First Line of Open Source IAM: Just Sudo That!

Peter Czanik
Open Source Evangelist
One Identity
@PCzanik
Overview

- What is sudo
- From aliases to plugins
- Alerting with syslog-ng
- What is new in 1.9?
What is sudo?

- Answers, depending on experience and size of environment:
  - A tool to complicate life
  - A prefix for administrative commands
  - A way to see who did what
What is sudo?

- Sudo allows a system administrator to delegate authority by giving certain users the ability to run some commands as root or another user while providing an audit trail of the commands and their arguments. ([https://www.sudo.ws/](https://www.sudo.ws/))

- A lot more, than just a prefix
It can make you a sandwich.

MAKE ME A SANDWICH.

SUDO MAKE ME A SANDWICH.

WHAT? MAKE IT YOURSELF

OKAY.
Basic / etc / sudoers

%wheel ALL=(ALL) ALL

- Who
- Where
- As which user
- Which command
Aliases

- Simplify configuration
- Less error-prone

Host_Alias  WEBSERVERS = www1, www2, www3
User_Alias  ADMINS = smith, johnson, williams
Cmd_Alias   REBOOT = /sbin/halt, /sbin/reboot, /sbin/poweroff

ADMINS  WEBSERVERS = REBOOT
Defaults

- Changes the default behavior:
  
  Defaults secure_path="/usr/sbin:/usr/bin:/sbin:/bin"
  Defaults env_keep = "LANG LC_ADDRESS LC_CTYPE"
  Defaults !insults

- Can be user/host/etc specific
  
  Defaults:%wheel insults
Insults

• Fun, but not always PC :)

czanik@linux-mewy:~> sudo ls
[sudo] password for root:
Hold it up to the light --- not a brain in sight!
[sudo] password for root:
My pet ferret can type better than you!
[sudo] password for root:
sudo: 3 incorrect password attempts
czanik@linux-mewy:~>
Digest verification

peter ALL =
sha244:11925141bb22866afdf257ce7790bd6275feda80b3b241c108b79c88 /usr/bin/passwd

• Modified binaries do not run
• Difficult to maintain
• Additional layer of protection
Session recording

- Recording the terminal
- Play it back
- Difficult to modify (not cleartext)
- Easy to delete (saved locally) with unlimited access
  - Stay tuned :)
Session Recording - Demo
Plugin-based architecture

- Starting with version 1.8
- Replace or extend functionality
- Both open source and commercial
Plugin-based architecture

- **sudo_pair**
  - Making sure that no user can enter commands on their own
  - Terminate session on suspicious activity
  - Developed in Rust
  - https://github.com/square/sudo_pair
sudo_pair - Demo
Configuration hints

- Use visudo for syntax check
- Use EDITOR to use another text editor :) 
- A syntactically correct config still does not mean that you can execute anything :) 
- root password (even for Ubuntu!)
Configuration

- Read from top to bottom
- Start with generic
- Add exceptions at the end
Sample Configuration

Defaults !visiblepw
Defaults always_set_home
Defaults match_group_by_gid
Defaults always_query_group_plugin
Defaults env_reset
Defaults env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS"
Defaults env_keep += "MAIL PS1 PS2 QTDIR USERAME LANG LC_ADDRESS LC_CTYPE"
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
Defaults:%wheel insults
Defaults !insults
Defaults log_output
Where is the problem?

(There was a common mistake)
Central management

- Puppet, Ansible, etc.
  - Not real-time
  - Users can modify locally
  - Error-prone

- LDAP
  - Propagates real-time
  - Can’t be modified locally
  - Many limitations
Logging and alerting

- Email alerts
- All events to syslog
  - Make sure logs are centralized
  - Using syslog-ng sudo logs are automatically parsed and you can also do alerting to slack, Splunk, Elasticsearch, etc.
- Debug logs
  - Debug rules
  - Report problems
syslog-ng

● Logging
  Recording events, such as:
    • Jan 14 11:38:48 linux-0jbu sshd[7716]: Accepted publickey for root from 127.0.0.1 port 48806 ssh2

● syslog-ng
  Enhanced logging daemon with a focus on portability and high-performance central log collection. Originally developed in C.
Configuring syslog-ng

- “Don’t Panic”
- Simple and logical, even if it looks difficult at first
- Pipeline model:
  - Many different building blocks (sources, destinations, filters, parsers, etc.)
  - Connected into a pipeline using “log” statements
syslog-ng.conf: getting started

@version:3.23
@include "scl.conf"

# this is a comment :)

options {flush_lines (0); keep_hostname (yes);};

source s_sys { system(); internal();};
destination d_mesg { file("/var/log/messages"); };  
filter f_default { level(info..emerg) and not (facility(mail)); };  
log { source(s_sys); filter(f_default); destination(d_mesg); };
syslog-ng.conf: sudo building blocks

filter f_sudo {program(sudo)};

destination d_test {
    file("/var/log/sudo.json"
        template("$(format-json --scope nv_pairs --scope dot_nv_pairs --scope rfc5424)\n\n")
    );
}

destination d_slack {
    slack(hook-
        url("https://hooks.slack.com/services/TF8LZ3CSF/BF8CJKVT3/C2qdnM XCwDD3ATOFVMyxMyHB")
    );
}
syslog-ng.conf: sudo log statement

# name-value pairs come from the sudo parser

log {
    source(s_sys);
    filter(f_sudo);
    if (match("czanik" value(".sudo.SUBJECT"))) {
        destination { file("/var/log/sudo_filtered"); };
        destination(d_slack);
    };
    destination(d_test);
};
sudo logs in Slack
Coming to sudo 1.9

- Recording Service: collect sudo IO logs centrally
- Audit Plugin (ToDo)
- Approval Plugin framework (ToDo)
- Python support for plugins
Recording Service

- Collect sudo IO logs centrally
- Streamed in real-time, securely
- Convenient, available, secure
Audit plugin

- Not user visible
- API to access to all kinds of sudo logs
- Useful from Python
- Logging/Alerting to Elasticsearch, cloud providers, etc.
  - without external tools (like syslog-ng)
Approval Plugin framework

- Session approval
- No 3\textsuperscript{rd} party plugin necessary (like sudo\_pair, developed in Rust: https://github.com/square/sudo\_pair/)
- Using Python you can connect sudo with ticketing systems
  - Allow session only with open ticket
Python Support

● Extend sudo using Python
● Using the same API-s as C plugins

● API: https://www.sudo.ws/man/sudo_plugin.man.html

● No development environment or compilation is needed
Policy plugin API

- Decides who can do what
- Only one allowed
- Enabled in /etc/sudo.conf

- Example: only allow to run the command “id”
import sudo

class SudoPolicyPlugin(sudo.Plugin):
    def check_policy(self, argv, env_add):
        cmd = argv[0]  # the first argument is the command name
        if cmd != "id":  # Example for a simple reject:
            sudo.log_error("You are not allowed to run this command!")
            return sudo.RC_REJECT

        command_info_out = (  # setup command to execute
            "command=/usr/bin/id",  # Absolute path of command
            "runas_uid=0",  # The user id
            "runas_gid=0"  # The group id
        )
        return(sudo.RC_ACCEPT, command_info_out, argv, env_add)
Policy plugin API example: screenshot

[czanik@centos7 ~]$ sudo ls
You are not allowed to run this command!

[czanik@centos7 ~]$ sudo id
uid=0(root) gid=0(root) groups=0(root)
IO logs API

- Access input and output from user sessions
- Only one Python implementation is allowed
- Python examples:
  - Break connection if a given text appears on screen
  - Break connection if “rm -fr” is typed on the command line
  - Ask for the reason of the session
import sudo

class MyIOPPlugin(sudo.Plugin):
    def log_ttyout(self, buf):
        if "MySecret" in buf:
            sudo.log_info("Don't look at my secret!")
        return sudo.RC_REJECT
IO logs API example 1 (output check): screenshot

[czanik@centos7 ~]$ sudo -s
[root@centos7 czanik]# cd /root/
[root@centos7 ~]# ls
DoNotEnter kick.py_v1 policy.py_v1 sng
kick.py policy.py __pycache__ sudo

[root@centos7 ~]# cd DoNotEnter/
[root@centos7 DoNotEnter]# ls
Don't look at my secret!

    Hangup

[czanik@centos7 ~]$
import sudo

class MyI0Plugin(sudo.Plugin):
    def __init__(self, version: str, plugin_options, **kwargs):
        self.collected_buf = ''

    def log_ttyout(self, buf):
        self.collected_buf += buf
        if "rm -fr" in self.collected_buf:
            sudo.log_info("Oops. 'rm -fr' is dangerous! Kicking you out...")
            return sudo.RC_REJECT

        # drop all the string until last enter:
        last_enter_pos = self.collected_buf.rfind("\n")
        if last_enter_pos == 0:
            self.collected_buf = ''
IO logs API example 2 (input check): screenshot

[czanik@centos7 ~]$ sudo -s
[root@centos7 czanik]# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[root@centos7 czanik]# rm -f
Oops. 'rm -fr' is dangerous! Kicking you out...

Hangup
import sudo

class ReasonLoggerI0Plugin(sudo.Plugin):
    def open(self, argv, command_info):
        try:
            conv_timeout = 120  # in seconds
            sudo.log_info("Please provide your reason for executing '{0}'.format(argv[0]))
            message1 = sudo.ConvMessage(sudo.CONV_PROMPT_ECHO_ON, "Reason: ", conv_timeout)
            message2 = sudo.ConvMessage(sudo.CONV_PROMPT_MASK, "Secret reason: ", conv_timeout)
            reply1, reply2 = sudo.conv(message1, message2)

            with open("/tmp/sudo_reasons.txt", "a") as file:
                print("Executed", ' '.join(argv), file=file)
                print("Reason:", reply1, file=file)
                print("Hidden reason:", reply2, file=file)

        except sudo.ConversationInterrupted:
            sudo.log_error("You did not answer in time")
        return sudo.RC_REJECT
[czanik@centos7 ~]$ sudo -s
Please provide your reason for executing '/bin/bash'
Reason: my public reason
Secret reason: **************
[root@centos7 czanik]#
Group plugin API

- Allows non-Unix group lookups
- Example: can check if admin is on duty
- Python example: no password is used if user part of mygroup

Defaults group_plugin="python_plugin.so \ 
ModulePath=/root/group.py \ 
ClassName=SudoGroupPlugin"

%:mygroup ALL=(ALL) NOPASSWD: ALL
import sudo

class SudoGroupPlugin(sudo.Plugin):
    def query(self, user: str, group: str, user_pwd):
        hardcoded_user_groups = {
            "testgroup": [ "testuser1", "testuser2" ],
            "mygroup": [ "czanik" ]
        }
        group_has_user = user in hardcoded_user_groups.get(group, [])
        return sudo.RC_ACCEPT if group_has_user else sudo.RC_REJECT
Not just a prefix, but...

- **1.8**
  - Fine tuned permissions
  - Aliases / Defaults / Digest verification
  - Session recording / Logging and alerting
  - LDAP
  - Plugins

- **1.9**
  - Python plugin
  - Logging API
  - Central session recording collection
Apply it!

- Test features of existing 1.8 installation
  - Digest verification, session recording, etc.
- Enter tested features to production
- Install 1.9 on a test system
  - Central session recording
  - Extend sudo using Python
Questions?

- Sudo website: [https://www.sudo.ws/](https://www.sudo.ws/)
- My email: peter.czanik@oneidentity.com
- Twitter: @Pczanik