Rocky Enterprise Linux 9.2 Manual Pages on command 'sudo.conf.5'

$ man sudo.conf.5

SUDO.CONF(5) BSD File Formats Manual SUDO.CONF(5)

NAME

sudo.conf ? configuration for sudo front end

DESCRIPTION

The sudo.conf file is used to configure the sudo front end. It specifies the security policy and I/O logging plugins, debug flags as well as plugin-agnostic path names and settings.

The sudo.conf file supports the following directives, described in detail below.

Plugin a security policy or I/O logging plugin

Path a plugin-agnostic path

Set a front end setting, such as disable_coredump or group_source

Debug debug flags to aid in debugging sudo, sudoreplay, visudo, and the sudoers plugin.

The pound sign (?#?) is used to indicate a comment. Both the comment character and any text after it, up to the end of the line, are ignored.

Long lines can be continued with a backslash (?\?) as the last character on the line. Note that leading white space is removed from the beginning

of lines even when the continuation character is used.

Non-comment lines that don't begin with Plugin, Path, Debug, or Set are silently ignored.

The sudo.conf file is always parsed in the ?C? locale.

Plugin configuration

sudo supports a plugin architecture for security policies and input/out? put logging. Third parties can develop and distribute their own policy and I/O logging plugins to work seamlessly with the sudo front end.

Plugins are dynamically loaded based on the contents of sudo.conf.

A Plugin line consists of the Plugin keyword, followed by the symbol_name and the path to the dynamic shared object that contains the plugin. The symbol_name is the name of the approval_plugin, audit_plugin, io_plugin, or policy_plugin struct contained in the plugin. If a plugin implements multiple plugin types, there must be a Plugin line for each unique symbol name. The path may be fully qualified or relative. If not fully quali? fied, it is relative to the directory specified by the plugin_dir Path setting, which defaults to /usr/libexec/sudo. In other words:

```
Plugin sudoers_policy sudoers.so
```

is equivalent to:

```
Plugin sudoers_policy /usr/libexec/sudo/sudoers.so
```

If the plugin was compiled statically into the sudo binary instead of be? ing installed as a dynamic shared object, the path should be specified without a leading directory, as it does not actually exist in the file system. For example:

```
Plugin sudoers_policy sudoers.so
```

Starting with sudo 1.8.5, any additional parameters after the path are passed as arguments to the plugin's open function. For example, to over? ride the compile-time default sudoers file mode:

```
Plugin sudoers_policy sudoers.so sudoers_mode=0440
```

See the sudoers(5) manual for a list of supported arguments.

The same dynamic shared object may contain multiple plugins, each with a different symbol name. The file must be owned by uid 0 and only writable by its owner. Because of ambiguities that arise from composite policies,

only a single policy plugin may be specified. This limitation does not apply to I/O plugins.

If no sudo.conf file is present, or if it contains no Plugin lines, the sudoers plugin will be used as the default security policy, for I/O log? ging (if enabled by the policy) and for auditing. This is equivalent to the following:

```
Plugin sudoers_policy sudoers.so
Plugin sudoers_io sudoers.so
Plugin sudoers_audit sudoers.so
```

Starting with sudo version 1.9.1, some of the logging functionality of the sudoers plugin has been moved from the policy plugin to an audit plugin. To maintain compatibility with sudo.conf files from older sudo versions, if sudoers is configured as the security policy, it will be used as an audit plugin as well. This guarantees that the logging behav? ior will be consistnet with that of sudo versions 1.9.0 and below.

For more information on the sudo plugin architecture, see the sudo_plugin(5) manual.

Path settings

A Path line consists of the Path keyword, followed by the name of the path to set and its value. For example:

```
Path noexec /usr/libexec/sudo/sudo_noexec.so
Path askpass /usr/X11R6/bin/ssh-askpass
```

If no path name is specified, features relying on the specified setting will be disabled. Disabling Path settings is only supported in sudo ver? sion 1.8.16 and higher.

The following plugin-agnostic paths may be set in the /etc/sudo.conf file:

askpass The fully qualified path to a helper program used to read the user's password when no terminal is available. This may be the case when sudo is executed from a graphical (as opposed to text-based) application. The program specified by askpass should display the argument passed to it as the prompt and write the user's password to the standard output. The value of

askpass may be overridden by the SUDO_ASKPASS environment vari?
able.

devsearch
An ordered, colon-separated search path of directories to look in for device nodes. This is used when mapping the process's tty device number to a device name on systems that do not pro? vide such a mechanism. Sudo will not recurse into sub-directo? ries. If terminal devices may be located in a sub-directory of /dev, that path must be explicitly listed in devsearch. The default value is
/dev/pts:/dev/vt:/dev/term:/dev/zcons:/dev/pty:/dev
This option is ignored on systems that support either the devname() or _ttyname_dev() functions, for example BSD, macOS and Solaris.

noexec The fully-qualified path to a shared library containing wrap? pers for the execl(), execle(), execlp(), exect(), execv(), execve(), execvP(), execvp(), execvpe(), fexecve(), popen(), posix_spawn(), posix_spawnp(), system(), and wordexp() library functions that prevent the execution of further commands. This is used to implement the noexec functionality on systems that support LD_PRELOAD or its equivalent. The default value is /usr/libexec/sudo/sudo_noexec.so.

plugin_dir
The default directory to use when searching for plugins that are specified without a fully qualified path name. The default value is /usr/libexec/sudo.

sesh The fully-qualified path to the sesh binary. This setting is only used when sudo is built with SELinux support. The default value is /usr/libexec/sudo/sesh.

Other settings
The sudo.conf file also supports the following front end settings:

disable_coredump
Core dumps of sudo itself are disabled by default to prevent

the disclosure of potentially sensitive information. To aid in debugging sudo crashes, you may wish to re-enable core dumps by setting ?disable_coredump? to false in sudo.conf as follows:

Set disable_coredump false

All modern operating systems place restrictions on core dumps from set-user-ID processes like sudo so this option can be en? abled without compromising security. To actually get a sudo core file you will likely need to enable core dumps for set-user-ID processes. On BSD and Linux systems this is accom? plished in the sysctl(8) command. On Solaris, the coreadm(1m) command is used to configure core dump behavior.

This setting is only available in sudo version 1.8.4 and higher.

developer_mode

By default sudo refuses to load plugins which can be modified by other than the root user. The plugin should be owned by root and write access permissions should be disabled for ?group? and ?other?. To make development of a plugin easier, you can disable that by setting ?developer_mode? option to true in sudo.conf as follows:

Set developer_mode true

Please note that this creates a security risk, so it is not recommended on critical systems such as a desktop machine for daily use, but is intended to be used in development environ? ments (VM, container, etc). Before enabling developer mode, ensure you understand the implications.

This setting is only available in sudo version 1.9.0 and higher.

group_source

sudo passes the invoking user's group list to the policy and I/O plugins. On most systems, there is an upper limit to the number of groups that a user may belong to simultaneously (typ? ically 16 for compatibility with NFS). On systems with the

getconf(1) utility, running:

    getconf NGROUPS_MAX

will return the maximum number of groups.
However, it is still possible to be a member of a larger number
of groups--they simply won't be included in the group list re?
turned by the kernel for the user. Starting with sudo version
1.8.7, if the user's kernel group list has the maximum number
of entries, sudo will consult the group database directly to
determine the group list. This makes it possible for the secu?
rity policy to perform matching by group name even when the
user is a member of more than the maximum number of groups.
The group_source setting allows the administrator to change
this default behavior. Supported values for group_source are:

static Use the static group list that the kernel returns.
        Retrieving the group list this way is very fast but
        it is subject to an upper limit as described above.
        It is ?static? in that it does not reflect changes to
        the group database made after the user logs in. This
        was the default behavior prior to sudo 1.8.7.

dynamic Always query the group database directly. It is
        ?dynamic? in that changes made to the group database
        after the user logs in will be reflected in the group
        list. On some systems, querying the group database
        for all of a user's groups can be time consuming when
        querying a network-based group database. Most oper?
        ating systems provide an efficient method of perform?
        ing such queries. Currently, sudo supports efficient
        group queries on AIX, BSD, HP-UX, Linux and Solaris.

adaptive Only query the group database if the static group
        list returned by the kernel has the maximum number of
        entries. This is the default behavior in sudo 1.8.7
        and higher.

For example, to cause sudo to only use the kernel's static list

of groups for the user:

Set group_source static

This setting is only available in sudo version 1.8.7 and higher.

max_groups

The maximum number of user groups to retrieve from the group database. Values less than one will be ignored. This setting is only used when querying the group database directly. It is intended to be used on systems where it is not possible to de? tect when the array to be populated with group entries is not sufficiently large. By default, sudo will allocate four times the system's maximum number of groups (see above) and retry with double that number if the group database query fails. This setting is only available in sudo version 1.8.7 and higher. It should not be required in sudo versions 1.8.24 and higher and may be removed in a later release.

probe_interfaces

By default, sudo will probe the system's network interfaces and pass the IP address of each enabled interface to the policy plugin. This makes it possible for the plugin to match rules based on the IP address without having to query DNS. On Linux systems with a large number of virtual interfaces, this may take a non-negligible amount of time. If IP-based matching is not required, network interface probing can be disabled as fol? lows:

Set probe_interfaces false

This setting is only available in sudo version 1.8.10 and higher.

Debug flags

sudo versions 1.8.4 and higher support a flexible debugging framework that can help track down what sudo is doing internally if there is a problem.

A Debug line consists of the Debug keyword, followed by the name of the

program (or plugin) to debug (sudo, visudo, sudoreplay, sudoers), the de?
bug file name and a comma-separated list of debug flags. The debug flag
syntax used by sudo and the sudoers plugin is subsystem@priority but a
plugin is free to use a different format so long as it does not include a
comma (?,?).

For example:

    Debug sudo /var/log/sudo_debug all@warn,plugin@info

would log all debugging statements at the warn level and higher in addi?
tion to those at the info level for the plugin subsystem.

As of sudo 1.8.12, multiple Debug entries may be specified per program.
Older versions of sudo only support a single Debug entry per program.
Plugin-specific Debug entries are also supported starting with sudo
1.8.12 and are matched by either the base name of the plugin that was
loaded (for example sudoers.so) or by the plugin's fully-qualified path
name. Previously, the sudoers plugin shared the same Debug entry as the
sudo front end and could not be configured separately.

The following priorities are supported, in order of decreasing severity:
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 priorities trace and debug also include function call tracing which
logs when a function is entered and when it returns. For example, the
following trace is for the get_user_groups() function located in
src/sudo.c:

    sudo[123] -> get_user_groups @ src/sudo.c:385
    sudo[123] <- get_user_groups @ src/sudo.c:429 := groups=10,0,5

When the function is entered, indicated by a right arrow ?->?, the pro?
gram, process ID, function, source file and line number are logged. When
the function returns, indicated by a left arrow ?<-?, the same informa?
tion is logged along with the return value. In this case, the return
value is a string.

The following subsystems are used by the sudo front-end:

all matches every subsystem

args command line argument processing

conv user conversation

edit sudoedit

event event subsystem

exec command execution

main sudo main function

netif network interface handling

pcomm communication with the plugin

plugin plugin configuration

pty pseudo-terminal related code

selinux SELinux-specific handling

util utility functions

utmp utmp handling

The sudoers(5) plugin includes support for additional subsystems.

FILES

/etc/sudo.conf sudo front end configuration

EXAMPLES

```
#
# Default /etc/sudo.conf file
#
# Sudo plugins:
#   Plugin plugin_name plugin_path plugin_options ...
#
# The plugin_path is relative to /usr/libexec/sudo unless
#   fully qualified.
# The plugin_name corresponds to a global symbol in the plugin
#   that contains the plugin interface structure.
# The plugin_options are optional.
#
# The sudoers plugin is used by default if no Plugin lines are present.
#Plugin sudoers_policy sudoers.so
#Plugin sudoers_io sudoers.so
```

```
#Plugin sudoers_audit sudoers.so
#
# Sudo askpass:
#   Path askpass /path/to/askpass
#
# An askpass helper program may be specified to provide a graphical
# password prompt for "sudo -A" support.  Sudo does not ship with its
# own askpass program but can use the OpenSSH askpass.
#
# Use the OpenSSH askpass
#Path askpass /usr/X11R6/bin/ssh-askpass
#
# Use the Gnome OpenSSH askpass
#Path askpass /usr/libexec/openssh/gnome-ssh-askpass
#
# Sudo device search path:
#   Path devsearch /dev/path1:/dev/path2:/dev
#
# A colon-separated list of paths to check when searching for a user's
# terminal device.
#
#Path devsearch /dev/pts:/dev/vt:/dev/term:/dev/zcons:/dev/pty:/dev
#
# Sudo noexec:
#   Path noexec /path/to/sudo_noexec.so
#
# Path to a shared library containing replacements for the execv(),
# execve() and fexecve() library functions that just return an error.
# This is used to implement the "noexec" functionality on systems that
# support LD_PRELOAD or its equivalent.
#
# The compiled-in value is usually sufficient and should only be changed
# if you rename or move the sudo_noexec.so file.
```

```
#
#Path noexec /usr/libexec/sudo/sudo_noexec.so
#
# Sudo plugin directory:
#   Path plugin_dir /path/to/plugins
#
# The default directory to use when searching for plugins that are
# specified without a fully qualified path name.
#
#Path plugin_dir /usr/libexec/sudo
#
# Sudo developer mode:
#   Set developer_mode true|false
#
# Allow loading of plugins that are owned by non-root or are writable
# by "group" or "other".  Should only be used during plugin development.
#Set developer_mode true
#
# Core dumps:
#   Set disable_coredump true|false
#
# By default, sudo disables core dumps while it is executing (they
# are re-enabled for the command that is run).
# To aid in debugging sudo problems, you may wish to enable core
# dumps by setting "disable_coredump" to false.
#
#Set disable_coredump false
#
# User groups:
#   Set group_source static|dynamic|adaptive
#
# Sudo passes the user's group list to the policy plugin.
# If the user is a member of the maximum number of groups (usually 16),
```

```
# sudo will query the group database directly to be sure to include
# the full list of groups.
#
# On some systems, this can be expensive so the behavior is configurable.
# The "group_source" setting has three possible values:
#   static   - use the user's list of groups returned by the kernel.
#   dynamic  - query the group database to find the list of groups.
#   adaptive - if user is in less than the maximum number of groups.
#              use the kernel list, else query the group database.
#
#Set group_source static
#
# Sudo interface probing:
#   Set probe_interfaces true|false
#
# By default, sudo will probe the system's network interfaces and
# pass the IP address of each enabled interface to the policy plugin.
# On systems with a large number of virtual interfaces this may take
# a noticeable amount of time.
#
#Set probe_interfaces false
#
# Sudo debug files:
#   Debug program /path/to/debug_log subsystem@priority[,subsyste@priority]
#
# Sudo and related programs support logging debug information to a file.
# The program is typically sudo, sudoers.so, sudoreplay or visudo.
#
# Subsystems vary based on the program; "all" matches all subsystems.
# Priority may be crit, err, warn, notice, diag, info, trace or debug.
# Multiple subsystem@priority may be specified, separated by a comma.
#
#Debug sudo /var/log/sudo_debug all@debug
```

#Debug sudoers.so /var/log/sudoers_debug all@debug

SEE ALSO

sudo_plugin(5), sudoers(5), sudo(8)

HISTORY

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

AUTHORS

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

Todd C. Miller

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

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.