Oracle Linux 10

Setting Up System Users and Authentication

G14602-01
June 2025

Oracle Linux 10 Setting Up System Users and Authentication,

G14602-01

Contents

Preface

Oracle Linux 10: Setting Up System Users and Authentication provides information about how to set up user accounts and authentication mechanisms on an Oracle Linux 10 release.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and

the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About System Authentication

Authentication is a way of implementing system security by verifying the identity of any user that tries to access the system.

A user signs in to the system by providing a username and a password, and the OS authenticates that user's identity by comparing this information to data stored on the system.

If the credentials match and the user account is active, the user is authenticated and can successfully access the system.

Authentication in Oracle Linux

In Oracle Linux, authentication is profile-based. Each profile uses different mechanisms to authenticate system access.

The following profiles are installed with Oracle Linux:

- `sssd`: Uses the System Security Services Daemon (`sssd`) service to perform system authentication. The `sssd` service is a client for many centralized directory and authentication providers such as Kerberos, Active Directory, FreeIPA, and LDAP.
- `winbind`: Uses the `winbind` service to perform system authentication. The `winbind` service is a client-side service that resolves user and group information on a Windows server, and lets Oracle Linux understand Windows users and groups.
- `local`: Uses system files to perform system authentication for local users. This is the default authentication profile in Oracle Linux 10.

You can adapt these existing profiles to suit the authentication needs of the organization. For example, you can configure the `sssd` profile to use different backend directory services.

You can also use profiles supplied by external vendors, or create custom profiles to enforce specific authentication requirements.

Profiles and Features

Each profile has associated features you can enable to make the profile's service use a specific authentication method, such as smart card authentication, fingerprint authentication, Kerberos, and so on.

After you select a profile to make it active and enable the features you want, `authselect` reads the appropriate configuration files for those features to run the relevant authentication processes. Every user who signs in to the host is authenticated based on that configured profile.

To see a full list of features available for a specific profile, use the `authselect show` command:

```
authselect show profile
```

The output of the command shows the optional features available for the named profile. For example, the following extract shows the optional features available in the `sssd` profile:

```
authselect show sssd


...
AVAILABLE OPTIONAL FEATURES
---------------------------

with-faillock::
    Enable account locking in case of too many consecutive
    authentication failures.

with-mkhomedir::
    Enable automatic creation of home directories for users on their
    first login.
...
```

This information is also available in the profile's corresponding `/usr/share/authselect/default/`*`profile`*`/README` file.

For more information on how profile files are organized, see the `authselect-profiles(5)` manual page.

About the authselect Utility

You configure authentication on the system using the `authselect` utility. The `authselect` utility manages system authentication profiles and is included in any Oracle Linux installation.

The `authselect` utility consists of the following components:

- The `authselect` command, which manages authentication profiles and their features. Only users with the appropriate administrator privileges can run this command.
- The profiles themselves, that enforce specific authentication mechanisms. These profiles include those supplied by Oracle, provided by vendors, or created by an organization.

The `authselect` utility stores these different profiles in separate directories:

- `/usr/share/authselect/default` contains the profiles provided by Oracle Linux.
- `/usr/share/authselect/vendor` contains the profiles that are provided by vendors. These profiles can override those that are in the `default` directory.
- `/etc/authselect/custom` contains any custom profiles you create.

> **Important:**
>
> The `authselect` utility applies the selected profile. However, `authselect` doesn't configure the service on which the profile is based. Consult the appropriate documentation to configure the profile's service. You must also ensure that the service is started and enabled.

For more details about the `authselect` utility, see the `authselect(8)` manual page.

2

Working With System Authentication Profiles

The `authselect` utility is preinstalled in Oracle Linux. You use its subcommands, arguments, and options to create and delete profiles, select a profile, and configure a profile's features. The tool applies system-wide changes for which you need administrative privileges.

Displaying Profile Information

Use the `authselect` utility to display information about a profile.

To display the active profile and its enabled features, type:

```
authselect current
```

The information displayed is similar to the following:

```
Profile ID: local
Enabled features:
- with-fingerprint
```

The output of the command in this example indicates that the `local` profile is active. The `with-fingerprint` feature is enabled so that authentication uses the `pam_fprintd` module to verify a user's fingerprints.

To display helpful information about any profile (not only the active one) and a list of the features available, use the `authselect show` command:

```
authselect show profile
```

Selecting a Profile

Use the `authselect` utility to select a profile and make it active.

To work with a specific profile and take advantage of its authentication features, you must first select it to be the active profile. You do this using the `authselect select` command.

The syntax is as follows:

```
sudo authselect select profile [features] [options]
```

- For example, to set the `sssd` profile:

  ```
  sudo authselect select sssd
  ```

- You can select a profile and enable one or more of its features at the same time. For example, to select the `sssd` profile and enable its `with-faillock` and `with-mkhomedir` features, use the following command:

  ```
  sudo authselect select sssd with-faillock with-mkhomedir
  ```

For more information about the , see the `authselect(8)` manual page.

Enabling Profile Features

Use the `authselect` utility to enable features in the active profile.

Specifying the features that are enabled for a profile affects how the system handles authentication. You enable profile features by either:

- Specifying extra features to enable in the active profile.
- Replacing enabled features in the active profile. This method is discussed in Selecting a Profile.

The following steps show how to enable extra features in the active profile.

1. Enabling extra features works only on the active profile. You can't enable features in unselected profiles. Identify the active profile using the following command:

   ```
   authselect current
   ```

2. Identify any requirements that the feature needs to work using the following command syntax:

   ```
   sudo authselect requirements profile feature
   ```

3. Satisfy any feature requirements that are listed in the output of the last command before proceeding.
4. Enable the feature:

   ```
   sudo authselect enable-feature feature
   ```

> **Note:**
>
> You can only enable features one at a time.

Example 2-1 Enable account locking and home directories

The following example shows how you can enable extra features in the `sssd` profile to use account locking and automatically create users' home directories.

1. Check the requirements for the `with-faillock` feature.

   The `with-faillock` feature automatically locks an account after too many authentication failures. Run the following command to list the feature's requirements:

   ```
   authselect requirements sssd with-faillock
   ```

Example output:

```
Make sure that SSSD service is configured and enabled. See SSSD
documentation for more information.
```

2. Check the requirements for the `with-mkhomedir` feature.

   The `with-mkhomedir` feature automatically creates the user's home directory when they first sign in. Run the following command to list the feature's requirements:

   ```
   authselect requirements sssd with-mkhomedir
   ```

   Example output:

   ```
   - with-mkhomedir is selected, make sure pam_oddjob_mkhomedir module
    is present and oddjobd service is enabled
    - systemctl enable oddjobd.service
    - systemctl start oddjobd.service
   ```

3. Satisfy the requirements of both features you want to enable.
4. Enable both profile features:

   ```
   sudo authselect enable-feature with-faillock
   ```

   ```
   sudo authselect enable-feature with-mkhomedir
   ```

5. Confirm that both profile features have been enabled in the active profile:

   ```
   authselect current
   ```

   ```
   Profile ID: sssd
   Enabled features:
   - with-fingerprint
   - with-silent-lastlog
   - with-faillock
   - with-mkhomedir
   ```

Example 2-2 Enable the PAM access feature

The following example shows how you can direct the system to check `/etc/security/access.conf` to authenticate and authorize users by enabling the `with-pamaccess` feature in the `local` profile.

1. Automatically enable PAM access:

   ```
   authselect requirements local with-pamaccess
   ```

   ```
   No requirements are specified.
   ```

2. Enable the PAM access profile feature:

```
sudo authselect enable-feature with-pamaccess
```

3. Confirm that the PAM access profile feature has been enabled in the active profile:

```
authselect current
```

```
Profile ID: local
Enabled features:
- with-fingerprint
- with-pamaccess
```

Disabling Profile Features

Use the `authselect` utility to disable features in the active profile.

Use the `authselect disable-feature` command to disable a feature in the active profile.

1. Disabling features works only on the active profile. You can't disable features in unselected profiles. Identify the active profile using the following command:

```
authselect current
```

2. Use the following command syntax to disable a feature in the active profle:

```
sudo authselect disable-feature feature
```

Example 2-3 Disable account locking and home directories features

In this example, you disable the `with-faillock` and `with-mkhomedir` features added to the `sssd` profile.

1. Confirm that the active profile is `sssd` and that the `with-faillock` and `with-mkhomedir` features are enabled, by entering the following command:

```
authselect current
```

```
Profile ID: sssd
Enabled features:
- with-fingerprint
- with-silent-lastlog
- with-faillock
- with-mkhomedir
```

2. Disable the `with-faillock` and `with-mkhomedir` features:

```
sudo authselect disable-feature with-faillock
```

```
sudo authselect disable-feature with-mkhomedir
```

3. Check that the features are no longer enabled in the active profile:

```
Profile ID: sssd
Enabled features:
- with-fingerprint
- with-silent-lastlog
```

Changing Existing Profiles

Adapt an existing profile to suit the authentication needs of the organization.

Profiles use settings stored in the `/etc/nsswitch.conf` file to enforce authentication and you can change these settings to customize authentication. For more information on the format and content of this file, view the `man` page:

```
man 5 nsswitch.conf
```

Don't edit `/etc/nsswitch.conf` directly. Instead, specify the new configuration settings in the `/etc/user-nsswitch.conf` file.

Use this file to

1. Ensure that the profile you want to change is the active profile. If required, select the profile to make it the current profile. For example:

   ```
   sudo authselect select sssd
   ```

2. Edit the `/etc/authselect/user-nsswitch.conf` file with the new configuration settings.

   Typically, this involves specifying the order and types of sources (such as `files`, `sss`, `ldap`, or `dns`) used for system databases such as `passwd`, `group`, or `hosts` to control where user, group, and host information is retrieved from.

   > **Note:**
   >
   > Don't try to change any of the following configurations in the file. If you do, they're ignored:
   >
   > - `passwd`
   > - `group`
   > - `netgroup`
   > - `automount`
   > - `services`

3. Apply the changes.

   ```
   sudo authselect apply-changes
   ```

This step applies the changes in the `/etc/authselect/user-nsswitch.conf` file to the `/etc/nsswitch.conf` file and affects the active profile.

> **Important:**
>
> If the system is part of an environment that uses either Identity Management or Active Directory, don't use `authselect` to manage authentication. When the host is made to join either Identity Management or Active Directory, their respective tools take care of managing authentication.

Creating Custom Profiles

Create a custom profile based on an existing profile.

If you don't want to use the profiles included in Oracle Linux or those provided by vendors, you can create a custom profile.

1. Create the new profile.

   Use the `authselect create-profile` command to create a profile. The syntax is:

   ```
   sudo authselect create-profile newprofile -b template --symlink-meta --
   symlink-pam
   ```

   newprofile
   The name for the custom profile.

   template
   The existing profile on which to base the new profile .

   --symlink-meta
   Creates symbolic links to the meta files in the original directory of the base template profile.

   --symlink-pam
   Creates symbolic links to the PAM templates in the original directory of the base template profile.

   This command creates an `/etc/authselect/custom/newprofile` directory that contains symbolic links to the files in the base profile's original directory. The only file that's **not** a symbolic link in this directory is `nsswitch.conf`.

2. Customize the profile's configuration settings.

   Edit the `/etc/authselect/custom/newprofile/nsswitch.conf` file to include the required configuration.

3. Select the new profile.

   Select the new, custom profile:

   ```
   sudo authselect select custom/newprofile
   ```

Running this command also creates a backup of the original `/etc/nsswitch.conf` file and replaces it with a symbolic link to the corresponding file in the custom profile's directory.

You can check this by comparing the symbolic link `/etc/nsswitch.conf` with the original `/etc/nsswitch.conf.bak` to verify that the original file's contents remain intact.

4. Enable any required features.

   See Enabling Profile Features for reference.

5. (Optional) Verify the profile's configuration.

   Run the following command to display information about the custom profile:

   ```
   authselect current
   ```

3
Using the Winbind Profile

Configure the Winbind profile to work with Windows users and groups.

Winbind is a client-side service that resolves user and group information on a Windows server. Use the Winbind profile to let Oracle Linux work with Windows users and groups.

1. Install the required packages.

   Install the `samba-winbind` package:

   ```
   sudo dnf install samba-winbind -y
   ```

2. Select `winbind` to be the active profile and enable the required features.

   The following command selects the `winbind` profile and enables the `with-faillock` and `with-mkhomedir` features:

   ```
   sudo authselect select winbind with-faillock with-mkhomedir


   Profile "winbind" was selected.
   The following nsswitch maps are overwritten by the profile:
   - passwd
   - group

   Make sure that winbind service is configured and enabled. See winbind
   documentation for
   more information.

   - with-mkhomedir is selected, make sure pam_oddjob_mkhomedir module
     is present and oddjobd service is enabled
     - systemctl enable oddjobd.service
     - systemctl start oddjobd.service
   ```

3. Satisfy the feature requirements of the profile.

   Using the output of the previous command, fulfill the requirements of the features you enabled for the profile.

4. Start the service.

   Start the `winbind` service and enable it to autostart when the system is rebooted.

   ```
   sudo systemctl enable --now winbind
   ```

> **Note:**
>
> If you change the features of a profile that's already active, the revised features replace whatever features were enabled before.

4

Using the System Security Services Daemon

The System Security Services Daemon (SSSD) feature provides a client system with access to remote identity and authentication providers. The SSSD acts as an intermediary between local clients and any backend provider that you configure.

The benefits of using SSSD include:

- **Reduced system load**: Clients don't have to contact the identification or authentication servers directly.
- **Offline authentication**: You can configure SSSD to maintain a cache of user identities and credentials.
- **Single sign-on**: If you configure SSSD to store network credentials, users only need to authenticate a single time per session with the local system to access network resources.

The SSSD service is installed and enabled automatically in Oracle Linux. The default configuration uses the Pluggable Authentication Modules (PAM) and the Name Service Switch (NSS) for managing system access and authentication. No further configuration is required, unless you want to use different authentication services or customize the configuration.

See https://sssd.io/ for more information about SSSD.

Customizing SSSD

By default, the SSSD service used by the `sssd` profile uses Pluggable Authentication Modules (PAM) and the Name Service Switch (NSS) for managing system access and authentication. As you enable extra features for the profile to customize SSSD authentication, you must also configure SSSD for the enabled feature.

Customize an SSSD configuration by creating configuration files within the `/etc/sssd/conf.d` directory. Each configuration file must have the `.conf` suffix.

Configuration files use ini-style syntax. The file is divided into sections, identified by square brackets. Each section contains parameters which are listed as `key = value` entries.

The following example shows how you might configure SSSD to authenticate against an LDAP provider that uses Kerberos:

1. Create a configuration file for the feature and store it in `/etc/sssd/conf.d`, for example `/etc/sssd/conf.d/00-ldap.conf`.
2. Configure `/etc/sssd/conf.d/00-ldap.conf` with the appropriate parameter definitions, for example:

```
[sssd]
config_file_version = 2
domains = LDAP
services = nss, pam

[domain/LDAP]
id_provider = ldap
```

```
ldap_uri = ldap://ldap.mydom.com
ldap_search_base = dc=mydom,dc=com

auth_provider = krb5
krb5_server = krbsvr.mydom.com
krb5_realm = MYDOM.COM
cache_credentials = true

min_id = 5000
max_id = 25000
enumerate = false

[nss]
filter_groups = root
filter_users = root
reconnection_retries = 3
entry_cache_timeout = 300

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

[sssd]

Contains configuration settings for SSSD monitor options, domains, and services. The SSSD monitor service manages the services that SSSD provides.

- `services` defines the services SSSD works with, which includes `nss` for the Name Service Switch and `pam` for Pluggable Authentication Modules.
- The `domains` entry specifies the names of the sections that define authentication domains.

[domain/LDAP]

Defines a domain for an LDAP identity provider that uses Kerberos authentication. Each domain defines where user information is stored, the authentication method, and any configuration options. SSSD can work with LDAP identity providers such as OpenLDAP, Red Hat Directory Server, IPA, and Microsoft Active Directory, and it can use either native LDAP or Kerberos authentication.

- `id_provider` specifies the type of provider (in this example, LDAP).
- `ldap_uri` specifies a comma-separated list of the Universal Resource Identifiers (URIs) of the LDAP servers, in order of preference, which SSSD can connect to.
- `ldap_search_base` specifies the base distinguished name (`dn`) that SSSD uses when performing LDAP user operations on a relative distinguished name (RDN) such as a common name (`cn`).
- `auth_provider` entry specifies the authentication provider (in this example, Kerberos).
- `krb5_server` specifies a comma-separated list of Kerberos servers, in order of preference, which SSSD can connect to.
- `krb5_realm` specifies the Kerberos realm.

- `cache_credentials` specifies if SSSD caches user credentials such as tickets, session keys, and other identifying information to enable offline authentication and single sign-on.

> **Note:**
>
> To enable SSSD to use Kerberos authentication with an LDAP server, you must configure the LDAP server to use both Simple Authentication and Security Layer (SASL) and the Generic Security Services API (GSSAPI). For more information about configuring SASL and GSSAPI for OpenLDAP, see https://www.openldap.org/doc/admin24/sasl.html.

- `min_id` and `max_id` specify upper and lower limits on the values of user and group IDs.
- `enumerate` specifies whether SSSD caches the complete list of users and groups that are available on the provider. The recommended setting is `False` unless a domain contains relatively few users or groups.

`[nss]`

Configures the Name Service Switch (NSS) module that integrates the SSSD database (SSS) with NSS.

- `filter_users` and `filter_groups` prevent NSS from extracting information about the specified users and groups being retrieved from SSS.
- `reconnection_retries` specifies the number of times that SSSD tries to reconnect if a data provider fails.
- `enum_cache_timeout` specifies the number of seconds SSSD caches user information requests for.

`[pam]`

Configures the PAM module that integrates SSSD with PAM.

- `offline_credentials_expiration` specifies the number of days for which to enable cached logins if the authentication provider is offline.
- `offline_failed_login_attempts` specifies how many failed sign-ins are allowed if the authentication provider is offline.
- `offline_failed_login_delay` specifies how many minutes after the maximum number of failed sign-ins before the user can try to sign-in again.

3. Change the mode of `/etc/sssd/conf.d/00-ldap.conf` to `0600`:

```
sudo chmod 0600 /etc/sssd/conf.d/00-ldap.conf
```

4. Ensure that the `sssd` service is running:

```
sudo systemctl status sssd
```

Start and enable the `sssd` service if required:

```
sudo systemctl enable --now sssd
```

5. Select the `sssd` profile.

```
sudo authselect select sssd
```

For more information about SSSD, see the README file: https://github.com/SSSD/sssd.

The manual pages provided for SSSD are comprehensive and provide detailed information on the options that are available. These include `sssd(8)`, `sssd.conf(5)`, `sssd-ldap(5)`, `sssd-krb5(5)`, and `sssd-ipa(5)`.

About Pluggable Authentication Modules

The Pluggable Authentication Modules (PAM) feature is an authentication mechanism used by the `sssd` profile that lets you configure how applications use authentication to verify the identity of a user. The PAM configuration files, in the `/etc/pam.d` directory, describe the authentication procedure for an application. The name of each configuration file is the same as, or similar to, the name of the application for the module provides authentication for. For example, the configuration files for `passwd` and `sudo` are named `passwd` and `sudo`.

Each PAM configuration file contains a list or *stack* of calls to authentication modules. For example, the following listing shows the default content of the `login` configuration file:

```
#%PAM-1.0
auth [user_unknown=ignore success=ok ignore=ignore default=bad]
pam_securetty.so
auth       include      system-auth
auth       include      postlogin
account    required     pam_nologin.so
account    include      system-auth
password   include      system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should only be followed by sessions to be executed in
the user context
session    required     pam_selinux.so open
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      system-auth
session    include      postlogin
-session   optional     pam_ck_connector.so
```

Comments in the file start with the `#` character. The remaining lines each define an operation type, a control flag, the name of a module such as `pam_rootok.so` or the name of an included configuration file such as `system-auth`, and any arguments to the module. PAM provides authentication modules as shared libraries in `/usr/lib64/security`.

For a particular operation type, PAM reads the stack from top to bottom and calls the modules listed in the configuration file. Each module generates a success or failure result when called.

The following operation types are available:

auth
The module tests whether a user is authenticated or authorized to use a service or application. For example, the module might request and verify a password. Such modules can also set credentials, such as a group membership or a Kerberos ticket.

account
The module tests whether an authenticated user is allowed access to a service or application. For example, the module might check if a user account has expired or if a user is only allowed to use a service at a specific time.

password
The module handles updates to an authentication token.

session
The module configures and manages user sessions, performing tasks such as mounting or unmounting a user's home directory.

If the operation type is preceded with a dash (-), PAM doesn't create a system log entry if the module is missing.

Except for `include`, the control flags tell PAM what to do with the result of running a module. The following control flags are defined for use:

optional
The module is required for authentication if it's the only module listed for a service.

required
The module must succeed for access to be granted. PAM continues to process the remaining modules in the stack whether the module succeeds or fails. PAM doesn't immediately inform the user of the failure.

requisite
The module must succeed for access to be granted. If the module succeeds, PAM continues to process the remaining modules in the stack. However, if the module fails, PAM notifies the user immediately and doesn't continue to process the remaining modules in the stack.

sufficient
If the module succeeds, PAM doesn't process any remaining modules of the same operation type. If the module fails, PAM processes the remaining modules of the same operation type to decide overall success or failure.

The control flag field can also define one or more rules that specify the action that PAM takes depending on the value that a module returns. Each rule takes the form *`value=action`*, and is surrounded square brackets, for example:

```
[user_unknown=ignore success=ok ignore=ignore default=bad]
```

If the result that's returned by a module matches a value, PAM uses the corresponding action, or, if there isn't a match, it uses the default action.

The `include` flag specifies that PAM must also consult the PAM configuration file specified as the argument.

For more information, see the `pam(8)` manual page. In addition, each PAM module has its own manual page, for example `pam_unix(8)`, `postlogin(5)`, and `system-auth(5)`.

5

Working With User and Group Accounts

By default, a new installation of Oracle Linux uses local user and group accounts for authentication, permissions handling, and access to resources. When working with local accounts for users and groups, you use three main commands: `useradd`, `groupadd`, and `usermod`. Use these commands and their various option to manage users and groups.

About User and Group Accounts

To implement system authentication, Oracle Linux uses two types of accounts: user and group. Together, these accounts store information about passwords, home directories for users, login shells, group memberships, and so on. The information is used to ensure that only authorized entities are granted access to the system. Users without credentials, or whose credentials don't match the information in these accounts, are locked out of the system.

By default, user and group information is stored locally in the system. However, in an enterprise environment that might have hundreds of servers and thousands of users, user and group account information is better stored in a central repository rather than in files on individual servers. User and group information is configured on a central server and then retrieved through services such as the Lightweight Directory Access Protocol (LDAP) or the Network Information Service (NIS). Central management of this information is more efficient than storing and configuring user and group information locally.

Local User and Group Information Storage

Unless you specify a different authentication mechanism, Oracle Linux verifies a user's identity by using the information stored in the `/etc/passwd` and `/etc/shadow` files.

The `/etc/passwd` file stores account information for each user such as their unique user ID (or *UID*, which is an integer), username, home directory, and login shell. A user signs in with their username, but the OS uses the associated UID. When the user signs in, they're placed in their home directory and their login shell runs.

The `/etc/shadow` file contains a cryptographic hash of the user's password that can only be viewed by an administrator.

The `/etc/group` file stores information about groups of users. A user belongs to one or more groups, and each group can contain one or more users. If you grant access privileges to a group, all members of the group receive the same access privileges. Each group account has a unique group ID (*GID*, also an integer) and an associated group name. The administrator can set a group password that a user must enter to become a member of the group. If a group doesn't have a password, a user can only join the group if the administrator adds that user as a member. A cryptographic hash of the group password is stored in `/etc/gshadow`.

By default, Oracle Linux implements the *user private group* (*UPG*) scheme where adding a user account also creates a corresponding group with the same name as the user, which the user is the only member of.

A user can use the `newgrp` command to override their current primary group. If the user has a password, they can add group membership on a permanent basis. See the `newgrp(1)` manual page.

The `/etc/login.defs` file defines parameters for password aging and related security policies.

For more information about the content of these files, see the `group(5)`, `gshadow(5)`, `login.defs(5)`, `passwd(5)`, and `shadow(5)` manual pages.

Creating User Accounts

Use the `useradd` and `passwd` commands to create a user and set a password.

1. To create a user account using the default settings, enter the following command:

   ```
   sudo useradd [options] username
   ```

   By default, if you specify a username argument with no other options, `useradd` creates a locked user account using the next available UID and assigns a user private group (UPG) rather than the value defined for `GROUP` as the user's group.

   > **Note:**
   >
   > Valid usernames contain only lowercase and uppercase letters, digits, underscores (_), or dashes (-). Dashes aren't allowed at the beginning of the username. The username can end with a dollar sign ($). Fully numeric usernames and usernames . or .. aren't allowed.

2. Assign a password to the account, by entering the following command.

   ```
   sudo passwd username
   ```

   The command prompts you to enter a password for the account.

   To change the password noninteractively (for example, from a script), use the `chpasswd` command instead:

   ```
   echo "username:password" | chpasswd
   ```

You can use the `newusers` command to create several user accounts at the same time. The `newusers` command reads the account information from a text file.

For more information about creating user accounts from the command line, see the `chpasswd(8)`, `newusers(8)`, `passwd(1)`, and `useradd(8)` manual pages.

To create user accounts with a web-based GUI, see Oracle Linux: Using the Cockpit Web Console.

Locking an Account

Prevent a user account from signing into the system.

To lock a user's account, use the `passwd -l` command:

```
sudo passwd -l username
```

To unlock the account, use the `passwd -u` command:

```
sudo passwd -u username
```

For more information, see the `passwd(1)` manual page.

Changing or Deleting User Accounts

Change a user account's access, such as the groups it belongs to, and delete a user account.

To change a user account, use the `usermod` command. The syntax is

```
sudo usermod [options] username
```

For example, to add a user to a group other than the user's default login group:

```
sudo usermod -aG groupname username
```

You can use the `groups` command to display the groups a user belongs to, for example:

```
sudo groups username
```

To delete a user's account, use the `userdel` command:

```
sudo userdel username
```

For more information, see the `groups(1)`, `userdel(8)` and `usermod(8)` manual pages.

Changing Default Settings for User Accounts

View and change the default settings for a user account.

To display the default settings for the current user account, use the following command:

```
useradd -D
```

The output of the command resembles the following:

```
GROUP=100
HOME=/home
INACTIVE=-1
```

```
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

- `INACTIVE`: Specifies after how many days the system locks an account if a user's password expires. If set to 0, the system locks the account immediately. If set to -1, the system doesn't lock the account.
- `SKEL`: Defines a template directory, the contents of which are copied to a new user's home directory. The contents of this directory matches the default shell defined by `SHELL`.

You can specify options to `useradd -D` to change the default settings for the current user account. For example, to change the defaults for `INACTIVE`, `HOME` and `SHELL`:

```
sudo useradd -D -f 3 -b /home2 -s /bin/sh
```

> **Note:**
>
> - If you change the default login shell, consider creating a `SKEL` template directory that contains contents that are appropriate to the new shell.
> - If you specify `/sbin/nologin` for a user's `SHELL`, that user can't sign in to the system directly but processes can run with that user's ID. This setting is typically used for services that run as users other than `root`.

The default settings are stored in the `/etc/default/useradd` file.

For more information, see Configuring Password Ageing and the `useradd(8)` manual page.

Creating Groups

Use groups to quickly assign the same permissions to several users.

To create a group, use the `groupadd` command.

```
sudo groupadd [options] groupname
```

Typically, you might want to use the `-g` option to specify the group ID (GID). This helps to maintain a consistent GID across systems. For example, to use `1000` as the GID for the `devgrp` group:

```
sudo groupadd -g 1000 devgrp
```

For more information, see the `groupadd(8)` manual page.

Changing or Deleting Groups

Rename or delete an existing group.

To change a group, use the `groupmod` command:

```
sudo groupmod [options] username
```

For example, to rename the `oldgrp` group to `newgrp`:

```
sudo groupmod -n newgrp oldgrp
```

To delete a group, use the `groupdel` command:

```
sudo groupdel groupname
```

For more information, see the `groupdel(8)` and `groupmod(8)` manual pages.

Configuring Group Access to Directories

Set a user's primary group to be different to their user private group (UPG), and grant access to files owned by the group.

A user whose primary group is a UPG has a `umask` of 0002. No other user has the same group.

Users whose primary group isn't a UPG have a `umask` of 0022 set by `/etc/profile` or `/etc/bashrc`, which prevents other users, including other members of the primary group, from changing any file that the user owns.

To grant users in the same group write access to files within the same directory, change the group ownership on the directory to the group, and set the `setgid` bit on the directory:

```
sudo chgrp groupname directory
sudo chmod g+s directory
```

Files that are created in such a directory have their group set to that of the directory rather than the primary group of the user who creates the file.

The restricted deletion bit prevents unprivileged users from removing or renaming a file in the directory unless they own either the file or the directory. To set the restricted deletion bit on a directory, use the following command:

```
sudo chmod a+t directory
```

For more information, see the `chmod(1)` manual page.

Configuring Password Ageing

Invalidate a user's password after a specified period.

To specify how users' passwords are aged, edit the following settings in the `/etc/login.defs` file:

Setting	Description
`PASS_MAX_DAYS`	Maximum number of days a password can be used before it must be changed. The default value is 99,999 days.
`PASS_MIN_DAYS`	Minimum number of days allowed between password changes. The default value is 0 days.
`PASS_WARN_AGE`	Number of days before a password expires that a warning is displayed. The default value is 7 days.

For more information, see the `login.defs(5)` manual page.

To change how long a user's account can be inactive before it's locked, use the `usermod` command. For example, to set the inactivity period to 30 days:

```
sudo usermod -f 30 username
```

To change the default inactivity period for new user accounts, use the `useradd` command:

```
sudo useradd -D -f 30
```

A value of -1 specifies that user accounts aren't locked because of inactivity.

For more information, see the `useradd(8)` and `usermod(8)` manual pages.

6

Granting sudo Access to Users

In Oracle Linux, only administrators can perform privileged tasks on the system.

To grant users extra privileges, an administrator can use the `visudo` command to either create a configuration file in the `/etc/sudoers.d` directory or change the `/etc/sudoers` file directly.

Privileges that an administrator assigns by adding configuration files in the `/etc/sudoers.d` directory are preserved between system upgrades and skipped automatically by the `sudo` command if they're invalid. Administrators can also change file ownership and permissions for each configuration file. For more information, see Adding User Authorizations in the sudoers.d Directory.

For information on assigning privileges by editing the `/etc/sudoers` file directly, see Adding User Authorizations in the sudoers File.

About Administrative Access on Oracle Linux

By default, any user can elevate to a `root` shell by running the `su` command and providing the `root` user password when prompted.

```
su

Password:
```

Any user can also perform individual administrative tasks in their current shell, but those commands can't be run until the user provides the `root` user password:

```
su -c "whoami"

Password:

root
```

> **Important:**
>
> Don't share the `root` user password with anyone else or let remote users sign in as the `root` user, both of these actions constitute poor and highly risky security practices.

Elevating to a `root` shell by using the `su` command might be adequate in single-user environments, because only one person needs to administer the system and know the `root`

user password. However, this approach is inadequate for shared systems with several users and administrators that require varying levels of access.

The `sudo` command is better suited for shared systems because any user can supply their own credentials when they elevate to a `root` shell:

```
sudo -s
```

Users exit from the `root` shell in the same way they would have if they had elevated directly with the `su` command and provided the `root` user password:

```
exit
```

In addition, users can run the `sudo` command to perform single administrative tasks with elevated permissions:

```
sudo whoami
```

```
root
```

For more information, see the `su(1)`, `sudo(8)` and `sudoers(5)` manual pages.

> **Note:**
>
> You can optionally disable the `root` user during the Oracle Linux installation process and grant `sudo` administrator privileges to the first user.
>
> In Oracle Linux 10, new users created during the installation process are granted administrative access by default.
>
> For more information, see Oracle Linux 10: Installing Oracle Linux.

Using the sudo Command

If a user has been granted `sudo` access then that user can run administrative commands with elevated privileges:

```
sudo command
```

Depending on the `sudoer` configuration, the user might also be prompted for a password.

Preserving environment variables

In some situations, a user might have set environment variables that they want to reuse or preserve while running elevated commands. They can do this by using the `-E` option.

Example 6-1 Accessing user proxy settings within a sudo session

For example, if the Oracle Linux system is connected to an enterprise intranet or virtual private network (VPN), proxy settings might be required to obtain outbound Internet access.

Terminal commands rely on the `http_proxy`, `https_proxy` and `no_proxy` environment variables. You can set them in the `$HOME/.bashrc` configuration file:

```
export http_proxy=http://proxy.example.com:8080
export https_proxy=https://proxy.example.com:8080
export no_proxy=localhost,127.0.0.1
```

Run the `source` command to refresh the session environment variables without signing out:

```
source $HOME/.bashrc
```

The `sudo` command can use the proxy settings that you have configured as environment variables within the user's session. For example, to run the `curl` command with administrative privileges:

```
sudo -E curl https://www.example.com
```

> **Note:**
>
> An administrator can optionally set system-wide proxy environment variables by configuring them in a shell script and then saving that file in the `/etc/profile.d/` directory.
>
> For more information about configuring network settings, see Oracle Linux 10: Setting Up Networking With NetworkManager.

Elevating to a root shell

You can also use `sudo` access to start an elevated `root` shell. The `-s` option elevates the user to a `root` shell as the `root` user. The `-i` option elevates the user to a `root` shell while preserving both the user profile and shell configuration:

```
sudo -i
```

Exiting sudo

When you have finished running administrative commands, exit the `root` shell and return to the standard user privilege level by using the `exit` command.

Using the visudo Command

To edit the `/etc/sudoers` file in the `vi` text editor without risking any change conflicts from other users on the system, use the `visudo` command:

```
sudo visudo
```

To learn more about how to configure the the `/etc/sudoers` file, see Adding User Authorizations in the sudoers File and the `visudo(8)` manual page.

Administrators can also use the `visudo` command to manage permission files for individual users in the `/etc/sudoers.d/` directory. For more information, see Adding User Authorizations in the sudoers.d Directory.

Adding User Authorizations in the sudoers.d Directory

To set privileges for a specific user, add a file for them in the `/etc/sudoers.d` directory. For example, to set `sudo` permissions for the user `alice`:

```
sudo visudo -f /etc/sudoers.d/alice
```

You can append permissions to `/etc/sudoers.d/alice` in the following format:

```
usernamehostname=command
```

`username` is the name of the user, `hostname` is the name of any hosts for which you're defining permissions, and `command` is the command you're giving the user permission to run, specifying the full executable path and allowed options. If you don't specify options, then the user can run the command with full options.

For example, to grant the user `alice` permission to install packages with the `sudo dnf` command on all hosts:

```
alice           ALL=/usr/bin/dnf
```

You can also add several comma separated commands on the same line. To let the user `alice` run the `sudo dnf` and `sudo yum` commands on all hosts:

```
alice           ALL=/usr/bin/dnf, /usr/bin/yum
```

The `alice` user still needs to use `sudo` when they run privileged commands:

```
sudo dnf install package
```

Use `ALL=(ALL)` in `/etc/sudoers.d/username` to specify that a user can run specified commands as any user, typically root, on any host by using `sudo`. For example, the following grants full root privileges to the user `alice`:

```
alice           ALL=(ALL)
```

The following lets `alice` run the `/usr/bin/dnf` command with `sudo` as any user, but doesn't grant full root privileges or the ability to run other commands with `sudo`:

```
alice           ALL=(ALL) /usr/bin/dnf
```

Adding User Authorizations in the sudoers File

To set user privileges directly in the `/etc/sudoers` file, run the `visudo` command without specifying a file location:

```
sudo visudo
```

You can append permissions to the `/etc/sudoers` file in the same format that you would use if you were adding those permissions to user files in the `/etc/sudoers.d/` directory.

In both cases, you can use aliases to assign broader permission categories instead of specifying each command individually. The `ALL` alias functions as a wildcard for all permissions, so to set the user `bob` to have sudo permission for all commands on all hosts:

```
bob         ALL=(ALL)       ALL
```

Other category aliases are listed in the `/etc/sudoers` file and the `sudoers(5)` manual page. You can create custom aliases using the following format:

```
Cmnd_Alias ALIAS = command
```

You can also add several aliases on the same line, separated by commas. For example, to grant the user `alice` permission to manage system services and software packages:

```
Cmnd_Alias SOFTWARE=/bin/rpm, /usr/bin/up2date, /usr/bin/yum
Cmnd_Alias SERVICES=/sbin/service, /sbin/chkconfig, /usr/bin/systemctl
start, /usr/bin/systemctl stop, /usr/bin/systemctl reload, /usr/bin/systemctl
restart, /usr/bin/systemctl status, /usr/bin/systemctl enable, /usr/bin/
systemctl disable
alice       ALL= SERVICES, SOFTWARE
```

Both users still need to use `sudo` when they run privileged commands:

```
sudo systemctl restart service
```

Using Groups to Manage sudo Access

Assign sudo permissions to groups and add users as group members.

Instead of specifying different levels of `sudo` access for each individual user you can optionally manage `sudo` access at group level by adding the `%` symbol to the group name.

For example, to define permissions for an existing group called `example` in the `/etc/sudoers.d/` directory and then add the user `alice` to that group:

1. Create the `/etc/sudoers.d/example` file by using the `visudo` command:

   ```
   sudo visudo /etc/sudoers.d/example
   ```

2. Grant the `example` group permissions to manage system services and software packages:

```
%example        ALL= SERVICES, SOFTWARE
```

3. Add the the `alice` user to the `example` group:

```
sudo usermod -aG example alice
```

Or, you can set group permissions directly in the `/etc/sudoers` file. For example, to grant the user `bob` full `sudo` access on all hosts, enable the existing group `wheel`, and then add the user `bob` to it:

1. Open the `/etc/sudoers` file by using the `visudo` command without specifying a target file:

```
sudo visudo
```

2. Remove the comment `#` symbol from the beginning of the following line in the `/etc/sudoers` file:

```
%wheel          ALL=(ALL)       ALL
```

3. Add the `bob` user to the `wheel` group to grant them full `sudo` access on all hosts:

```
sudo usermod -aG wheel bob
```