Analyze your Suricata logs in real time using syslog-ng

Peter Czanik / syslog-ng, a One Identity business
About me

Peter Czanik from Hungary
Evangelist at One Identity: syslog-ng upstream
syslog-ng packaging, support, advocacy

syslog-ng originally developed by Balabit, now part of One Identity
Overview

- What is syslog-ng
- The four roles of syslog-ng
- Message parsing
- Enriching messages
- Blacklist filtering
- Configuring syslog-ng for Suricata
Inspiration

- Turris Omnia
- Linux on ARM
- SOHO router, firewall & container platform
- Runs syslog-ng & Suricata
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.
Main syslog-ng roles

Collector
Processor
Filter
Storage (or forwarder)
Role: data collector

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

A wide variety of platform-specific sources:
- /dev/log & co
- Journal, Sun streams

Receive 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
Role: processing

Classify, normalize, and structure logs with built-in parsers:
- CSV-parser, DB-parser (PatternDB), **JSON parser**, key=value parser, Python parser and more to come

Rewrite messages:
- For example: anonymization

Reformatting messages using templates:
- Destination might need a specific format (ISO date, JSON, etc.)

Enrich data:
- GeoIP
- Additional fields based on message content
Role: data filtering

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: destinations

Traditional:
- File, network, SQL, etc.

Big data:
- Hadoop
- MongoDB
- Elasticsearch
- Kafka

Others:
- HTTP(S)
- Java / Python
Which is the most used version?

- Project started in 1998
- RHEL EPEL has version 3.5
- Latest stable version is 3.18, released a month ago
Greenland or right-whale, he is the best existing authority. But scarce by knew nothing and says nothing of the great sperm whale, compared with which the Greenland whale is almost unworthy mentioning. And here be it said, that the Greenland whale is an usurper upon the throne of the seas. He is not even by any means the largest of the whales. Yet, owing to the long priority of his claims, and the profound ignorance which, till some seven years back, invested the then fabulous or utterly unknown sperm-whale, and which ignorance to this present day still reigns in all but some few scientific retreats and whale-shops; this usurpation has been every way complete. Reference to nearly all the leviathanic allusions in the great poets of past days, will satisfy you that the Greenland whale, without one rival, was to them the monarch of the seas. But the time has at last come for a new proclamation. This is Chasing Cross; hear ye! good people all,—the Greenland whale is deposed,—the great sperm whale now reigns at.

There are only two books in being which at all pretend to put the living sperm whale before you, and at the same time, in the remotest degree succeed in the attempt. Those books are Bdale’s and Bennett’s, both in their time surgeons to English South Sea whale-ships, and both exact and reliable men. The original matter touching the sperm whale to be found in their volumes is necessarily small; but so far as it goes, it is of excellent quality, though
Freeform log messages

**Most log messages are: date + hostname + text**

Mar 11 13:37:56 linux-6965 sshd[4547]: Accepted keyboard-interactive/pam for root from 127.0.0.1 port 46048 ssh2

- Text = English sentence with some variable parts
- Easy to read by a human
- Difficult to create alerts or reports
Solution: structured logging

Events represented as name-value pairs

Example: an ssh login:

app=sshd user=root source_ip=192.168.123.45

syslog-ng: name-value pairs inside

- Date, facility, priority, program name, pid, etc.

Parsers in syslog-ng can turn unstructured and some structured data (CSV, JSON) into name-value pairs
Enriching logs

Additional name-value pairs based on message content

GeoIP: find the geo-location of an IP address
■ Country name or longitude/latitude
■ Detect anomalies
■ Display locations on a map

Add metadata from CSV files
■ For example: host role, contact person
■ Less time spent on locating extra information
■ More accurate alerts or dashboards
The inlist() filter

Filtering based on white- or blacklisting

- Compares a single field with a list of values
- One value per line in text file
- case sensitive (up to 3.17)

Use cases

- Low cost SIEM: alerting based on spammer / C&C / etc. IP address lists
- Filtering based on a list of application names
Configuration

- “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:3.18
@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); };

@include "/etc/syslog-ng/conf.d/*.conf"
Suricata.conf: source, JSON parsing

# receive Suricata logs
source s_suricata {
    tcp(ip("0.0.0.0") port("514") flags(no-parse));
};

# parse JSON into name-value pairs
parser p_json {
    json-parser (prefix("suricata."));
};
Suricata.conf: GeoIP

```conf
parser p_geoip2 {
    geoip2( "${suricata.dest_ip}" , prefix( "parsed.dest." ) database("/usr/share/GeoIP/GeoLite2-City.mmdb") );
}

rewrite r_geoip2 {
    set("${parsed.dest.location.latitude},${parsed.dest.location.longitude}", value( "parsed.dest.ll" ), condition(not "${parsed.dest.location.latitude}" == ""))
}
```
Suricata.conf: destinations

destination d_suricata { file("/var/log/suricata.log" template("$(format-json --key suricata.* --key parsed.* --key ISODATE)\n")};

destination d_elastic {
  elasticsearch2 (
    cluster("syslog-ng") client_mode("http") index("syslog") time-zone(UTC) type("syslog") flush-limit(1) server("192.168.1.187")
    template("$(format-json --key suricata.* --key parsed.* --key ISODATE)")
    persist-name(elasticsearch-syslog)
  )
};
Suricata.conf: more parsers

# resolve non-local destination IP addresses using Python parser
parser p resolver {
  python(class("SngResolver"));
};

# add-contextual-data based on local IP address
parser p localsrc_info {
  add-contextual-data(selector("${suricata.src_ip}"), default-selector("unknown"), database("/etc/syslog-ng/conf.d/context-info-db.csv"), prefix("parsed.src."));
};
Suricata.conf: inline Python code

```python
python {
import socket

class SngResolver(object):
    def parse(self, log_message):
        ipaddr_b = log_message['suricata.dest_ip']
        ipaddr = ipaddr_b.decode('utf-8')
        try:
            resolved = socket.gethostbyaddr(ipaddr)
            hostname = resolved[0]
            log_message['parsed.dest.hostname'] = hostname
        except:
            pass
        return True
};
```
Suricata.conf: log statement 1.

log {
  # receive Suricata logs
  source(s_suricata);
  # parse JSON into name-value pairs
  parser(p_json);
  # resolve non-local destination IP addresses
  # using Python parser
  if (not match("^192.168" value("suricata.dest_ip"))) {
    parser(p_resolver);
  }
}
Suricata.conf: log statement 2.

# add-contextual-data based on local IP address
if (match("^192.168" value("suricata.src_ip"))){
    parser(p_localsrc_info);
};

# send alert if someone is reading slashdot
if (match("slashdot.org" value("suricata.tls.sni"))){
    destination { file("/var/log/slashdot");
    # ToDo: change to smtp destination
};
Suricata.conf: log statement 3.

# talking to a malware C&C
if {
    filter { in-list("/etc/syslog-ng/conf.d/malwarecc.list", value("suricata.dest_ip"));
    rewrite { set("Problem", value("parsed.malware")); };
} else {
    rewrite { set("OK", value("parsed.malware")); };
};
# add GeoIP information
parser(p_geoip2);
rewrite(r_geoip2);

```plaintext
# save results locally
destination(d_suricata);

# save results to Elasticsearch
destination(d_elastic);
```

```
What’s new in syslog-ng

- Disk-based buffering
- Grouping-by(): generic correlation
- Python bindings
- Elasticsearch REST API support
- HTTP(s) destination
- Wildcard file source
- Performance and memory usage improvements
- Many more :-)

#GetIAMRight | One Identity - Restricted - Confidential
syslog-ng benefits

- High-performance reliable log collection
- Simplified architecture
  - Single application for both syslog and application data
- Easier-to-use data
  - Parsed and presented in a ready-to-use format
- Lower load on destinations
  - Efficient message filtering and routing
Join the community!

- syslog-ng: http://syslog-ng.org/
- Source on GitHub: https://github.com/balabit/syslog-ng
- Mailing list: https://lists.balabit.hu/pipermail/syslog-ng/
- Gitter: https://gitter.im/balabit/syslog-ng
Questions?

syslog-ng blog: https://syslog-ng.com/community/

My e-mail: peter.czanik@oneidentity.com

Twitter: https://twitter.com/PCzanik