Oracle Linux 9 Setting Up System Users and Authentication





Oracle Linux 9 Setting Up System Users and Authentication,

F56873-03

Copyright $\ensuremath{@}$ 2022, Oracle and/or its affiliates.

Contents

Preface

	Conventions	,		
	Documentation Accessibility	\		
	Access to Oracle Support for Accessibility	\		
	Diversity and Inclusion	\		
1	About System Authentication			
	Authentication in Oracle Linux	1-1		
	Profiles and Supported Features	1-1		
	About the authselect Utility	1-3		
2	Working With System Authentication Profiles			
	Displaying Profile Information	2-1		
	Configuring Profile Features	2-1		
	Selecting the Winbind Profile	2-3		
	Modifying Ready-Made Profiles	2-4		
	Creating Custom Profiles	2-5		
3	Using the System Security Services Daemon			
	Customizing SSSD	3-1		
	About Pluggable Authentication Modules	3-4		
4	Working With User and Group Accounts			
	About User and Group Accounts	4-1		
	Where User and Group Information Is Stored Locally	4-1		
	Creating User Accounts	4-2		
	Locking an Account	4-2		
	Modifying or Deleting User Accounts	4-3		
	Changing Default Settings for User Accounts	4-3		



Creating Groups	4-4	
Modifying or Deleting Groups	4-4	
Configuring Group Access Modes to Directories	4-4	
Configuring Password Ageing	4-5	
Granting sudo Access to Users		
About Administrative Access on Oracle Linux	5-1	
Using the sudo Command	5-2	
Using the visudo Command	5-3	
Adding User Authorizations in the sudoers.d Directory	5-3	
Adding User Authorizations in the sudoers File	5-4	
Using Groups to Manage User Authorizations	5-4	



Preface

Oracle Linux 9: Setting Up System Users and Authentication provides information about how to set up user accounts and authentication mechanisms on an Oracle Linux 9 release.

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/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

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 an entity, such as a user, to a system. A user logs in by providing a user name and a password, and the operating system authenticates the user's identity by comparing this information to data stored on the system. If the login 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. An Oracle Linux installation automatically includes profiles, each of which has predefined features. Profiles and their associated features determine which mechanisms to use to authenticate system access.

The following profiles are available:

- sssd profile: Uses the sssd service to perform system authentication.
- winbind profile: Uses the winbind service to perform system authentication.
- The minimal profile: Uses system files to perform system authentication for local users.

After an Oracle Linux installation, the sssd profile is selected by default to manage authentication on the system. This profile covers most authentication cases including PAM authentication, Kerberos, and so on.

System authentication is not restricted to using only the profiles in Oracle Linux. If preferred, you can also use profiles that might be supplied by vendors. Additionally, you can create your own customized profile to enforce authentication that complies with your company's requirements.

As an added flexibility, you can also reconfigure profiles by revising their active features. For example, you can set the profile to use a variety of alternate back-end directory services such as LDAP, FreeIPA, and Active Directory. Additionally, you can use SSSD in conjunction with a directory service to centralize and simplify user and group management in an environment where multiple users and systems with different access requirements exist.

Profiles and Supported Features

Each profile has associated features you can enable to make the profile's service perform a particular method of authentication, such as smart card authentication, fingerprint authentication, kerberos, and so on. After you select a profile and enable preferred features, authselect automatically reads the appropriate configuration files of those features to run the relevant authentication processes. Every user who logs in to the host is authenticated based on that configured profile.

The following tables shows the profiles and their corresponding supported features:



Table 1-1 Features Supported by sssd Profile

Feature Name	Description
with-faillock	Lock the account after too many authentication failures.
with-mkhomedir	Create home directory on user's first log in.
with-ecryptfs	Enable automatic per-user ecryptfs.
with-smartcard	Authenticate smart cards through SSSD.
with-smartcard-lock-on-removal	Lock the screen when the smart card is removed. Requires that with-smartcard is also enabled.
with-smartcard-required	Only smart card authentication is operative; others, including password, are disabled. Requires that with-smartcard is also enabled.
with-fingerprint	Authenticate through fingerprint reader.
with-silent-lastlog	Disable generation of pam_lostlog messages during login
with-sudo	Enable sudo to use SSSD for rules besides /etc/sudoers.
with-pamaccess	Refer to /etc/access.conf for account authorization.
without-nullock	Do not add the nullock parameter to pam_unix

Table 1-2 Features Supported by winbind Profile

Feature Name	Description
with-faillock	Lock the account after too many authentication failures.
with-mkhomedir	Create home directory on user's first log in.
with-ecryptfs	Enable automatic per-user ecryptfs.
with-fingerprint	Authenticate through fingerprint reader.
with-krb5	Use Kerberos authentication.
with-silent-lastlog	Disable generation of pam_lostlog messages during login
with-pamaccess	Refer to /etc/access.conf for account authorization.
without-nullock	Do not add the nullock parameter to
	pam_unix

For details about each profile, refer to the profile's corresponding /usr/share/authselect/default/profile/README file. See also the authselect-profiles (5) manual page.



About the authselect Utility

The authselect utility is the Oracle Linux tool for configuring authentication on the system. Specifically, the tool manages system authentication profiles. It is automatically included in any Oracle Linux 8 installation.

The authselect utility consists of the following components:

- authselect command to manage system authentication. Only users with the appropriate administrator privileges can run this command.
- Profiles that apply specific authentication mechanisms. These profiles can be those supplied by Oracle, provided by vendors, or created by your own company.

To efficiently manage the variety of supported profiles, authselect stores different types of profiles in corresponding files:

- /usr/share/authselect/default contains the Oracle-supplied 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 profiles you create for your specific environment.

Important:

The authselect utility applies the specifications in your selected profile. However, authselect does not modify the configuration files of the service on which the profile is based. If, for example, you use the <code>sssd</code> profile, you must configure SSSD for the service to function properly. Consult the proper documentation to configure the profile's service. You must also ensure that the service is started and enabled.

For more details about the utility, refer to the authselect (8) manual page.



Working With System Authentication Profiles

The authselect command has various subcommands, arguments, and options to create, delete, switch to a different profile, and modify profile features. A user must have the appropriate privileges to be able to use this configuration tool.

Displaying Profile Information

To determine which profile is currently active in a system, type:

sudo authselect current
Profile ID: sssd
Enabled features:
 with-fingerprint
 with-silent-lastlog

The output of the command indicates that the ssd profile is currently active. At a minimum, authentication with the fingerprint reader is enforced through pam_fprintd. Additionally, no pam lastlog message is displayed on the screen when users log in.

Configuring Profile Features

Enabled features of a profile determine the manner of authentication on the system. You can enable profile features in one of two ways:

- Specify additional features to be enabled in the current profile.
- Replace current features of a selected profile. This method is discussed in Selecting the Winbind Profile.

Enabling Profile Features

1. (Optional): Identify the current profile.

Enabling additional features works only on the current profile. The procedure does not work on unselected profiles.

```
sudo authselect current
```

2. If necessary, identify the feature requirements for the feature to work properly.

```
sudo authselect requirements profile feature
```

- Complete the indicated listed feature requirements as needed.
- 4. Enable the feature.

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

Note that you can only enable features one at a time.

Disabling Profile Features

Use the disable-feature subcommand.

sudo authselect disable-feature feature

Example 2-1 Adding Functionalities to a Profile

The following example shows how you can set account locking and define home directories as additional features of the default sssd profile.

1. Determine the requirements to automatically lock an account after too many authentication failures (with-faillock):

```
sudo authselect requirements sssd with-faillock

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

2. Determine the requirements to automatically create a user home directory at the user's first time log in (with-mkhomedir).

```
sudo authselect requirements sssd with-mkhomedir

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

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

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

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

systemctl enable oddjobd.servicesystemctl start oddjobd.service

5. Confirm that both profile features have been enabled:

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

Example 2-2 Enabling 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. In this case, the PAM access feature needs to be added as an enabled feature for sssd.

Automatically enable PAM access:

```
sudo authselect requirements sssd with-pamaccess
```



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

2. Enable the PAM access profile feature:

sudo authselect enable-feature sssd with-pamaccess

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

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



The prevous example assumes that you have configured /etc/security/ access.conf so that the feature functions correctly. For more information, see the access.conf (5) manual page.

Selecting the Winbind Profile

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

1. Install the samba-winbind package.

sudo dnf install samba-winbind -y

2. Select the winbind profile.

When selecting a profile, you can enable multiple features in the same command, for example:

sudo authselect select winbind with-faillock with-mkhomedir [options]
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

For other options you can use with the authselect select command, see the authselect (8) manual page.

- 3. Fulfill the requirements of the features you enabled for the profile.
- 4. Start the winbind service.

```
sudo systemctl start winbind
sudo systemctl enable winbind
```



If you modify features of an already current and active profile, the revised features will replace whatever features were previously enabled.

Modifying Ready-Made Profiles

Profiles also use information stored in the /etc/nsswitch.conf file to enforce authentication. However, to modify and customize a ready-made profile, specify its configuration properties in the /etc/user-nsswitch.conf file. Do not edit the /etc/nsswitch.conf directly.

1. If necessary, select the profile to make it current, for example:

sudo authselect select sssd

2. Edit the /etc/authselect/user-nsswitch.conf file as required.

Note:

Do not modify the any of following configurations in the file. If you do, those modifications will be ignored:

- passwd
- group
- netgroup
- automount
- services
- 3. Apply the changes.

sudo authselect apply-changes

The changes in /etc/authselect/user-nsswitch.conf are applied to /etc/nsswitch.conf and will be used by the current profile.

Important:

If the system is part of an environment that uses either Identity Management or Active Directory, do not 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 of the environment.

Creating Custom Profiles

If you do not want to use the profiles included in Oracle Linux or those provided by vendors, you can create your own specific profile.

1. Create the profile.

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

newprofile

Name of your custom profile.

template

Base to be used for the custom profile, which is either sssd or winbind.

--symlink-meta

Creates symbolic links to the meta files in the original directory of the template profile you are using as base.

--symlink-pam

Creates symbolic links to the PAM templates in the original directory of the template profile you are using as base.

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

- Edit the /etc/authselect/custom/newprofile/nsswitch.conf file according to your preference.
- 3. Select your custom profile.

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

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 your custom profile's directory.

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

4. Enable features for your new profile as needed.

See Configuring Profile Features for reference.

5. (Optional) Verify the configuration of the custom profile.

sudo authselect current



Using the System Security Services Daemon

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

The benefits of configuring SSSD include the following:

- Reduced system load
 Clients do not 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 access

If you configure SSSD to store network credentials, users need only authenticate once per session with the local system to access network resources.

Because the Oracle Linux sssd profile is used by default, the SSSD service is also automatically installed and enabled on a newly installed system. The default configuration uses the Pluggable Authentication Modules (PAM) and the Name Service Switch (NSS) for managing access and authentication on a system. No further configuration is required, unless you wish to use different authentication services or wish to customize the configuration to use alternative values to the default settings.

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 access and authentication on a system. As you enable additional features for the profile to customize SSSD authentication, you must also configure SSSD for the enabled feature.

You customize an SSSD configuration by creating configuration files within the <code>/etc/sssd/conf.d</code> directory. Each configuration file must have the <code>.conf</code> suffix to enable it when SSSD is started. Configuration files use ini-style syntax as format. The content consist of sections, which are identified by square brackets, and parameters, lwhich are isted as key = value entries. The man pages provided for SSSD are comprehensive and provide detailed information on the options that are available.

The following example demonstrates how you might configure SSSD to authenticate against an LDAP provider with Kerberos configured:

- 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 parameter definitions, such as the following:

```
[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 supported services, which should include nss for the Name Service Switch and pam for Pluggable Authentication Modules.
- The domains entry specifies the name 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, to which SSSD can connect.
- ldap_search_base specifies the base distinguished name (dn) that SSSD should use 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, to which SSSD can connect.
- krb5 realm specifies the Kerberos realm.
- cache_credentials specifies if SSSD caches user credentials such as tickets, session keys, and other identifying information to support offline authentication and single sign-on.

Note:

To allow 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 SSS database 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 should attempt to reconnect if a data provider crashes.
- enum_cache_timeout specifies the number of seconds for which SSSD caches user information requests.

[pam]

Configures the PAM module that integrates SSSD with PAM.

- offline_credentials_expiration specifies the number of days for which to allow cached logins if the authentication provider is offline.
- offline_failed_login_attempts specifies how many failed login attempts are allowed if the authentication provider is offline.
- offline_failed_login_delay specifies how many minutes after the limit of allowed failed login attempts have been exceeded before a new login attempt is permitted.
- 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. If it is not started yet, enable the SSSD service:
- 5. Select the sssd profile.



```
sudo authselect select sssd
```

For more information about the SSSD service, see the <code>sssd(8)</code> manual page as well as https://pagure.io/SSSD/sssd/. Additionally, you can consult <code>sssd.conf(5)</code>, <code>sssd-ldap(5)</code>, <code>sssd-ipa(5)</code>, and other manual pages.

About Pluggable Authentication Modules

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

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
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 a # 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 defined for use:

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 allowed to use a service at a given 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 does not add an create a system log entry if the module is missing.

With the exception of 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 is the only module listed for a service.

required

The module must succeed for access to be granted. PAM continues to execute the remaining modules in the stack whether the module succeeds or fails. PAM does not immediately inform the user of the failure.

requisite

The module must succeed for access to be granted. If the module succeeds, PAM continues to execute the remaining modules in the stack. However, if the module fails, PAM notifies the user immediately and does not continue to execute the remaining modules in the stack.

sufficient

If the module succeeds, PAM does not process any remaining modules of the same operation type. If the module fails, PAM processes the remaining modules of the same operation type to determine overall success or failure.

The control flag field can also define one or more rules that specify the action that PAM should take depending on the value that a module returns. Each rule takes the form value=action, and the rules are enclosed in square brackets, for example:

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

If the result that is returned by a module matches a value, PAM uses the corresponding action, or, if there is no match, it uses the default action.

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

Most authentication modules and PAM configuration files have their own manual pages. Relevant files are stored in the /usr/share/doc/pam directory.

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).



4

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. Through these commands and their different options, you can add or delete users and groups, as well as modify user or group settings.

About User and Group Accounts

To implement system authentication, Oracle Linux uses two types of accounts: user and group. Together, these accounts hold information such as passwords, home directories for users, login shells, group settings and memberships, and so on. The information is used to ensure that only authorized logins are granted access to the system. Users without credentials, or whose credentials do not match the information in these accounts, are locked out of the system.

By default, user and group information is located 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.

Where User and Group Information Is Stored Locally

Unless you select a different authentication mechanism during installation or use the authselect command to create an authentication profile, Oracle Linux verifies a user's identity by using the information that is stored in the /etc/passwd and /etc/shadow files.

The /etc/passwd file stores account information for each user such as his or her unique user ID (or *UID*, which is an integer), user name, home directory, and login shell. A user logs in using his or her user name, but the operating system uses the associated UID. When the user logs in, he or she is placed in his or her home directory and his or her login shell runs.

The /etc/group file stores information about groups of users. A user also belongs to one or more groups, and each group can contain one or more users. If you can 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*, again an integer) and an associated group name.

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

By default, both users and groups use shadow passwords, which are cryptographically hashed and stored in /etc/shadow and /etc/gshadow respectively. These shadow password files are readable only by the administraor. The administrator can set a group password that a

user must enter to become a member of the group. If a group does not have a password, a user can only join the group if the administrator adds that user as a member.

A user can use the newgrp command to log into a new group or to change the current group ID during a login section. If the user has a password, he or she 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

1. Type the following command:

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

You can specify options to change the account's settings from the default ones.

By default, if you specify a user name argument with no additional 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.

2. Assign a password to the account.

```
sudo passwd username
```

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

If you want to change the password non-interactively (for example, from a script), use the chpasswd command instead:

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

Alternatively, you can use the newusers command to create a number of user accounts at the same time.

For more information, see the chpasswd(8), newusers(8), passwd(1), and useradd(8) manual pages.

To create users by using the web-based GUI, see Oracle Linux: Using the Cockpit Web Console.

Locking an Account

To lock a user's account, use the passwd -1 command.

```
sudo passwd -1 username
```

To unlock the account, use the passwd -u command.

```
sudo passwd -u username
```

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



Modifying or Deleting User Accounts

To modify a user account, use the usermod command.

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

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

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

You can use the groups command to display the groups to which a user belongs, 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

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

```
sudo useradd -D

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 does not lock the account.

SKEL: Defines a template directory, whose contents are copied to a newly created user's home directory. The contents of this directory should match the default shell defined by SHELL.

You can specify options to useradd -D to change the default settings for user accounts. 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, you would probably also create a new 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 cannot log into 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

To create a group, use the groupadd command.

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

Typically, you might want to use the $\neg g$ option to specify the group ID (GID). For example:

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

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

Modifying or Deleting Groups

To modify a group, use the groupmod command:

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

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

```
sudo groupdel username
```

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

Configuring Group Access Modes to Directories

Users whose primary group is not a UPG have a <code>umask</code> of 0022 set by <code>/etc/profile</code> or <code>/etc/bashrc</code>, which prevents other users, including other members of the primary group, from modifying any file that the user owns.

A user whose primary group is a UPG has a umask of 0002. It is assumed that no other user has the same group.

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 <code>setgid</code> 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:

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

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

Configuring Password Ageing

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 for which a password can be used before it must be changed. The default value is 99,999 days.
PASS_MIN_DAYS	Minimum number of days that is allowed between password changes. The default value is 0 days.
PASS_WARN_AGE	Number of days warning that is given before a password expires. 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 is 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 are not locked due to inactivity.

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



5

Granting sudo Access to Users

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

To grant users additional privileges, an administrator can use the <code>visudo</code> command to either create a new configuration file in the <code>/etc/sudoers.d</code> directory or modify the <code>/etc/sudoers</code> file.

Privileges that an administrator assigns by using configuration files in the /etc/sudoers.d directory are preserved between system upgrades and skipped automatically by the sudo command if they are 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.

Alternatively, an administrator can assign privileges directly in the /etc/sudoers file by using the visudo command. For more information, 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 provide the root user password:

```
su
Password:
```

Any user can also perform single administrative tasks in the same shell, but those commands cannot be executed until that user provides the root user password:

```
su -c "whoami"
Password:
```

Elevating to a \mathtt{root} shell by using the \mathtt{su} command can work well for single user environments and workstations because only one person needs to administer the system and know the \mathtt{root} user password. However, that does not scale well for shared systems with multiple users and administrators that require varying levels of access.

Oracle strongly recommends against sharing your root user password with anyone else or allowing remote users to sign in as the root user, both of which constitute poor and highly risky security practice.

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 can subsequently terminate their 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
```

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



You can optionally disable the root user during the Oracle Linux 8 installation process and grant sudo administrator privileges to the first user. For more information, see Oracle Linux 9: Installing Oracle Linux.

Using the sudo Command

If your user has been granted sudo access then you can run administrative commands with elevated privileges:

```
sudo command
```

Depending on your sudoer configuration, you may also be prompted for a password.

In some situations, you may have set environment variables that you want to reuse or preserve while running elevated commands, and you can do so by using the -E option.

For example, you may have connected your Oracle Linux system to an enterprise intranet or virtual private network (VPN) for which you also need proxy settings for outbound Internet access.

The environment variables on which terminal commands rely for proxy access are https_proxy and no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set them in your no_proxy, and you can set the your no_proxy, and you can set the your <a href="http_pr

```
export http_proxy=http://example.com
export https_proxy=https://example.com
export no proxy=localhost,127.0.0.1
```

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

```
source ~/.bashrc
```

You can now safely pass those proxy settings to elevated commands. For example, to run the curl command with administrative privileges:

```
sudo -E curl https://docs.oracle.com
```





An administrator can optionally set system-wide proxy environment variables by configuring them in a shell script and then saving that file in the <code>/etc/profile.d/</code> directory.

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

```
sudo -i
```

When you have finished running administrative commands, terminate the root shell and return to your usual privilege level by using the exit command.

For more information about configuring network settings, see Oracle Linux 9: Setting Up Networking.

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 <code>visudo</code> command to manage permission files for individual users in the <code>/etc/sudoers.d/</code> 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:

```
username hostname=command
```

username is the name of the user, hostname is the name of any hosts for which you are defining permissions, and command is the permitted command with full executable path and options. If you do not 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 allow the user alice to run both 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

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 exactly the same format that you would if you were adding those permissions to user files in the /etc/sudoers.d/ directory.

In both cases, you can use aliases to permit broader permission categories instead of specifying each command individually. The \mathtt{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
```

Additional aliased categories are listed in the /etc/sudoers file and the sudoers (5) manual page. You can create your own aliases in the following format:

```
Cmnd Alias ALIAS = command
```

In addition, you can also add several comma separated aliases on the same line. For example, to grant the user alice permission to manage system services and software packages:

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

```
\verb+sudo+ systemctl restart $\mathit{service}$\\
```

Using Groups to Manage User Authorizations

Instead of specifying different levels of \mathtt{sudo} access for each individual user you can optionally manage \mathtt{sudo} access at group level by adding the \$ symbol to the group name.

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

Alternatively, 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:

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

