Red Hat Enterprise Linux 8

Using authselect on a Red Hat Enterprise Linux host

Understanding, selecting, modifying, and creating authselect profiles

Last Updated: 2020-04-28

Red Hat Enterprise Linux 8 Using authselect on a Red Hat Enterprise Linux host

Understanding, selecting, modifying, and creating authselect profiles

Legal Notice

Abstract

This documentation collection provides instructions on how to use authselect on a Red Hat Enterprise Linux 8 host.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. CONFIGURING USER AUTHENTICATION USING AUTHSELECT

1.1. WHAT IS AUTHSELECT USED FOR

Authselect is a utility that simplifies the configuration of user authentication on a Red Hat Enterprise Linux host. **Authselect** offers two ready-made profiles that can be universally used with all modern identity management systems:

- the **sssd** profile
- the **winbind** profile

For legacy compatibility reasons, the **nis** profile is also available.

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

WARNING

Do not use **authselect** if your host is part of Red Hat Enterprise Linux Identity Management or Active Directory. The **ipa-client-install** command, called when joining your host to a Red Hat Identity Management domain, takes full care of configuring authentication on your host. Similarly the **realm join** command, called when joining your host to an Active Directory domain, takes full care of configuring authentication on your host.

The **authconfig** utility, used in previous Red Hat Enterprise Linux versions, created and modified many different configuration files, making troubleshooting a difficult task. **Authselect** makes testing and troubleshooting easy because it only modifies files in these directories:

- **/etc/nsswitch.conf**
- **/etc/pam.d/*** files
- **/etc/dconf/db/distro.d/*** files

The Name Service Switch (NSS) configuration file, **/etc/nsswitch.conf**, is used by the GNU C Library and certain other applications 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.

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. This dynamic configuration is set by the contents of the configuration files in the **/etc/pam.d/** directory, which list the PAMs that will do the authentication tasks required by this service, and the appropriate behavior of the PAM-API in the event that individual PAMs fail.

Once an **authselect** profile is selected for a given host, the profile will be applied to every user logging into the host.

1.2. 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.

Procedure

1. 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**. Run the command as root:

   ```
   # authselect select sssd
   ```

2. Optionally, review the contents of the **/etc/nsswitch.conf** file:

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

   The content of the **/etc/nsswitch.conf** file shows that selecting the **sssd** profile means that the system first uses **sssd** if information concerning one of the first five items is requested. 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 does not normally get consulted at all. The only exception is the case of the **root** user, which is never handled by **sssd** but by **files**.

3. Optionally, review the contents of the **/etc/pam.d/system-auth** file:

   ```
   # 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
   ...
   ```

–

Among other things, the **/etc/pam.d/system-auth** file contains information about:

- user password lockout condition
- the possibility to authenticate with a smart card
- the possibility to authenticate with fingerprints
  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 Section 1.5, "Converting your scripts from authconfig to authselect" or the authselect-migration(7) man page.

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.

If adjusting a ready-made profile by adding one of the **authselect select** command-line options described above is not enough for your use case, you can:

- modify a ready-made profile by changing the **/etc/authselect/user-nsswitch.conf** file. For details, see Section 1.3, "Modifying a ready-made authselect profile" .
- create your own custom profile. For details, see Section 1.4, "Creating and deploying your own authselect profile".

1.3. MODIFYING A READY-MADE AUTHSELECT PROFILE

As a system administrator, you can modify one of the default profiles, the **sssd**, **winbind**, or the **nis** profile, 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 permissible changes to the profile being transferred from **/etc/authselect/user-nsswitch.conf** to the **/etc/nsswitch.conf** file but unacceptable changes being 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.

3. Apply the changes from the **/etc/authselect/user-nsswitch.conf** file:

   ```
   # authselect apply-changes
   ```

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

1.4. CREATING AND DEPLOYING YOUR OWN AUTHSELECT PROFILE

As a system administrator, you can create and deploy a custom profile by customizing one of the default profiles, the **sssd**, **winbind**, or the **nis** profile. This is particularly useful if Section 1.3, "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
   ```

   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 has created 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

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. Optionally, 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
```

1.5. 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)

Table 1.1, "Relation of authconfig options to authselect profiles" and Table 1.2, "Authselect profile option equivalents of authconfig options" show the **authselect** equivalents of **authconfig** options.

Table 1.1. Relation of authconfig options to authselect profiles

Authconfig options	Authselect profile
--enableldap --enableldapauth	**sssd**

--enablesssd --enablesssdauth	**sssd**
--enablekrb5	**sssd**
--enablewinbind --enablewinbindauth	**winbind**
--enablenis	**nis**

Table 1.2. 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

Table 1.3, "Examples of authselect commands equivalents to authconfig commands" shows example transformations of Kickstart calls to **authconfig** into Kickstart calls to **authselect**.

Table 1.3. Examples of authselect commands 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 --enableecryptfs --enablepamaccess --updateall	authselect select sssd with-ecryptfs with-pamaccess
authconfig --enablewinbind --enablewinbindauth --winbindjoin=Administrator --updateall	realm join -U Administrator --client-software=winbind **WINBINDDOMAIN**

CHAPTER 2. UNDERSTANDING SSSD AND ITS BENEFITS

2.1. 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

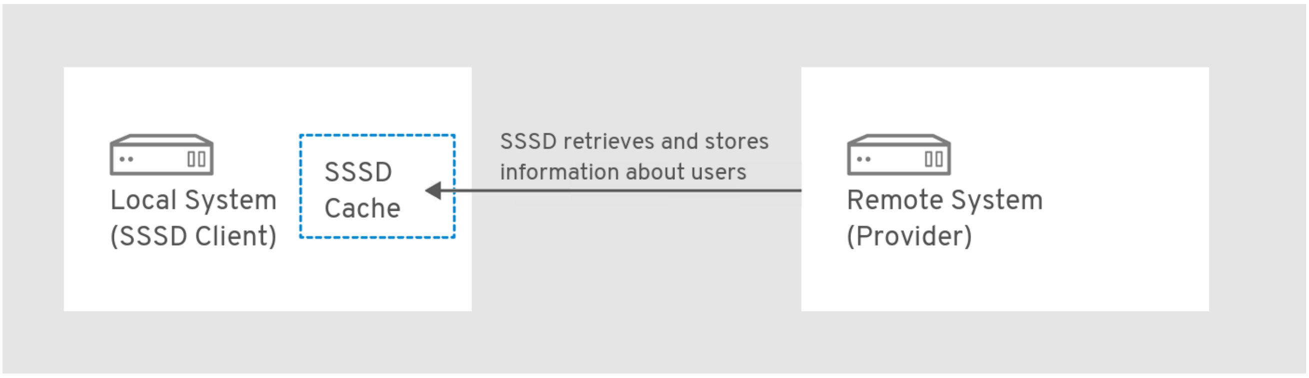
SSSD works in two stages:

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 2.1. How SSSD works



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

2.2. 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.

2.3. 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.

2.4. IDENTITY AND AUTHENTICATION PROVIDERS FOR SSSD

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 in order 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

Available Combinations of Identity and Authentication Providers

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

Table 2.1. Available Combinations of Identity and Authentication Providers

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

Identity Provider	Authentication Provider
LDAP	Kerberos
Proxy	Proxy
Proxy	LDAP
Proxy	Kerberos

[a] An extension of the LDAP provider type.

Additional resources

- You can configure SSSD using the **authselect** utility. For more details about using **authselect**, see Chapter 1, *Configuring user authentication using authselect*.
- If your host is enrolled in Identity Management (IdM) that is in a trust agreement with an Active Directory (AD) forest, you can list and verify the status of the domains using the **sssctl** utility. For more details, see Chapter 8, *Querying domain information using SSSD*.
- You can use the **sssctl** utility to create access control reports and display user data. For more details, see Chapter 7, *Reporting on user access on hosts using SSSD*.

CHAPTER 3. CONFIGURING SSSD TO USE LDAP AND REQUIRE TLS AUTHENTICATION

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

The System Security Services Daemon (SSSD) is a daemon that manages identity data retrieval and authentication on a RHEL 8 host. A system administrator can configure the SSSD on the host to use a standalone LDAP server database as the user account database. Examples of an LDAP server include the OpenLDAP server and the Red Hat 389 Directory Server. In this chapter, the scenario also includes the requirement that the connection with the LDAP server must be encrypted with a TLS certificate.

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 in this chapter.

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.

3.2. 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 and to specify the following client configuration:

- The RHEL system uses an OpenLDAP server as the user account database.
- The RHEL system uses the System Security Services Daemon (SSSD) as the service responsible for retrieving the user data.
- The RHEL system uses a TLS certificate to encrypt the connection with the OpenLDAP server.

NOTE

You can alternatively use the steps in this procedure to configure your RHEL system as a client of the Red Hat 389 Directory Server.

Prerequisites

- The OpenLDAP server is installed.
- On the host you want to become a client of the OpenLDAP server, you have root credentials.
- On the host you want to become a client of the OpenLDAP server, the **/etc/sssd/sssd.conf** file has been created and configured to specify **ldap** as the **autofs_provider** and the **id_provider**.
- You have the TLS certificate of the OpenLDAP server stored in a **PEM** format.

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 LDAP server certificate into the **/etc/openldap/cacerts** folder.

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 **/etc/openldap/ldap.conf**, specify the location of the OpenLDAP server certificate by adding a line pointing the **TLS_CACERT** parameter to **/etc/openldap/cacerts/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:

```
[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/default]** 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 SSSD:

   ```
   # 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 8 to communicate using these protocols:

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

    For more details, see the Deprecated Functionality section in the RHEL 8.0 Release Notes .

Verification steps

- Verify login 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 4. CONFIGURING RHEL TO USE AD AS AN AUTHENTICATION PROVIDER

4.1. A STANDALONE RHEL HOST USING 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 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 on an IdM-AD trust, see Planning a cross-forest trust between IdM and AD and Installing a trust between IdM and AD.

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

Complete this procedure to enable the user named **AD_user** to log in to the **rhel8_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 **rhel8_host**.
- The **AD_user** user account exists in the **example.com** domain.
- The Kerberos realm is **EXAMPLE.COM**.
- **rhel8_host** has not been joined to AD using the **realm join** command.

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 = 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 steps

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 **rhel8_host** using the credentials from the **EXAMPLE.COM** Kerberos domain.

CHAPTER 5. CONFIGURING SMART CARDS USING AUTHSELECT

This chapter describes how to configure your smart card to achieve one of the following aims:"

- Enable both password and smart card authentication
- Disable password and enable smart card authentication
- Enable lock on removal

Prerequisites

- Authselect installed
  The authselect tool configures user authentication on Linux hosts and you can use it to configure smart card authentication parameters. For details about authselect, see Explaining authselect.
- Smart Card or USB device supported by RHEL 8
  For details, see Smart Card support in RHEL8 .

5.1. CERTIFICATES ELIGIBLE FOR SMART CARDS

Before you can configure a smart card with **authselect**, you must import a certificate into your card. You can use the following tools to generate the certificate:

- Active Directory (AD)
- Identity Management (IdM)
  For details about how to create IdM certificates, see Requesting a new user certificate and exporting it to the client.
- Red Hat Certificate System (RHCS)
  For details, see Managing Smart Cards with the Enterprise Security Client .
- Local Certification Authority. You can use a certificate generated by the Local Certification Authority if the user is not part of a domain or for testing purposes.
  For details about how to create and import local certificates into a smart card, Configuring and importing local certificates to a smart card.

5.2. ENABLING USER PASSWORD AUTHENTICATION TO CONFIGURE SMART CARD AUTHENTICATION

This section describes how to enable both smart card and password authentication on your system.

Prerequisites

- The Smart card contains your certificate and private key.
- The card is inserted into the reader and connected to the computer.
- The **authselect** tool is installed on your system.

Procedure

- Enter the following command to allow smart card and password authentication:

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

At this point, smart card authentication is enabled, however, password authentication will work if you forget your smart card at home.

5.3. CONFIGURING AUTHSELECT TO ENFORCE SMART CARD AUTHENTICATION

The **authselect** tool enables you to configure smart card authentication on your system and to disable the default password authentication. The **authselect** command must include the following options:

- **with-smartcard** – enabling smart card authentication
- **with-smartcard-required** – enabling exclusive smart card authentication (authentication with a password is disabled)

Prerequisites

- Smart card contains your certificate and private key.
- The card is inserted into the reader and connected to the computer.
- The **authselect** tool is installed on your local system.

Procedure

- Enter the following command to enforce smart card authentication:

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

At this point, you can only log in with a smart card. Password authentication will not be working any more.

5.4. CONFIGURING SMART CARD AUTHENTICATION WITH LOCK ON REMOVAL

The **authselect** service enables you to configure your smart card authentication to lock your screen instantly after removing the smart card from the reader. The **authselect** command must include the following variables:

- **with-smartcard** – enabling smart card authentication
- **with-smartcard-required** – enabling exclusive smart card authentication (authentication with a password is disabled)
- **with-smartcard-lock-on-removal** – enforcing log out after the smart card removal

Prerequisites

- Smart card contains your certificate and private key.
- The card is inserted into the reader and connected to the computer.
- The **authselect** tool is installed on your local system.

Procedure

- Enter the following command to enable smart card authentication, disable password authentication, and enforce lock on removal:

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

Now, when you remove the card, the screen locks. You must re–insert your smart card to unlock it.

CHAPTER 6. CONFIGURING AND IMPORTING LOCAL CERTIFICATES TO A SMART CARD

This chapter describes a scenario where:

- The host is not connected to a domain.
- You want to authenticate with a smart card on this host.
- You want to configure SSH access using smart card authentication.
- You want to configure the smart card with **authselect**.

Use the following configuration to accomplish this scenario:

- Obtain a user certificate for the user who wants to authenticate with a smart card. The certificate should be generated by a trustworthy Certification Authority used in the domain.
  If you cannot get the certificate, you can generate a user certificate signed by a local certificate authority for testing purposes,
- Store the certificate and private key in a smart card.
- Configure the smart card authentication for SSH access.

IMPORTANT

If a host can be part of the domain, add the host to the domain and use certificates generated by Active Directory or Identity Management Certification Authority.

For details about how to create IdM certificates for a smart card, see Configuring Identity Management for smart card authentication.

Prerequisites

- Authselect installed
  The authselect tool configures user authentication on Linux hosts and you can use it to configure smart card authentication parameters. For details about authselect, see Explaining authselect.
- Smart Card or USB device supported by RHEL 8
  For details, see Smart Card support in RHEL8 .

6.1. CREATING LOCAL CERTIFICATES

This section describes how to perform these tasks:

- Generate the OpenSSL certificate authority
- Create a certificate signing request

WARNING

The following steps are intended for testing purpose only. Certificates generated by a local self-signed Certificate Authority are not as secure as using AD, IdM, or RHCS Certification Authority. You should use a certificate generated by your enterprise Certification Authority even if the host is not part of the domain.

Procedure

1. Create a directory where you can generate the certificate, for example:

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

2. Set up the certificate (copy this text to your command line in the **ca** directory):

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

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

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

default_days     = 365
default_crl_days = 30
default_md       = sha256

policy           = policy_any
email_in_dn      = no

name_opt         = ca_default
cert_opt         = ca_default
copy_extensions  = copy

[ usr_cert ]
authorityKeyIdentifier = keyid, issuer

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

[ policy_any ]
```

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

[ req ]
distinguished_name = req_distinguished_name
prompt             = no

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

3. Create the following directories:

```
# mkdir certs crl newcerts
```

4. Create the following files:

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

5. Write the number 01 in the serial file:

```
# echo 01 > serial
```

This command writes a number 01 in the serial file. It is a serial number of the certificate. With each new certificate released by this CA the number increases by one.

6. Create an OpenSSL root CA key:

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

7. Create a self-signed root Certification Authority certificate:

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

8. Create the key for your username:

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

This key is generated in the local system which is not secure, therefore, remove the key from the system when the key is stored in the card.

You can create a key directly in the smart card as well. For doing this, follow instructions created by the manufacturer of your smart card.

9. Create the certificate signing request configuration file (copy this text to your command line in the ca directory):

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

```
[ req ]
distinguished_name = req_distinguished_name
prompt = no

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

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

10. Create a certificate signing request for your example.user certificate:

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

11. Configure the new certificate. Expiration period is set to 1 year:

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

At this point, the certification authority and certificates are successfully generated and prepared for import into a smart card.

6.2. COPYING CERTIFICATES TO THE SSSD DIRECTORY

Gnome Desktop Manager (GDM) requires SSSD. If you use GDM, you need to copy the PEM certificate to the **/etc/sssd/pki** directory.

Prerequisites

- The local CA authority and certificates have been generated

Procedure

1. Ensure that you have SSSD installed on the system.

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

2. Create a **/etc/sssd/pki** directory:

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

3. Copy the **rootCA.crt** as a PEM file in the **/etc/sssd/pki/** directory:

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

Now you have successfully generated the certificate authority and certificates, and you have saved them in the **/etc/sssd/pki** directory.

NOTE

If you want to share the Certificate Authority certificates with another application, you can change a location in sssd.conf:

- SSSD PAM responder: **pam_cert_db_path** in the **[pam]** section
- SSSD ssh responder: **ca_db** in the **[ssh]** section

For details, see man page for **sssd.conf**.

Red Hat recommends to keep the default path and use a dedicated Certificate Authority certificate file for SSSD to make sure that only Certificate Authorities trusted for authentication are listed here.

6.3. INSTALLING TOOLS FOR MANAGING AND USING SMART CARDS

To configure your smart card, you need tools which can generate certificates and store them on a smart card.

You must:

- Install the **gnutls-utils** package which helps you to manage certificates.
- Install the **opensc** package which provides a set of libraries and utilities to work with smart cards.
- Start the **pcscd** service which communicates with the smart card reader.

Procedure

1. Install the **opensc** and **gnutls-utils** packages:

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

2. Start the **pcscd** service.

   ```
   # systemctl start pcscd
   ```

Verify that the **pcscd** service is up and running.

6.4. STORING A CERTIFICATE ON A SMART CARD

This section describes smart card configuration with the **pkcs15-init** tool, which helps you to configure:

- Erasing your smart card
- Setting new PINs and optional PIN Unblocking Keys (PUKs)
- Creating a new slot on the smart card
- Storing the certificate, private key, and public key in the slot
- Locking the smart card settings (some smart cards require this type of finalization)

Prerequisites

- The **opensc** package, which includes the **pkcs15-init** tool is installed.
  For details, see Installing tools for managing and using smart cards .
- The card is inserted in the reader and connected to the computer.
- You have the private key, public key, and certificate to store on the smart card. In this procedure, **testuser.key**, **testuserpublic.key**, and **testuser.crt** are the names used for the private key, public key, and the certificate.
- Your current smart card user PIN and Security Officer PIN (SO-PIN)

Procedure

1. Erase your smart card and authenticate yourself with your PIN:

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

   The card has been erased.

2. Initialize your smart card, set your user PIN and PUK, and your Security Officer PIN and PUK:

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

   The **pcks15-init** tool creates a new slot on the smart card.

3. Set the label and the authentication ID for the slot:

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

   The label is set to a human-readable value, in this case, **testuser**. The **auth-id** must be two hexadecimal values, in this case it is set to **01**.

4. Store and label the private key in the new slot on the smart card:

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

NOTE

The value you specify for **--id** must be the same when storing your private key, and certificate. If you do not specify a value for **--id**, a more complicated value is calculated by the tool and it is therefore easier to define your own value.

5. Store and label the certificate in the new slot on the smart card:

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

6. (Optional) Store and label the public key in the new slot on the smart card:

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

NOTE

If the public key corresponds to a private key and/or certificate, you should specify the same ID as that private key and/or certificate.

7. (Optional) Some smart cards require you to finalize the card by locking the settings:

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

At this stage, your smart card includes the certificate, private key, and public key in the newly created slot. You have also created your user PIN and PUK and the Security Officer PIN and PUK.

6.5. CONFIGURING SSH ACCESS USING SMART CARD AUTHENTICATION

SSH connections require authentication. You can use a password or a certificate. This section describes:

- the configuration necessary for enabling authentication using a certificate stored on a smart card
- the lock on removal configuration using the **authselect** tool

The lock on removal configuration enforces log out after the smart card removal.

For details about configuring smart cards with **authselect**, see Configuring smart cards using authselect.

Prerequisites

- The smart card contains your certificate and private key.

- The card is inserted in the reader and connected to the computer.
- SSSD is installed and configured.
- Your username matches the Common Name (CN) or User ID (UID) in the certificate's SUBJECT.
- The **pcscd** service is running on your local machine.
  For details, see Installing tools for managing and using smart cards .

Procedure

1. Create a new directory for SSH keys in the home directory of the user who uses smart card authentication:

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

1. Run the **ssh-keygen** command to read the existing public key for the private key stored in the smart card. The command also appends the key to the **authorized_keys** file. It enables the SSH access authenticated by smart card.

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

2. SSH requires access right configuration for the **/.ssh** directory and the **authorized_keys** file. To set or change the access rights, enter:

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

3. Optionally, display the keys:

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

   The terminal displays the keys.

4. Verify that the smart card authentication is enabled in the **/etc/sssd/sssd.conf** file:
   In the **[pam]** section, enable the pam certificate authentication module: **pam_cert_auth = True**

   If the **sssd.conf** file has not been created yet, you can create the minimal functional configuration by copying the following script to the command line:

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

   [nss]

   [pam]
   pam_cert_auth = True
   ```

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

5. To use the SSH keys, configure the authentication with the **authselect** command:

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

Now, you can verify the SSH access with the following command:

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

If the configuration is successful, you are prompted to enter the smart card PIN.

The configuration works now locally. For smart card authentication working on remote servers, you must copy the public key and distribute it to **authorized_keys** files located on all servers on which you want to use SSH.

CHAPTER 7. 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.

7.1. THE SSSCTL COMMAND

sssctl is a command-line tool using Security System Services Daemon (SSSD) 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

- For details about **sssctl**, enter:

```
# sssctl --help
```

7.2. 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** is available on RHEL 7 and RHEL 8 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
```

7.3. 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 and RHEL 8 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

- For details on **sssctl user-checks**, use the following command:

```
sssctl user-checks --help
```

CHAPTER 8. QUERYING DOMAIN INFORMATION USING SSSD

Security System Services Daemon (SSSD) can list domains in Identity Management (IdM), including Active Directory domains in the cross-forest trust. You can also verify the status of each of the listed domains:

- Listing domains using the sssctl command
- Verifying the domain status using the sssctl command

8.1. LISTING DOMAINS USING SSSCTL

The **sssctl domain-list** command helps 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** is available on RHEL 7 and RHEL 8 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.

8.2. VERIFYING THE DOMAIN STATUS USING SSSCTL

The **sssctl domain-status** command helps 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** is available on RHEL 7 and RHEL 8 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: master.idm.example.com

   Discovered IPA servers:
   - master.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
```