



Rocky Enterprise Linux 9.2 Manual Pages on command 'sudoedit.8'

C:~>man sudoedit.8

SUDO(8) BSD System Manager's Manual SUDO(8)

NAME

sudo, sudoedit ? execute a command as another user

SYNOPSIS

sudo -h | -K | -k | -V

sudo -v [-ABknS] [-g group] [-h host] [-p prompt] [-u user]

sudo -l [-ABknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]

sudo [-ABbEHnPS] [-C num] [-g group] [-h host] [-p prompt] [-r role] [-t type]

[-T timeout] [-u user] [VAR=value] [-i | -s] [command]

sudoedit [-ABknS] [-C num] [-g group] [-h host] [-p prompt] [-T timeout] [-u user]

file ...

DESCRIPTION

sudo allows a permitted user to execute a command as the superuser or another user,

as specified by the security policy. The invoking user's real (not effective) user-

ID is used to determine the user name with which to query the security policy.

sudo supports a plugin architecture for security policies and input/output logging.

Third parties can develop and distribute their own policy and I/O logging plugins to

work seamlessly with the sudo front end. The default security policy is sudoers,

which is configured via the file /etc/sudoers, or via LDAP. See the Plugins section

for more information.

The security policy determines what privileges, if any, a user has to run sudo. The

policy may require that users authenticate themselves with a password or another au?

thentication mechanism. If authentication is required, sudo will exit if the user's password is not entered within a configurable time limit. This limit is policy-specific; the default password prompt timeout for the sudoers security policy is 0 minutes.

Security policies may support credential caching to allow the user to run sudo again for a period of time without requiring authentication. By default, the sudoers policy caches credentials on a per-terminal basis for 15 minutes. See the `timestamp_type` and `timestamp_timeout` options in `sudoers(5)` for more information. By running sudo with the `-v` option, a user can update the cached credentials without running a command.

When invoked as `sudoedit`, the `-e` option (described below), is implied.

Security policies may log successful and failed attempts to use sudo. If an I/O plugin is configured, the running command's input and output may be logged as well.

The options are as follows:

`-A, --askpass`

Normally, if sudo requires a password, it will read it from the user's terminal. If the `-A` (askpass) option is specified, a (possibly graphical) helper program is executed to read the user's password and output the password to the standard output. If the `SUDO_ASKPASS` environment variable is set, it specifies the path to the helper program. Otherwise, if `sudo.conf(5)` contains a line specifying the askpass program, that value will be used. For example:

```
# Path to askpass helper program
Path askpass /usr/X11R6/bin/ssh-askpass
```

If no askpass program is available, sudo will exit with an error.

`-B, --bell` Ring the bell as part of the password prompt when a terminal is present.

This option has no effect if an askpass program is used.

`-b, --background`

Run the given command in the background. Note that it is not possible to use shell job control to manipulate background processes started by sudo. Most interactive commands will fail to work properly in background mode.

`-C num, --close-from=num`

Close all file descriptors greater than or equal to `num` before executing

a command. Values less than three are not permitted. By default, sudo will close all open file descriptors other than standard input, standard output and standard error when executing a command. The security policy may restrict the user's ability to use this option. The sudoers policy only permits use of the -C option when the administrator has enabled the `closefrom_override` option.

`-E, --preserve-env`

Indicates to the security policy that the user wishes to preserve their existing environment variables. The security policy may return an error if the user does not have permission to preserve the environment.

`--preserve-env=list`

Indicates to the security policy that the user wishes to add the comma-separated list of environment variables to those preserved from the user's environment. The security policy may return an error if the user does not have permission to preserve the environment.

`-e, --edit` Edit one or more files instead of running a command. In lieu of a path name, the string "sudeedit" is used when consulting the security policy.

If the user is authorized by the policy, the following steps are taken:

1. Temporary copies are made of the files to be edited with the owner set to the invoking user.
2. The editor specified by the policy is run to edit the temporary files. The sudoers policy uses the `SUDO_EDITOR`, `VISUAL` and `EDITOR` environment variables (in that order). If none of `SUDO_EDITOR`, `VISUAL` or `EDITOR` are set, the first program listed in the editor `sudoers(5)` option is used.
3. If they have been modified, the temporary files are copied back to their original location and the temporary versions are removed.

To help prevent the editing of unauthorized files, the following restrictions are enforced unless explicitly allowed by the security policy:

- ? Symbolic links may not be edited (version 1.8.15 and higher).
- ? Symbolic links along the path to be edited are not followed when the parent directory is writable by the invoking user unless that user is root (version 1.8.16 and higher).

? Files located in a directory that is writable by the invoking user may not be edited unless that user is root (version 1.8.16 and higher).

Users are never allowed to edit device special files.

If the specified file does not exist, it will be created. Note that un? like most commands run by sudo, the editor is run with the invoking user's environment unmodified. If, for some reason, sudo is unable to update a file with its edited version, the user will receive a warning and the edited copy will remain in a temporary file.

`-g group, --group=group`

Run the command with the primary group set to group instead of the primary group specified by the target user's password database entry. The group may be either a group name or a numeric group-ID (GID) prefixed with the `#` character (e.g., `#0` for GID 0). When running a command as a GID, many shells require that the `#` be escaped with a backslash (`\#`).

If no `-u` option is specified, the command will be run as the invoking user. In either case, the primary group will be set to group. The sudoers policy permits any of the target user's groups to be specified via the `-g` option as long as the `-P` option is not in use.

`-H, --set-home`

Request that the security policy set the HOME environment variable to the home directory specified by the target user's password database entry.

Depending on the policy, this may be the default behavior.

`-h, --help` Display a short help message to the standard output and exit.

`-H host, --host=host`

Run the command on the specified host if the security policy plugin supports remote commands. Note that the sudoers plugin does not currently support running remote commands. This may also be used in conjunction with the `-l` option to list a user's privileges for the remote host.

`-i, --login`

Run the shell specified by the target user's password database entry as a login shell. This means that login-specific resource files such as `.profile`, `.bash_profile` or `.login` will be read by the shell. If a command is specified, it is passed to the shell for execution via the

shell's -c option. If no command is specified, an interactive shell is executed. sudo attempts to change to that user's home directory before running the shell. The command is run with an environment similar to the one a user would receive at log in. Note that most shells behave differently when a command is specified as compared to an interactive session; consult the shell's manual for details. The Command environment section in the sudoers(5) manual documents how the -i option affects the environment in which a command is run when the sudoers policy is in use.

-K, --remove-timestamp

Similar to the -k option, except that it removes the user's cached credentials entirely and may not be used in conjunction with a command or other option. This option does not require a password. Not all security policies support credential caching.

-k, --reset-timestamp

When used without a command, invalidates the user's cached credentials. In other words, the next time sudo is run a password will be required. This option does not require a password and was added to allow a user to revoke sudo permissions from a .logout file.

When used in conjunction with a command or an option that may require a password, this option will cause sudo to ignore the user's cached credentials. As a result, sudo will prompt for a password (if one is required by the security policy) and will not update the user's cached credentials.

Not all security policies support credential caching.

-l, --list If no command is specified, list the allowed (and forbidden) commands for the invoking user (or the user specified by the -U option) on the current host. A longer list format is used if this option is specified multiple times and the security policy supports a verbose output format. If a command is specified and is permitted by the security policy, the fully-qualified path to the command is displayed along with any command line arguments. If a command is specified but not allowed by the policy, sudo will exit with a status value of 1.

-n, --non-interactive

Avoid prompting the user for input of any kind. If a password is required for the command to run, sudo will display an error message and exit.

-P, --preserve-groups

Preserve the invoking user's group vector unaltered. By default, the sudoers policy will initialize the group vector to the list of groups the target user is a member of. The real and effective group-IDs, however, are still set to match the target user.

-p prompt, --prompt=prompt

Use a custom password prompt with optional escape sequences. The following

percent (???) escape sequences are supported by the sudoers policy:

%H expanded to the host name including the domain name (on if the machine's host name is fully qualified or the `fqdn` option is set in `sudoers(5)`)

%h expanded to the local host name without the domain name

%p expanded to the name of the user whose password is being requested (respects the `rootpw`, `targetpw`, and `runaspw` flags in `sudoers(5)`)

%U expanded to the login name of the user the command will be run as (defaults to `root` unless the `-u` option is also specified)

%u expanded to the invoking user's login name

%% two consecutive ??? characters are collapsed into a single ??? character

The custom prompt will override the default prompt specified by either the security policy or the `SUDO_PROMPT` environment variable. On systems that use PAM, the custom prompt will also override the prompt specified by a PAM module unless the `passprompt_override` flag is disabled in `sudoers`.

-r role, --role=role

Run the command with an SELinux security context that includes the specified role.

-S, --stdin

Write the prompt to the standard error and read the password from the standard input instead of using the terminal device.

`-s, --shell`

Run the shell specified by the SHELL environment variable if it is set or the shell specified by the invoking user's password database entry. If a command is specified, it is passed to the shell for execution via the shell's `-c` option. If no command is specified, an interactive shell is executed. Note that most shells behave differently when a command is specified as compared to an interactive session; consult the shell's manual for details.

`-t type, --type=type`

Run the command with an SELinux security context that includes the specified type. If no type is specified, the default type is derived from the role.

`-U user, --other-user=user`

Used in conjunction with the `-l` option to list the privileges for user instead of for the invoking user. The security policy may restrict listing other users' privileges. The sudoers policy only allows root or a user with the ALL privilege on the current host to use this option.

`-T timeout, --command-timeout=timeout`

Used to set a timeout for the command. If the timeout expires before the command has exited, the command will be terminated. The security policy may restrict the ability to set command timeouts. The sudoers policy requires that user-specified timeouts be explicitly enabled.

`-u user, --user=user`

Run the command as a user other than the default target user (usually root). The user may be either a user name or a numeric user-ID (UID) prefixed with the `#` character (e.g., `#0` for UID 0). When running commands as a UID, many shells require that the `#` be escaped with a backslash (`\#`). Some security policies may restrict UIDs to those listed in the password database. The sudoers policy allows UIDs that are not in the password database as long as the `targetpw` option is not set. Other security policies may not support this.

`-V, --version`

Print the sudo version string as well as the version string of the security

curity policy plugin and any I/O plugins. If the invoking user is already root the -V option will display the arguments passed to configure when sudo was built and plugins may display more verbose information such as default options.

-v, --validate

Update the user's cached credentials, authenticating the user if necessary. For the sudoers plugin, this extends the sudo timeout for another 15 minutes by default, but does not run a command. Not all security policies support cached credentials.

-- The -- option indicates that sudo should stop processing command line arguments.

Environment variables to be set for the command may also be passed on the command line in the form of VAR=value, e.g., LD_LIBRARY_PATH=/usr/local/pkg/lib. Variables passed on the command line are subject to restrictions imposed by the security policy plugin. The sudoers policy subjects variables passed on the command line to the same restrictions as normal environment variables with one important exception. If the setenv option is set in sudoers, the command to be run has the SETENV tag set or the command matched is ALL, the user may set variables that would otherwise be forbidden. See sudoers(5) for more information.

COMMAND EXECUTION

When sudo executes a command, the security policy specifies the execution environment for the command. Typically, the real and effective user and group and IDs are set to match those of the target user, as specified in the password database, and the group vector is initialized based on the group database (unless the -P option was specified).

The following parameters may be specified by security policy:

- ? real and effective user-ID
- ? real and effective group-ID
- ? supplementary group-IDs
- ? the environment list
- ? current working directory
- ? file creation mode mask (umask)
- ? SELinux role and type

? scheduling priority (aka nice value)

Process model

There are two distinct ways sudo can run a command.

If an I/O logging plugin is configured or if the security policy explicitly requests it, a new pseudo-terminal (?pty?) is allocated and fork(2) is used to create a second sudo process, referred to as the monitor. The monitor creates a new terminal session with itself as the leader and the pty as its controlling terminal, calls fork(2), sets up the execution environment as described above, and then uses the execve(2) system call to run the command in the child process. The monitor exists to relay job control signals between the user's existing terminal and the pty the command is being run in. This makes it possible to suspend and resume the command. Without the monitor, the command would be in what POSIX terms an "orphaned process group" and it would not receive any job control signals from the kernel. When the command exits or is terminated by a signal, the monitor passes the command's exit status to the main sudo process and exits. After receiving the command's exit status, the main sudo passes the command's exit status to the security policy's close function and exits.

If no pty is used, sudo calls fork(2), sets up the execution environment as described above, and uses the execve(2) system call to run the command in the child process. The main sudo process waits until the command has completed, then passes the command's exit status to the security policy's close function and exits. As a special case, if the policy plugin does not define a close function, sudo will execute the command directly instead of calling fork(2) first. The sudoers policy plugin will only define a close function when I/O logging is enabled, a pty is required, or the pam_session or pam_setcred options are enabled. Note that pam_session and pam_setcred are enabled by default on systems using PAM.

Signal handling

When the command is run as a child of the sudo process, sudo will relay signals it receives to the command. The SIGINT and SIGQUIT signals are only relayed when the command is being run in a new pty or when the signal was sent by a user process, not the kernel. This prevents the command from receiving SIGINT twice each time the user enters control-C. Some signals, such as SIGSTOP and SIGKILL, cannot be caught and thus will not be relayed to the command. As a general rule, SIGTSTP should be used instead of SIGSTOP when you wish to suspend a command being run by sudo.

As a special case, sudo will not relay signals that were sent by the command it is running. This prevents the command from accidentally killing itself. On some systems, the `reboot(8)` command sends `SIGTERM` to all non-system processes other than itself before rebooting the system. This prevents sudo from relaying the `SIGTERM` signal it received back to `reboot(8)`, which might then exit before the system was actually rebooted, leaving it in a half-dead state similar to single user mode. Note, however, that this check only applies to the command run by sudo and not any other processes that the command may create. As a result, running a script that calls `reboot(8)` or `shutdown(8)` via sudo may cause the system to end up in this undefined state unless the `reboot(8)` or `shutdown(8)` are run using the `exec()` family of functions instead of `system()` (which interposes a shell between the command and the calling process).

If no I/O logging plugins are loaded and the policy plugin has not defined a `close()` function, set a command timeout or required that the command be run in a new pty, sudo may execute the command directly instead of running it as a child process.

Plugins

Plugins may be specified via `Plugin` directives in the `sudo.conf(5)` file. They may be loaded as dynamic shared objects (on systems that support them), or compiled directly into the sudo binary. If no `sudo.conf(5)` file is present, or it contains no `Plugin` lines, sudo will use the traditional sudoers security policy and I/O logging. See the `sudo.conf(5)` manual for details of the `/etc/sudo.conf` file and the `sudo_plugin(5)` manual for more information about the sudo plugin architecture.

EXIT VALUE

Upon successful execution of a command, the exit status from sudo will be the exit status of the program that was executed. If the command terminated due to receipt of a signal, sudo will send itself the same signal that terminated the command.

If the `-l` option was specified without a command, sudo will exit with a value of 0 if the user is allowed to run sudo and they authenticated successfully (as required by the security policy). If a command is specified with the `-l` option, the exit value will only be 0 if the command is permitted by the security policy, otherwise it will be 1.

If there is an authentication failure, a configuration/permission problem or if the given command cannot be executed, sudo exits with a value of 1. In the latter case,

the error string is printed to the standard error. If sudo cannot stat(2) one or more entries in the user's PATH, an error is printed to the standard error. (If the directory does not exist or if it is not really a directory, the entry is ignored and no error is printed.) This should not happen under normal circumstances. The most common reason for stat(2) to return "permission denied" is if you are running an automounter and one of the directories in your PATH is on a machine that is currently unreachable.

SECURITY NOTES

sudo tries to be safe when executing external commands.

To prevent command spoofing, sudo checks "." and "" (both denoting current directory) last when searching for a command in the user's PATH (if one or both are in the PATH). Note, however, that the actual PATH environment variable is not modified and is passed unchanged to the program that sudo executes.

Users should never be granted sudo privileges to execute files that are writable by the user or that reside in a directory that is writable by the user. If the user can modify or replace the command there is no way to limit what additional commands they can run.

Please note that sudo will normally only log the command it explicitly runs. If a user runs a command such as sudo su or sudo sh, subsequent commands run from that shell are not subject to sudo's security policy. The same is true for commands that offer shell escapes (including most editors). If I/O logging is enabled, subsequent commands will have their input and/or output logged, but there will not be traditional logs for those commands. Because of this, care must be taken when giving users access to commands via sudo to verify that the command does not inadvertently give the user an effective root shell. For more information, please see the Preventing shell escapes section in sudoers(5).

To prevent the disclosure of potentially sensitive information, sudo disables core dumps by default while it is executing (they are re-enabled for the command that is run). This historical practice dates from a time when most operating systems allowed set-user-ID processes to dump core by default. To aid in debugging sudo crashes, you may wish to re-enable core dumps by setting "disable_coredump" to false in the sudo.conf(5) file as follows:

```
Set disable_coredump false
```

See the sudo.conf(5) manual for more information.

ENVIRONMENT

sudo utilizes the following environment variables. The security policy has control over the actual content of the command's environment.

EDITOR Default editor to use in -e (sudoedit) mode if neither SUDO_EDITOR nor VISUAL is set.

MAIL Set to the mail spool of the target user when the -i option is specified or when env_reset is enabled in sudoers (unless MAIL is present in the env_keep list).

HOME Set to the home directory of the target user when the -i or -H options are specified, when the -s option is specified and set_home is set in sudoers, when always_set_home is enabled in sudoers, or when env_reset is enabled in sudoers and HOME is not present in the env_keep list.

LOGNAME Set to the login name of the target user when the -i option is specified, when the set_logname option is enabled in sudoers or when the env_reset option is enabled in sudoers (unless LOGNAME is present in the env_keep list).

PATH May be overridden by the security policy.

SHELL Used to determine shell to run with -s option.

SUDO_ASKPASS Specifies the path to a helper program used to read the password if no terminal is available or if the -A option is specified.

SUDO_COMMAND Set to the command run by sudo.

SUDO_EDITOR Default editor to use in -e (sudoedit) mode.

SUDO_GID Set to the group-ID of the user who invoked sudo.

SUDO_PROMPT Used as the default password prompt unless the -p option was specified.

SUDO_PS1 If set, PS1 will be set to its value for the program being run.

SUDO_UID Set to the user-ID of the user who invoked sudo.

SUDO_USER Set to the login name of the user who invoked sudo.

USER Set to the same value as LOGNAME, described above.

VISUAL Default editor to use in -e (sudoedit) mode if SUDO_EDITOR is not set.

FILES

`/etc/sudo.conf` sudo front end configuration

EXAMPLES

Note: the following examples assume a properly configured security policy.

To get a file listing of an unreadable directory:

```
$ sudo ls /usr/local/protected
```

To list the home directory of user yaz on a machine where the file system holding ~yaz is not exported as root:

```
$ sudo -u yaz ls ~yaz
```

To edit the index.html file as user www:

```
$ sudoedit -u www ~www/htdocs/index.html
```

To view system logs only accessible to root and users in the adm group:

```
$ sudo -g adm more /var/log/syslog
```

To run an editor as jim with a different primary group:

```
$ sudoedit -u jim -g audio ~jim/sound.txt
```

To shut down a machine:

```
$ sudo shutdown -r +15 "quick reboot"
```

To make a usage listing of the directories in the /home partition. Note that this runs the commands in a sub-shell to make the cd and file redirection work.

```
$ sudo sh -c "cd /home ; du -s * | sort -rn > USAGE"
```

DIAGNOSTICS

Error messages produced by sudo include:

editing files in a writable directory is not permitted

By default, sudoedit does not permit editing a file when any of the parent directories are writable by the invoking user. This avoids a race condition that could allow the user to overwrite an arbitrary file. See the `sudoedit_checkdir` option in `sudoers(5)` for more information.

editing symbolic links is not permitted

By default, sudoedit does not follow symbolic links when opening files. See the `sudoedit_follow` option in `sudoers(5)` for more information.

effective uid is not 0, is sudo installed setuid root?

sudo was not run with root privileges. The sudo binary must be owned by the root user and have the set-user-ID bit set. Also, it must not be located on a

file system mounted with the `?nosuid?` option or on an NFS file system that maps uid 0 to an unprivileged uid.

effective uid is not 0, is sudo on a file system with the 'nosuid' option set or an NFS file system without root privileges?

sudo was not run with root privileges. The sudo binary has the proper owner and permissions but it still did not run with root privileges. The most common reason for this is that the file system the sudo binary is located on is mounted with the `?nosuid?` option or it is an NFS file system that maps uid 0 to an unprivileged uid.

fatal error, unable to load plugins

An error occurred while loading or initializing the plugins specified in `sudo.conf(5)`.

invalid environment variable name

One or more environment variable names specified via the `-E` option contained an equal sign (`?=?`). The arguments to the `-E` option should be environment variable names without an associated value.

no password was provided

When sudo tried to read the password, it did not receive any characters. This may happen if no terminal is available (or the `-S` option is specified) and the standard input has been redirected from `/dev/null`.

a terminal is required to read the password

sudo needs to read the password but there is no mechanism available for it to do so. A terminal is not present to read the password from, sudo has not been configured to read from the standard input, the `-S` option was not used, and no `askpass` helper has been specified either via the `sudo.conf(5)` file or the `SUDO_ASKPASS` environment variable.

no writable temporary directory found

`sudoedit` was unable to find a usable temporary directory in which to store its intermediate files.

sudo must be owned by uid 0 and have the `setuid` bit set

sudo was not run with root privileges. The sudo binary does not have the correct owner or permissions. It must be owned by the root user and have the `set-user-ID` bit set.

sudoedit is not supported on this platform

It is only possible to run sudoedit on systems that support setting the effective user-ID.

timed out reading password

The user did not enter a password before the password timeout (5 minutes by default) expired.

you do not exist in the passwd database

Your user-ID does not appear in the system passwd database.

you may not specify environment variables in edit mode

It is only possible to specify environment variables when running a command.

When editing a file, the editor is run with the user's environment unmodified.

SEE ALSO

su(1), stat(2), login_cap(3), passwd(5), sudo.conf(5), sudo_plugin(5), sudoers(5), sudoers_timestamp(5), sudoreplay(8), visudo(8)

HISTORY

See the HISTORY file in the sudo distribution (<https://www.sudo.ws/history.html>) for a brief history of sudo.

AUTHORS

Many people have worked on sudo over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the sudo distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to sudo.

CAVEATS

There is no easy way to prevent a user from gaining a root shell if that user is allowed to run arbitrary commands via sudo. Also, many programs (such as editors) allow the user to run commands via shell escapes, thus avoiding sudo's checks. However, on most systems it is possible to prevent shell escapes with the sudoers(5) plugin's noexec functionality.

It is not meaningful to run the cd command directly via sudo, e.g.,

```
$ sudo cd /usr/local/protected
```

since when the command exits the parent process (your shell) will still be the same.

Please see the EXAMPLES section for more information.

Running shell scripts via sudo can expose the same kernel bugs that make set-user-ID shell scripts unsafe on some operating systems (if your OS has a /dev/fd/ directory, set-user-ID shell scripts are generally safe).

BUGS

If you feel you have found a bug in sudo, please submit a bug report at

<https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see

<https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo is provided ?AS IS? and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with sudo or

<https://www.sudo.ws/license.html> for complete details.

Sudo 1.8.31

October 20, 2019

Sudo 1.8.31