Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'sudoers.5' command

$ man sudoers.5

SUDOERS(5) BSD File Formats Manual SUDOERS(5)

NAME

sudoers ? default sudo security policy plugin

DESCRIPTION

The sudoers policy plugin determines a user's sudo privileges. It is the default sudo policy plugin. The policy is driven by the /etc/sudoers file or, optionally in LDAP. The policy format is described in detail in the SUDOERS FILE FORMAT section. For information on storing sudoers pol? icy information in LDAP, please see sudoers.ldap(5).

Configuring sudo.conf for sudoers

sudo consults the sudo.conf(5) file to determine which policy and I/O logging plugins to load. If no sudo.conf(5) file is present, or if it contains no Plugin lines, sudoers will be used for policy decisions and I/O logging. To explicitly configure sudo.conf(5) to use the sudoers plugin, the following configuration can be used.

Plugin sudoers_audit sudoers.so

Plugin sudoers_policy sudoers.so

Plugin sudoers_io sudoers.so

Starting with sudo 1.8.5, it is possible to specify optional arguments to the sudoers plugin in the sudo.conf(5) file. Plugin arguments, if any, should be listed after the path to the plugin (i.e., after sudoers.so). The arguments are only effective for the plugin that opens (and parses) the sudoers file.

For sudo version 1.9.1 and higher, this is the sudoers_audit plugin. For older versions, it is the sudoers_policy plugin. Multiple arguments may be specified, separated by white space. For example:

Plugin sudoers_audit sudoers.so sudoers_mode=0400 error_recovery=false

The following plugin arguments are supported:

error_recovery=bool

The error_recovery argument can be used to control whether sudoers should attempt to recover from syntax errors in the sudoers file. If set to true (the default), sudoers will try to recover from a syntax error by discarding the portion of the line that contains the error until the end of the line. A value of false will disable error recovery. Prior to version 1.9.3, no error recovery was performed.

ldap_conf=pathname

The ldap_conf argument can be used to override the default path to the ldap.conf file.

ldap_secret=pathname

The ldap_secret argument can be used to override the default path to the ldap.secret file.

sudoers_file=pathname

The sudoers_file argument can be used to override the default path to the sudoers file.

sudoers_uid=uid

The sudoers_uid argument can be used to override the default owner of the sudoers file. It should be specified as a numeric user-ID.

sudoers_gid=gid

The sudoers_gid argument can be used to override the default group of the sudoers file. It must be specified as a numeric group-ID (not a group name).

sudoers_mode=mode

The sudoers_mode argument can be used to override the default file mode for the sudoers file. It should be specified as an

octal value.

For more information on configuring sudo.conf(5), please refer to its manual.

User Authentication

The sudoers security policy requires that most users authenticate themselves before they can use sudo. A password is not required if the invoking user is root, if the target user is the same as the invoking user, or if the policy has disabled authentication for the user or command. Unlike su(1), when sudoers requires authentication, it validates the invoking user's credentials, not the target user's (or root's) credentials. This can be changed via the rootpw, targetpw and runaspw flags, described later.

If a user who is not listed in the policy tries to run a command via sudo, mail is sent to the proper authorities. The address used for such mail is configurable via the mailto Defaults entry (described later) and defaults to root.

Note that no mail will be sent if an unauthorized user tries to run sudo with the -l or -v option unless there is an authentication error and either the mail_always or mail_badpass flags are enabled. This allows users to determine for themselves whether or not they are allowed to use sudo. By default, all attempts to run sudo (successful or not) are logged, regardless of whether or not mail is sent.

If sudo is run by root and the SUDO_USER environment variable is set, the sudoers policy will use this value to determine who the actual user is. This can be used by a user to log commands through sudo even when a root shell has been invoked. It also allows the -e option to remain useful even when invoked via a sudo-run script or program. Note, however, that the sudoers file lookup is still done for root, not the user specified by SUDO_USER.

sudoers uses per-user time stamp files for credential caching. Once a user has been authenticated, a record is written containing the user-ID that was used to authenticate, the terminal session ID, the start time of the session leader (or parent process) and a time stamp (using a mono?

tonic clock if one is available). The user may then use sudo without a password for a short period of time (5 minutes unless overridden by the timestamp_timeout option). By default, sudoers uses a separate record for each terminal, which means that a user's login sessions are authenti? cated separately. The timestamp_type option can be used to select the type of time stamp record sudoers will use.

Logging

By default, sudoers logs both successful and unsuccessful attempts (as well as errors). The log_allowed and log_denied flags can be used to control this behavior. Messages can be logged to syslog(3), a log file, or both. The default is to log to syslog(3) but this is configurable via the syslog and logfile settings. See LOG FORMAT for a description of the log file format.

sudoers is also capable of running a command in a pseudo-terminal and logging all input and/or output. The standard input, standard output and standard error can be logged even when not associated with a terminal. I/O logging is not on by default but can be enabled using the log_input and log_output options as well as the LOG_INPUT and LOG_OUTPUT command tags. See I/O LOG FILES for details on how I/O log files are stored.

Starting with version 1.9, the log_servers setting may be used to send event and I/O log data to a remote server running sudo_logsrvd or another service that implements the protocol described by sudo_logsrv.proto(5).

Command environment

Since environment variables can influence program behavior, sudoers pro? vides a means to restrict which variables from the user's environment are inherited by the command to be run. There are two distinct ways sudoers can deal with environment variables.

By default, the env_reset flag is enabled. This causes commands to be executed with a new, minimal environment. On AIX (and Linux systems without PAM), the environment is initialized with the contents of the /etc/environment file. The HOME, MAIL, SHELL, LOGNAME and USER environ? ment variables are initialized based on the target user and the SUDO_* variables are set based on the invoking user. Additional variables, such

as DISPLAY, PATH and TERM, are preserved from the invoking user's envi?
ronment if permitted by the env_check or env_keep options. A few envi?
ronment variables are treated specially. If the PATH and TERM variables
are not preserved from the user's environment, they will be set to de?
fault values. The LOGNAME and USER are handled as a single entity. If
one of them is preserved (or removed) from the user's environment, the
other will be as well. If LOGNAME and USER are to be preserved but only
one of them is present in the user's environment, the other will be set
to the same value. This avoids an inconsistent environment where one of
the variables describing the user name is set to the invoking user and
one is set to the target user. Environment variables with a value begin?
ning with () are removed unless both the name and value parts are matched
by env_keep or env_check, as they may be interpreted as functions by the
bash shell. Prior to version 1.8.11, such variables were always removed.
If, however, the env_reset flag is disabled, any variables not explicitly
denied by the env_check and env_delete options are allowed and their val?
ues are inherited from the invoking process. Prior to version 1.8.21,
environment variables with a value beginning with () were always removed.
Beginning with version 1.8.21, a pattern in env_delete is used to match
bash shell functions instead. Since it is not possible to block all po?
tentially dangerous environment variables, use of the default env_reset
behavior is encouraged.
Environment variables specified by env_check, env_delete, or env_keep may
include one or more ?*? characters which will match zero or more charac?
ters. No other wildcard characters are supported.
By default, environment variables are matched by name. However, if the
pattern includes an equal sign (?=?), both the variables name and value
must match. For example, a bash shell function could be matched as fol?
lows:

    env_keep += "BASH_FUNC_my_func%%=()*"

Without the ?=()*? suffix, this would not match, as bash shell functions
are not preserved by default.
The complete list of environment variables that are preserved or removed,

as modified by global Defaults parameters in sudoers, is displayed when sudo is run by root with the -V option. Please note that the list of en? vironment variables to remove varies based on the operating system sudo is running on.

Other sudoers options may influence the command environment, such as always_set_home, secure_path, set_logname, and set_home.

On systems that support PAM where the pam_env module is enabled for sudo, variables in the PAM environment may be merged in to the environment. If a variable in the PAM environment is already present in the user's envi? ronment, the value will only be overridden if the variable was not pre? served by sudoers. When env_reset is enabled, variables preserved from the invoking user's environment by the env_keep list take precedence over those in the PAM environment. When env_reset is disabled, variables present the invoking user's environment take precedence over those in the PAM environment unless they match a pattern in the env_delete list.

Note that the dynamic linker on most operating systems will remove vari? ables that can control dynamic linking from the environment of set-user-ID executables, including sudo. Depending on the operating system this may include _RLD*, DYLD_*, LD_*, LDR_*, LIBPATH, SHLIB_PATH, and others. These type of variables are removed from the environment before sudo even begins execution and, as such, it is not possible for sudo to preserve them.

As a special case, if the -i option (initial login) is specified, sudoers will initialize the environment regardless of the value of env_reset.

The DISPLAY, PATH and TERM variables remain unchanged; HOME, MAIL, SHELL, USER, and LOGNAME are set based on the target user. On AIX (and Linux systems without PAM), the contents of /etc/environment are also included.

All other environment variables are removed unless permitted by env_keep or env_check, described above.

Finally, the restricted_env_file and env_file files are applied, if present. The variables in restricted_env_file are applied first and are subject to the same restrictions as the invoking user's environment, as detailed above. The variables in env_file are applied last and are not

subject to these restrictions. In both cases, variables present in the files will only be set to their specified values if they would not con? flict with an existing environment variable.

SUDOERS FILE FORMAT

The sudoers file is composed of two types of entries: aliases (basically variables) and user specifications (which specify who may run what).

When multiple entries match for a user, they are applied in order. Where there are multiple matches, the last match is used (which is not neces? sarily the most specific match).

The sudoers file grammar will be described below in Extended Backus-Naur Form (EBNF). Don't despair if you are unfamiliar with EBNF; it is fairly simple, and the definitions below are annotated.

Quick guide to EBNF

EBNF is a concise and exact way of describing the grammar of a language. Each EBNF definition is made up of production rules. E.g.,

symbol ::= definition | alternate1 | alternate2 ...

Each production rule references others and thus makes up a grammar for the language. EBNF also contains the following operators, which many readers will recognize from regular expressions. Do not, however, con? fuse them with ?wildcard? characters, which have different meanings.

? Means that the preceding symbol (or group of symbols) is optional. That is, it may appear once or not at all.

* Means that the preceding symbol (or group of symbols) may appear zero or more times.

+ Means that the preceding symbol (or group of symbols) may appear one or more times.

Parentheses may be used to group symbols together. For clarity, we will use single quotes (') to designate what is a verbatim character string (as opposed to a symbol name).

Aliases

There are four kinds of aliases: User_Alias, Runas_Alias, Host_Alias and Cmnd_Alias. Beginning with sudo 1.9.0, Cmd_Alias may be used in place of Cmnd_Alias if desired.

Alias ::= 'User_Alias' User_Alias_Spec (':' User_Alias_Spec)* |

'Runas_Alias' Runas_Alias_Spec (':' Runas_Alias_Spec)* |

'Host_Alias' Host_Alias_Spec (':' Host_Alias_Spec)* |

'Cmnd_Alias' Cmnd_Alias_Spec (':' Cmnd_Alias_Spec)* |

'Cmd_Alias'   Cmnd_Alias_Spec (':' Cmnd_Alias_Spec)*

User_Alias ::= NAME

User_Alias_Spec ::= User_Alias '=' User_List

Runas_Alias ::= NAME

Runas_Alias_Spec ::= Runas_Alias '=' Runas_List

Host_Alias ::= NAME

Host_Alias_Spec ::= Host_Alias '=' Host_List

Cmnd_Alias ::= NAME

Cmnd_Alias_Spec ::= Cmnd_Alias '=' Cmnd_List

NAME ::= [A-Z]([A-Z][0-9]_)*

Each alias definition is of the form

Alias_Type NAME = item1, item2, ...

where Alias_Type is one of User_Alias, Runas_Alias, Host_Alias, or

Cmnd_Alias. A NAME is a string of uppercase letters, numbers, and under?

score characters (?_?). A NAME must start with an uppercase letter. It

is possible to put several alias definitions of the same type on a single

line, joined by a colon (?:?). E.g.,

Alias_Type NAME = item1, item2, item3 : NAME = item4, item5

It is a syntax error to redefine an existing alias. It is possible to

use the same name for aliases of different types, but this is not recom?

mended.

The definitions of what constitutes a valid alias member follow.

User_List ::= User |

User ',' User_List

User ::= '!'* user name |

'!'* #uid |

'!'* %group |

'!'* %#gid |

'!'* +netgroup |

'!'* %:nonunix_group |

'!'* %:#nonunix_gid |

'!'* User_Alias

A User_List is made up of one or more user names, user-IDs (prefixed with ?#?), system group names and IDs (prefixed with ?%? and ?%#? respec? tively), netgroups (prefixed with ?+?), non-Unix group names and IDs (prefixed with ?%:? and ?%:#? respectively) and User_Aliases. Each list item may be prefixed with zero or more ?!? operators. An odd number of ?!? operators negate the value of the item; an even number just cancel each other out. User netgroups are matched using the user and domain members only; the host member is not used when matching.

A user name, uid, group, gid, netgroup, nonunix_group or nonunix_gid may be enclosed in double quotes to avoid the need for escaping special char? acters. Alternately, special characters may be specified in escaped hex mode, e.g., \x20 for space. When using double quotes, any prefix charac? ters must be included inside the quotes.

The actual nonunix_group and nonunix_gid syntax depends on the underlying group provider plugin. For instance, the QAS AD plugin supports the fol? lowing formats:

? Group in the same domain: "%:Group Name"

? Group in any domain: "%:Group Name@FULLY.QUALIFIED.DOMAIN"

? Group SID: "%:S-1-2-34-5678901234-5678901234-5678901234-567"

See GROUP PROVIDER PLUGINS for more information.

Note that quotes around group names are optional. Unquoted strings must use a backslash (?\?) to escape spaces and special characters. See Other special characters and reserved words for a list of characters that need to be escaped.

Runas_List ::= Runas_Member |

Runas_Member ',' Runas_List

Runas_Member ::= '!'* user name |

'!'* #uid |

'!'* %group |

'!'* %#gid |

'!'* %:nonunix_group |

'!'* %:#nonunix_gid |

'!'* +netgroup |

'!'* Runas_Alias

A Runas_List is similar to a User_List except that instead of User_Aliases it can contain Runas_Aliases. Note that user names and groups are matched as strings. In other words, two users (groups) with the same user (group) ID are considered to be distinct. If you wish to match all user names with the same user-ID (e.g., root and toor), you can use a user-ID instead of a name (#0 in the example given). Note that the user-ID or group-ID specified in a Runas_Member need not be listed in the password or group database.

Host_List ::= Host |

Host ',' Host_List

Host ::= '!'* host name |

'!'* ip_addr |

'!'* network(/netmask)? |

'!'* +netgroup |

'!'* Host_Alias

A Host_List is made up of one or more host names, IP addresses, network numbers, netgroups (prefixed with ?+?) and other aliases. Again, the value of an item may be negated with the ?!? operator. Host netgroups are matched using the host (both qualified and unqualified) and domain members only; the user member is not used when matching. If you specify a network number without a netmask, sudo will query each of the local host's network interfaces and, if the network number corresponds to one of the hosts's network interfaces, will use the netmask of that inter? face. The netmask may be specified either in standard IP address nota? tion (e.g., 255.255.255.0 or ffff:ffff:ffff:ffff::), or CIDR notation (number of bits, e.g., 24 or 64). A host name may include shell-style wildcards (see the Wildcards section below), but unless the host name command on your machine returns the fully qualified host name, you'll need to use the fqdn flag for wildcards to be useful. Note that sudo

only inspects actual network interfaces; this means that IP address
127.0.0.1 (localhost) will never match. Also, the host name ?localhost?
will only match if that is the actual host name, which is usually only
the case for non-networked systems.

```
digest ::= [A-Fa-f0-9]+ |
          [A-Za-z0-9\+/=]+
Digest_Spec ::= "sha224" ':' digest |
               "sha256" ':' digest |
               "sha384" ':' digest |
               "sha512" ':' digest
Digest_List ::= Digest_Spec |
               Digest_Spec ',' Digest_List
Cmnd_List ::= Cmnd |
             Cmnd ',' Cmnd_List
command name ::= file name |
                file name args |
                file name """"
Edit_Spec ::= "sudoedit" file name+
Cmnd ::= Digest_List? '!'* command name |
        '!'* directory |
        '!'* Edit_Spec |
        '!'* Cmnd_Alias
```

A Cmnd_List is a list of one or more command names, directories, and
other aliases. A command name is a fully qualified file name which may
include shell-style wildcards (see the Wildcards section below). A sim?
ple file name allows the user to run the command with any arguments they
wish. However, you may also specify command line arguments (including
wildcards). Alternately, you can specify "" to indicate that the command
may only be run without command line arguments. A directory is a fully
qualified path name ending in a ?/?. When you specify a directory in a
Cmnd_List, the user will be able to run any file within that directory
(but not in any sub-directories therein).

If a Cmnd has associated command line arguments, then the arguments in

the Cmnd must match exactly those given by the user on the command line (or match the wildcards if there are any). Note that the following char? acters must be escaped with a ?\? if they are used in command arguments: ?,?, ?:?, ?=?, ?\?. The built-in command ?sudoedit? is used to permit a user to run sudo with the -e option (or as sudoedit). It may take com? mand line arguments just as a normal command does. Note that ?sudoedit? is a command built into sudo itself and must be specified in the sudoers file without a leading path. If a leading path is present, for example /usr/bin/sudoedit, the path name will be silently converted to ?sudoedit?. A fully-qualified path for sudoedit is treated as an error by visudo.

A command name may be preceded by a Digest_List, a comma-separated list of one or more Digest_Spec entries. If a Digest_List is present, the command will only match successfully if it can be verified using one of the SHA-2 digests in the list. Starting with version 1.9.0, the ALL re? served word can be used in conjunction with a Digest_List. The following digest formats are supported: sha224, sha256, sha384 and sha512. The string may be specified in either hex or base64 format (base64 is more compact). There are several utilities capable of generating SHA-2 di? gests in hex format such as openssl, shasum, sha224sum, sha256sum, sha384sum, sha512sum.

For example, using openssl:

```
$ openssl dgst -sha224 /bin/ls
SHA224(/bin/ls)= 118187da8364d490b4a7debbf483004e8f3e053ec954309de2c41a25
```

It is also possible to use openssl to generate base64 output:

```
$ openssl dgst -binary -sha224 /bin/ls | openssl base64
EYGH2oNk1JC0p9679IMATo8+BT7JVDCd4sQaJQ==
```

Warning, if the user has write access to the command itself (directly or via a sudo command), it may be possible for the user to replace the com? mand after the digest check has been performed but before the command is executed. A similar race condition exists on systems that lack the fexecve(2) system call when the directory in which the command is located is writable by the user. See the description of the fdexec setting for

more information on how sudo executes commands that have an associated digest.

Command digests are only supported by version 1.8.7 or higher.

Defaults

Certain configuration options may be changed from their default values at run-time via one or more Default_Entry lines. These may affect all users on any host, all users on a specific host, a specific user, a specific command, or commands being run as a specific user. Note that per-command entries may not include command line arguments. If you need to specify arguments, define a Cmnd_Alias and reference that instead.

```
Default_Type ::= 'Defaults' |
                 'Defaults' '@' Host_List |
                 'Defaults' ':' User_List |
                 'Defaults' '!' Cmnd_List |
                 'Defaults' '>' Runas_List
Default_Entry ::= Default_Type Parameter_List
Parameter_List ::= Parameter |
                   Parameter ',' Parameter_List
Parameter ::= Parameter '=' Value |
              Parameter '+=' Value |
              Parameter '-=' Value |
              '!'* Parameter
```

Parameters may be flags, integer values, strings, or lists. Flags are implicitly boolean and can be turned off via the ?!? operator. Some in? teger, string and list parameters may also be used in a boolean context to disable them. Values may be enclosed in double quotes ("") when they contain multiple words. Special characters may be escaped with a back? slash (?\?).

Lists have two additional assignment operators, += and -=. These opera? tors are used to add to and delete from a list respectively. It is not an error to use the -= operator to remove an element that does not exist in a list.

Defaults entries are parsed in the following order: generic, host, user

and runas Defaults first, then command defaults. If there are multiple Defaults settings of the same type, the last matching setting is used.

The following Defaults settings are parsed before all others since they may affect subsequent entries: fqdn, group_plugin, runas_default, sudoers_locale.

See SUDOERS OPTIONS for a list of supported Defaults parameters.

User specification

```
User_Spec ::= User_List Host_List '=' Cmnd_Spec_List \
              (':' Host_List '=' Cmnd_Spec_List)*
Cmnd_Spec_List ::= Cmnd_Spec |
                   Cmnd_Spec ',' Cmnd_Spec_List
Cmnd_Spec ::= Runas_Spec? Option_Spec* Tag_Spec* Cmnd
Runas_Spec ::= '(' Runas_List? (':' Runas_List)? ')'
Option_Spec ::= (SELinux_Spec | Date_Spec | Timeout_Spec | Chdir_Spec | Chroot_Spec)
SELinux_Spec ::= ('ROLE=role' | 'TYPE=type')
Date_Spec ::= ('NOTBEFORE=timestamp' | 'NOTAFTER=timestamp')
Timeout_Spec ::= 'TIMEOUT=timeout'
Chdir_Spec ::= 'CWD=directory'
Chroot_Spec ::= 'CHROOT=directory'
Tag_Spec ::= ('EXEC:' | 'NOEXEC:' | 'FOLLOW:' | 'NOFOLLOW' |
              'LOG_INPUT:' | 'NOLOG_INPUT:' | 'LOG_OUTPUT:' |
              'NOLOG_OUTPUT:' | 'MAIL:' | 'NOMAIL:' | 'PASSWD:' |
              'NOPASSWD:' | 'SETENV:' | 'NOSETENV:')
```

A user specification determines which commands a user may run (and as what user) on specified hosts. By default, commands are run as root, but this can be changed on a per-command basis.

The basic structure of a user specification is ?who where = (as_whom) what?. Let's break that down into its constituent parts:

Runas_Spec

A Runas_Spec determines the user and/or the group that a command may be run as. A fully-specified Runas_Spec consists of two Runas_Lists (as de? fined above) separated by a colon (?:?) and enclosed in a set of paren? theses. The first Runas_List indicates which users the command may be

run as via the -u option. The second defines a list of groups that can be specified via the -g option in addition to any of the target user's groups. If both Runas_Lists are specified, the command may be run with any combination of users and groups listed in their respective Runas_Lists. If only the first is specified, the command may be run as any user in the list but no -g option may be specified. If the first Runas_List is empty but the second is specified, the command may be run as the invoking user with the group set to any listed in the Runas_List. If both Runas_Lists are empty, the command may only be run as the invok? ing user. If no Runas_Spec is specified the command may be run as root and no group may be specified.

A Runas_Spec sets the default for the commands that follow it. What this means is that for the entry:

dgb boulder = (operator) /bin/ls, /bin/kill, /usr/bin/lprm

The user dgb may run /bin/ls, /bin/kill, and /usr/bin/lprm on the host boulder?but only as operator. E.g.,

$ sudo -u operator /bin/ls

It is also possible to override a Runas_Spec later on in an entry. If we modify the entry like so:

dgb boulder = (operator) /bin/ls, (root) /bin/kill, /usr/bin/lprm

Then user dgb is now allowed to run /bin/ls as operator, but /bin/kill and /usr/bin/lprm as root.

We can extend this to allow dgb to run /bin/ls with either the user or group set to operator:

dgb boulder = (operator : operator) /bin/ls, (root) /bin/kill,\
/usr/bin/lprm

Note that while the group portion of the Runas_Spec permits the user to run as command with that group, it does not force the user to do so. If no group is specified on the command line, the command will run with the group listed in the target user's password database entry. The following would all be permitted by the sudoers entry above:

$ sudo -u operator /bin/ls

$ sudo -u operator -g operator /bin/ls

$ sudo -g operator /bin/ls

In the following example, user tcm may run commands that access a modem device file with the dialer group.

```
tcm     boulder = (:dialer) /usr/bin/tip, /usr/bin/cu,\
        /usr/local/bin/minicom
```

Note that in this example only the group will be set, the command still runs as user tcm. E.g.

$ sudo -g dialer /usr/bin/cu

Multiple users and groups may be present in a Runas_Spec, in which case the user may select any combination of users and groups via the -u and -g options. In this example:

```
alan    ALL = (root, bin : operator, system) ALL
```

user alan may run any command as either user root or bin, optionally set? ting the group to operator or system.

Option_Spec

A Cmnd may have zero or more options associated with it. Options may consist of SELinux roles and/or types, start and/or end dates and command timeouts. Once an option is set for a Cmnd, subsequent Cmnds in the Cmnd_Spec_List, inherit that option unless it is overridden by another option. Note that the option names are reserved words in sudoers. This means that none of the valid option names (see below) can be used when declaring an alias.

SELinux_Spec

On systems with SELinux support, sudoers file entries may optionally have an SELinux role and/or type associated with a command. If a role or type is specified with the command it will override any default values speci? fied in sudoers. A role or type specified on the command line, however, will supersede the values in sudoers.

Date_Spec

sudoers rules can be specified with a start and end date via the NOTBEFORE and NOTAFTER settings. The time stamp must be specified in Generalized Time as defined by RFC 4517. The format is effectively yyyymmddHHMMSSZ where the minutes and seconds are optional. The ?Z? suf?

fix indicates that the time stamp is in Coordinated Universal Time (UTC).
It is also possible to specify a timezone offset from UTC in hours and minutes instead of a ?Z?. For example, ?-0500? would correspond to East? ern Standard time in the US. As an extension, if no ?Z? or timezone off? set is specified, local time will be used.
The following are all valid time stamps:

20170214083000Z

2017021408Z

20160315220000-0500

20151201235900

Timeout_Spec

A command may have a timeout associated with it. If the timeout expires before the command has exited, the command will be terminated. The time? out may be specified in combinations of days, hours, minutes and seconds with a single-letter case-insensitive suffix that indicates the unit of time. For example, a timeout of 7 days, 8 hours, 30 minutes and 10 sec? onds would be written as 7d8h30m10s. If a number is specified without a unit, seconds are assumed. Any of the days, minutes, hours or seconds may be omitted. The order must be from largest to smallest unit and a unit may not be specified more than once.
The following are all valid timeout values: 7d8h30m10s, 14d, 8h30m, 600s, 3600. The following are invalid timeout values: 12m2w1d, 30s10m4h, 1d2d3h.
This setting is only supported by version 1.8.20 or higher.

Chdir_Spec

The working directory that the command will be run in can be specified using the CWD setting. The directory must be a fully-qualified path name beginning with a ?/? or ?~? character, or the special value ?*?. A value of ?*? indicates that the user may specify the working directory by run? ning sudo with the -D option. By default, commands are run from the in? voking user's current working directory, unless the -i option is given.
Path names of the form ~user/path/name are interpreted as being relative to the named user's home directory. If the user name is omitted, the

path will be relative to the runas user's home directory.

This setting is only supported by version 1.9.3 or higher.

Chroot_Spec

The root directory that the command will be run in can be specified using the CHROOT setting. The directory must be a fully-qualified path name beginning with a ?/? or ?~? character, or the special value ?*?. A value of ?*? indicates that the user may specify the root directory by running sudo with the -R option. This setting can be used to run the command in a chroot(2) ?sandbox? similar to the chroot(8) utility. Path names of the form ~user/path/name are interpreted as being relative to the named user's home directory. If the user name is omitted, the path will be relative to the runas user's home directory.

This setting is only supported by version 1.9.3 or higher.

Tag_Spec

A command may have zero or more tags associated with it. The following tag values are supported: EXEC, NOEXEC, FOLLOW, NOFOLLOW, LOG_INPUT, NOLOG_INPUT, LOG_OUTPUT, NOLOG_OUTPUT, MAIL, NOMAIL, PASSWD, NOPASSWD, SETENV, and NOSETENV. Once a tag is set on a Cmnd, subsequent Cmnds in the Cmnd_Spec_List, inherit the tag unless it is overridden by the oppo? site tag (in other words, PASSWD overrides NOPASSWD and NOEXEC overrides EXEC).

EXEC and NOEXEC

If sudo has been compiled with noexec support and the underlying oper? ating system supports it, the NOEXEC tag can be used to prevent a dy? namically-linked executable from running further commands itself.

In the following example, user aaron may run /usr/bin/more and /usr/bin/vi but shell escapes will be disabled.

aaron shanty = NOEXEC: /usr/bin/more, /usr/bin/vi

See the Preventing shell escapes section below for more details on how NOEXEC works and whether or not it will work on your system.

FOLLOW and NOFOLLOW Starting with version 1.8.15, sudoedit will not open a file that is a symbolic link unless the sudoedit_follow flag is en? abled. The FOLLOW and NOFOLLOW tags override the value of

sudoedit_follow and can be used to permit (or deny) the editing of sym?
bolic links on a per-command basis. These tags are only effective for
the sudoedit command and are ignored for all other commands.

LOG_INPUT and NOLOG_INPUT

These tags override the value of the log_input flag on a per-command
basis. For more information, see the description of log_input in the
SUDOERS OPTIONS section below.

LOG_OUTPUT and NOLOG_OUTPUT

These tags override the value of the log_output flag on a per-command
basis. For more information, see the description of log_output in the
SUDOERS OPTIONS section below.

MAIL and NOMAIL

These tags provide fine-grained control over whether mail will be sent
when a user runs a command by overriding the value of the
mail_all_cmnds flag on a per-command basis. They have no effect when
sudo is run with the -l or -v options. A NOMAIL tag will also override
the mail_always and mail_no_perms options. For more information, see
the descriptions of mail_all_cmnds, mail_always, and mail_no_perms in
the SUDOERS OPTIONS section below.

PASSWD and NOPASSWD

By default, sudo requires that a user authenticate him or herself be?
fore running a command. This behavior can be modified via the NOPASSWD
tag. Like a Runas_Spec, the NOPASSWD tag sets a default for the com?
mands that follow it in the Cmnd_Spec_List. Conversely, the PASSWD tag
can be used to reverse things. For example:

ray rushmore = NOPASSWD: /bin/kill, /bin/ls, /usr/bin/lprm

would allow the user ray to run /bin/kill, /bin/ls, and /usr/bin/lprm
as root on the machine rushmore without authenticating himself. If we
only want ray to be able to run /bin/kill without a password the entry
would be:

ray rushmore = NOPASSWD: /bin/kill, PASSWD: /bin/ls, /usr/bin/lprm

Note, however, that the PASSWD tag has no effect on users who are in
the group specified by the exempt_group setting.

By default, if the NOPASSWD tag is applied to any of a user's entries for the current host, the user will be able to run ?sudo -l? without a password. Additionally, a user may only run ?sudo -v? without a pass? word if all of the user's entries for the current host have the NOPASSWD tag. This behavior may be overridden via the verifypw and listpw options.

SETENV and NOSETENV

These tags override the value of the setenv flag on a per-command ba? sis. Note that if SETENV has been set for a command, the user may dis? able the env_reset flag from the command line via the -E option. Addi? tionally, environment variables set on the command line are not subject to the restrictions imposed by env_check, env_delete, or env_keep. As such, only trusted users should be allowed to set variables in this manner. If the command matched is ALL, the SETENV tag is implied for that command; this default may be overridden by use of the NOSETENV tag.

Wildcards

sudo allows shell-style wildcards (aka meta or glob characters) to be used in host names, path names and command line arguments in the sudoers file. Wildcard matching is done via the glob(3) and fnmatch(3) functions as specified by IEEE Std 1003.1 (?POSIX.1?).

* Matches any set of zero or more characters (including white space).

? Matches any single character (including white space).

[...] Matches any character in the specified range.

[!...] Matches any character not in the specified range.

\x For any character ?x?, evaluates to ?x?. This is used to es? cape special characters such as: ?*?, ???, ?[?, and ?]?.

Note that these are not regular expressions. Unlike a regular expression there is no way to match one or more characters within a range. Character classes may be used if your system's glob(3) and fnmatch(3) functions support them. However, because the ?:? character has special meaning in sudoers, it must be escaped. For example:

/bin/ls [[\:alpha\:]]*

Would match any file name beginning with a letter.

Note that a forward slash (?/?) will not be matched by wildcards used in the file name portion of the command. This is to make a path like:

/usr/bin/*

match /usr/bin/who but not /usr/bin/X11/xterm.

When matching the command line arguments, however, a slash does get matched by wildcards since command line arguments may contain arbitrary strings and not just path names.

Wildcards in command line arguments should be used with care. Command line arguments are matched as a single, concatenated string. This mean a wildcard character such as ??? or ?*? will match across word boundaries, which may be unexpected. For example, while a sudoers entry like:

%operator ALL = /bin/cat /var/log/messages*

will allow command like:

$ sudo cat /var/log/messages.1

It will also allow:

$ sudo cat /var/log/messages /etc/shadow

which is probably not what was intended. In most cases it is better to do command line processing outside of the sudoers file in a scripting language.

Exceptions to wildcard rules

The following exceptions apply to the above rules:

"" If the empty string "" is the only command line argument in the sudoers file entry it means that command is not allowed to be run with any arguments.

sudoedit Command line arguments to the sudoedit built-in command should always be path names, so a forward slash (?/?) will not be matched by a wildcard.

Including other files from within sudoers

It is possible to include other sudoers files from within the sudoers file currently being parsed using the @include and @includedir direc?

tives. For compatibility with sudo versions prior to 1.9.1, #include and #includedir are also accepted.

An include file can be used, for example, to keep a site-wide sudoers file in addition to a local, per-machine file. For the sake of this ex? ample the site-wide sudoers file will be /etc/sudoers and the per-machine one will be /etc/sudoers.local. To include /etc/sudoers.local from within /etc/sudoers one would use the following line in /etc/sudoers:

    @include /etc/sudoers.local

When sudo reaches this line it will suspend processing of the current file (/etc/sudoers) and switch to /etc/sudoers.local. Upon reaching the end of /etc/sudoers.local, the rest of /etc/sudoers will be processed. Files that are included may themselves include other files. A hard limit of 128 nested include files is enforced to prevent include file loops.

The path to the include file may contain white space if it is escaped with a backslash (?\?). Alternately, the entire path may be enclosed in double quotes (""), in which case no escaping is necessary. To include a literal backslash in the path, ?\\? should be used.

If the path to the include file is not fully-qualified (does not begin with a ?/?), it must be located in the same directory as the sudoers file it was included from. For example, if /etc/sudoers contains the line:

    @include sudoers.local

the file that will be included is /etc/sudoers.local.

The file name may also include the %h escape, signifying the short form of the host name. In other words, if the machine's host name is ?xerxes?, then

    @include /etc/sudoers.%h

will cause sudo to include the file /etc/sudoers.xerxes.

The @includedir directive can be used to create a sudoers.d directory that the system package manager can drop sudoers file rules into as part of package installation. For example, given:

    @includedir /etc/sudoers.d

sudo will suspend processing of the current file and read each file in /etc/sudoers.d, skipping file names that end in ?~? or contain a ?.?

character to avoid causing problems with package manager or editor tempo?
rary/backup files. Files are parsed in sorted lexical order. That is,
/etc/sudoers.d/01_first will be parsed before /etc/sudoers.d/10_second.
Be aware that because the sorting is lexical, not numeric,
/etc/sudoers.d/1_whoops would be loaded after /etc/sudoers.d/10_second.
Using a consistent number of leading zeroes in the file names can be used
to avoid such problems. After parsing the files in the directory, con?
trol returns to the file that contained the @includedir directive.
Note that unlike files included via @include, visudo will not edit the
files in a @includedir directory unless one of them contains a syntax er?
ror. It is still possible to run visudo with the -f flag to edit the
files directly, but this will not catch the redefinition of an alias that
is also present in a different file.

Other special characters and reserved words

The pound sign (?#?) is used to indicate a comment (unless it is part of
a #include directive or unless it occurs in the context of a user name
and is followed by one or more digits, in which case it is treated as a
user-ID). Both the comment character and any text after it, up to the
end of the line, are ignored.
The reserved word ALL is a built-in alias that always causes a match to
succeed. It can be used wherever one might otherwise use a Cmnd_Alias,
User_Alias, Runas_Alias, or Host_Alias. Attempting to define an alias
named ALL will result in a syntax error. Please note that using ALL can
be dangerous since in a command context, it allows the user to run any
command on the system.
The following option names permitted in an Option_Spec are also consid?
ered reserved words: CHROOT, ROLE, TYPE, TIMEOUT, CWD, NOTBEFORE and
NOTAFTER. Attempting to define an alias with the same name as one of the
options will result in a syntax error.
An exclamation point (?!?) can be used as a logical not operator in a
list or alias as well as in front of a Cmnd. This allows one to exclude
certain values. For the ?!? operator to be effective, there must be
something for it to exclude. For example, to match all users except for

root one would use:

ALL,!root

If the ALL, is omitted, as in:

!root

it would explicitly deny root but not match any other users. This is different from a true ?negation? operator.

Note, however, that using a ?!? in conjunction with the built-in ALL alias to allow a user to run ?all but a few? commands rarely works as in? tended (see SECURITY NOTES below).

Long lines can be continued with a backslash (?\?) as the last character on the line.

White space between elements in a list as well as special syntactic char? acters in a User Specification (?=?, ?:?, ?(?, ?)?) is optional.

The following characters must be escaped with a backslash (?\?) when used as part of a word (e.g., a user name or host name): ?!?, ?=?, ?:?, ?,?, ?(?, ?)?, ?\?.

SUDOERS OPTIONS

sudo's behavior can be modified by Default_Entry lines, as explained ear? lier. A list of all supported Defaults parameters, grouped by type, are listed below.

Boolean Flags:

always_query_group_plugin

If a group_plugin is configured, use it to resolve groups of the form %group as long as there is not also a system group of the same name. Normally, only groups of the form %:group are passed to the group_plugin. This flag is off by default.

always_set_home If enabled, sudo will set the HOME environment variable to the home directory of the target user (which is the root user unless the -u option is used). This flag is largely obsolete and has no effect unless the env_reset flag has been disabled or HOME is present in the env_keep list, both of which are strongly discouraged.

This flag is off by default.

authenticate If set, users must authenticate themselves via a pass? word (or other means of authentication) before they may run commands. This default may be overridden via the PASSWD and NOPASSWD tags. This flag is on by default.

case_insensitive_group
If enabled, group names in sudoers will be matched in a case insensitive manner. This may be necessary when users are stored in LDAP or AD. This flag is on by de? fault.

case_insensitive_user
If enabled, user names in sudoers will be matched in a case insensitive manner. This may be necessary when groups are stored in LDAP or AD. This flag is on by default.

closefrom_override
If set, the user may use the -C option which overrides the default starting point at which sudo begins closing open file descriptors. This flag is off by default.

compress_io If set, and sudo is configured to log a command's input or output, the I/O logs will be compressed using zlib. This flag is on by default when sudo is compiled with zlib support.

exec_background By default, sudo runs a command as the foreground process as long as sudo itself is running in the fore? ground. When the exec_background flag is enabled and the command is being run in a pseudo-terminal (due to I/O logging or the use_pty flag), the command will be run as a background process. Attempts to read from the controlling terminal (or to change terminal settings) will result in the command being suspended with the SIGTTIN signal (or SIGTTOU in the case of terminal set? tings). If this happens when sudo is a foreground

process, the command will be granted the controlling terminal and resumed in the foreground with no user in? tervention required. The advantage of initially run? ning the command in the background is that sudo need not read from the terminal unless the command explic? itly requests it. Otherwise, any terminal input must be passed to the command, whether it has required it or not (the kernel buffers terminals so it is not possible to tell whether the command really wants the input). This is different from historic sudo behavior or when the command is not being run in a pseudo-terminal. For this to work seamlessly, the operating system must support the automatic restarting of system calls. Un? fortunately, not all operating systems do this by de? fault, and even those that do may have bugs. For exam? ple, macOS fails to restart the tcgetattr() and tcsetattr() system calls (this is a bug in macOS). Furthermore, because this behavior depends on the com? mand stopping with the SIGTTIN or SIGTTOU signals, pro? grams that catch these signals and suspend themselves with a different signal (usually SIGTOP) will not be automatically foregrounded. Some versions of the linux su(1) command behave this way. This flag is off by de? fault.

This setting is only supported by version 1.8.7 or higher. It has no effect unless I/O logging is enabled or the use_pty flag is enabled.

env_editor If set, visudo will use the value of the SUDO_EDITOR, VISUAL or EDITOR environment variables before falling back on the default editor list. Note that visudo is typically run as root so this flag may allow a user with visudo privileges to run arbitrary commands as root without logging. An alternative is to place a

colon-separated list of ?safe? editors int the editor variable. visudo will then only use SUDO_EDITOR, VISUAL or EDITOR if they match a value specified in editor. If the env_reset flag is enabled, the SUDO_EDITOR, VISUAL and/or EDITOR environment variables must be present in the env_keep list for the env_editor flag to function when visudo is invoked via sudo. This flag is on by default.

env_reset If set, sudo will run the command in a minimal environ? ment containing the TERM, PATH, HOME, MAIL, SHELL, LOGNAME, USER and SUDO_* variables. Any variables in the caller's environment or in the file specified by the restricted_env_file setting that match the env_keep and env_check lists are then added, followed by any variables present in the file specified by the env_file setting (if any). The contents of the env_keep and env_check lists, as modified by global Defaults parame? ters in sudoers, are displayed when sudo is run by root with the -V option. If the secure_path setting is en? abled, its value will be used for the PATH environment variable. This flag is on by default.

fast_glob Normally, sudo uses the glob(3) function to do shell-style globbing when matching path names. However, since it accesses the file system, glob(3) can take a long time to complete for some patterns, especially when the pattern references a network file system that is mounted on demand (auto mounted). The fast_glob flag causes sudo to use the fnmatch(3) function, which does not access the file system to do its matching. The disadvantage of fast_glob is that it is unable to match relative path names such as ./ls or ../bin/ls. This has security implications when path names that in? clude globbing characters are used with the negation

operator, ?!?, as such rules can be trivially bypassed. As such, this flag should not be used when the sudoers file contains rules that contain negated path names which include globbing characters. This flag is off by default.

fqdn Set this flag if you want to put fully qualified host names in the sudoers file when the local host name (as returned by the hostname command) does not contain the domain name. In other words, instead of myhost you would use myhost.mydomain.edu. You may still use the short form if you wish (and even mix the two). This flag is only effective when the ?canonical? host name, as returned by the getaddrinfo() or gethostbyname() function, is a fully-qualified domain name. This is usually the case when the system is configured to use DNS for host name resolution.
If the system is configured to use the /etc/hosts file in preference to DNS, the ?canonical? host name may not be fully-qualified. The order that sources are queried for host name resolution is usually specified in the /etc/nsswitch.conf, /etc/netsvc.conf, /etc/host.conf, or, in some cases, /etc/resolv.conf file. In the /etc/hosts file, the first host name of the entry is considered to be the ?canonical? name; subsequent names are aliases that are not used by sudoers. For example, the following hosts file line for the machine ?xyzzy? has the fully-qualified domain name as the ?canonical? host name, and the short version as an alias.

```
192.168.1.1 xyzzy.sudo.ws xyzzy
```

If the machine's hosts file entry is not formatted properly, the fqdn flag will not be effective if it is queried before DNS.
Beware that when using DNS for host name resolution,

turning on fqdn requires sudoers to make DNS lookups which renders sudo unusable if DNS stops working (for example if the machine is disconnected from the net? work). Also note that just like with the hosts file, you must use the ?canonical? name as DNS knows it. That is, you may not use a host alias (CNAME entry) due to performance issues and the fact that there is no way to get all aliases from DNS.

This flag is off by default.

ignore_audit_errors

Allow commands to be run even if sudoers cannot write to the audit log. If enabled, an audit log write fail? ure is not treated as a fatal error. If disabled, a command may only be run after the audit event is suc? cessfully written. This flag is only effective on sys? tems for which sudoers supports audit logging, includ? ing FreeBSD, Linux, macOS and Solaris. This flag is on by default.

ignore_dot If set, sudo will ignore "." or "" (both denoting cur? rent directory) in the PATH environment variable; the PATH itself is not modified. This flag is on by de? fault.

ignore_iolog_errors

Allow commands to be run even if sudoers cannot write to the I/O log (local or remote). If enabled, an I/O log write failure is not treated as a fatal error. If disabled, the command will be terminated if the I/O log cannot be written to. This flag is off by default.

ignore_logfile_errors

Allow commands to be run even if sudoers cannot write to the log file. If enabled, a log file write failure is not treated as a fatal error. If disabled, a com? mand may only be run after the log file entry is suc?

cessfully written. This flag only has an effect when sudoers is configured to use file-based logging via the logfile setting. This flag is on by default.

ignore_local_sudoers

If set via LDAP, parsing of /etc/sudoers will be skipped. This is intended for Enterprises that wish to prevent the usage of local sudoers files so that only LDAP is used. This thwarts the efforts of rogue opera? tors who would attempt to add roles to /etc/sudoers. When this flag is enabled, /etc/sudoers does not even need to exist. Since this flag tells sudo how to be? have when no specific LDAP entries have been matched, this sudoOption is only meaningful for the cn=defaults section. This flag is off by default.

ignore_unknown_defaults

If set, sudo will not produce a warning if it encoun? ters an unknown Defaults entry in the sudoers file or an unknown sudoOption in LDAP. This flag is off by de? fault.

insults If set, sudo will insult users when they enter an in? correct password. This flag is off by default.

log_allowed If set, sudoers will log commands allowed by the policy to the system audit log (where supported) as well as to syslog and/or a log file. This flag is on by default. This setting is only supported by version 1.8.29 or higher.

log_denied If set, sudoers will log commands denied by the policy to the system audit log (where supported) as well as to syslog and/or a log file. This flag is on by default. This setting is only supported by version 1.8.29 or higher.

log_host If set, the host name will be included in log entries written to the file configured by the logfile setting.

This flag is off by default.

log_input If set, sudo will run the command in a pseudo-terminal and log all user input. If the standard input is not connected to the user's tty, due to I/O redirection or because the command is part of a pipeline, that input is also captured and stored in a separate log file. Anything sent to the standard input will be consumed, regardless of whether or not the command run via sudo is actually reading the standard input. This may have unexpected results when using sudo in a shell script that expects to process the standard input. For more information about I/O logging, see the I/O LOG FILES section. This flag is off by default.

log_output If set, sudo will run the command in a pseudo-terminal and log all output that is sent to the screen, similar to the script(1) command. For more information about I/O logging, see the I/O LOG FILES section. This flag is off by default.

log_server_keepalive

If set, sudo will enable the TCP keepalive socket op? tion on the connection to the log server. This enables the periodic transmission of keepalive messages to the server. If the server does not respond to a message, the connection will be closed and the running command will be terminated unless the ignore_iolog_errors flag (I/O logging enabled) or the ignore_log_errors flag (I/O logging disabled) is set. This flag is on by de? fault.

This setting is only supported by version 1.9.0 or higher.

log_server_verify

If set, the server certificate received during the TLS handshake must be valid and it must contain either the

server name (from log_servers) or its IP address. If either of these conditions is not met, the TLS hand? shake will fail. This flag is on by default. This setting is only supported by version 1.9.0 or higher.

log_year If set, the four-digit year will be logged in the (non-syslog) sudo log file. This flag is off by default.

long_otp_prompt When validating with a One Time Password (OTP) scheme such as S/Key or OPIE, a two-line prompt is used to make it easier to cut and paste the challenge to a lo? cal window. It's not as pretty as the default but some people find it more convenient. This flag is off by default.

mail_all_cmnds Send mail to the mailto user every time a user attempts to run a command via sudo (this includes sudoedit). No mail will be sent if the user runs sudo with the -l or -v option unless there is an authentication error and the mail_badpass flag is also set. This flag is off by default.

mail_always Send mail to the mailto user every time a user runs sudo. This flag is off by default.

mail_badpass Send mail to the mailto user if the user running sudo does not enter the correct password. If the command the user is attempting to run is not permitted by sudoers and one of the mail_all_cmnds, mail_always, mail_no_host, mail_no_perms or mail_no_user flags are set, this flag will have no effect. This flag is off by default.

mail_no_host If set, mail will be sent to the mailto user if the in? voking user exists in the sudoers file, but is not al? lowed to run commands on the current host. This flag is off by default.

mail_no_perms If set, mail will be sent to the mailto user if the in?

voking user is allowed to use sudo but the command they are trying is not listed in their sudoers file entry or is explicitly denied. This flag is off by default.

mail_no_user If set, mail will be sent to the mailto user if the in? voking user is not in the sudoers file. This flag is on by default.

match_group_by_gid

By default, sudoers will look up each group the user is a member of by group-ID to determine the group name (this is only done once). The resulting list of the user's group names is used when matching groups listed in the sudoers file. This works well on systems where the number of groups listed in the sudoers file is larger than the number of groups a typical user belongs to. On systems where group lookups are slow, where users may belong to a large number of groups, and where the number of groups listed in the sudoers file is rel? atively small, it may be prohibitively expensive and running commands via sudo may take longer than normal. On such systems it may be faster to use the match_group_by_gid flag to avoid resolving the user's group-IDs to group names. In this case, sudoers must look up any group name listed in the sudoers file and use the group-ID instead of the group name when deter? mining whether the user is a member of the group. Note that if match_group_by_gid is enabled, group data? base lookups performed by sudoers will be keyed by group name as opposed to group-ID. On systems where there are multiple sources for the group database, it is possible to have conflicting group names or group-IDs in the local /etc/group file and the remote group database. On such systems, enabling or disabling match_group_by_gid can be used to choose whether group

database queries are performed by name (enabled) or ID (disabled), which may aid in working around group entry conflicts.
The match_group_by_gid flag has no effect when sudoers data is stored in LDAP. This flag is off by default.
This setting is only supported by version 1.8.18 or higher.

netgroup_tuple If set, netgroup lookups will be performed using the full netgroup tuple: host name, user name and domain (if one is set). Historically, sudo only matched the user name and domain for netgroups used in a User_List and only matched the host name and domain for netgroups used in a Host_List. This flag is off by default.

noexec If set, all commands run via sudo will behave as if the NOEXEC tag has been set, unless overridden by an EXEC tag. See the description of EXEC and NOEXEC above as well as the Preventing shell escapes section at the end of this manual. This flag is off by default.

pam_acct_mgmt On systems that use PAM for authentication, sudo will perform PAM account validation for the invoking user by default. The actual checks performed depend on which PAM modules are configured. If enabled, account vali? dation will be performed regardless of whether or not a password is required. This flag is on by default.
This setting is only supported by version 1.8.28 or higher.

pam_rhost On systems that use PAM for authentication, sudo will set the PAM remote host value to the name of the local host when the pam_rhost flag is enabled. On Linux sys? tems, enabling pam_rhost may result in DNS lookups of the local host name when PAM is initialized. On So? laris versions prior to Solaris 8, pam_rhost must be enabled if pam_ruser is also enabled to avoid a crash

in the Solaris PAM implementation.
This flag is off by default on systems other than So?
laris.
This setting is only supported by version 1.9.0 or
higher.

pam_ruser On systems that use PAM for authentication, sudo will set the PAM remote user value to the name of the user that invoked sudo when the pam_ruser flag is enabled.
This flag is on by default.
This setting is only supported by version 1.9.0 or
higher.

pam_session On systems that use PAM for authentication, sudo will create a new PAM session for the command to be run in.
Unless sudo is given the -i or -s options, PAM session modules are run with the ?silent? flag enabled. This prevents last login information from being displayed for every command on some systems. Disabling pam_session may be needed on older PAM implementations or on operating systems where opening a PAM session changes the utmp or wtmp files. If PAM session support is disabled, resource limits may not be updated for the command being run. If pam_session, pam_setcred, and use_pty are disabled, log_servers has not been set and I/O logging has not been configured, sudo will execute the command directly instead of running it as a child process. This flag is on by default.
This setting is only supported by version 1.8.7 or
higher.

pam_setcred On systems that use PAM for authentication, sudo will attempt to establish credentials for the target user by default, if supported by the underlying authentication system. One example of a credential is a Kerberos ticket. If pam_session, pam_setcred, and use_pty are

disabled, log_servers has not been set and I/O logging has not been configured, sudo will execute the command directly instead of running it as a child process. This flag is on by default. This setting is only supported by version 1.8.8 or higher.

passprompt_override
If set, the prompt specified by passprompt or the SUDO_PROMPT environment variable will always be used and will replace the prompt provided by a PAM module or other authentication method. This flag is off by de? fault.

path_info Normally, sudo will tell the user when a command could not be found in their PATH environment variable. Some sites may wish to disable this as it could be used to gather information on the location of executables that the normal user does not have access to. The disadvan? tage is that if the executable is simply not in the user's PATH, sudo will tell the user that they are not allowed to run it, which can be confusing. This flag is on by default.

preserve_groups By default, sudo will initialize the group vector to the list of groups the target user is in. When preserve_groups is set, the user's existing group vec? tor is left unaltered. The real and effective group-IDs, however, are still set to match the target user. This flag is off by default.

pwfeedback By default, sudo reads the password like most other Unix programs, by turning off echo until the user hits the return (or enter) key. Some users become confused by this as it appears to them that sudo has hung at this point. When pwfeedback is set, sudo will provide visual feedback when the user presses a key. Note that

this does have a security impact as an onlooker may be able to determine the length of the password being en? tered. This flag is off by default.

requiretty If set, sudo will only run when the user is logged in to a real tty. When this flag is set, sudo can only be run from a login session and not via other means such as cron(8) or cgi-bin scripts. This flag is off by de? fault.

root_sudo If set, root is allowed to run sudo too. Disabling this prevents users from ?chaining? sudo commands to get a root shell by doing something like ?sudo sudo /bin/sh?. Note, however, that turning off root_sudo will also prevent root from running sudoedit. Dis? abling root_sudo provides no real additional security; it exists purely for historical reasons. This flag is on by default.

rootpw If set, sudo will prompt for the root password instead of the password of the invoking user when running a command or editing a file. This flag is off by de? fault.

runas_allow_unknown_id

If enabled, allow matching of runas user and group IDs that are not present in the password or group data? bases. In addition to explicitly matching unknown user or group IDs in a Runas_List, this option also allows the ALL alias to match unknown IDs. This flag is off by default.

This setting is only supported by version 1.8.30 or higher. Older versions of sudo always allowed matching of unknown user and group IDs.

runas_check_shell

If enabled, sudo will only run commands as a user whose shell appears in the /etc/shells file, even if the in?

voking user's Runas_List would otherwise permit it. If no /etc/shells file is present, a system-dependent list of built-in default shells is used. On many operating systems, system users such as ?bin?, do not have a valid shell and this flag can be used to prevent com? mands from being run as those users. This flag is off by default.
This setting is only supported by version 1.8.30 or higher.

runaspw If set, sudo will prompt for the password of the user defined by the runas_default option (defaults to root) instead of the password of the invoking user when run? ning a command or editing a file. This flag is off by default.

selinux If enabled, the user may specify an SELinux role and/or type to use when running the command, as permitted by the SELinux policy. If SELinux is disabled on the sys? tem, this flag has no effect. This flag is on by de? fault.

set_home If enabled and sudo is invoked with the -s option, the HOME environment variable will be set to the home di? rectory of the target user (which is the root user un? less the -u option is used). This flag is largely ob? solete and has no effect unless the env_reset flag has been disabled or HOME is present in the env_keep list, both of which are strongly discouraged. This flag is off by default.

set_logname Normally, sudo will set the LOGNAME and USER environ? ment variables to the name of the target user (usually root unless the -u option is given). However, since some programs (including the RCS revision control sys? tem) use LOGNAME to determine the real identity of the user, it may be desirable to change this behavior.

This can be done by negating the set_logname option. Note that set_logname will have no effect if the env_reset option has not been disabled and the env_keep list contains LOGNAME or USER. This flag is on by de? fault.

set_utmp When enabled, sudo will create an entry in the utmp (or utmpx) file when a pseudo-terminal is allocated. A pseudo-terminal is allocated by sudo when it is running in a terminal and one or more of the log_input, log_output or use_pty flags is enabled. By default, the new entry will be a copy of the user's existing utmp entry (if any), with the tty, time, type and pid fields updated. This flag is on by default.

setenv Allow the user to disable the env_reset option from the command line via the -E option. Additionally, environ? ment variables set via the command line are not subject to the restrictions imposed by env_check, env_delete, or env_keep. As such, only trusted users should be al? lowed to set variables in this manner. This flag is off by default.

shell_noargs If set and sudo is invoked with no arguments it acts as if the -s option had been given. That is, it runs a shell as root (the shell is determined by the SHELL en? vironment variable if it is set, falling back on the shell listed in the invoking user's /etc/passwd entry if not). This flag is off by default.

stay_setuid Normally, when sudo executes a command the real and ef? fective UIDs are set to the target user (root by de? fault). This option changes that behavior such that the real UID is left as the invoking user's UID. In other words, this makes sudo act as a set-user-ID wrap? per. This can be useful on systems that disable some potentially dangerous functionality when a program is

run set-user-ID. This option is only effective on sys? tems that support either the setreuid(2) or setresuid(2) system call. This flag is off by default.

sudoedit_checkdir

If set, sudoedit will check all directory components of the path to be edited for writability by the invoking user. Symbolic links will not be followed in writable directories and sudoedit will refuse to edit a file lo? cated in a writable directory. These restrictions are not enforced when sudoedit is run by root. On some systems, if all directory components of the path to be edited are not readable by the target user, sudoedit will be unable to edit the file. This flag is on by default.

This setting was first introduced in version 1.8.15 but initially suffered from a race condition. The check for symbolic links in writable intermediate directories was added in version 1.8.16.

sudoedit_follow By default, sudoedit will not follow symbolic links when opening files. The sudoedit_follow option can be enabled to allow sudoedit to open symbolic links. It may be overridden on a per-command basis by the FOLLOW and NOFOLLOW tags. This flag is off by default.

This setting is only supported by version 1.8.15 or higher.

syslog_pid When logging via syslog(3), include the process ID in the log entry. This flag is off by default.

This setting is only supported by version 1.8.21 or higher.

targetpw If set, sudo will prompt for the password of the user specified by the -u option (defaults to root) instead of the password of the invoking user when running a command or editing a file. Note that this flag pre?

cludes the use of a user-ID not listed in the passwd database as an argument to the -u option. This flag is off by default.

tty_tickets If set, users must authenticate on a per-tty basis. With this flag enabled, sudo will use a separate record in the time stamp file for each terminal. If disabled, a single record is used for all login sessions. This option has been superseded by the timestamp_type option.

umask_override If set, sudo will set the umask as specified in the sudoers file without modification. This makes it pos? sible to specify a umask in the sudoers file that is more permissive than the user's own umask and matches historical behavior. If umask_override is not set, sudo will set the umask to be the union of the user's umask and what is specified in sudoers. This flag is off by default.

use_netgroups If set, netgroups (prefixed with ?+?), may be used in place of a user or host. For LDAP-based sudoers, net? group support requires an expensive sub-string match on the server unless the NETGROUP_BASE directive is present in the /etc/sudo-ldap.conf file. If netgroups are not needed, this option can be disabled to reduce the load on the LDAP server. This flag is on by de? fault.

use_pty If set, and sudo is running in a terminal, the command will be run in a pseudo-terminal (even if no I/O log? ging is being done). If the sudo process is not at? tached to a terminal, use_pty has no effect. A malicious program run under sudo may be capable of injecting commands into the user's terminal or running a background process that retains access to the user's terminal device even after the main program has fin?

ished executing. By running the command in a separate pseudo-terminal, this attack is no longer possible. This flag is off by default.

user_command_timeouts
If set, the user may specify a timeout on the command line. If the timeout expires before the command has exited, the command will be terminated. If a timeout is specified both in the sudoers file and on the com? mand line, the smaller of the two timeouts will be used. See the Timeout_Spec section for a description of the timeout syntax. This flag is off by default. This setting is only supported by version 1.8.20 or higher.

utmp_runas If set, sudo will store the name of the runas user when updating the utmp (or utmpx) file. By default, sudo stores the name of the invoking user. This flag is off by default.

visiblepw By default, sudo will refuse to run if the user must enter a password but it is not possible to disable echo on the terminal. If the visiblepw flag is set, sudo will prompt for a password even when it would be visi? ble on the screen. This makes it possible to run things like ?ssh somehost sudo ls? since by default, ssh(1) does not allocate a tty when running a command. This flag is off by default.

Integers:

closefrom Before it executes a command, sudo will close all open file descriptors other than standard input, standard output and standard error (ie: file descriptors 0-2). The closefrom option can be used to specify a different file descriptor at which to start closing. The default is 3.

command_timeout The maximum amount of time a command is allowed to run

before it is terminated. See the Timeout_Spec section for a description of the timeout syntax.
This setting is only supported by version 1.8.20 or higher.

log_server_timeout
The maximum amount of time to wait when connecting to a log server or waiting for a server response. See the Timeout_Spec section for a description of the timeout syntax. The default value is 30 seconds.
This setting is only supported by version 1.9.0 or higher.

maxseq The maximum sequence number that will be substituted for the ?%{seq}? escape in the I/O log file (see the iolog_dir description below for more information).
While the value substituted for ?%{seq}? is in base 36, maxseq itself should be expressed in decimal. Values larger than 2176782336 (which corresponds to the base 36 sequence number ?ZZZZZZ?) will be silently truncated to 2176782336. The default value is 2176782336.
Once the local sequence number reaches the value of maxseq, it will ?roll over? to zero, after which sudoers will truncate and re-use any existing I/O log path names.
This setting is only supported by version 1.8.7 or higher.

passwd_tries The number of tries a user gets to enter his/her pass? word before sudo logs the failure and exits. The de? fault is 3.

syslog_maxlen On many systems, syslog(3) has a relatively small log buffer. IETF RFC 5424 states that syslog servers must support messages of at least 480 bytes and should sup? port messages up to 2048 bytes. By default, sudoers creates log messages up to 980 bytes which corresponds

to the historic BSD syslog implementation which used a 1024 byte buffer to store the message, date, hostname and program name. To prevent syslog messages from be? ing truncated, sudoers will split up log messages that are larger than syslog_maxlen bytes. When a message is split, additional parts will include the string ?(command continued)? after the user name and before the continued command line arguments.
This setting is only supported by version 1.8.19 or higher.

Integers that can be used in a boolean context:

loglinelen Number of characters per line for the file log. This value is used to decide when to wrap lines for nicer log files. This has no effect on the syslog log file, only the file log. The default is 80 (use 0 or negate the option to disable word wrap).

passwd_timeout Number of minutes before the sudo password prompt times out, or 0 for no timeout. The timeout may include a fractional component if minute granularity is insuffi? cient, for example 2.5. The default is 5.

timestamp_timeout
Number of minutes that can elapse before sudo will ask for a passwd again. The timeout may include a frac? tional component if minute granularity is insufficient, for example 2.5. The default is 5. Set this to 0 to always prompt for a password. If set to a value less than 0 the user's time stamp will not expire until the system is rebooted. This can be used to allow users to create or delete their own time stamps via ?sudo -v? and ?sudo -k? respectively.

umask File mode creation mask to use when running the com? mand. Negate this option or set it to 0777 to prevent sudoers from changing the umask. Unless the

umask_override flag is set, the actual umask will be the union of the user's umask and the value of the umask setting, which defaults to 0022. This guarantees that sudo never lowers the umask when running a com? mand.

If umask is explicitly set in sudoers, it will override any umask setting in PAM or login.conf. If umask is not set in sudoers, the umask specified by PAM or lo? gin.conf will take precedence. The umask setting in PAM is not used for sudoedit, which does not create a new PAM session.

Strings:

authfail_message Message that is displayed after a user fails to authen? ticate. The message may include the ?%d? escape which will expand to the number of failed password attempts. If set, it overrides the default message, %d incorrect password attempt(s).

badpass_message Message that is displayed if a user enters an incorrect password. The default is Sorry, try again. unless in? sults are enabled.

editor A colon (?:?) separated list of editors path names used by sudoedit and visudo. For sudoedit, this list is used to find an editor when none of the SUDO_EDITOR, VISUAL or EDITOR environment variables are set to an editor that exists and is executable. For visudo, it is used as a white list of allowed editors; visudo will choose the editor that matches the user's SUDO_EDITOR, VISUAL or EDITOR environment variable if possible, or the first editor in the list that exists and is exe? cutable if not. Unless invoked as sudoedit, sudo does not preserve the SUDO_EDITOR, VISUAL or EDITOR environ? ment variables unless they are present in the env_keep list or the env_reset option is disabled. The default

is /bin/vi.

iolog_dir The top-level directory to use when constructing the path name for the input/output log directory. Only used if the log_input or log_output options are enabled or when the LOG_INPUT or LOG_OUTPUT tags are present for a command. The session sequence number, if any, is stored in the directory. The default is /var/log/sudo-io.

The following percent (?%?) escape sequences are sup? ported:

%{seq}
expanded to a monotonically increasing base-36 sequence number, such as 0100A5, where every two digits are used to form a new directory, e.g., 01/00/A5

%{user}
expanded to the invoking user's login name

%{group}
expanded to the name of the invoking user's real group-ID

%{runas_user}
expanded to the login name of the user the com? mand will be run as (e.g., root)

%{runas_group}
expanded to the group name of the user the com? mand will be run as (e.g., wheel)

%{hostname}
expanded to the local host name without the do? main name

%{command}
expanded to the base name of the command being run

In addition, any escape sequences supported by the sys?

tem's strftime(3) function will be expanded.
To include a literal ?%? character, the string ?%%? should be used.

iolog_file The path name, relative to iolog_dir, in which to store input/output logs when the log_input or log_output op? tions are enabled or when the LOG_INPUT or LOG_OUTPUT tags are present for a command. Note that iolog_file may contain directory components. The default is ?%{seq}?.
See the iolog_dir option above for a list of supported percent (?%?) escape sequences.
In addition to the escape sequences, path names that end in six or more Xs will have the Xs replaced with a unique combination of digits and letters, similar to the mktemp(3) function.
If the path created by concatenating iolog_dir and iolog_file already exists, the existing I/O log file will be truncated and overwritten unless iolog_file ends in six or more Xs.

iolog_flush If set, sudo will flush I/O log data to disk after each write instead of buffering it. This makes it possible to view the logs in real-time as the program is execut? ing but may significantly reduce the effectiveness of I/O log compression. This flag is off by default.
This setting is only supported by version 1.8.20 or higher.

iolog_group The group name to look up when setting the group-ID on new I/O log files and directories. If iolog_group is not set, the primary group-ID of the user specified by iolog_user is used. If neither iolog_group nor iolog_user are set, I/O log files and directories are created with group-ID 0.
This setting is only supported by version 1.8.19 or

higher.

iolog_mode The file mode to use when creating I/O log files. Mode bits for read and write permissions for owner, group or other are honored, everything else is ignored. The file permissions will always include the owner read and write bits, even if they are not present in the speci? fied mode. When creating I/O log directories, search (execute) bits are added to match the read and write bits specified by iolog_mode. Defaults to 0600 (read and write by user only).
This setting is only supported by version 1.8.19 or higher.

iolog_user The user name to look up when setting the user and group-IDs on new I/O log files and directories. If iolog_group is set, it will be used instead of the user's primary group-ID. By default, I/O log files and directories are created with user and group-ID 0.
This setting can be useful when the I/O logs are stored on a Network File System (NFS) share. Having a dedi? cated user own the I/O log files means that sudoers does not write to the log files as user-ID 0, which is usually not permitted by NFS.
This setting is only supported by version 1.8.19 or higher.

lecture_status_dir
The directory in which sudo stores per-user lecture status files. Once a user has received the lecture, a zero-length file is created in this directory so that sudo will not lecture the user again. This directory should not be cleared when the system reboots. The de? fault is /var/db/sudo/lectured.

log_server_cabundle
The path to a certificate authority bundle file, in PEM

format, to use instead of the system's default certifi? cate authority database when authenticating the log server. The default is to use the system's default certificate authority database. This setting has no effect unless log_servers is set and the remote log server is secured with TLS.
This setting is only supported by version 1.9.0 or higher.

log_server_peer_cert
The path to the client's certificate file, in PEM for? mat. This setting is required when log_servers is set and the remote log server is secured with TLS.
This setting is only supported by version 1.9.0 or higher.

log_server_peer_key
The path to the client's private key file, in PEM for? mat. This setting is required when log_servers is set and the remote log server is secured with TLS.
This setting is only supported by version 1.9.0 or higher.

mailsub Subject of the mail sent to the mailto user. The es? cape %h will expand to the host name of the machine. Default is ?*** SECURITY information for %h ***?.

noexec_file As of sudo version 1.8.1 this option is no longer sup? ported. The path to the noexec file should now be set in the sudo.conf(5) file.

pam_login_service
On systems that use PAM for authentication, this is the service name used when the -i option is specified. The default value is ?sudo-i?. See the description of pam_service for more information.
This setting is only supported by version 1.8.8 or higher.

pam_service On systems that use PAM for authentication, the service name specifies the PAM policy to apply. This usually corresponds to an entry in the pam.conf file or a file in the /etc/pam.d directory. The default value is ?sudo?.
This setting is only supported by version 1.8.8 or higher.

passprompt The default prompt to use when asking for a password; can be overridden via the -p option or the SUDO_PROMPT environment variable. The following percent (?%?) es? cape sequences are supported:
%H expanded to the local host name including the do? main name (only if the machine's host name is fully qualified or the fqdn option is set)
%h expanded to the local host name without the do? main name
%p expanded to the user whose password is being asked for (respects the rootpw, targetpw and runaspw flags in sudoers)
%U expanded to the login name of the user the com? mand will be run as (defaults to root)
%u expanded to the invoking user's login name
%% two consecutive % characters are collapsed into a single % character
On systems that use PAM for authentication, passprompt will only be used if the prompt provided by the PAM module matches the string ?Password: ? or ?username's Password: ?. This ensures that the passprompt setting does not interfere with challenge-response style au? thentication. The passprompt_override flag can be used to change this behavior.
The default value is ?[sudo] password for %p: ?.

role The default SELinux role to use when constructing a new

security context to run the command. The default role may be overridden on a per-command basis in the sudoers file or via command line options. This option is only available when sudo is built with SELinux support.

runas_default The default user to run commands as if the -u option is not specified on the command line. This defaults to root.

sudoers_locale Locale to use when parsing the sudoers file, logging commands, and sending email. Note that changing the locale may affect how sudoers is interpreted. Defaults to ?C?.

timestamp_type sudoers uses per-user time stamp files for credential caching. The timestamp_type option can be used to specify the type of time stamp record used. It has the following possible values:

global A single time stamp record is used for all of a user's login sessions, regardless of the termi? nal or parent process ID. An additional record is used to serialize password prompts when sudo is used multiple times in a pipeline, but this does not affect authentication.

ppid A single time stamp record is used for all pro? cesses with the same parent process ID (usually the shell). Commands run from the same shell (or other common parent process) will not re? quire a password for timestamp_timeout minutes (5 by default). Commands run via sudo with a different parent process ID, for example from a shell script, will be authenticated separately.

tty One time stamp record is used for each termi? nal, which means that a user's login sessions are authenticated separately. If no terminal is present, the behavior is the same as ppid.

Commands run from the same terminal will not require a password for timestamp_timeout min? utes (5 by default).

kernel The time stamp is stored in the kernel as an attribute of the terminal device. If no termi? nal is present, the behavior is the same as ppid. Negative timestamp_timeout values are not supported and positive values are limited to a maximum of 60 minutes. This is currently only supported on OpenBSD.

The default value is tty.

This setting is only supported by version 1.8.21 or higher.

timestampdir The directory in which sudo stores its time stamp files. This directory should be cleared when the sys? tem reboots. The default is /run/sudo/ts.

timestampowner The owner of the lecture status directory, time stamp directory and all files stored therein. The default is root.

type The default SELinux type to use when constructing a new security context to run the command. The default type may be overridden on a per-command basis in the sudoers file or via command line options. This option is only available when sudo is built with SELinux support.

Strings that can be used in a boolean context:

env_file The env_file option specifies the fully qualified path to a file containing variables to be set in the environment of the program being run. Entries in this file should either be of the form ?VARIABLE=value? or ?export VARIABLE=value?. The value may optionally be enclosed in single or double quotes. Variables in this file are only added if the vari? able does not already exist in the environment. This file is considered to be part of the security policy, its con?

tents are not subject to other sudo environment restric?
tions such as env_keep and env_check.

exempt_group Users in this group are exempt from password and PATH re?
quirements. The group name specified should not include a
% prefix. This is not set by default.

fdexec Determines whether sudo will execute a command by its path
or by an open file descriptor. It has the following possi?
ble values:
always Always execute by file descriptor.
never Never execute by file descriptor.
digest_only
Only execute by file descriptor if the command has
an associated digest in the sudoers file.
The default value is digest_only. This avoids a time of
check versus time of use race condition when the command is
located in a directory writable by the invoking user.
Note that fdexec will change the first element of the argu?
ment vector for scripts ($0 in the shell) due to the way
the kernel runs script interpreters. Instead of being a
normal path, it will refer to a file descriptor. For exam?
ple, /dev/fd/4 on Solaris and /proc/self/fd/4 on Linux. A
workaround is to use the SUDO_COMMAND environment variable
instead.
The fdexec setting is only used when the command is matched
by path name. It has no effect if the command is matched
by the built-in ALL alias.
This setting is only supported by version 1.8.20 or higher.
If the operating system does not support the fexecve(2)
system call, this setting has no effect.

group_plugin A string containing a sudoers group plugin with optional
arguments. The string should consist of the plugin path,
either fully-qualified or relative to the /usr/libexec/sudo
directory, followed by any configuration arguments the

plugin requires. These arguments (if any) will be passed to the plugin's initialization function. If arguments are present, the string must be enclosed in double quotes (""). For more information see GROUP PROVIDER PLUGINS.

lecture This option controls when a short lecture will be printed along with the password prompt. It has the following pos? sible values:

always Always lecture the user.

never Never lecture the user.

once Only lecture the user the first time they run sudo.

If no value is specified, a value of once is implied. Negating the option results in a value of never being used. The default value is once.

lecture_file Path to a file containing an alternate sudo lecture that will be used in place of the standard lecture if the named file exists. By default, sudo uses a built-in lecture.

listpw This option controls when a password will be required when a user runs sudo with the -l option. It has the following possible values:

all All the user's sudoers file entries for the cur? rent host must have the NOPASSWD flag set to avoid entering a password.

always The user must always enter a password to use the -l option.

any At least one of the user's sudoers file entries for the current host must have the NOPASSWD flag set to avoid entering a password.

never The user need never enter a password to use the -l option.

If no value is specified, a value of any is implied. Negating the option results in a value of never being used. The default value is any.

log_format The event log format. Supported log formats are:

json Logs in JSON format. JSON log entries contain the full user details as well as the execution environment if the command was allowed. Due to limitations of the protocol, JSON events sent via syslog may be truncated.

sudo Traditional sudo-style logs, see LOG FORMAT for a description of the log file format.

This setting affects logs sent via syslog(3) as well as the file specified by the logfile setting, if any. The default value is sudo.

logfile Path to the sudo log file (not the syslog log file). Set? ting a path turns on logging to a file; negating this op? tion turns it off. By default, sudo logs via syslog.

mailerflags Flags to use when invoking mailer. Defaults to -t.

mailerpath Path to mail program used to send warning mail. Defaults to the path to sendmail found at configure time.

mailfrom Address to use for the ?from? address when sending warning and error mail. The address should be enclosed in double quotes ("") to protect against sudo interpreting the @ sign. Defaults to the name of the user running sudo.

mailto Address to send warning and error mail to. The address should be enclosed in double quotes ("") to protect against sudo interpreting the @ sign. Defaults to root.

restricted_env_file

The restricted_env_file option specifies the fully quali? fied path to a file containing variables to be set in the environment of the program being run. Entries in this file should either be of the form ?VARIABLE=value? or ?export VARIABLE=value?. The value may optionally be enclosed in single or double quotes. Variables in this file are only added if the variable does not already exist in the envi? ronment. Unlike env_file, the file's contents are not trusted and are processed in a manner similar to that of

the invoking user's environment. If env_reset is enabled, variables in the file will only be added if they are matched by either the env_check or env_keep list. If env_reset is disabled, variables in the file are added as long as they are not matched by the env_delete list. In either case, the contents of restricted_env_file are pro? cessed before the contents of env_file.

runchroot If set, sudo will use this value for the root directory when running a command. The special value ?*? will allow the user to specify the root directory via sudo's -R op? tion. See the Chroot_Spec section for more details. It is only possible to use runchroot as a command-specific Defaults setting if the command exists with the same path both inside and outside the chroot jail. This restriction does not apply to generic, host or user-based Defaults set? tings or to a Cmnd_Spec that includes a Chroot_Spec. This setting is only supported by version 1.9.3 or higher.

runcwd If set, sudo will use this value for the working directory when running a command. The special value ?*? will allow the user to specify the working directory via sudo's -D op? tion. See the Chdir_Spec section for more details. This setting is only supported by version 1.9.3 or higher.

secure_path If set, sudo will use this value in place of the user's PATH environment variable. This option can be used to re? set the PATH to a known good value that contains directo? ries for system administrator commands such as /usr/sbin. Users in the group specified by the exempt_group option are not affected by secure_path. This option is not set by de? fault.

syslog Syslog facility if syslog is being used for logging (negate to disable syslog logging). Defaults to authpriv. The following syslog facilities are supported: authpriv (if your OS supports it), auth, daemon, user, local0, local1,

local2, local3, local4, local5, local6, and local7.

syslog_badpri

Syslog priority to use when the user is not allowed to run a command or when authentication is unsuccessful. Defaults to alert.

The following syslog priorities are supported: alert, crit, debug, emerg, err, info, notice, warning, and none. Negat? ing the option or setting it to a value of none will dis? able logging of unsuccessful commands.

syslog_goodpri

Syslog priority to use when the user is allowed to run a command and authentication is successful. Defaults to notice.

See syslog_badpri for the list of supported syslog priori? ties. Negating the option or setting it to a value of none will disable logging of successful commands.

verifypw This option controls when a password will be required when a user runs sudo with the -v option. It has the following possible values:

all All the user's sudoers file entries for the current host must have the NOPASSWD flag set to avoid en? tering a password.

always The user must always enter a password to use the -v option.

any At least one of the user's sudoers file entries for the current host must have the NOPASSWD flag set to avoid entering a password.

never The user need never enter a password to use the -v option.

If no value is specified, a value of all is implied.

Negating the option results in a value of never being used.

The default value is all.

Lists that can be used in a boolean context:

env_check Environment variables to be removed from the user's en?
vironment unless they are considered ?safe?. For all
variables except TZ, ?safe? means that the variable's
value does not contain any ?%? or ?/? characters. This
can be used to guard against printf-style format vul?
nerabilities in poorly-written programs. The TZ vari?
able is considered unsafe if any of the following are
true:

? It consists of a fully-qualified path name, option?
ally prefixed with a colon (?:?), that does not
match the location of the zoneinfo directory.

? It contains a .. path element.

? It contains white space or non-printable characters.

? It is longer than the value of PATH_MAX.

The argument may be a double-quoted, space-separated
list or a single value without double-quotes. The list
can be replaced, added to, deleted from, or disabled by
using the =, +=, -=, and ! operators respectively. Re?
gardless of whether the env_reset option is enabled or
disabled, variables specified by env_check will be pre?
served in the environment if they pass the aforemen?
tioned check. The global list of environment variables
to check is displayed when sudo is run by root with the
-V option.

env_delete Environment variables to be removed from the user's en?
vironment when the env_reset option is not in effect.
The argument may be a double-quoted, space-separated
list or a single value without double-quotes. The list
can be replaced, added to, deleted from, or disabled by
using the =, +=, -=, and ! operators respectively. The
global list of environment variables to remove is dis?
played when sudo is run by root with the -V option.
Note that many operating systems will remove poten?

tially dangerous variables from the environment of any set-user-ID process (such as sudo).

env_keep Environment variables to be preserved in the user's environment when the env_reset option is in effect. This allows fine-grained control over the environment sudo-spawned processes will receive. The argument may be a double-quoted, space-separated list or a single value without double-quotes. The list can be replaced, added to, deleted from, or disabled by using the =, +=, -=, and ! operators respectively. The global list of variables to keep is displayed when sudo is run by root with the -V option.
Preserving the HOME environment variable has security implications since many programs use it when searching for configuration or data files. Adding HOME to env_keep may enable a user to run unrestricted commands via sudo and is strongly discouraged. Users wishing to edit files with sudo should run sudoedit (or sudo -e) to get their accustomed editor configuration instead of invoking the editor directly.

log_servers A list of one or more servers to use for remote event and I/O log storage, separated by white space. Log servers must be running sudo_logsrvd or another service that implements the protocol described by sudo_logsrv.proto(5).
Server addresses should be of the form ?host[:port][(tls)]?. The host portion may be a host name, an IPv4 address, or an IPv6 address in square brackets.
If the optional tls flag is present, the connection will be secured with Transport Layer Security (TLS) version 1.2 or 1.3. Versions of TLS prior to 1.2 are not supported.

If a port is specified, it may either be a port number or a well-known service name as defined by the system service name database. If no port is specified, port 30343 will be used for plaintext connections and port 30344 will be used for TLS connections.

When log_servers is set, event log data will be logged both locally (see the syslog and log_file settings) as well as remotely, but I/O log data will only be logged remotely. If multiple hosts are specified, they will be attempted in reverse order. If no log servers are available, the user will not be able to run a command unless either the ignore_iolog_errors flag (I/O logging enabled) or the ignore_log_errors flag (I/O logging disabled) is set. Likewise, if the connection to the log server is interrupted while sudo is running, the command will be terminated unless the ignore_iolog_errors flag (I/O logging enabled) or the ignore_log_errors flag (I/O logging disabled) is set. This setting is only supported by version 1.9.0 or higher.

GROUP PROVIDER PLUGINS

The sudoers plugin supports its own plugin interface to allow non-Unix group lookups which can query a group source other than the standard Unix group database. This can be used to implement support for the nonunix_group syntax described earlier.

Group provider plugins are specified via the group_plugin setting. The argument to group_plugin should consist of the plugin path, either fully-qualified or relative to the /usr/libexec/sudo directory, followed by any configuration options the plugin requires. These options (if specified) will be passed to the plugin's initialization function. If options are present, the string must be enclosed in double quotes ("").

The following group provider plugins are installed by default:

group_file

The group_file plugin supports an alternate group file that uses the same syntax as the /etc/group file. The path to the group file should be specified as an option to the plugin. For example, if the group file to be used is /etc/sudo-group:

Defaults group_plugin="group_file.so /etc/sudo-group"

system_group

The system_group plugin supports group lookups via the standard C library functions getgrnam() and getgrid(). This plugin can be used in instances where the user belongs to groups not present in the user's supplemental group vector. This plugin takes no options:

Defaults group_plugin=system_group.so

The group provider plugin API is described in detail in sudo_plugin(5).

LOG FORMAT

sudoers can log events in either JSON or sudo format, this section de? scribes the sudo log format. Depending on sudoers configuration, sudoers can log events via syslog(3), to a local log file, or both. The log for? mat is almost identical in both cases.

Accepted command log entries

Commands that sudo runs are logged using the following format (split into multiple lines for readability):

date hostname progname: username : TTY=ttyname ; PWD=cwd ; \
USER=runasuser ; GROUP=runasgroup ; TSID=logid ; \
ENV=env_vars COMMAND=command

Where the fields are as follows:

date The date the command was run. Typically, this is in the format ?MMM, DD, HH:MM:SS?. If logging via syslog(3), the actual date format is controlled by the syslog daemon. If logging to a file and the log_year option is enabled, the date will also include the year.

hostname The name of the host sudo was run on. This field is only present when logging via syslog(3).

progname The name of the program, usually sudo or sudoedit. This

field is only present when logging via syslog(3).

username The login name of the user who ran sudo.

ttyname The short name of the terminal (e.g., ?console?, ?tty01?, or ?pts/0?) sudo was run on, or ?unknown? if there was no terminal present.

cwd The current working directory that sudo was run in.

runasuser The user the command was run as.

runasgroup The group the command was run as if one was specified on the command line.

logid An I/O log identifier that can be used to replay the com? mand's output. This is only present when the log_input or log_output option is enabled.

env_vars A list of environment variables specified on the command line, if specified.

command The actual command that was executed.

Messages are logged using the locale specified by sudoers_locale, which defaults to the ?C? locale.

Denied command log entries

If the user is not allowed to run the command, the reason for the denial will follow the user name. Possible reasons include:

user NOT in sudoers

The user is not listed in the sudoers file.

user NOT authorized on host

The user is listed in the sudoers file but is not allowed to run com? mands on the host.

command not allowed

The user is listed in the sudoers file for the host but they are not allowed to run the specified command.

3 incorrect password attempts

The user failed to enter their password after 3 tries. The actual num? ber of tries will vary based on the number of failed attempts and the value of the passwd_tries option.

a password is required

The -n option was specified but a password was required.

sorry, you are not allowed to set the following environment variables

The user specified environment variables on the command line that were not allowed by sudoers.

Error log entries

If an error occurs, sudoers will log a message and, in most cases, send a message to the administrator via email. Possible errors include:

parse error in /etc/sudoers near line N

sudoers encountered an error when parsing the specified file. In some cases, the actual error may be one line above or below the line number listed, depending on the type of error.

problem with defaults entries

The sudoers file contains one or more unknown Defaults settings. This does not prevent sudo from running, but the sudoers file should be checked using visudo.

timestamp owner (username): No such user

The time stamp directory owner, as specified by the timestampowner set? ting, could not be found in the password database.

unable to open/read /etc/sudoers

The sudoers file could not be opened for reading. This can happen when the sudoers file is located on a remote file system that maps user-ID 0 to a different value. Normally, sudoers tries to open the sudoers file using group permissions to avoid this problem. Consider either chang? ing the ownership of /etc/sudoers or adding an argument like ?sudoers_uid=N? (where ?N? is the user-ID that owns the sudoers file) to the end of the sudoers Plugin line in the sudo.conf(5) file.

unable to stat /etc/sudoers

The /etc/sudoers file is missing.

/etc/sudoers is not a regular file

The /etc/sudoers file exists but is not a regular file or symbolic link.

/etc/sudoers is owned by uid N, should be 0

The sudoers file has the wrong owner. If you wish to change the

sudoers file owner, please add ?sudoers_uid=N? (where ?N? is the user-ID that owns the sudoers file) to the sudoers Plugin line in the sudo.conf(5) file.

/etc/sudoers is world writable

The permissions on the sudoers file allow all users to write to it. The sudoers file must not be world-writable, the default file mode is 0440 (readable by owner and group, writable by none). The default mode may be changed via the ?sudoers_mode? option to the sudoers Plugin line in the sudo.conf(5) file.

/etc/sudoers is owned by gid N, should be 1

The sudoers file has the wrong group ownership. If you wish to change the sudoers file group ownership, please add ?sudoers_gid=N? (where ?N? is the group-ID that owns the sudoers file) to the sudoers Plugin line in the sudo.conf(5) file.

unable to open /run/sudo/ts/username

sudoers was unable to read or create the user's time stamp file. This can happen when timestampowner is set to a user other than root and the mode on /run/sudo is not searchable by group or other. The default mode for /run/sudo is 0711.

unable to write to /run/sudo/ts/username

sudoers was unable to write to the user's time stamp file.

/run/sudo/ts is owned by uid X, should be Y

The time stamp directory is owned by a user other than timestampowner. This can occur when the value of timestampowner has been changed. sudoers will ignore the time stamp directory until the owner is cor? rected.

/run/sudo/ts is group writable

The time stamp directory is group-writable; it should be writable only by timestampowner. The default mode for the time stamp directory is 0700. sudoers will ignore the time stamp directory until the mode is corrected.

Notes on logging via syslog

By default, sudoers logs messages via syslog(3). The date, hostname, and

progname fields are added by the system's syslog() function, not sudoers itself. As such, they may vary in format on different systems.

The maximum size of syslog messages varies from system to system. The syslog_maxlen setting can be used to change the maximum syslog message size from the default value of 980 bytes. For more information, see the description of syslog_maxlen.

Notes on logging to a file

If the logfile option is set, sudoers will log to a local file, such as /var/log/sudo. When logging to a file, sudoers uses a format similar to syslog(3), with a few important differences:

1. The progname and hostname fields are not present.
2. If the log_year option is enabled, the date will also include the year.
3. Lines that are longer than loglinelen characters (80 by default) are word-wrapped and continued on the next line with a four character indent. This makes entries easier to read for a human being, but makes it more difficult to use grep(1) on the log files. If the loglinelen option is set to 0 (or negated with a ?!?), word wrap will be disabled.

I/O LOG FILES

When I/O logging is enabled, sudo will run the command in a pseudo-termi? nal and log all user input and/or output, depending on which options are enabled. I/O can be logged either to the local machine or to a remote log server. For local logs, I/O is logged to the directory specified by the iolog_dir option (/var/log/sudo-io by default) using a unique session ID that is included in the sudo log line, prefixed with ?TSID=?. The iolog_file option may be used to control the format of the session ID.

For remote logs, the log_servers setting is used to specify one or more log servers running sudo_logsrvd or another server that implements the protocol described by sudo_logsrv.proto(5).

For both local and remote I/O logs, each log is stored in a separate di? rectory that contains the following files:

log A text file containing information about the command. The

first line consists of the following colon-delimited fields:
the time the command was run, the name of the user who ran
sudo, the name of the target user, the name of the target group
(optional), the terminal that sudo was run from, and the number
of lines and columns of the terminal. The second and third
lines contain the working directory the command was run from
and the path name of the command itself (with arguments if
present).

log.json A JSON-formatted file containing information about the command.
This is similar to the log file but contains additional infor?
mation and is easily extensible. The log.json file will be
used by sudoreplay(8) in preference to the log file if it ex?
ists. The file may contain the following elements:

timestamp
A JSON object containing time the command was run.
It consists of two values, seconds and nanoseconds.

columns The number of columns of the terminal the command ran
on, or zero if no terminal was present.

command The fully-qualified path of the command that was run.

lines The number of lines of the terminal the command ran
on, or zero if no terminal was present.

runargv A JSON array representing the command's argument vec?
tor as passed to the execve(2) system call.

runenv A JSON array representing the command's environment
as passed to the execve(2) system call.

rungid The group ID the command ran as. This element is
only present when the user specifies a group on the
command line.

rungroup The name of the group the command ran as. This ele?
ment is only present when the user specifies a group
on the command line.

runuid The user ID the command ran as.

runuser The name of the user the command ran as.

submitcwd

The current working directory at the time sudo was run.

submithost

The name of the host the command was run on.

submituser

The name of the user who ran the command via sudo.

ttyname The path name of the terminal the user invoked sudo from. If the command was run in a pseudo-terminal, ttyname will be different from the terminal the com? mand actually ran in.

timing Timing information used to replay the session. Each line con? sists of the I/O log entry type and amount of time since the last entry, followed by type-specific data. The I/O log entry types and their corresponding type-specific data are:

0 standard input, number of bytes in the entry

1 standard output, number of bytes in the entry

2 standard error, number of bytes in the entry

3 terminal input, number of bytes in the entry

4 terminal output, number of bytes in the entry

5 window change, new number lines and columns

6 bug compatibility for sudo 1.8.7 terminal output

7 command suspend or resume, signal received

ttyin Raw input from the user's terminal, exactly as it was received. No post-processing is performed. For manual viewing, you may wish to convert carriage return characters in the log to line feeds. For example: ?gunzip -c ttyin | tr "\r" "\n"?

stdin The standard input when no terminal is present, or input redi? rected from a pipe or file.

ttyout Output from the pseudo-terminal (what the command writes to the screen). Note that terminal-specific post-processing is per? formed before the data is logged. This means that, for exam? ple, line feeds are usually converted to line feed/carriage re?

turn pairs and tabs may be expanded to spaces.

stdout The standard output when no terminal is present, or output redirected to a pipe or file.

stderr The standard error redirected to a pipe or file.

All files other than log are compressed in gzip format unless the compress_io flag has been disabled. Due to buffering, it is not normally possible to display the I/O logs in real-time as the program is execut? ing. The I/O log data will not be complete until the program run by sudo has exited or has been terminated by a signal. The iolog_flush flag can be used to disable buffering, in which case I/O log data is written to disk as soon as it is available. The output portion of an I/O log file can be viewed with the sudoreplay(8) utility, which can also be used to list or search the available logs.

Note that user input may contain sensitive information such as passwords (even if they are not echoed to the screen), which will be stored in the log file unencrypted. In most cases, logging the command output via log_output or LOG_OUTPUT is all that is required.

Since each session's I/O logs are stored in a separate directory, tradi? tional log rotation utilities cannot be used to limit the number of I/O logs. The simplest way to limit the number of I/O is by setting the maxseq option to the maximum number of logs you wish to store. Once the I/O log sequence number reaches maxseq, it will be reset to zero and sudoers will truncate and re-use any existing I/O logs.

FILES

/etc/sudo.conf Sudo front end configuration

/etc/sudoers List of who can run what

/etc/group Local groups file

/etc/netgroup List of network groups

/var/log/sudo-io I/O log files

/run/sudo/ts Directory containing time stamps for the sudoers security policy

/var/db/sudo/lectured Directory containing lecture status files for the sudoers security policy

/etc/environment Initial environment for -i mode on AIX and Linux systems

EXAMPLES

Below are example sudoers file entries. Admittedly, some of these are a bit contrived. First, we allow a few environment variables to pass and then define our aliases:

```
# Run X applications through sudo; HOME is used to find the
# .Xauthority file.  Note that other programs use HOME to find
# configuration files and this may lead to privilege escalation!
Defaults env_keep += "DISPLAY HOME"
# User alias specification
User_Alias      FULLTIMERS = millert, mikef, dowdy
User_Alias      PARTTIMERS = bostley, jwfox, crawl
User_Alias      WEBADMIN = will, wendy, wim
# Runas alias specification
Runas_Alias     OP = root, operator
Runas_Alias     DB = oracle, sybase
Runas_Alias     ADMINGRP = adm, oper
# Host alias specification
Host_Alias      SPARC = bigtime, eclipse, moet, anchor :\
                SGI = grolsch, dandelion, black :\
                ALPHA = widget, thalamus, foobar :\
                HPPA = boa, nag, python
Host_Alias      CUNETS = 128.138.0.0/255.255.0.0
Host_Alias      CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
Host_Alias      SERVERS = primary, mail, www, ns
Host_Alias      CDROM = orion, perseus, hercules
# Cmnd alias specification
Cmnd_Alias      DUMPS = /usr/bin/mt, /usr/sbin/dump, /usr/sbin/rdump,\
                        /usr/sbin/restore, /usr/sbin/rrestore,\
                        sha224:0GomF8mNN3wlDt1HD9XldjJ3SNgpFdbjO1+NsQ== \
                        /home/operator/bin/start_backups
Cmnd_Alias      KILL = /usr/bin/kill
```

```
Cmnd_Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias      SHUTDOWN = /usr/sbin/shutdown
Cmnd_Alias      HALT = /usr/sbin/halt
Cmnd_Alias      REBOOT = /usr/sbin/reboot
Cmnd_Alias      SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh,\
                         /usr/local/bin/tcsh, /usr/bin/rsh,\
                         /usr/local/bin/zsh
Cmnd_Alias      SU = /usr/bin/su
Cmnd_Alias      PAGERS = /usr/bin/more, /usr/bin/pg, /usr/bin/less
Here we override some of the compiled in default values.  We want sudo to
log via syslog(3) using the auth facility in all cases and for commands
to be run with the target user's home directory as the working directory.
We don't want to subject the full time staff to the sudo lecture and we
want to allow them to run commands in a chroot(2) ?sandbox? via the -R
option.  User millert need not provide a password and we don't want to
reset the LOGNAME or USER environment variables when running commands as
root.  Additionally, on the machines in the SERVERS Host_Alias, we keep
an additional local log file and make sure we log the year in each log
line since the log entries will be kept around for several years.
Lastly, we disable shell escapes for the commands in the PAGERS
Cmnd_Alias (/usr/bin/more, /usr/bin/pg and /usr/bin/less).  Note that
this will not effectively constrain users with sudo ALL privileges.
# Override built-in defaults
Defaults                syslog=auth,runcwd=~
Defaults>root           !set_logname
Defaults:FULLTIMERS     !lecture,runchroot=*
Defaults:millert        !authenticate
Defaults@SERVERS        log_year, logfile=/var/log/sudo.log
Defaults!PAGERS         noexec
The User specification is the part that actually determines who may run
what.
root            ALL = (ALL) ALL
%wheel          ALL = (ALL) ALL
```

We let root and any user in group wheel run any command on any host as
any user.

FULLTIMERS ALL = NOPASSWD: ALL

Full time sysadmins (millert, mikef, and dowdy) may run any command on
any host without authenticating themselves.

PARTTIMERS ALL = ALL

Part time sysadmins bostley, jwfox, and crawl) may run any command on any
host but they must authenticate themselves first (since the entry lacks
the NOPASSWD tag).

jack CSNETS = ALL

The user jack may run any command on the machines in the CSNETS alias
(the networks 128.138.243.0, 128.138.204.0, and 128.138.242.0). Of those
networks, only 128.138.204.0 has an explicit netmask (in CIDR notation)
indicating it is a class C network. For the other networks in CSNETS,
the local machine's netmask will be used during matching.

lisa CUNETS = ALL

The user lisa may run any command on any host in the CUNETS alias (the
class B network 128.138.0.0).

operator ALL = DUMPS, KILL, SHUTDOWN, HALT, REBOOT, PRINTING,\
sudoedit /etc/printcap, /usr/oper/bin/

The operator user may run commands limited to simple maintenance. Here,
those are commands related to backups, killing processes, the printing
system, shutting down the system, and any commands in the directory
/usr/oper/bin/. Note that one command in the DUMPS Cmnd_Alias includes a
sha224 digest, /home/operator/bin/start_backups. This is because the di?
rectory containing the script is writable by the operator user. If the
script is modified (resulting in a digest mismatch) it will no longer be
possible to run it via sudo.

joe ALL = /usr/bin/su operator

The user joe may only su(1) to operator.

pete HPPA = /usr/bin/passwd [A-Za-z]*, !/usr/bin/passwd *root*

%opers ALL = (: ADMINGRP) /usr/sbin/

Users in the opers group may run commands in /usr/sbin/ as themselves

with any group in the ADMINGRP Runas_Alias (the adm and oper groups).

The user pete is allowed to change anyone's password except for root on the HPPA machines. Because command line arguments are matched as a sin? gle, concatenated string, the ?*? wildcard will match multiple words.

This example assumes that passwd(1) does not take multiple user names on the command line. Note that on GNU systems, options to passwd(1) may be specified after the user argument. As a result, this rule will also al? low:

```
    passwd username --expire
```

which may not be desirable.

```
bob             SPARC = (OP) ALL : SGI = (OP) ALL
```

The user bob may run anything on the SPARC and SGI machines as any user listed in the OP Runas_Alias (root and operator.)

```
jim             +biglab = ALL
```

The user jim may run any command on machines in the biglab netgroup. sudo knows that ?biglab? is a netgroup due to the ?+? prefix.

```
+secretaries    ALL = PRINTING, /usr/bin/adduser, /usr/bin/rmuser
```

Users in the secretaries netgroup need to help manage the printers as well as add and remove users, so they are allowed to run those commands on all machines.

```
fred            ALL = (DB) NOPASSWD: ALL
```

The user fred can run commands as any user in the DB Runas_Alias (oracle or sybase) without giving a password.

```
john            ALPHA = /usr/bin/su [!-]*, !/usr/bin/su *root*
```

On the ALPHA machines, user john may su to anyone except root but he is not allowed to specify any options to the su(1) command.

```
jen             ALL, !SERVERS = ALL
```

The user jen may run any command on any machine except for those in the SERVERS Host_Alias (primary, mail, www and ns).

```
jill            SERVERS = /usr/bin/, !SU, !SHELLS
```

For any machine in the SERVERS Host_Alias, jill may run any commands in the directory /usr/bin/ except for those commands belonging to the SU and SHELLS Cmnd_Aliases. While not specifically mentioned in the rule, the

commands in the PAGERS Cmnd_Alias all reside in /usr/bin and have the noexec option set.

steve CSNETS = (operator) /usr/local/op_commands/

The user steve may run any command in the directory /usr/local/op_com?mands/ but only as user operator.

matt valkyrie = KILL

On his personal workstation, valkyrie, matt needs to be able to kill hung processes.

WEBADMIN www = (www) ALL, (root) /usr/bin/su www

On the host www, any user in the WEBADMIN User_Alias (will, wendy, and wim), may run any command as user www (which owns the web pages) or sim?ply su(1) to www.

ALL CDROM = NOPASSWD: /sbin/umount /CDROM,\
/sbin/mount -o nosuid\,nodev /dev/cd0a /CDROM

Any user may mount or unmount a CD-ROM on the machines in the CDROM Host_Alias (orion, perseus, hercules) without entering a password. This is a bit tedious for users to type, so it is a prime candidate for encap?sulating in a shell script.

SECURITY NOTES

Limitations of the ?!? operator

It is generally not effective to ?subtract? commands from ALL using the ?!? operator. A user can trivially circumvent this by copying the de?sired command to a different name and then executing that. For example:

bill ALL = ALL, !SU, !SHELLS

Doesn't really prevent bill from running the commands listed in SU or SHELLS since he can simply copy those commands to a different name, or use a shell escape from an editor or other program. Therefore, these kind of restrictions should be considered advisory at best (and rein?forced by policy).

In general, if a user has sudo ALL there is nothing to prevent them from creating their own program that gives them a root shell (or making their own copy of a shell) regardless of any ?!? elements in the user specifi?cation.

Security implications of fast_glob

If the fast_glob option is in use, it is not possible to reliably negate commands where the path name includes globbing (aka wildcard) characters. This is because the C library's fnmatch(3) function cannot resolve rela? tive paths. While this is typically only an inconvenience for rules that grant privileges, it can result in a security issue for rules that sub? tract or revoke privileges.

For example, given the following sudoers file entry:

```
john    ALL = /usr/bin/passwd [a-zA-Z0-9]*, /usr/bin/chsh [a-zA-Z0-9]*,\
            /usr/bin/chfn [a-zA-Z0-9]*, !/usr/bin/* root
```

User john can still run /usr/bin/passwd root if fast_glob is enabled by changing to /usr/bin and running ./passwd root instead.

Preventing shell escapes

Once sudo executes a program, that program is free to do whatever it pleases, including run other programs. This can be a security issue since it is not uncommon for a program to allow shell escapes, which lets a user bypass sudo's access control and logging. Common programs that permit shell escapes include shells (obviously), editors, paginators, mail and terminal programs.

There are two basic approaches to this problem:

restrict Avoid giving users access to commands that allow the user to run arbitrary commands. Many editors have a restricted mode where shell escapes are disabled, though sudoedit is a better solution to running editors via sudo. Due to the large number of programs that offer shell escapes, restricting users to the set of programs that do not is often unworkable.

noexec Many systems that support shared libraries have the ability to override default library functions by pointing an environment variable (usually LD_PRELOAD) to an alternate shared library. On such systems, sudo's noexec functionality can be used to prevent a program run by sudo from executing any other pro? grams. Note, however, that this applies only to dynamically-linked executables. Statically-linked executables and executa?

bles running under binary emulation are not affected.

The noexec feature is known to work on SunOS, Solaris, *BSD, Linux, IRIX, Tru64 UNIX, macOS, HP-UX 11.x and AIX 5.3 and above. It should be supported on most operating systems that support the LD_PRELOAD environment variable. Check your oper? ating system's manual pages for the dynamic linker (usually ld.so, ld.so.1, dyld, dld.sl, rld, or loader) to see if LD_PRELOAD is supported.

On Solaris 10 and higher, noexec uses Solaris privileges in? stead of the LD_PRELOAD environment variable.

To enable noexec for a command, use the NOEXEC tag as docu? mented in the User Specification section above. Here is that example again:

aaron shanty = NOEXEC: /usr/bin/more, /usr/bin/vi

This allows user aaron to run /usr/bin/more and /usr/bin/vi with noexec enabled. This will prevent those two commands from executing other commands (such as a shell). If you are unsure whether or not your system is capable of supporting noexec you can always just try it out and check whether shell escapes work when noexec is enabled.

Note that restricting shell escapes is not a panacea. Programs running as root are still capable of many potentially hazardous operations (such as changing or overwriting files) that could lead to unintended privilege escalation. In the specific case of an editor, a safer approach is to give the user permission to run sudoedit (see below).

Secure editing

The sudoers plugin includes sudoedit support which allows users to se? curely edit files with the editor of their choice. As sudoedit is a built-in command, it must be specified in the sudoers file without a leading path. However, it may take command line arguments just as a nor? mal command does. Wildcards used in sudoedit command line arguments are expected to be path names, so a forward slash (?/?) will not be matched by a wildcard.

Unlike other sudo commands, the editor is run with the permissions of the invoking user and with the environment unmodified. More information may be found in the description of the -e option in sudo(8).

For example, to allow user operator to edit the ?message of the day? file:

```
operator    sudoedit /etc/motd
```

The operator user then runs sudoedit as follows:

```
$ sudoedit /etc/motd
```

The editor will run as the operator user, not root, on a temporary copy of /etc/motd. After the file has been edited, /etc/motd will be updated with the contents of the temporary copy.

Users should never be granted sudoedit permission to edit a file that re? sides in a directory the user has write access to, either directly or via a wildcard. If the user has write access to the directory it is possible to replace the legitimate file with a link to another file, allowing the editing of arbitrary files. To prevent this, starting with version 1.8.16, symbolic links will not be followed in writable directories and sudoedit will refuse to edit a file located in a writable directory un? less the sudoedit_checkdir option has been disabled or the invoking user is root. Additionally, in version 1.8.15 and higher, sudoedit will refuse to open a symbolic link unless either the sudoedit_follow option is enabled or the sudoedit command is prefixed with the FOLLOW tag in the sudoers file.

Time stamp file checks

sudoers will check the ownership of its time stamp directory (/run/sudo/ts by default) and ignore the directory's contents if it is not owned by root or if it is writable by a user other than root. Older versions of sudo stored time stamp files in /tmp; this is no longer rec? ommended as it may be possible for a user to create the time stamp them? selves on systems that allow unprivileged users to change the ownership of files they create.

While the time stamp directory should be cleared at reboot time, not all systems contain a /run or /var/run directory. To avoid potential prob?

lems, sudoers will ignore time stamp files that date from before the ma?
chine booted on systems where the boot time is available.
Some systems with graphical desktop environments allow unprivileged users to change the system clock. Since sudoers relies on the system clock for time stamp validation, it may be possible on such systems for a user to run sudo for longer than timestamp_timeout by setting the clock back. To combat this, sudoers uses a monotonic clock (which never moves backwards) for its time stamps if the system supports it.
sudoers will not honor time stamps set far in the future. Time stamps with a date greater than current_time + 2 * TIMEOUT will be ignored and sudoers will log and complain.
If the timestamp_type option is set to ?tty?, the time stamp record in?
cludes the device number of the terminal the user authenticated with.
This provides per-terminal granularity but time stamp records may still outlive the user's session.
Unless the timestamp_type option is set to ?global?, the time stamp record also includes the session ID of the process that last authenti?
cated. This prevents processes in different terminal sessions from using the same time stamp record. On systems where a process's start time can be queried, the start time of the session leader is recorded in the time stamp record. If no terminal is present or the timestamp_type option is set to ?ppid?, the start time of the parent process is used instead. In most cases this will prevent a time stamp record from being re-used with?
out the user entering a password when logging out and back in again.

DEBUGGING

Versions 1.8.4 and higher of the sudoers plugin support a flexible debug?
ging framework that can help track down what the plugin is doing inter?
nally if there is a problem. This can be configured in the sudo.conf(5) file.
The sudoers plugin uses the same debug flag format as the sudo front-end: subsystem@priority.
The priorities used by sudoers, in order of decreasing severity, are:
crit, err, warn, notice, diag, info, trace and debug. Each priority,

when specified, also includes all priorities higher than it. For exam?
ple, a priority of notice would include debug messages logged at notice
and higher.

The following subsystems are used by the sudoers plugin:

alias User_Alias, Runas_Alias, Host_Alias and Cmnd_Alias processing

all matches every subsystem

audit BSM and Linux audit code

auth user authentication

defaults sudoers file Defaults settings

env environment handling

ldap LDAP-based sudoers

logging logging support

match matching of users, groups, hosts and netgroups in the sudoers
file

netif network interface handling

nss network service switch handling in sudoers

parser sudoers file parsing

perms permission setting

plugin The equivalent of main for the plugin.

pty pseudo-terminal related code

rbtree redblack tree internals

sssd SSSD-based sudoers

util utility functions

For example:

Debug sudo /var/log/sudo_debug match@info,nss@info

For more information, see the sudo.conf(5) manual.

SEE ALSO

ssh(1), su(1), fnmatch(3), glob(3), mktemp(3), strftime(3), sudo.conf(5),
sudo_plugin(5), sudoers.ldap(5), sudoers_timestamp(5), sudo(8), visudo(8)

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

The sudoers file should always be edited by the visudo utility which locks the file and checks for syntax errors. If sudoers contains syntax errors, sudo may refuse to run, which is a serious problem if sudo is your only method of obtaining superuser privileges. Recent versions of sudoers will attempt to recover after a syntax error by ignoring the rest of the line after encountering an error. Older versions of sudo will not run if sudoers contains a syntax error.

When using netgroups of machines (as opposed to users), if you store fully qualified host name in the netgroup (as is usually the case), you either need to have the machine's host name be fully qualified as re? turned by the hostname command or use the fqdn option in sudoers.

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, includ? ing, 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.