
ONE IDENTITY
by Quest

Security alerting made easy using Python

Peter Czanik

Open Source Evangelist

One Identity

@PCzanik

Overview

- What is alerting?
- sudo
- syslog-ng
- Alerting: the hard way
- Using the Apprise Python library

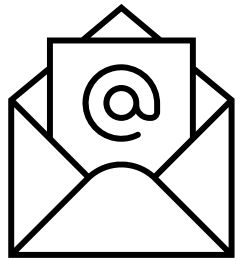
Alerting

- Alerting is a signal that warns of attack or danger
- Could be an air raid or chemical disaster

Alerting

- Sudo or syslog-ng: when something suspicious / important happens

- Traditionally by e-mail, now rather Discord, Slack, etc.



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/)
- A lot more than just a prefix

A basic /etc/sudoers

%wheel ALL=(ALL) ALL

- Who
- Where
- As which user
- Which command

Session recording

- Recording the terminal
- Playback
- Difficult to modify (not cleartext)
- Saved locally; therefore, easy to delete with unlimited access

- Sudo 1.9: central session recording

LDAP for central management

- Propagates in real-time
- Can’t be modified locally
- Many limitations

Python support

- Extending sudo using Python
- Using the same APIs as C plugins

- API: https://www.sudo.ws/man/sudo_plugin.man.html
  - Python plugin documentation: https://www.sudo.ws/man/sudo_plugin_python.man.html

- No development environment or compilation is needed

IO logs API

- Accessing input and output from user sessions
- Only one Python implementation is allowed

- Python examples:
  - Breaking connection if a given text appears on screen
  - Breaking connection if “rm -fr” is typed in the command line

IO logs API example: code

```
import sudo

class MyIOPlugin(sudo.Plugin):
    def log_ttyout(self, buf):
        if "MySecret" in buf:
            sudo.log_info("Don't look at my secret!")
            return sudo.RC_REJECT
```

syslog-ng

syslog-ng

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

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

Role: collecting data

Collect system and application logs together: contextual data for either side

A wide variety of platform-specific sources:

- /dev/log & co
- Journal, Sun streams

Syslog messages over the network:

- Legacy or RFC5424, UDP/TCP/TLS

Logs or any kind of text data from applications:

- Through files, sockets, pipes, application output, etc.

Python source: Jolly Joker

- HTTP server, Kafka source, etc.

Role: processing

Classifying, normalizing, and structuring logs with built-in parsers:

- CSV-parser, PatternDB, JSON parser, key=value parser

Rewriting messages:

- For example: anonymization

Reformatting messages using templates:

- Destination might need a specific format (ISO date, JSON, etc.)

Enriching data:

- GeoIP
- Additional fields based on message content

Python parser:

- all of the above, enriching logs from databases and also filtering

Role: filtering data

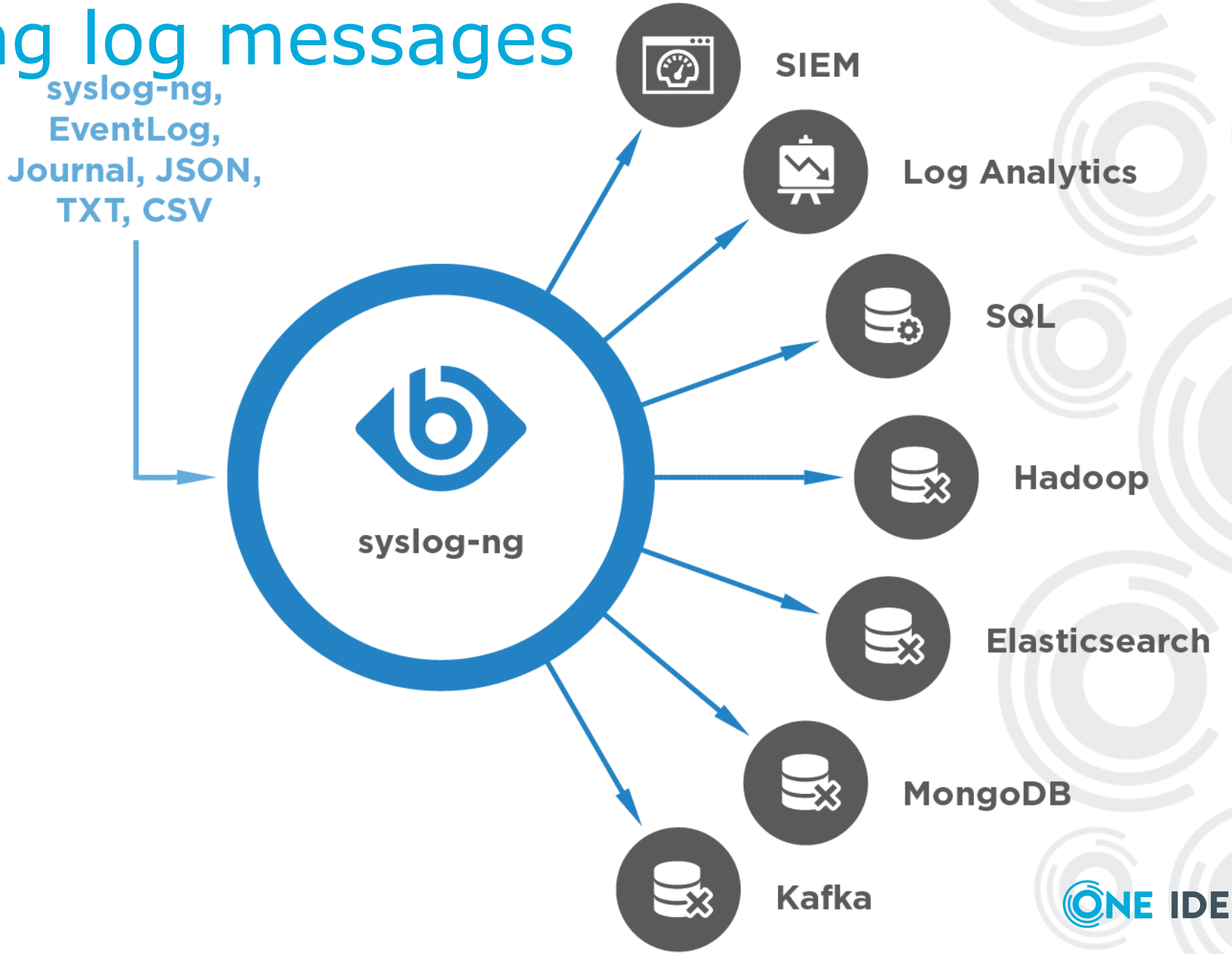
Main uses:

- Discarding surplus logs (not storing debug-level messages)
- Message routing (login events to SIEM)

Many possibilities:

- Based on message content, parameters, or macros
- Using comparisons, wildcards, regular expressions, and functions
- Combining all of these with Boolean operators

Role: storing log messages



ONE IDENTITY by Quest

What is common between a BMW i3 and a Kindle?

ONE IDENTITY by Quest

Alerting: the hard way

- Both sudo and syslog-ng support e-mail alerts
- Syslog-ng also supports http()
- Read documentation or reverse engineer protocols
- Painful, often not possible

Alerting to Discord using http()

- Relatively simple
- Read Discord documentation
- Create webhook
- Check Curl example
- Read syslog-ng http() documentation

- Create Discord destination

- See: https://www.syslog-ng.com/community/b/blog/posts/first-steps-of-sending-alerts-to-discord-and-others-from-syslog-ng-http-and-apprise

Alerting to Discord using http()

```
destination d_http {
    http(
        url("https://discord.com/api/webhooks/xxx/yyy")
        method("POST")
        user-agent("syslog-ng User Agent")
        headers("Content-Type: application/json")
        body('{"username": "test", "content": "${ISODATE} ${MESSAGE}"}')
    );
};
```

Alerting to other services?

- Start it over from scratch…

Or use Apprise

- Sudo and syslog-ng support Python
- Python libraries available to many services
- Still difficult

- Apprise:
  - https://github.com/caronc/apprise
  - Python library sending notifications to 50+ of services

Getting started is easy: syslog-ng configuration

```
source s_net {
  tcp(port(514) flags(syslog-protocol));
};
destination d_python_apprise {
    python(
        class("AppRise")
        value-pairs(scope(rfc5424))
        options(url "https://discord.com/api/webhooks/xxx/yyy")
    );
};
log {
    source(s_net);
    destination(d_python_apprise);
};
```

Getting started is easy: embedded Python code

```
python {
import apprise
class AppRise(object):
    def init(self, options):
#       print(options)
        self.apobj = apprise.Apprise()
        self.apobj.add(options["url"])
        return True
    def send(self, msg):
#       print(msg['MESSAGE'].decode('utf-8'))
        self.apobj.notify(
            body=msg['MESSAGE'].decode('utf-8'),
            title=msg['DATE'].decode('utf-8'),
        )
        return True
};
```

Making it easier and more flexible to use

- Already 50+ services supported, but…
- Blocks: hides implementation details, easier
- Templates: content is configurable, flexible
- Learn more at: https://www.syslog-ng.com/community/b/blog/posts/sending-alerts-to-discord-and-others-from-syslog-ng-using-apprise-blocks-and-python-templates

Alerting on sudo events

- Using the Audit API from Python
- Most services are throttled: slows sudo down
  - Aggregate messages
  - Send only what is important
  - Or be patient ☺

Alerting on sudo events 4/1

```
import sudo
import os
import apprise
VERSION = 1.0

class SudoAuditPlugin(sudo.Plugin):
    def __init__(self, plugin_options, user_info, **kwargs):
        # For loading multiple times, an optional "Id" can be specified
        # as argument to identify the log lines
        self.apobj = apprise.Apprise()
        self.apobj.add('https://discord.com/api/webhooks/xxx/yyy')
        self.message = ''
        self.approved = False
        self.sync = sudo.options_as_dict(plugin_options).get("sync", "no")
        plugin_id = sudo.options_as_dict(plugin_options).get("Id", "")
        self._log_line_prefix = "(AUDIT{}) ".format(plugin_id)

        user_info_dict = sudo.options_as_dict(user_info)
        user = user_info_dict.get("user", "???")
        uid = user_info_dict.get("uid", "???")
        self._log("-- Started by user {} ({}) -- ".format(user, uid))
```

Alerting on sudo events 4/2

```
    def __del__(self):
        self._log("-- Finished --")

    def open(self, submit_optind: int, submit_argv: tuple) -> int:
        # To cut out the sudo options, use "submit_optind":
        program_args = submit_argv[submit_optind:]
        if program_args:
            self._log("Requested command: " + " ".join(program_args))

    def accept(self, plugin_name, plugin_type,
               command_info, run_argv, run_envp) -> int:
        info = sudo.options_as_dict(command_info)
        cmd = list(run_argv)
        cmd[0] = info.get("command")
        self._log("Accepted command: {}".format(" ".join(cmd)))
        self._log("  By the plugin: {} (type={})".format(
            plugin_name, self.__plugin_type_str(plugin_type)))
        if plugin_type == sudo.PLUGIN_TYPE.SUDO:
            self.approved = True
#       self._log("  Environment: " + " ".join(run_envp))
```

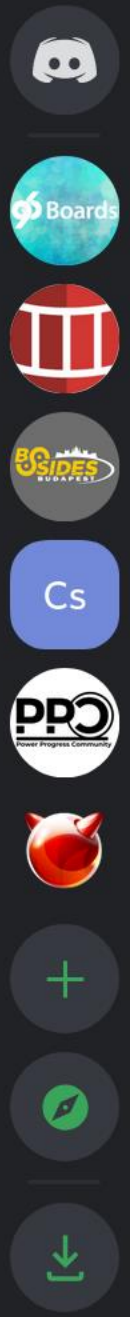
Alerting on sudo events 4/3

```
def reject(self, plugin_name, plugin_type, audit_msg, command_info) -> int:
    self._log("Rejected by plugin {} (type={}): {}".format(
        plugin_name, self.__plugin_type_str(plugin_type), audit_msg))

def error(self, plugin_name, plugin_type, audit_msg, command_info) -> int:
    self._log("Plugin {} (type={}) got an error: {}".format(
        plugin_name, self.__plugin_type_str(plugin_type), audit_msg))

def close(self, status_kind: int, status: int) -> None:
    if status_kind == sudo.EXIT_REASON.NO_STATUS:
        self._log("The command was not executed")

    elif status_kind == sudo.EXIT_REASON.WAIT_STATUS:
        if os.WIFEXITED(status):
            self._log("Command returned with exit code "
                      "{}".format(os.WEXITSTATUS(status)))
```

```
    elif os.WIFSIGNALED(status):
            self._log("Command exited due to signal "
                      "{}".format(os.WTERMSIG(status)))
    else:
            raise sudo.PluginError("Failed to understand wait exit status")
    elif status_kind == sudo.EXIT_REASON.EXEC_ERROR:
        self._log("Sudo has failed to execute the command, "
                  "execve returned {}".format(status))

    elif status_kind == sudo.EXIT_REASON.SUDO_ERROR:
        self._log("Sudo has run into an error: {}".format(status))

    else:
        raise Exception("Command returned unknown status kind {}".format(
            status_kind))
```

Alerting on sudo events 4/4

```
def _log(self, string):
        # For the example, we just log to output (this could be a file)
        sudo.log_info(self._log_line_prefix, string)
        self.message = self.message + string + '\n'
        if self.sync == "yes":
            self.apobj.notify(body=self.message,title='sudo',)
            self.message=''
        else:
            if self.approved:
                self.apobj.notify(body=self.message,title='sudo',)
                self.message=''
                self.approved = False
            if string == '-- Finished --':
                self.apobj.notify(body=self.message,title='sudo',)

    @staticmethod
    def __plugin_type_str(plugin_type):
        return sudo.PLUGIN_TYPE(plugin_type).name
```



CzP's server

An adventure begins.
Let's add some friends!

Invite People

TEXT CHANNELS

general

VOICE CHANNELS

General

CzP
#3042

general

Search

Command returned with exit code 0

sudo
-- Finished --

sudo
-- Started by user czanik (1000) --
Accepted command: /bin/bash
By the plugin: sudoers_policy (type=POLICY)
Accepted command: /bin/bash
By the plugin: sudo (type=SUDO)
Command returned with exit code 0

sudo
-- Finished --

sudo
-- Started by user czanik (1000) --
Accepted command: /bin/bash
By the plugin: sudoers_policy (type=POLICY)
Accepted command: /bin/bash
By the plugin: sudo (type=SUDO)
Command returned with exit code 0

sudo
-- Finished --

Message #general

GIF

ONLINE — 1

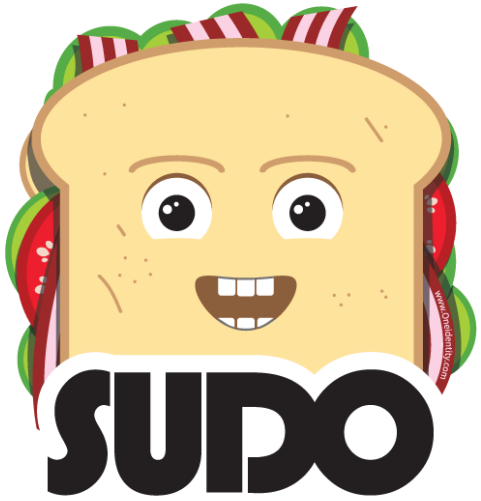
CzP

Summary

- Alerting lets you act on interesting IT events
- E-mail is no more enough
- Python support in sudo and syslog-ng enhances alerting possibilities
- Apprise allows you to send alerts to 50+ services

Questions?

- Sudo website: https://www.sudo.ws/
- Syslog-ng website: https://syslog-ng.com/
- My email: peter.czanik@oneidentity.com
- Twitter: @Pczanik



ONE IDENTITY by Quest


ONE IDENTITY
by Quest