NAME

sudo − execute a command as another user

SYNOPSIS

sudo −V | **−h** | **−l** | **−L** | **−v** | **−k** | **−K** | **−s** | [**−H**] [**−S**] [**−b**] | [**−p** prompt] [**−c** class|−] [**−u** username/#uid] *command*

DESCRIPTION

sudo allows a permitted user to execute a *command* as the superuser or another user, as specified in the sudoers file. The real and effective uid and gid are set to match those of the target user as specified in the passwd file (the group vector is also initialized when the target user is not root). By default, **sudo** requires that users authenticate themselves with a password (NOTE: this is the user's password, not the root password). Once a user has been authenticated, a timestamp is updated and the user may then use sudo without a password for a short period of time (five minutes by default).

sudo determines who is an authorized user by consulting the file */etc/sudoers*. By giving **sudo** the `-v` flag a user can update the time stamp without running a *command.* The password prompt itself will also time out if the user's password is not entered with N minutes (again, this is defined at configure time and defaults to 5 minutes).

If a user that is not listed in the *sudoers* file tries to run a command via **sudo**, mail is sent to the proper authorities, as defined at configure time (defaults to root). Note that the mail will not be sent if an unauthorized user tries to run sudo with the `-l` or `-v` flags. This allows users to determine for themselves whether or not they are allowed to use **sudo**.

sudo can log both successful an unsuccessful attempts (as well as errors) to *syslog* (3), a log file, or both. By default **sudo** will log via *syslog* (3) but this is changeable at configure time.

OPTIONS

sudo accepts the following command line options:

-V The `-V` (*version*) option causes **sudo** to print the version number and exit.

-l The `-l` (*list*) option will list out the allowed (and forbidden) commands for the user on the current host.

-L The `-L` (*list* defaults) option will list out the parameters that may be set in a *Defaults* line along with a short description for each. This option is useful in conjunction with *grep* (1).

-h The `-h` (*help*) option causes **sudo** to print a usage message and exit.

-v If given the `-v` (*validate*) option, **sudo** will update the user's timestamp, prompting for the user's password if necessary. This extends the **sudo** timeout to for another N minutes (where N is defined at installation time and defaults to 5 minutes) but does not run a command.

-k The `-k` (*kill*) option to **sudo** invalidates the user's timestamp by setting the time on it to the epoch. 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.

-K The `-K` (sure *kill*) option to **sudo** removes the user's timestamp entirely. This option does not require a password.

-b The `-b` (*background*) option tells **sudo** to run the given command in the background. Note that if you use the `-b` option you cannot use shell job control to manipulate the command.

-p The `-p` (*prompt*) option allows you to override the default password prompt and use a custom one. If the password prompt contains the `%u` escape, `%u` will be replaced with the user's login name. Similarly, `%h` will be replaced with the local hostname.

-c The `-c` (*class*) option causes **sudo** to run the specified command with resources limited by the specified login class. The *class* argument can be either a class name as defined in /etc/login.conf, or a single '−' character. Specifying the *class* as '−' means that the command should be run restricted by the default login capibilities of the user the command is run as. If the *class* argument specifies an existing user class, the command must be run as root, or the **sudo** command must be run from a shell that is

already root. This option is only available on systems with BSD login classes where **sudo** has been configured with the --with-logincap option.

-u The `-u` (*user*) option causes **sudo** to run the specified command as a user other than *root*. To specify a *uid* instead of a *username*, use ‘‘#uid’’.

-s The `-s` (*shell*) option runs the shell specified by the *SHELL* environment variable if it is set or the shell as specified in *passwd* (5).

-H The `-H` (*HOME*) option sets the *HOME* environment variable to the homedir of the target user (root by default) as specified in *passwd* (5). By default, **sudo** does not modify *HOME*.

-S The `-S` (*stdin*) option causes **sudo** to read the password from standard input instead of the terminal device.

-- The `--` flag indicates that **sudo** should stop processing command line arguments. It is most useful in conjunction with the `-s` flag.

RETURN VALUES

sudo quits with an exit value of 1 if there is a configuration/permission problem or if **sudo** cannot execute the given command. In the latter case the error string is printed to stderr. If **sudo** cannot *stat* (2) one or more entries in the user’s `PATH` an error is printed on stderr. (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. Variables that control how dynamic loading and binding is done can be used to subvert the program that **sudo** runs. To combat this the `LD_*`, `_RLD_*`, `SHLIB_PATH` (HP−UX only), and `LIBPATH` (AIX only) environment variables are removed from the environment passed on to all commands executed. **sudo** will also remove the `IFS`, `ENV`, `BASH_ENV`, `KRB_CONF`, `KRB5_CONFIG`, `LOCALDOMAIN`, `RES_OPTIONS` and `HOSTALIASES` variables as they too can pose a threat.

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.

For security reasons, if your OS supports shared libraries and does not disable user-defined library search paths for setuid programs (most do), you should either use a linker option that disables this behavior or link **sudo** statically.

sudo will check the ownership of its timestamp directory (*/var/run/sudo* by default) and ignore the directory’s contents if it is not owned by root and only writable by root. On systems that allow non-root users to give away files via *chown* (2), if the timestamp directory is located in a directory writable by anyone (eg: */tmp*), it is possible for a user to create the timestamp directory before **sudo** is run. However, because **sudo** checks the ownership and mode of the directory and its contents, the only damage that can be done is to ‘‘hide’’ files by putting them in the timestamp dir. This is unlikely to happen since once the timestamp dir is owned by root and inaccessible by any other user the user placing files there would be unable to get them back out. To get around this issue you can use a directory that is not world-writable for the timestamps (*/var/adm/sudo* for instance) or create */var/run/sudo* with the appropriate owner (root) and permissions (0700) in the system startup files.

sudo will not honor timestamps set far in the future. Timestamps with a date greater than current_time + 2 * `TIMEOUT` will be ignored and sudo will log and complain. This is done to keep a user from creating his/her own timestamp with a bogus date on system that allow users to give away files.

EXAMPLES

Note: the following examples assume suitable *sudoers* (5) entries.

To get a file listing of an unreadable directory:

```
% sudo ls /usr/local/protected
```

To list the home directory of user yazza on a machine where the filesystem holding ~yazza is not exported as root:

```
% sudo -u yazza ls ~yazza
```

To edit the *index.html* file as user www:

```
% sudo -u www vi ~www/htdocs/index.html
```

To shutdown 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"
```

ENVIRONMENT

sudo utilizes the following environment variables:

PATH	Set to a sane value if SECURE_PATH is set
SHELL	Used to determine shell to run with -s option
USER	Set to the target user (root unless the -u option is specified)
HOME	In -s or -H mode (or if sudo was configured with the --enable-shell-sets-home option), set to homedir of the target user.
SUDO_PROMPT	Used as the default password prompt
SUDO_COMMAND	Set to the command run by sudo
SUDO_USER	Set to the login of the user who invoked sudo
SUDO_UID	Set to the uid of the user who invoked sudo
SUDO_GID	Set to the gid of the user who invoked sudo
SUDO_PS1	If set, PS1 will be set to its value

FILES

/etc/sudoers	List of who can run what
/var/run/sudo	Directory containing timestamps

AUTHORS

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

```
Todd Miller
Chris Jepeway
```

See the HISTORY file in the **sudo** distribution for a short history of **sudo**.

BUGS

If you feel you have found a bug in sudo, please submit a bug report at http://www.courtesan.com/sudo/bugs/

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** for complete details.

CAVEATS

There is no easy way to prevent a user from gaining a root shell if that user has access to commands allowing shell escapes.

If users have sudo ALL there is nothing to prevent them from creating their own program that gives them a root shell regardless of any '!' elements in the user specification.

Running shell scripts via **sudo** can expose the same kernel bugs that make setuid shell scripts unsafe on some operating systems (if your OS supports the /dev/fd/ directory, setuid shell scripts are generally safe).

SEE ALSO

login_cap (3), *sudoers* (5), *visudo* (8), *su* (1).