before being accepted.

7. The **pam_unix.so** module manages the session and logs the user name and service type at the beginning and end of each session. Logs are collected by the **systemd-journald** service and can be viewed by using the **journalctl** command. Logs are also stored in **/var/log/secure**. This module can be supplemented by stacking it with other session modules for additional functionality.

CHAPTER 12. ELIMINATING TYPOGRAPHICAL ERRORS IN LOCAL SSSD CONFIGURATION

You can test if the **/etc/sssd/sssd.conf** file on your host contains any typographical errors using the **sssctl config-check** command.

Prerequisites

- You are logged in as root.
- The **sssd-tools** package is installed.

Procedure

1. Enter the **sssctl config-check** command:

```
# sssctl config-check

Issues identified by validators: 1
[rule/allowed_domain_options]: Attribute 'ldap_search' is not allowed in section
'domain/example1'. Check for typos.

Messages generated during configuration merging: 0

Used configuration snippet files: 0
```

2. Open the **/etc/sssd/sssd.conf** file and correct the typo. If you, for example, received the error message in the previous step, replace **ldap_search** with **ldap_search_base**:

```
[...]
[domain/example1]
ldap_search_base = dc=example,dc=com
[...]
```

3. Save the file.
4. Restart SSSD:

```
# systemctl restart sssd
```

Verification

- Enter the **sssctl config-check** command:

```
# sssctl config-check

Issues identified by validators: 0

Messages generated during configuration merging: 0

Used configuration snippet files: 0
```

The **/etc/sssd/sssd.conf** file now has no typographical errors.

CHAPTER 13. TROUBLESHOOTING AUTHENTICATION WITH SSSD IN IDM

Authentication in an Identity Management (IdM) environment involves many components:

On the IdM client:

- The SSSD service.
- The Name Services Switch (NSS).
- Pluggable Authentication Modules (PAM).

On the IdM server:

- The SSSD service.
- The IdM Directory Server.
- The IdM Kerberos Key Distribution Center (KDC).

If you are authenticating as an Active Directory (AD) user:

- The Directory Server on an AD Domain Controller.
- The Kerberos server on an AD Domain Controller.

To authenticate users, you must be able to perform the following functions with the SSSD service:

- Retrieve user information from the authentication server.
- Prompt the user for their credentials, pass those credentials to the authentication server, and process the outcome.

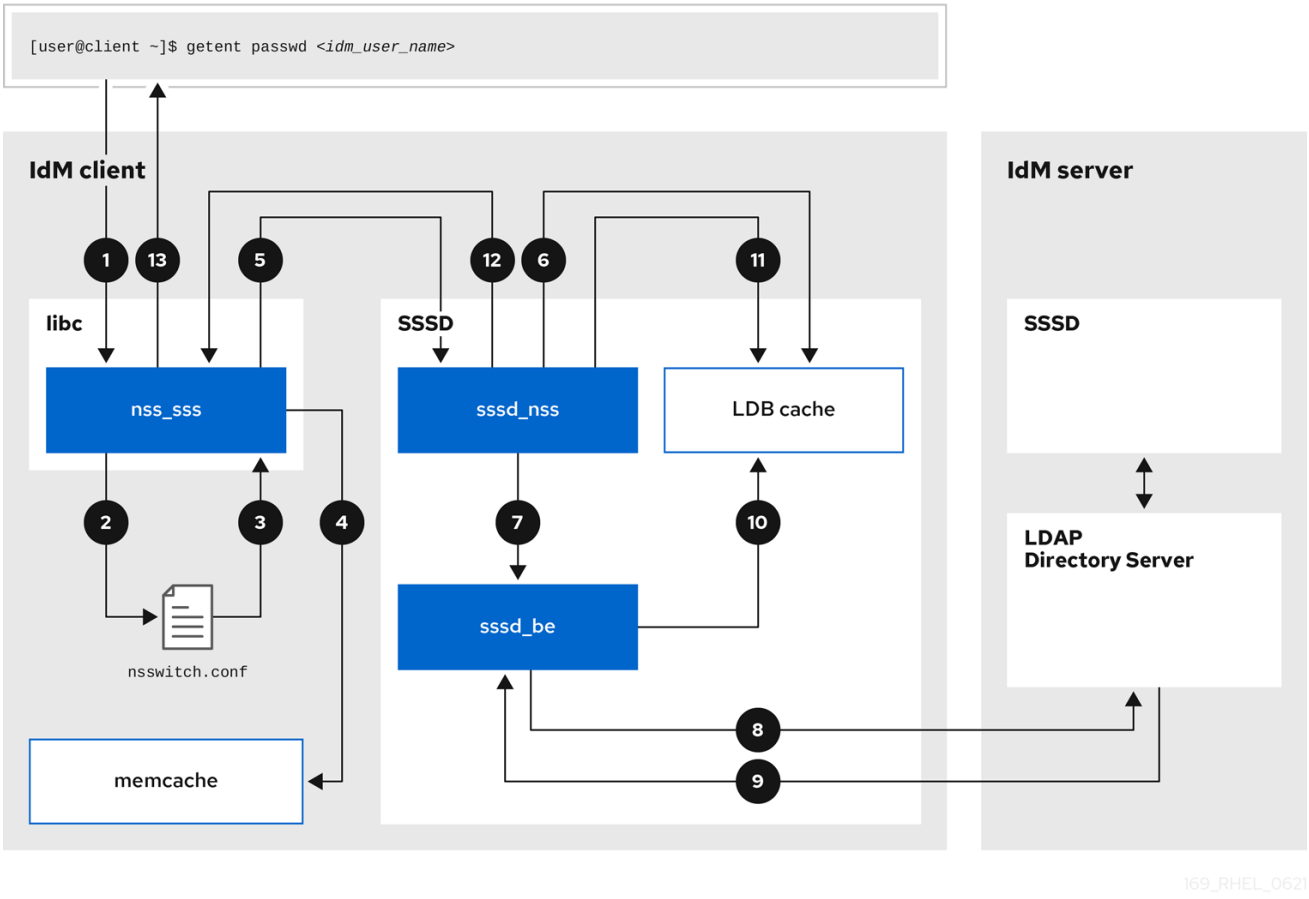
To learn more about how information flows between the SSSD service and servers that store user information, so you can troubleshoot failing authentication attempts in your environment, see the following:

1. Data flow when retrieving IdM user information with SSSD
2. Data flow when retrieving AD user information with SSSD
3. Data flow when authenticating as a user with SSSD in IdM
4. Narrowing the scope of authentication issues
5. SSSD log files and logging levels
6. Enabling detailed logging for SSSD in the sssd.conf file
7. Enabling detailed logging for SSSD with the sssctl command
8. Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM server

9. Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM client
10. Tracking client requests in the SSSD backend
11. Tracking client requests using the log analyzer tool

13.1. DATA FLOW WHEN RETRIEVING IDM USER INFORMATION WITH SSSD

The following diagram is a simplification of the information flow between an IdM client and an IdM server during a request for IdM user information with the command **getent passwd *<idm_user_name>***.



1. The **getent** command triggers the **getpwnam** call from the **libc** library.
2. The **libc** library references the **/etc/nsswitch.conf** configuration file to check which service is responsible for providing user information, and discovers the entry **sss** for the SSSD service.
3. The **libc** library opens the **nss_sss** module.
4. The nss_sss module checks the memory-mapped cache for the user information. If the data is present in the cache, the **nss_sss** module returns it.
5. If the user information is not in the memory-mapped cache, the request is passed to the SSSD **sssd_nss** responder process.

6. The SSSD service checks its cache. If the data is present in the cache and valid, the **sssd_nss** responder reads the data from the cache and returns it to the application.

7. If the data is not present in the cache or it is expired, the **sssd_nss** responder queries the appropriate back-end process and waits for a reply. The SSSD service uses the IPA backend in an IdM environment, enabled by the setting **id_provider=ipa** in the **sssd.conf** configuration file.

8. The **sssd_be** back-end process connects to the IdM server and requests the information from the IdM LDAP Directory Server.

9. The SSSD back-end on the IdM server responds to the SSSD back-end process on the IdM client.

10. The SSSD back-end on the client stores the resulting data in the SSSD cache and alerts the responder process that the cache has been updated.

11. The **sssd_nss** front-end responder process retrieves the information from the SSSD cache.

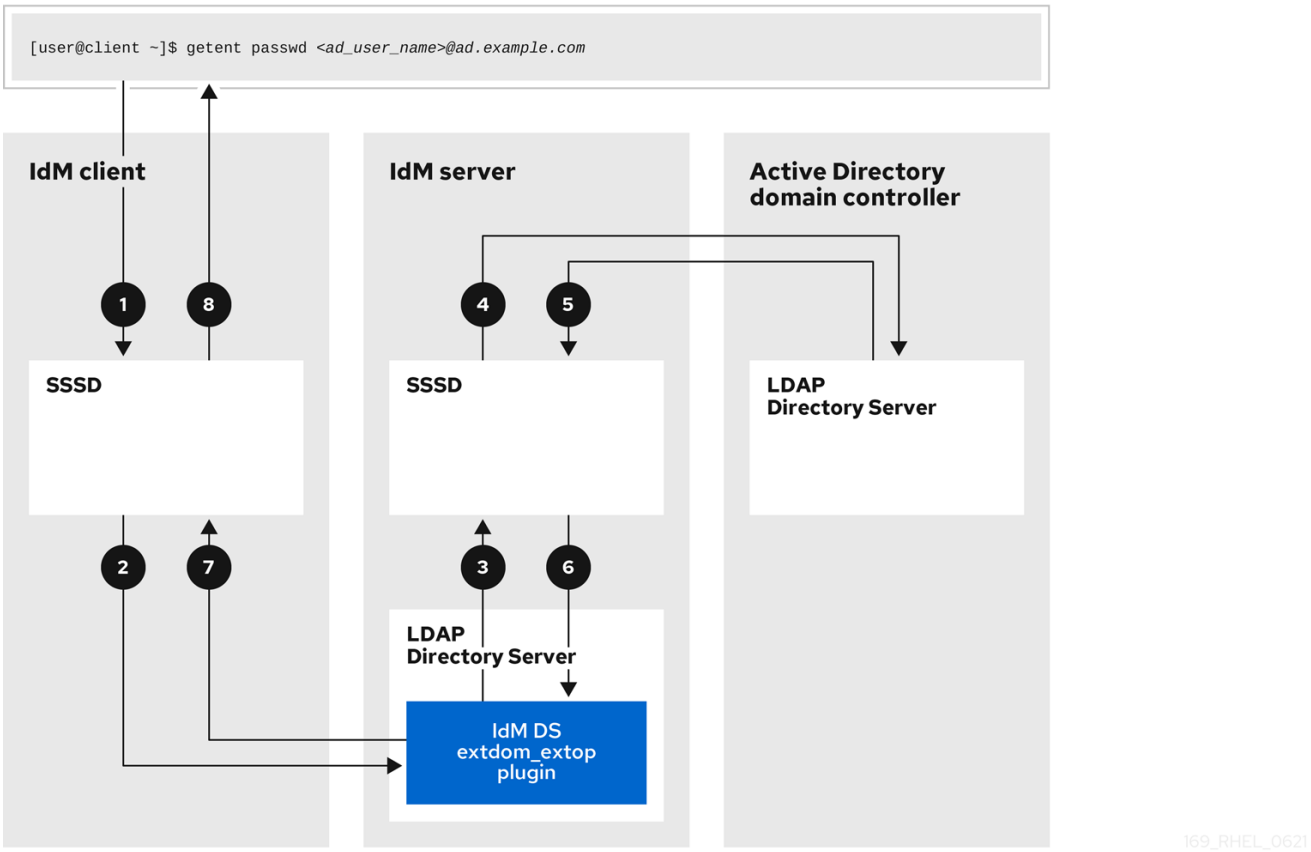
12. The **sssd_nss** responder sends the user information to the **nss_sss** responder, completing the request.

13. The **libc** library returns the user information to the application that requested it.

13.2. DATA FLOW WHEN RETRIEVING AD USER INFORMATION WITH SSSD

If you have established a cross-forest trust between your IdM environment and an Active Directory (AD) domain, the information flow when retrieving AD user information about an IdM client is very similar to the information flow when retrieving IdM user information, with the additional step of contacting the AD user database.

The following diagram is a simplification of the information flow when a user requests information about an AD user with the command **getent passwd *<ad_user_name@ad.example.com>***. This diagram does not include the internal details discussed in the Data flow when retrieving IdM user information with SSSD section. It focuses on the communication between the SSSD service on an IdM client, the SSSD service on an IdM server, and the LDAP database on an AD Domain Controller.



1. The IdM client looks to its local SSSD cache for AD user information.
2. If the IdM client does not have the user information, or the information is stale, the SSSD service on the client contacts the **extdom_extop** plugin on the IdM server to perform an LDAP extended operation and requests the information.
3. The SSSD service on the IdM server looks for the AD user information in its local cache.
4. If the IdM server does not have the user information in its SSSD cache, or its information is stale, it performs an LDAP search to request the user information from an AD Domain Controller.
5. The SSSD service on the IdM server receives the AD user information from the AD domain controller and stores it in its cache.
6. The **extdom_extop** plugin receives the information from the SSSD service on the IdM server, which completes the LDAP extended operation.
7. The SSSD service on the IdM client receives the AD user information from the LDAP extended operation.
8. The IdM client stores the AD user information in its SSSD cache and returns the information to the application that requested it.

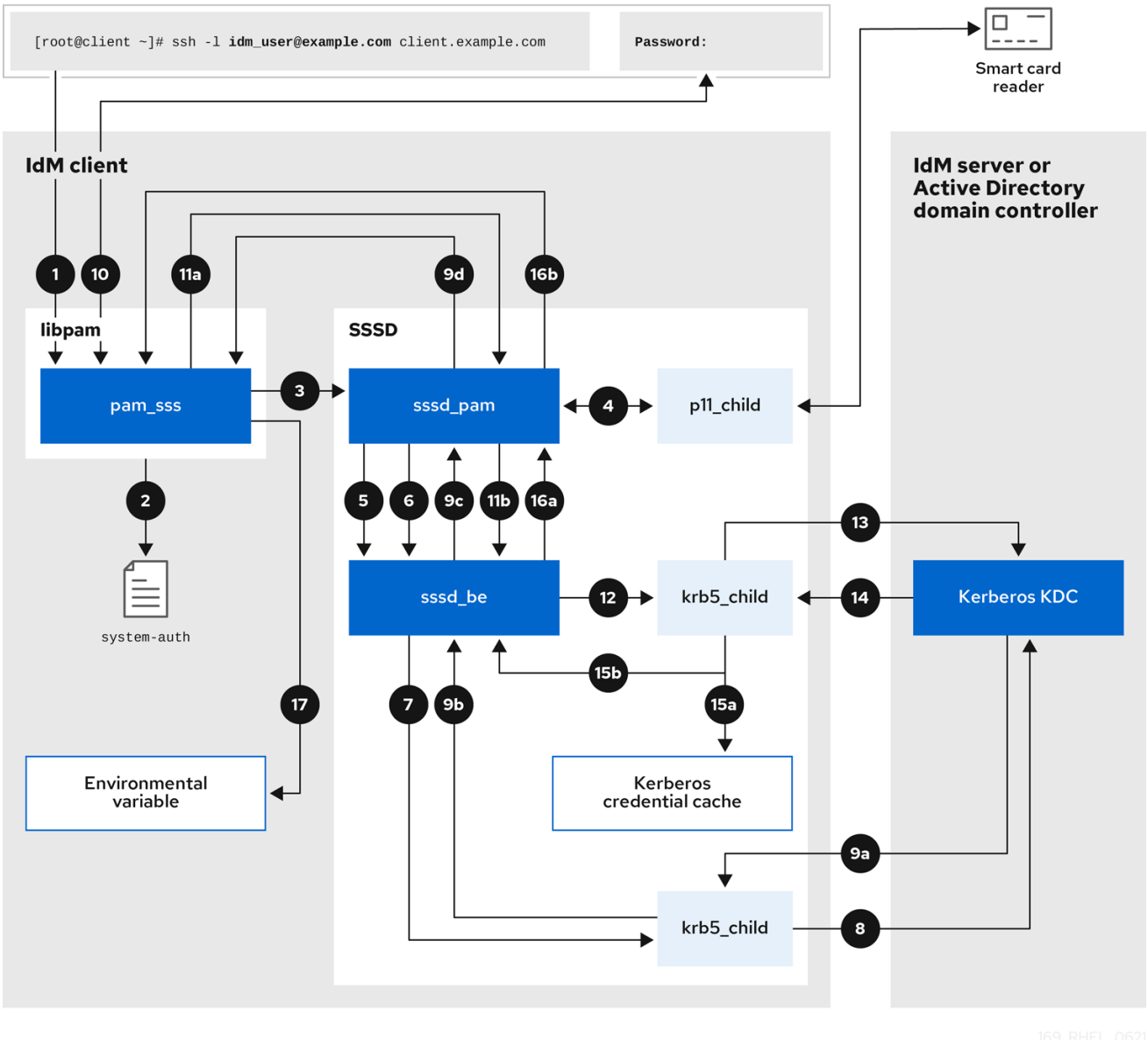
13.3. DATA FLOW WHEN AUTHENTICATING AS A USER WITH SSSD IN IDM

Authenticating as a user on an IdM server or client involves the following components:

- The service that initiates the authentication request, such as the sshd service.

- The Pluggable Authentication Module (PAM) library and its modules.
- The SSSD service, its responders, and back-ends.
- A smart card reader, if smart card authentication is configured.
- The authentication server:
  - IdM users are authenticated against an IdM Kerberos Key Distribution Center (KDC).
  - Active Directory (AD) users are authenticated against an AD Domain Controller (DC).

The following diagram is a simplification of the information flow when a user needs to authenticate during an attempt to log in locally to a host via the SSH service on the command line.



1. The authentication attempt with the **ssh** command triggers the **libpam** library.
2. The **libpam** library references the PAM file in the **/etc/pam.d/** directory that corresponds to the service requesting the authentication attempt. In this example involving authenticating via the SSH service on the local host, the **libpam** library checks the **/etc/pam.d/system-auth** configuration file and discovers the **pam_sss.so** entry for the SSSD PAM:

```
auth    sufficient    pam_sss.so
```

3. To determine which authentication methods are available, the **libpam** library opens the **pam_sss** module and sends an **SSS_PAM_PREAUTH** request to the **sssd_pam** PAM responder of the SSSD service.

4. If smart card authentication is configured, the SSSD service spawns a temporary **p11_child** process to check for a smart card and retrieve certificates from it.

5. If smart card authentication is configured for the user, the **sssd_pam** responder attempts to match the certificate from the smart card with the user. The **sssd_pam** responder also performs a search for the groups that the user belongs to, since group membership might affect access control.

6. The **sssd_pam** responder sends an **SSS_PAM_PREAUTH** request to the **sssd_be** back-end responder to see which authentication methods the server supports, such as passwords or 2-factor authentication. In an IdM environment, where the SSSD service uses the IPA responder, the default authentication method is Kerberos. For this example, the user authenticates with a simple Kerberos password.

7. The **sssd_be** responder spawns a temporary **krb5_child** process.

8. The **krb5_child** process contacts the KDC on the IdM server and checks for available authentication methods.

9. The KDC responds to the request:

    a. The **krb5_child** process evaluates the reply and sends the results back to the **sssd_be** backend process.

    b. The **sssd_be** backend process receives the result.

    c. The **sssd_pam** responder receives the result.

    d. The **pam_sss** module receives the result.

10. If password authentication is configured for the user, the **pam_sss** module prompts the user for their password. If smart card authentication is configured, the **pam_sss** module prompts the user for their smart card PIN.

11. The module sends an **SSS_PAM_AUTHENTICATE** request with the user name and password, which travels to:

    a. The **sssd_pam** responder.

    b. The **sssd_be** back-end process.

12. The **sssd_be** process spawns a temporary **krb5_child** process to contact the KDC.

13. The **krb5_child** process attempts to retrieve a Kerberos Ticket Granting Ticket (TGT) from the KDC with the user name and password the user provided.

14. The **krb5_child** process receives the result of the authentication attempt.

15. The **krb5_child** process:

    a. Stores the TGT in a credential cache.

b. Returns the authentication result to the **sssd_be** back-end process.

16. The authentication result travels from the **sssd_be** process to:

    a. The **sssd_pam** responder.

    b. The **pam_sss** module.

17. The **pam_sss** module sets an environment variable with the location of the user's TGT so other applications can reference it.

13.4. NARROWING THE SCOPE OF AUTHENTICATION ISSUES

To successfully authenticate a user, you must be able to retrieve user information with the SSSD service from the database that stores user information. The following procedure describes steps to test different components of the authentication process so you can narrow the scope of authentication issues when a user is unable to log in.

Procedure

1. Verify that the SSSD service and its processes are running.

```
[root@client ~]# pstree -a | grep sssd
  |-sssd -i --logger=files
  |   |-sssd_be --domain implicit_files --uid 0 --gid 0 --logger=files
  |   |-sssd_be --domain example.com --uid 0 --gid 0 --logger=files
  |   |-sssd_ifp --uid 0 --gid 0 --logger=files
  |   |-sssd_nss --uid 0 --gid 0 --logger=files
  |   |-sssd_pac --uid 0 --gid 0 --logger=files
  |   |-sssd_pam --uid 0 --gid 0 --logger=files
  |   |-sssd_ssh --uid 0 --gid 0 --logger=files
  |   `-sssd_sudo --uid 0 --gid 0 --logger=files
  |-sssd_kcm --uid 0 --gid 0 --logger=files
```

2. Verify that the client can contact the user database server via the IP address.

```
[user@client ~]$ ping <IP_address_of_the_database_server>
```

   If this step fails, check that your network and firewall settings allow direct communication between IdM clients and servers. See Using and configuring firewalld.

3. Verify that the client can discover and contact the IdM LDAP server (for IdM users) or AD domain controller (for AD users) via the fully qualified host name.

```
[user@client ~]$ dig -t SRV _ldap._tcp.example.com @<name_server>
[user@client ~]$ ping <fully_qualified_host_name_of_the_server>
```

   If this step fails, check your Dynamic Name Service (DNS) settings, including the **/etc/resolv.conf** file. See Configuring the order of DNS servers.

NOTE

By default, the SSSD service attempts to automatically discover LDAP servers and AD DCs through DNS service (SRV) records. Alternatively, you can restrict the SSSD service to use specific servers by setting the following options in the **sssd.conf** configuration file:

- **ipa_server = *<fully_qualified_host_name_of_the_server>***
- **ad_server = *<fully_qualified_host_name_of_the_server>***
- **ldap_uri = *<fully_qualified_host_name_of_the_server>***

If you use these options, verify you can contact the servers listed in them.

4. Verify that the client can authenticate to the LDAP server and retrieve user information with **ldapsearch** commands.

   a. If your LDAP server is an IdM server, like **server.example.com**, retrieve a Kerberos ticket for the host and perform the database search authenticating with the host Kerberos principal:

   ```
   [user@client ~]$ kinit -k 'host/client.example.com@EXAMPLE.COM'
   [user@client ~]$ ldapsearch -LLL -Y GSSAPI -h server.example.com -b
   “dc=example,dc=com” uid=<user_name>
   ```

   b. If your LDAP server is an Active Directory (AD) Domain Controller (DC), like **server.ad.example.com**, retrieve a Kerberos ticket for the host and perform the database search authenticating with the host Kerberos principal:

   ```
   [user@client ~]$ kinit -k 'CLIENT$@AD.EXAMPLE.COM'
   [user@client ~]$ ldapsearch -LLL -Y GSSAPI -h server.ad.example.com -b
   “dc=example,dc=com” sAMAccountname=<user_name>
   ```

   c. If your LDAP server is a plain LDAP server, and you have set the **ldap_default_bind_dn** and **ldap_default_authtok** options in the **sssd.conf** file, authenticate as the same **ldap_default_bind_dn** account:

   ```
   [user@client ~]$ ldapsearch -xLLL -D "cn=ldap_default_bind_dn_value" -W -h
   ldapserver.example.com -b “dc=example,dc=com” uid=<user_name>
   ```

   If this step fails, verify that your database settings allow your host to search the LDAP server.

5. Since the SSSD service uses Kerberos encryption, verify you can obtain a Kerberos ticket as the user that is unable to log in.

   a. If your LDAP server is an IdM server:

   ```
   [user@client ~]$ kinit <user_name>
   ```

   b. If LDAP server database is an AD server:

   ```
   [user@client ~]$ kinit <user_name@AD.EXAMPLE.COM>
   ```

If this step fails, verify that your Kerberos server is operating properly, all servers have their times synchronized, and that the user account is not locked.

6. Verify you can retrieve user information about the command line.

```
[user@client ~]$ getent passwd <user_name>
[user@client ~]$ id <user_name>
```

If this step fails, verify that the SSSD service on the client can receive information from the user database:

a. Review errors in the **/var/log/messages** log file.

b. Enable detailed logging in the SSSD service, collect debugging logs, and review the logs for indications to the source of the issue.

c. Optional: Open a Red Hat Technical Support case and provide the troubleshooting information you have gathered.

7. If you are allowed to run **sudo** on the host, use the **sssctl** utility to verify the user is allowed to log in.

```
[user@client ~]$ sudo sssctl user-checks -a auth -s ssh <user_name>
```

If this step fails, verify your authorization settings, such as your PAM configuration, IdM HBAC rules, and IdM RBAC rules:

a. Ensure that the user's UID is equal to or higher than **UID_MIN**, which is defined in the **/etc/login.defs** file.

b. Review authorization errors in the **/var/log/secure** and **/var/log/messages** log files.

c. Enable detailed logging in the SSSD service, collect debugging logs, and review the logs for indications to the source of the issue.

d. Optional: Open a Red Hat Technical Support case and provide the troubleshooting information you have gathered.

Additional resources

- Enabling detailed logging for SSSD in the sssd.conf file
- Enabling detailed logging for SSSD with the sssctl command
- Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM server
- Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM client

13.5. SSSD LOG FILES AND LOGGING LEVELS

Each SSSD service logs into its own log file in the **/var/log/sssd/** directory. For an IdM server in the **example.com** IdM domain, its log files might look like this:

```
[root@server ~]# ls -l /var/log/sssd/
```

```
total 620
-rw-------. 1 root root      0 Mar 29 09:21 krb5_child.log
-rw-------. 1 root root  14324 Mar 29 09:50 ldap_child.log
-rw-------. 1 root root 212870 Mar 29 09:50 sssd_example.com.log
-rw-------. 1 root root      0 Mar 29 09:21 sssd_ifp.log
-rw-------. 1 root root      0 Mar 29 09:21 sssd_implicit_files.log
-rw-------. 1 root root      0 Mar 29 09:21 sssd.log
-rw-------. 1 root root 219873 Mar 29 10:03 sssd_nss.log
-rw-------. 1 root root      0 Mar 29 09:21 sssd_pac.log
-rw-------. 1 root root  13105 Mar 29 09:21 sssd_pam.log
-rw-------. 1 root root   9390 Mar 29 09:21 sssd_ssh.log
-rw-------. 1 root root      0 Mar 29 09:21 sssd_sudo.log
```

13.5.1. SSSD log file purposes

krb5_child.log

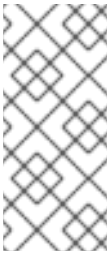
Log file for the short-lived helper process involved in Kerberos authentication.

ldap_child.log

Log file for the short-lived helper process involved in getting a Kerberos ticket for the communication with the LDAP server.

sssd_<example.com>.log

For each domain section in the **sssd.conf** file, the SSSD service logs information about communication with the LDAP server to a separate log file. For example, in an environment with an IdM domain named **example.com**, the SSSD service logs its information in a file named **sssd_example.com.log**. If a host is directly integrated with an AD domain named **ad.example.com**, information is logged to a file named **sssd_ad.example.com.log**.



NOTE

If you have an IdM environment and a cross-forest trust with an AD domain, information about the AD domain is still logged to the log file for the IdM domain.

Similarly, if a host is directly integrated to an AD domain, information about any child domains is written in the log file for the primary domain.

selinux_child.log

Log file for the short-lived helper process that retrieves and sets SELinux information.

sssd.log

Log file for SSSD monitoring and communicating with its responder and backend processes.

sssd_ifp.log

Log file for the InfoPipe responder, which provides a public D-Bus interface accessible over the system bus.

sssd_nss.log

Log file for the Name Services Switch (NSS) responder that retrieves user and group information.

sssd_pac.log

Log file for the Microsoft Privilege Attribute Certificate (PAC) responder, which collects the PAC from AD Kerberos tickets and derives information about AD users from the PAC, which avoids requesting it directly from AD.

sssd_pam.log

Log file for the Pluggable Authentication Module (PAM) responder.

sssd_ssh.log

Log file for the SSH responder process.

13.5.2. SSSD logging levels

Setting a debug level also enables all debug levels below it. For example, setting the debug level at 6 also enables debug levels 0 through 5.

Table 13.1. SSSD logging levels

Level	Description
0	**Fatal failures**. Errors that prevent the SSSD service from starting up or cause it to terminate.
1	**Critical failures**. Errors that do not terminate the SSSD service, but at least one major feature is not working properly.
2	**Serious failures**. Errors announcing that a particular request or operation has failed. **This is the default debug log level.**
3	**Minor failures**. Errors that cause the operation failures captured at level 2.
4	**Configuration** settings.
5	**Function** data.
6	Trace messages for **operation** functions.
7	Trace messages for **internal control** functions.
8	Contents of **function-internal** variables.
9	Extremely **low-level tracing** information.

13.6. ENABLING DETAILED LOGGING FOR SSSD IN THE SSSD.CONF FILE

By default, the SSSD service only logs serious failures (debug level 2), but it does not log at the level of detail necessary to troubleshoot authentication issues.

To enable detailed logging persistently across SSSD service restarts, add the option **debug_level=*<integer>*** in each section of the **/etc/sssd/sssd.conf** configuration file, where the ***<integer>*** value is a number between 0 and 9. Debug levels up to 3 log larger failures, and levels 8 and higher provide a large number of detailed log messages. **Level 6** is a good starting point for debugging authentication issues.

Prerequisites

- You need the root password to edit the **sssd.conf** configuration file and restart the SSSD service.

Procedure

1. Open the **/etc/sssd/sssd.conf** file in a text editor.
2. Add the **debug_level** option to every section of the file, and set the debug level to the verbosity of your choice.

```
[domain/example.com]
debug_level = 6
id_provider = ipa
...

[sssd]
debug_level = 6
services = nss, pam, ifp, ssh, sudo
domains = example.com

[nss]
debug_level = 6

[pam]
debug_level = 6

[sudo]
debug_level = 6

[ssh]
debug_level = 6

[pac]
debug_level = 6

[ifp]
debug_level = 6
```

3. Save and close the **sssd.conf** file.
4. Restart the SSSD service to load the new configuration settings.

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

Additional resources

- SSSD log files and logging levels

13.7. ENABLING DETAILED LOGGING FOR SSSD WITH THE SSSCTL COMMAND

By default, the SSSD service only logs serious failures (debug level 2), but it does not log at the level of detail necessary to troubleshoot authentication issues.

You can change the debug level of the SSSD service on the command line with the **sssctl debug-level *<integer>*** command, where the ***<integer>*** value is a number between 0 and 9. Debug levels up to 3 log larger failures, and levels 8 and higher provide a large number of detailed log messages. Level 6 is a good starting point for debugging authentication issues.

Prerequisites

- You need the root password to run the **sssctl** command.

Procedure

- Use the sssctl debug-level command to set the debug level of your choiceto your desired verbosity.

```
[root@server ~]# sssctl debug-level 6
```

Additional resources

- SSSD log files and logging levels

13.8. GATHERING DEBUGGING LOGS FROM THE SSSD SERVICE TO TROUBLESHOOT AUTHENTICATION ISSUES WITH AN IDM SERVER

If you experience issues when attempting to authenticate as an IdM user to an IdM server, enable detailed debug logging in the SSSD service on the server and gather logs of an attempt to retrieve information about the user.

Prerequisites

- You need the root password to run the **sssctl** command and restart the SSSD service.

Procedure

1. Enable detailed SSSD debug logging on the IdM server.

```
[root@server ~]# sssctl debug-level 6
```

2. Invalidate objects in the SSSD cache for the user that is experiencing authentication issues, so you do not bypass the LDAP server and retrieve information SSSD has already cached.

```
[root@server ~]# sssctl cache-expire -u idmuser
```

3. Minimize the troubleshooting dataset by removing older SSSD logs.

```
[root@server ~]# sssctl logs-remove
```

4. Attempt to switch to the user experiencing authentication problems, while gathering timestamps before and after the attempt. These timestamps further narrow the scope of the dataset.

```
[root@server sssd]# date; su idmuser; date
Mon Mar 29 15:33:48 EDT 2021
su: user idmuser does not exist
Mon Mar 29 15:33:49 EDT 2021
```

5. Optional: Lower the debug level if you do not wish to continue gathering detailed SSSD logs.

```
[root@server ~]# sssctl debug-level 2
```

6. Review SSSD logs for information about the failed request. For example, reviewing the **/var/log/sssd/sssd_example.com.log** file shows that the SSSD service did not find the user in the **cn=accounts,dc=example,dc=com** LDAP subtree. This might indicate that the user does not exist, or exists in another location.

```
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [dp_get_account_info_send] (0x0200):
Got request for [0x1][BE_REQ_USER][name=idmuser@example.com]
...
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [sdap_get_generic_ext_step] (0x0400):
calling ldap_search_ext with [(&(uid=idmuser)(objectclass=posixAccount)(uid=)(&
(uidNumber=)(!(uidNumber=0))))][cn=accounts,dc=example,dc=com].
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [sdap_get_generic_op_finished]
(0x0400): Search result: Success(0), no errmsg set
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [sdap_search_user_process] (0x0400):
Search for users, returned 0 results.
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [sysdb_search_by_name] (0x0400):
No such entry
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [sysdb_delete_user] (0x0400): Error: 2
(No such file or directory)
(Mon Mar 29 15:33:48 2021) [sssd[be[example.com]]] [sysdb_search_by_name] (0x0400):
No such entry
(Mon Mar 29 15:33:49 2021) [sssd[be[example.com]]]
[ipa_id_get_account_info_orig_done] (0x0080): Object not found, ending request
```

7. If you are unable to determine the cause of the authentication issue:

   a. Collect the SSSD logs you recently generated.

   ```
   [root@server ~]# sssctl logs-fetch sssd-logs-Mar29.tar
   ```

   b. Open a Red Hat Technical Support case and provide:

      i. The SSSD logs: **sssd-logs-Mar29.tar**

      ii. The console output, including the time stamps and user name, of the request that corresponds to the logs:

      ```
      [root@server sssd]# date; id idmuser; date
      Mon Mar 29 15:33:48 EDT 2021
      id: ‘idmuser’: no such user
      Mon Mar 29 15:33:49 EDT 2021
      ```

13.9. GATHERING DEBUGGING LOGS FROM THE SSSD SERVICE TO TROUBLESHOOT AUTHENTICATION ISSUES WITH AN IDM CLIENT

If you experience issues when attempting to authenticate as an IdM user to an IdM client, verify that you can retrieve user information about the IdM server. If you cannot retrieve the user information about an IdM server, you will not be able to retrieve it on an IdM client (which retrieves information from the IdM server).

After you have confirmed that authentication issues do not originate from the IdM server, gather SSSD debugging logs from both the IdM server and IdM client.

Prerequisites

- The user only has authentication issues on IdM clients, not IdM servers.
- You need the root password to run the **sssctl** command and restart the SSSD service.

Procedure

1. **On the client:** Open the **/etc/sssd/sssd.conf** file in a text editor.
2. **On the client:** Add the **ipa_server** option to the **[domain]** section of the file and set it to an IdM server. This avoids the IdM client autodiscovering other IdM servers, thus limiting this test to just one client and one server.

```
[domain/example.com]
ipa_server = server.example.com
...
```

3. **On the client:** Save and close the **sssd.conf** file.
4. **On the client:** Restart the SSSD service to load the configuration changes.

```
[root@client ~]# systemctl restart sssd
```

5. **On the server and client:** Enable detailed SSSD debug logging.

```
[root@server ~]# sssctl debug-level 6
```

```
[root@client ~]# sssctl debug-level 6
```

6. **On the server and client:** Invalidate objects in the SSSD cache for the user experiencing authentication issues, so you do not bypass the LDAP database and retrieve information SSSD has already cached.

```
[root@server ~]# sssctl cache-expire -u idmuser
```

```
[root@client ~]# sssctl cache-expire -u idmuser
```

7. **On the server and client:** Minimize the troubleshooting dataset by removing older SSSD logs.

```
[root@server ~]# sssctl logs-remove
```

```
[root@server ~]# sssctl logs-remove
```

8. **On the client:** Attempt to switch to the user experiencing authentication problems while gathering timestamps before and after the attempt. These timestamps further narrow the scope of the dataset.

```
[root@client sssd]# date; su idmuser; date
Mon Mar 29 16:20:13 EDT 2021
su: user idmuser does not exist
Mon Mar 29 16:20:14 EDT 2021
```

9. Optional: **On the server and client:** Lower the debug level if you do not wish to continue gathering detailed SSSD logs.

```
[root@server ~]# sssctl debug-level 0
```

```
[root@client ~]# sssctl debug-level 0
```

10. **On the server and client:** Review SSSD logs for information about the failed request.
    a. Review the request from the client in the client logs.
    b. Review the request from the client in the server logs.
    c. Review the result of the request in the server logs.
    d. Review the outcome of the client receiving the results of the request from the server.
11. If you are unable to determine the cause of the authentication issue:
    a. Collect the SSSD logs you recently generated on the IdM server and IdM client. Label them according to their hostname or role.

```
[root@server ~]# sssctl logs-fetch sssd-logs-server-Mar29.tar
```

```
[root@client ~]# sssctl logs-fetch sssd-logs-client-Mar29.tar
```

    b. Open a Red Hat Technical Support case and provide:
        i. The SSSD debug logs:
            A. **sssd-logs-server-Mar29.tar** from the server
            B. **sssd-logs-client-Mar29.tar** from the client
        ii. The console output, including the time stamps and user name, of the request that corresponds to the logs:

```
[root@client sssd]# date; su idmuser; date
Mon Mar 29 16:20:13 EDT 2021
su: user idmuser does not exist
Mon Mar 29 16:20:14 EDT 2021
```

13.10. TRACKING CLIENT REQUESTS IN THE SSSD BACKEND

SSSD processes requests asynchronously and as messages from different requests are added to the

same log file, you can use the unique request identifier and client ID to track client requests in the back-end logs. The unique request identifier is added to the debug logs in the form of **RID#<integer>** and the client ID in the form **[CID #<integer]**. This allows you to isolate logs pertaining to an individual request, and you can track requests from start to finish across log files from multiple SSSD components.

Prerequisites

- You have enabled debug logging and a request has been submitted from an IdM client.
- You must have root privileges to display the contents of the SSSD log files.

Procedure

1. To review your SSSD log file, open the log file using the **less** utility. For example, to view the **/var/log/sssd/sssd_example.com.log**:

```
[root@server ~]# less /var/log/sssd/sssd_example.com.log
```

2. Review the SSSD logs for information about the client request.

```
(2021-07-26 18:26:37): [be[testidm.com]] [dp_req_destructor] (0x0400): [RID#3] Number of
active DP request: 0
(2021-07-26 18:26:37): [be[testidm.com]] [dp_req_reply_std] (0x1000): [RID#3] DP Request
AccountDomain #3: Returning [Internal Error]: 3,1432158301,GetAccountDomain() not
supported
(2021-07-26 18:26:37): [be[testidm.com]] [dp_attach_req] (0x0400): [RID#4] DP Request
Account #4: REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-07-26 18:26:37): [be[testidm.com]] [dp_attach_req] (0x0400): [RID#4] Number of
active DP request: 1
```

This sample output from an SSSD log file shows the unique identifiers **RID#3** and **RID#4** for two different requests.

However, a single client request to the SSSD client interface often triggers multiple requests in the backend and as a result it is not a 1-to-1 correlation between client request and requests in the backend. Though the multiple requests in the backend have different RID numbers, each initial backend request includes the unique client ID so an administrator can track the multiple RID numbers to the single client request.

The following example shows one client request **[sssd.nss CID #1]** and the multiple requests generated in the backend, **[RID#5]** to **[RID#13]**:

```
(2021-10-29 13:24:16): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#5] DP Request [Account #5]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:16): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#6] DP Request [AccountDomain
#6]: REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:16): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#7] DP Request [Account #7]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:17): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#8] DP Request [Initgroups #8]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:17): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#9] DP Request [Account #9]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:17): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#10] DP Request [Account #10]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:17): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#11] DP Request [Account #11]:
```

```
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:17): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#12] DP Request [Account #12]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
(2021-10-29 13:24:17): [be[ad.vm]] [dp_attach_req] (0x0400): [RID#13] DP Request [Account #13]:
REQ_TRACE: New request. [sssd.nss CID #1] Flags [0x0001].
```

13.11. TRACKING CLIENT REQUESTS USING THE LOG ANALYZER TOOL

The System Security Services Daemon (SSSD) includes a log parsing tool that can be used to track requests from start to finish across log files from multiple SSSD components.

13.11.1. How the log analyzer tool works

Using the log parsing tool, you can track SSSD requests from start to finish across log files from multiple SSSD components. You run the analyzer tool using the **sssctl analyze** command.

The log analyzer tool helps you to troubleshoot NSS and PAM issues in SSSD and more easily review SSSD debug logs. You can extract and print SSSD logs related only to certain client requests across SSSD processes.

SSSD tracks user and group identity information (**id**, **getent**) separately from user authentication (**su**, **ssh**) information. The client ID (CID) in the NSS responder is independent of the CID in the PAM responder and you see overlapping numbers when analyzing NSS and PAM requests. Use the **--pam** option with the **sssctl analyze** command to review PAM requests.

NOTE

Requests returned from the SSSD memory cache are not logged and cannot be tracked by the log analyzer tool.

Additional resources

- **sudo sssctl analyze request --help**
- **sudo sssctl analyze --help**
- **sssd.conf** and **sssctl** man pages on your system

13.11.2. Running the log analyzer tool

Follow this procedure to use the log analyzer tool to track client requests in SSSD.

Prerequisites

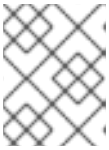
- You must set **debug_level** to at least 7 in the [$responder] section, and [domain/$domain] section of the **/etc/sssd/sssd.conf** file to enable log parsing functionality.
- Logs to analyze must be from a compatible version of SSSD built with **libtevent** chain ID support, that is SSSD in RHEL 8.5 and later.

Procedure

1. Run the log analyzer tool in **list** mode to determine the client ID of the request you are tracking, adding the **-v** option to display verbose output:

```
# sssctl analyze request list -v
```

A verbose list of recent client requests made to SSSD is displayed.



NOTE

If analyzing PAM requests, run the **sssctl analyze request list** command with the **--pam** option.

2. Run the log analyzer tool with the **show [unique client ID]** option to display logs pertaining to the specified client ID number:

```
# sssctl analyze request show 20
```

3. If required, you can run the log analyzer tool against log files, for example:

```
# sssctl analyze request --logdir=/tmp/var/log/sssd
```

Additional resources

- **sssctl analyze request list --help**
- **sssctl analyze request show --help**
- **sssctl** man page on your system

13.12. ADDITIONAL RESOURCES

- General SSSD Debugging Procedures

CHAPTER 14. CONFIGURING APPLICATIONS FOR A SINGLE SIGN-ON

Single sign-on (SSO) is an authentication scheme which allows you to log into multiple systems through a single log-in procedure. You can configure browsers and email clients to use Kerberos tickets, SSL certifications, or tokens as a means of authenticating users.

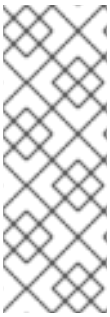
The configuration of different applications may vary. This chapter shows how to configure SSO authentication schema for the Mozilla Thunderbird email client and Mozilla Firefox web browser as the examples.

14.1. PREREQUISITES

- You have installed the following applications:
  - Mozilla Firefox *version 88*
  - Mozilla Thunderbird *version 78*

14.2. CONFIGURING FIREFOX TO USE KERBEROS FOR SINGLE SIGN-ON

You can configure Firefox to use Kerberos for single sign-on (SSO) to intranet sites and other protected websites. To do so, you first have to configure Firefox to send Kerberos credentials to the appropriate Key Distribution Center (KDC).



NOTE

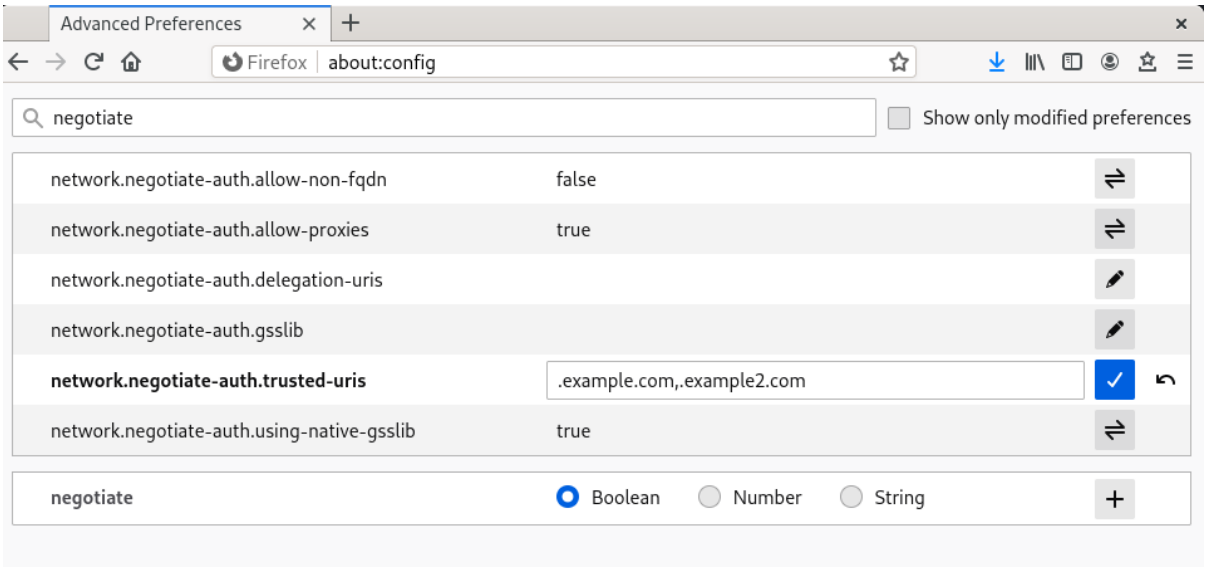
Even after Firefox is configured to pass Kerberos credentials, it still requires a valid Kerberos ticket to use. To generate a Kerberos ticket, use the **kinit** command and supply the user password for the user on the KDC.

```
[jsmith@host ~] $ kinit
Password for jsmith@EXAMPLE.COM:
```

Procedure

1. In the address bar of Firefox, type **about:config** to display the list of current configuration options.
2. In the **Filter** field, type **negotiate** to restrict the list of options.
3. Double-click the **network.negotiate-auth.trusted-uris** entry.
4. Enter the name of the domain against which to authenticate, including the preceding period (.). If you want to add multiple domains, enter them in a comma separated list.

Figure 14.1. Manual Firefox Configuration



Additional resources

- For information about configuring Firefox to use Kerberos in Identity Management, see the corresponding section in the Linux Domain Identity, Authentication, and Policy Guide.

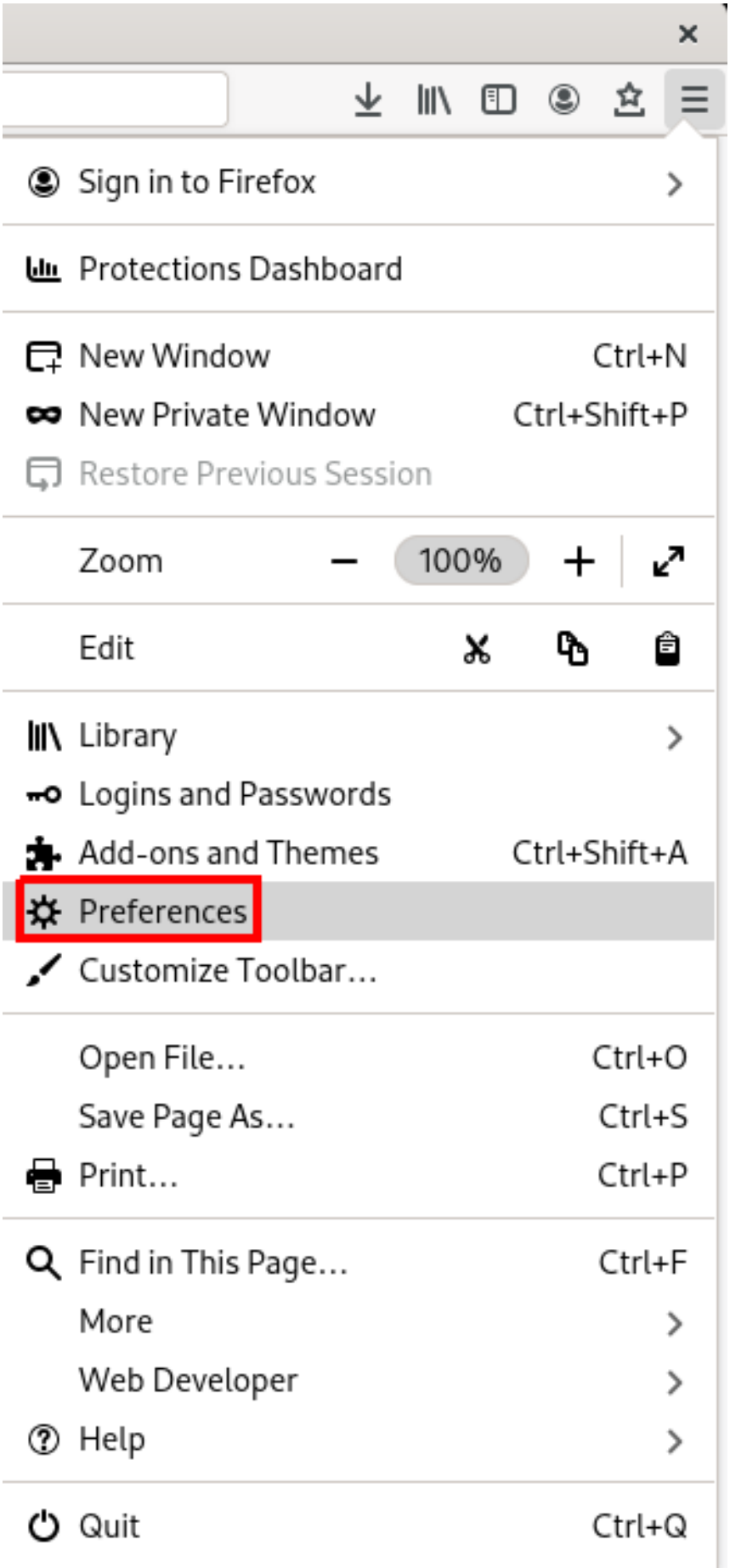
14.3. VIEWING CERTIFICATES IN FIREFOX

The following example shows how to view certificates in the Mozilla Firefox.

To view certificates in Firefox, you need to open the **Certificate Manager**.

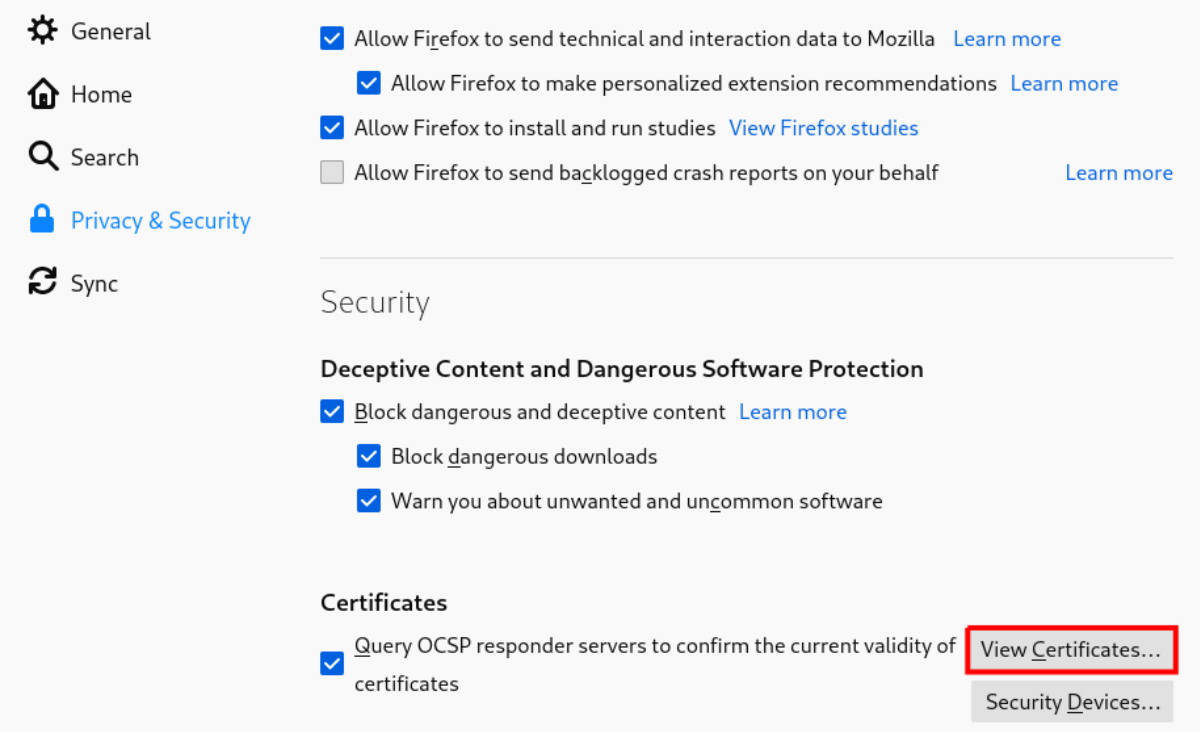
Procedure

1. In Mozilla Firefox, open the Firefox menu and select **Preferences**.



2. In the left panel, select the **Privacy & Security** section.

3. Scroll down to the **Certificates** section.
4. Click **View Certificates** to open the **Certificate Manager**.



14.4. IMPORTING CA CERTIFICATES IN FIREFOX

The following example shows how to import certificates in the Mozilla Firefox.

Prerequisites

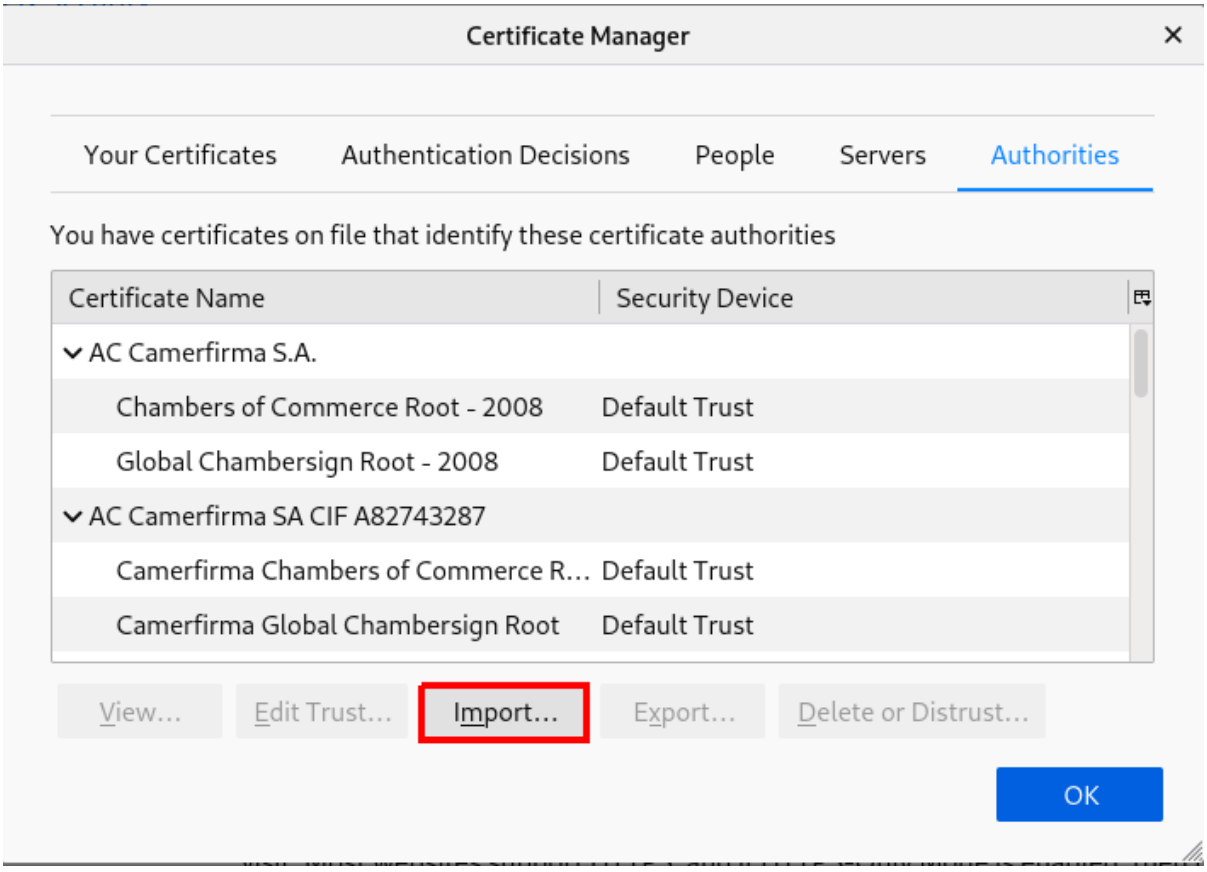
- You have a CA certificate on your device.

To import a CA certificate:

Procedure

1. Open **Certificate Manager**.
2. Select the **Authorities** tab and click **Import**.

Figure 14.2. Importing the CA Certificate in Firefox



3. Select the downloaded CA certificate from your device.

14.5. EDITING CERTIFICATE TRUST SETTINGS IN FIREFOX

The following example shows how to edit certificate settings in the Mozilla Firefox.

Prerequisites

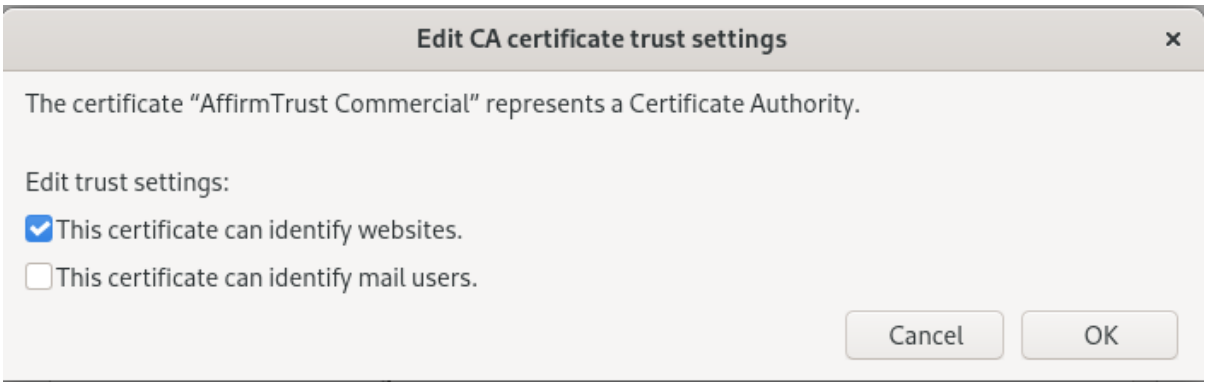
1. You have successfully imported a certificate.

To set the certificate trust settings:

Procedure

1. Open **Certificate Manager**.
2. Under the **Authorities** tab, select the appropriate certificate and click **Edit Trust**.
3. Edit the certificate trust settings.

Figure 14.3. Editing the Certificate Trust Settings in Firefox



14.6. IMPORTING PERSONAL CERTIFICATE FOR AUTHENTICATION IN FIREFOX

The following example shows how to import personal certificates for authentication in the Mozilla Firefox.

Prerequisites

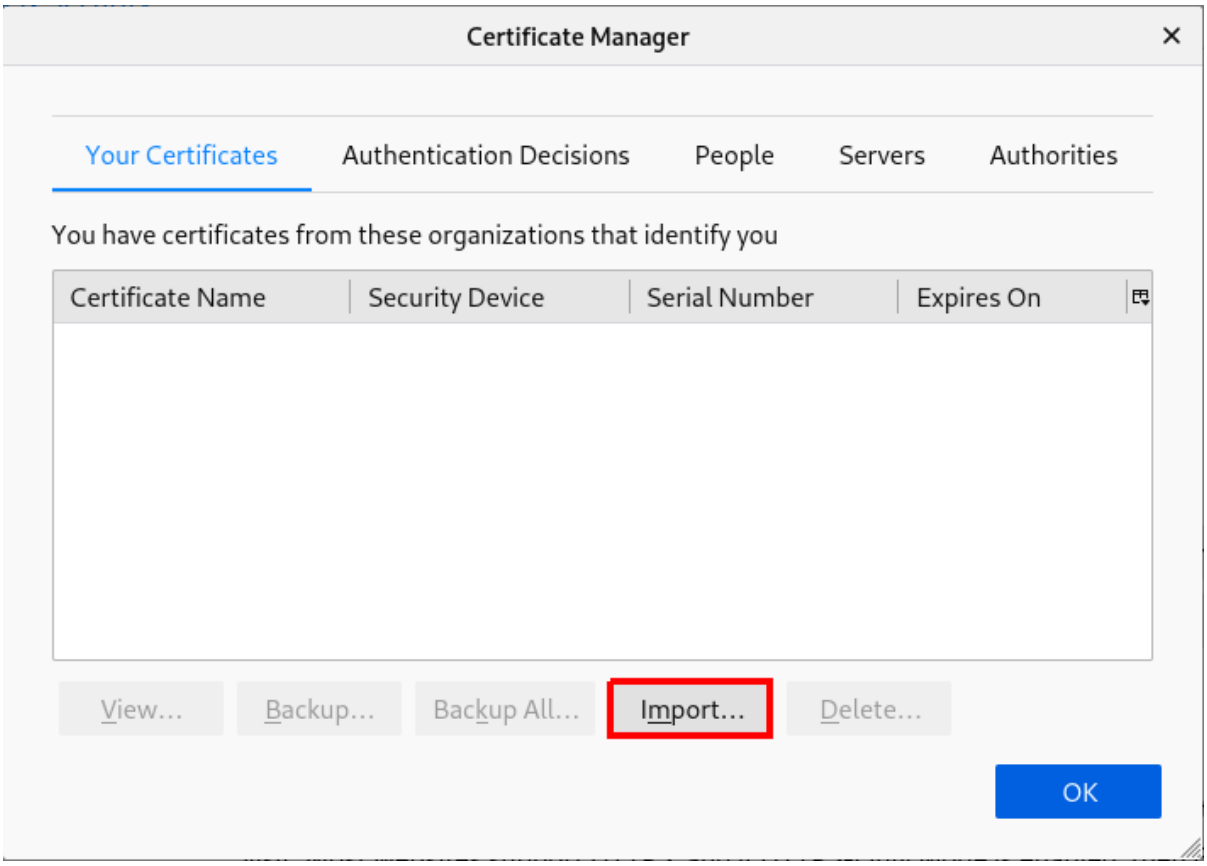
1. You have a personal certificate stored on your device.

To use a personal certificate for authentication:

Procedure

1. Open **Certificate Manager**.
2. Select the **Your Certificates** tab and click **Import**.

Figure 14.4. Importing a Personal Certificate for Authentication in Firefox



3. Select the appropriate certificate from your computer.

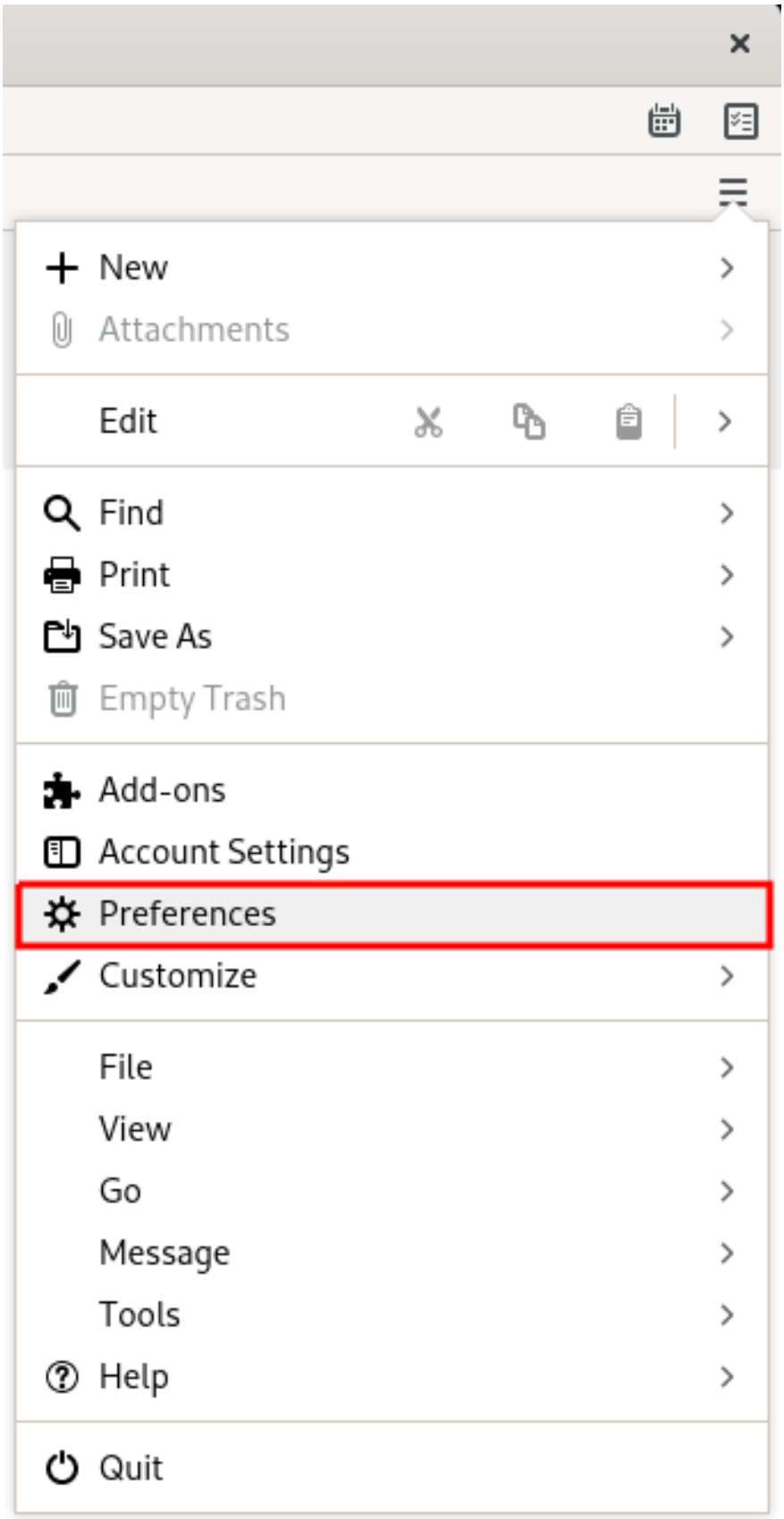
14.7. VIEWING CERTIFICATES IN THUNDERBIRD

The following example shows how to view certificates in the Mozilla Thunderbird email client.

Procedure

1. In Mozilla Thunderbird, open the main menu and select **Preferences**.

Figure 14.5. Selecting preferences from menu

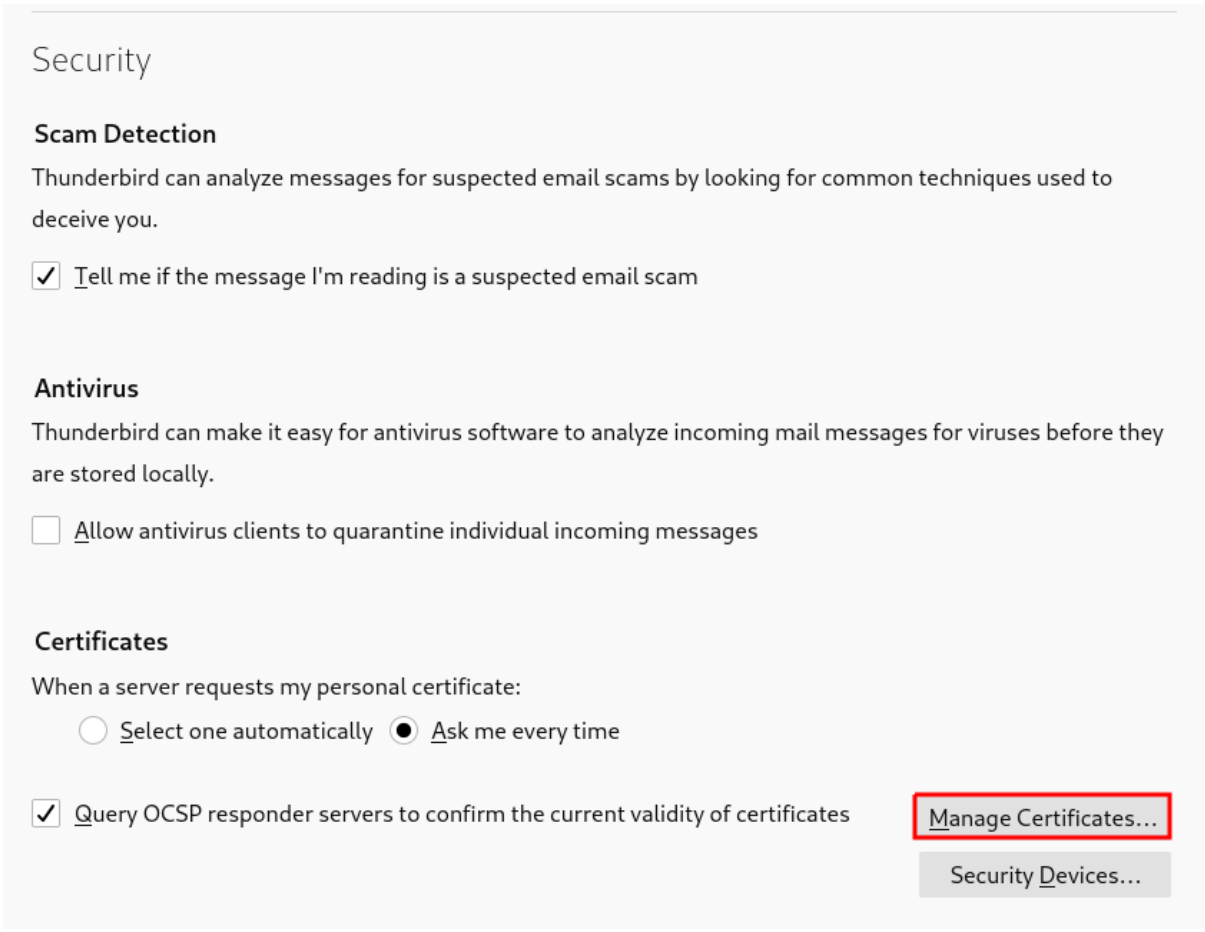


2. In the left panel, select the **Privacy & Security** section.

Figure 14.6. Selecting security section

3. Scroll down to the **Certificates** section.
4. Click **Manage Certificates** to open the **Certificate Manager**.

Figure 14.7. Opening certificate manager



14.8. IMPORTING CERTIFICATES IN THUNDERBIRD

The following example shows how to import certificates in the Mozilla Thunderbird email client.

Prerequisites

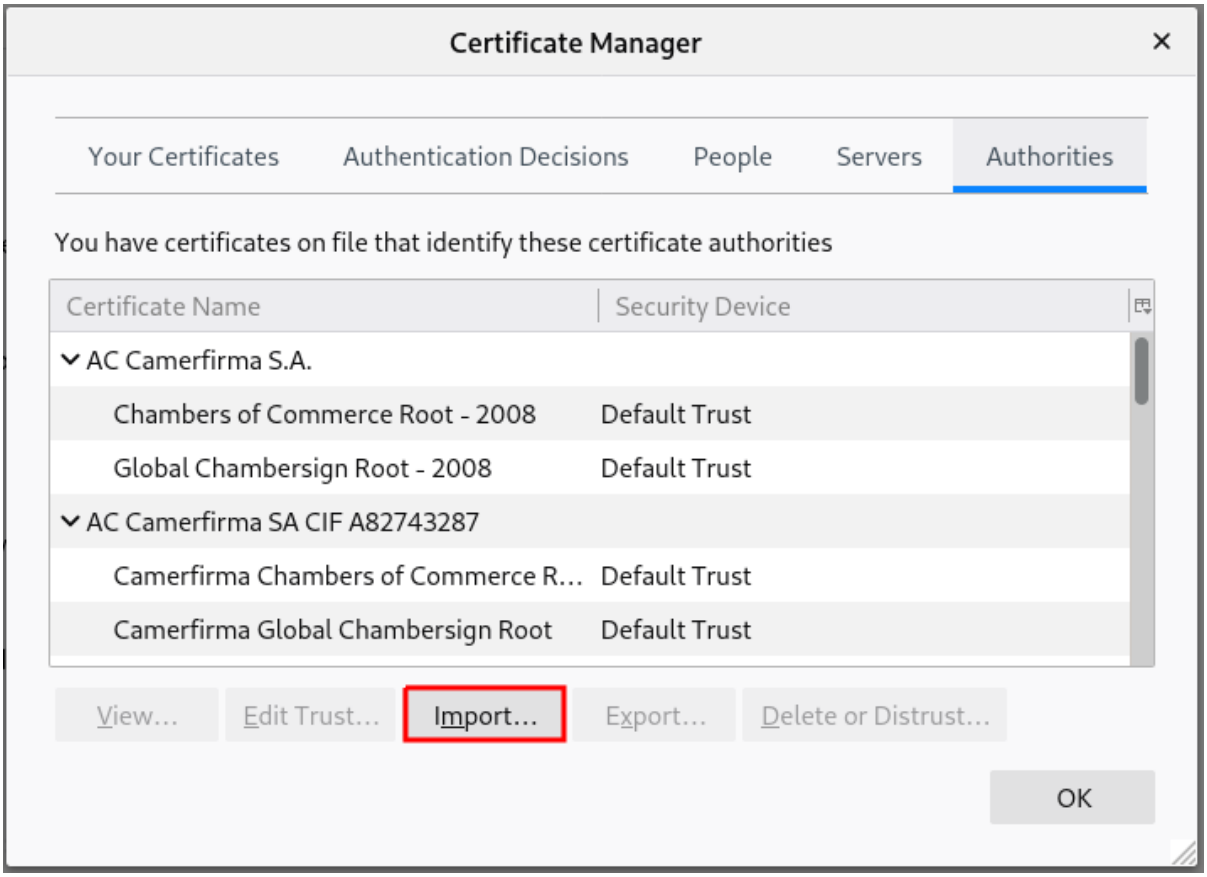
- You have a CA certificate stored on your device.

To import a CA certificate:

Procedure

1. Open **Certificate Manager**.
2. Select the **Authorities** tab and click **Import**.

   Figure 14.8. Importing the CA certificate in Thunderbird



3. Select the downloaded CA certificate.

14.9. EDITING CERTIFICATE TRUST SETTINGS IN THUNDERBIRD

The following example shows how to edit certificate settings in the Mozilla Thunderbird email client.

Prerequisites

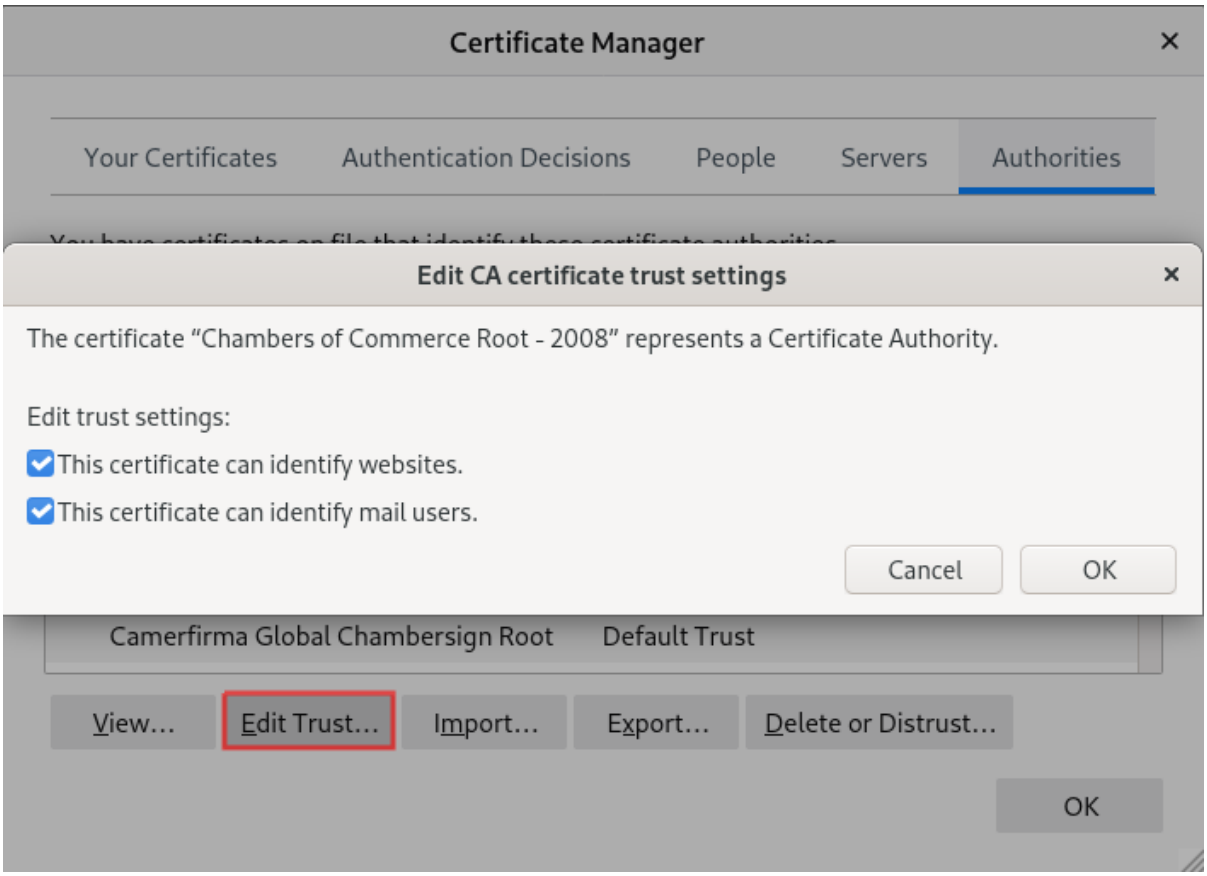
- You have successfully imported a certificate.

To set the certificate trust relationships:

Procedure

1. Open **Certificate Manager**.
2. Under the **Authorities** tab, select the appropriate certificate and click **Edit Trust**.
3. Edit the certificate trust settings.

Figure 14.9. Editing the certificate trust settings in Thunderbird



14.10. IMPORTING PERSONAL CERTIFICATE IN THUNDERBIRD

The following example shows how to import certificates for personal authentication in the Mozilla Thunderbird email client.

Prerequisites

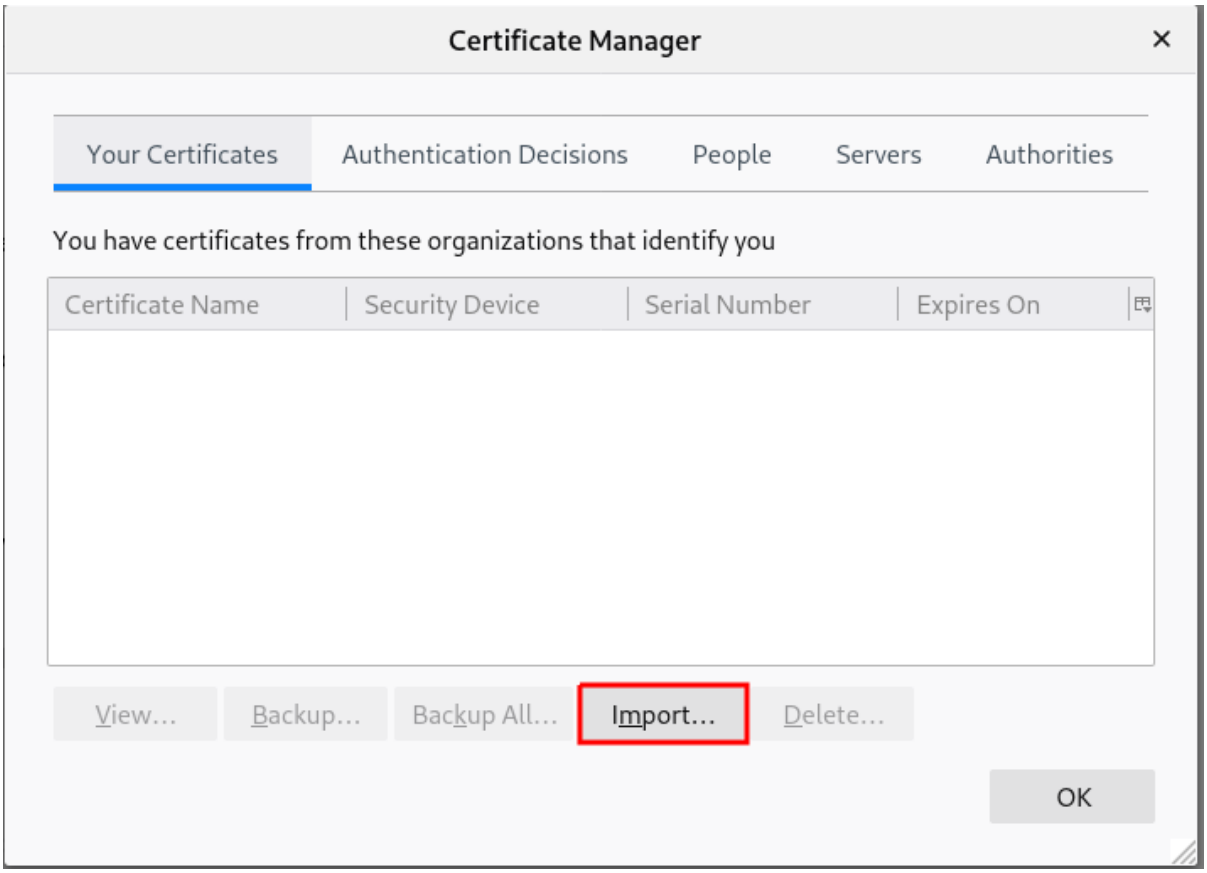
1. You have a personal certificate stored on your device.

To use a personal certificate for authentication:

Procedure

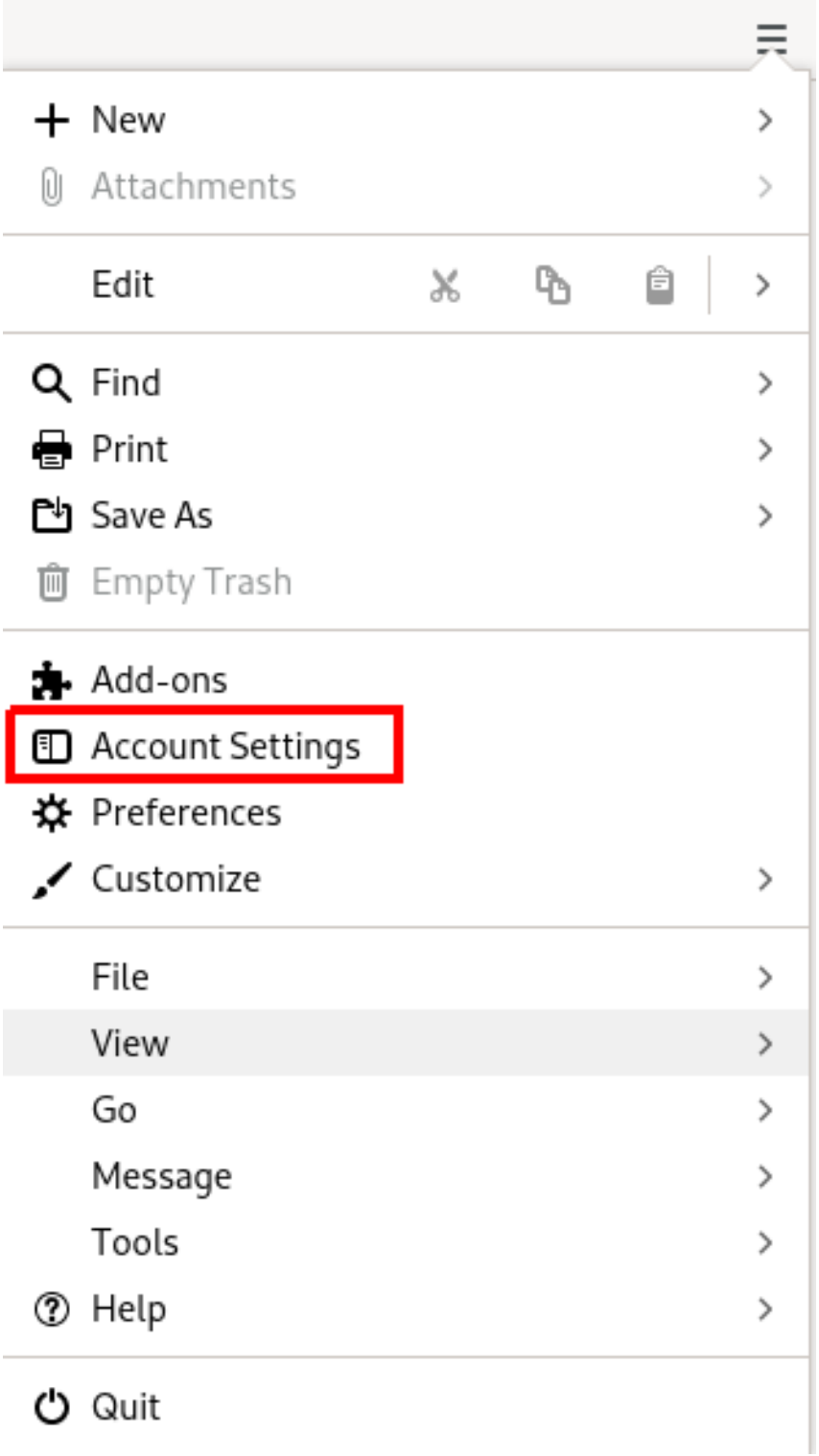
1. Open **Certificate Manager**.
2. Under the **Your Certificates** tab, click **Import**.

Figure 14.10. Importing a personal certificate for authentication in Thunderbird



3. Select the required certificate from your computer.
4. Close the **Certificate Manager**.
5. Open the main menu and select **Account Settings**.

Figure 14.11. Selecting account settings from menu



6. Select **End-To-End Encryption** in the left panel under your account email address.
Selecting end-to-end encryption section.

7. Under **S/MIME** section click the first **Select** button to choose your personal certificate to use for signing messages.

8. Under **S/MIME** section click the second **Select** button to choose your personal certificate to encrypt and decrypt messages.
Choosing certificate for signing and encryption/decryption.

NOTE

In case you forgot to import valid certificate, you can open **Certificate Manager** directly using the **Manage S/MIME certificates**.