Creating a dedicated log management layer

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

- Basics: central log collection
- Growing complexity: analytics for security & operations
- Reducing complexity: dedicated log management
- Implementation using syslog-ng

Back to basics

- Central log collection

Why central logging?

Ease of use

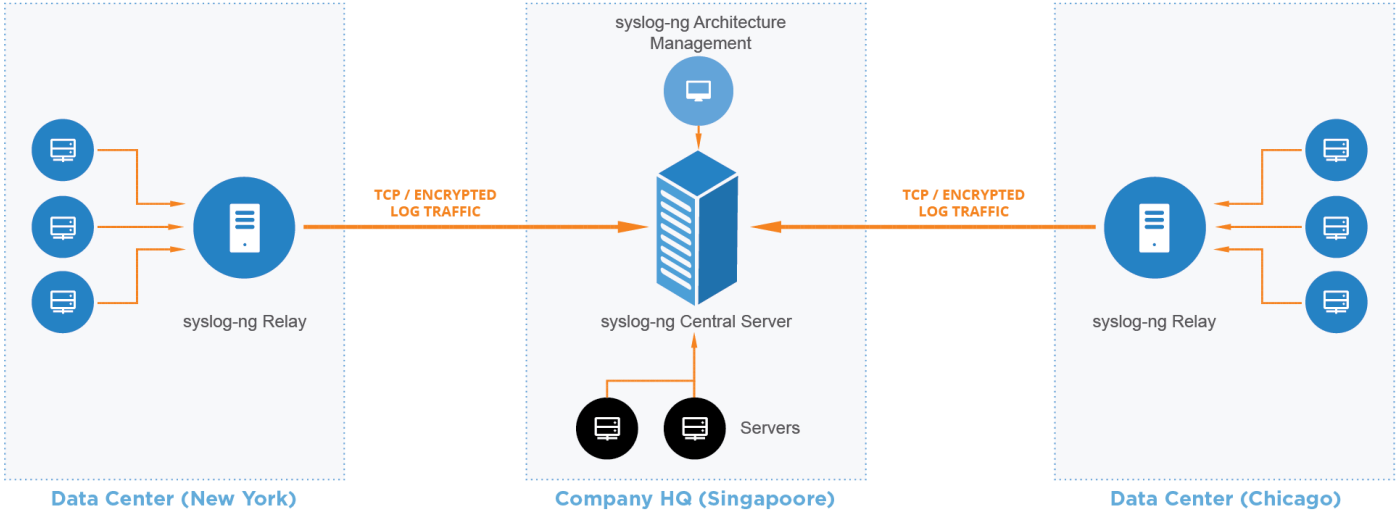
One place to check instead of many

Availability

Even if the sender machine is down

Security

Logs are available even if sender machine is compromised



Growing complexity

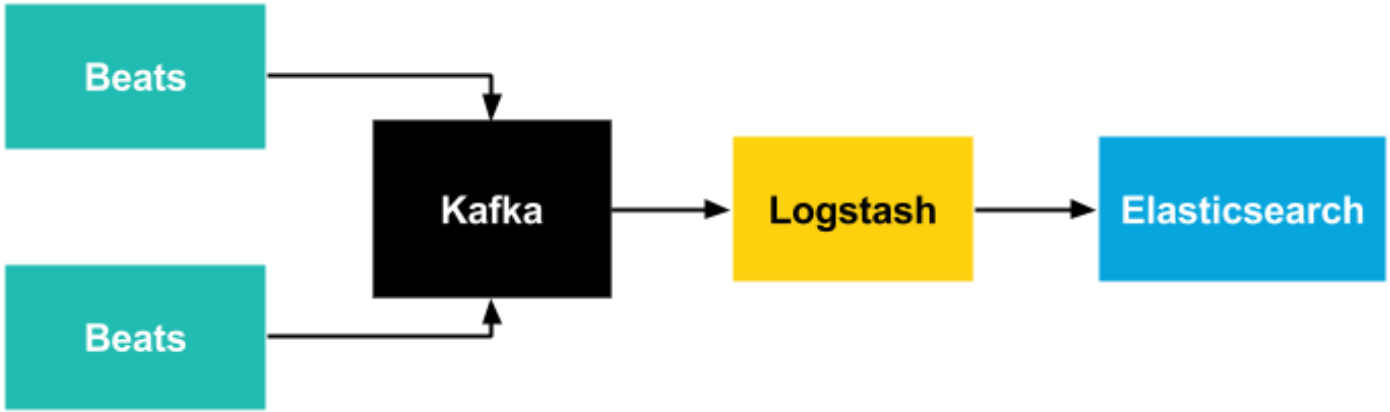
- Multiple analytics systems
- Wasting of resources

Multiple analytics systems

- Security, developers, operators use different analytics
- All come with log aggregation tools
- Some examples:
  - Elastic: Beats and Logstash
  - Splunk: forwarders
  - LaaS: collectors

Log aggregation

- Elastic stack on top of a local syslog:



- Also most LaaS, SIEM, etc. adds an additional layer on top of existing log management

Why is it a problem?

- More computing resources
- More network bandwidth (Cloud!)
- More human resources
- More security problems

Reducing complexity

- Unified log management layer

Using a unified log management layer

- Saves on computing, network & human resources
- Easier to push through security & operation teams
- Log management is separate from analytics
- Long term archiving separate from analytics

- Bonus: might save on analytics licensing and hardware costs

Implementing log management on syslog-ng

- What is syslog-ng
- Four roles: collecting, processing, filtering, store/forward
- Modes of operation
- Configuration

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.

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: Jolly Joker

- HTTP server, Kafka source, etc.

Role: processing

Classify, normalize, and structure logs with built-in parsers:

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

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

Python parser:

- all of above, enrich logs from databases and also filtering

Role: data filtering

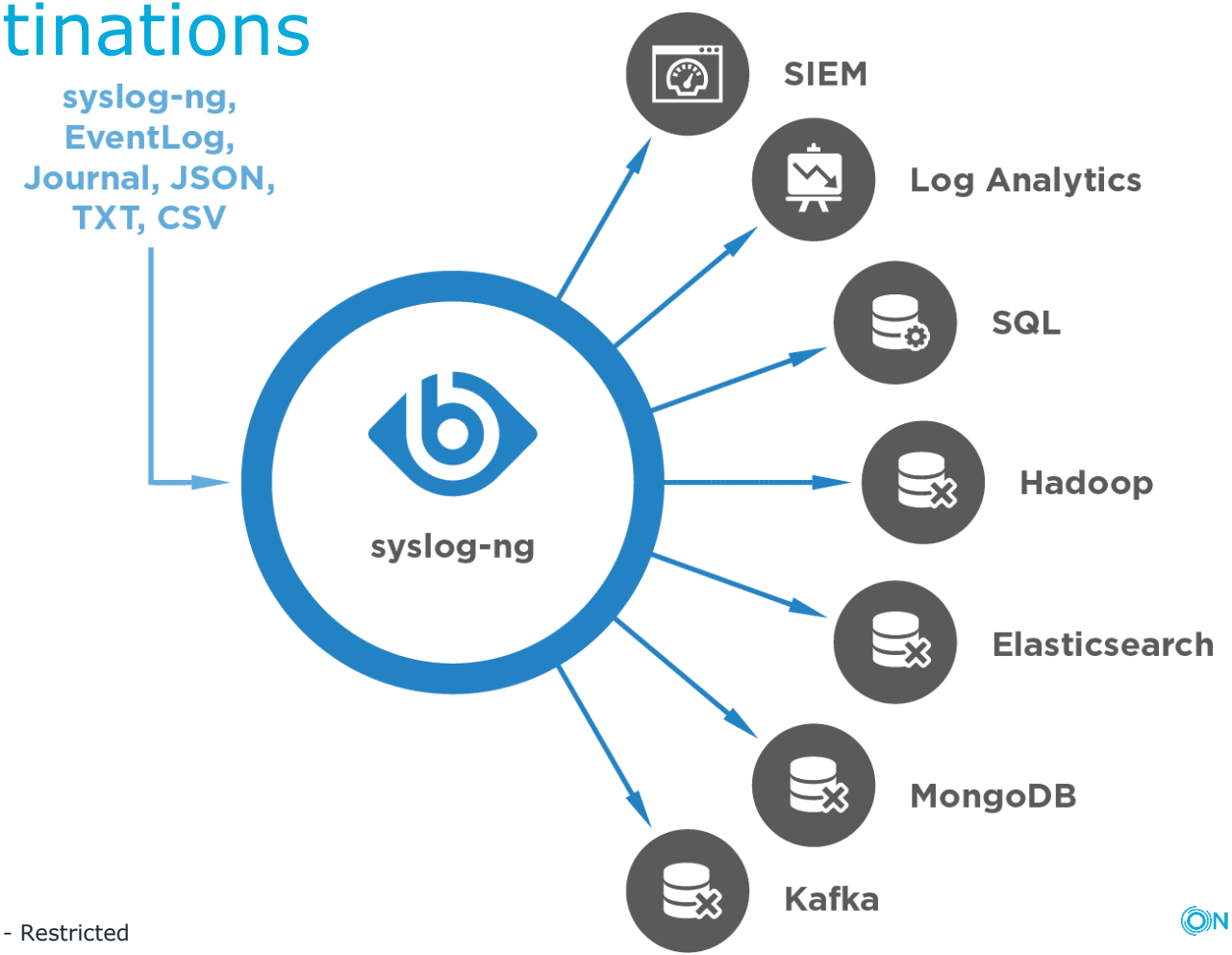
Main uses:

- 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



MODES OF OPERATION

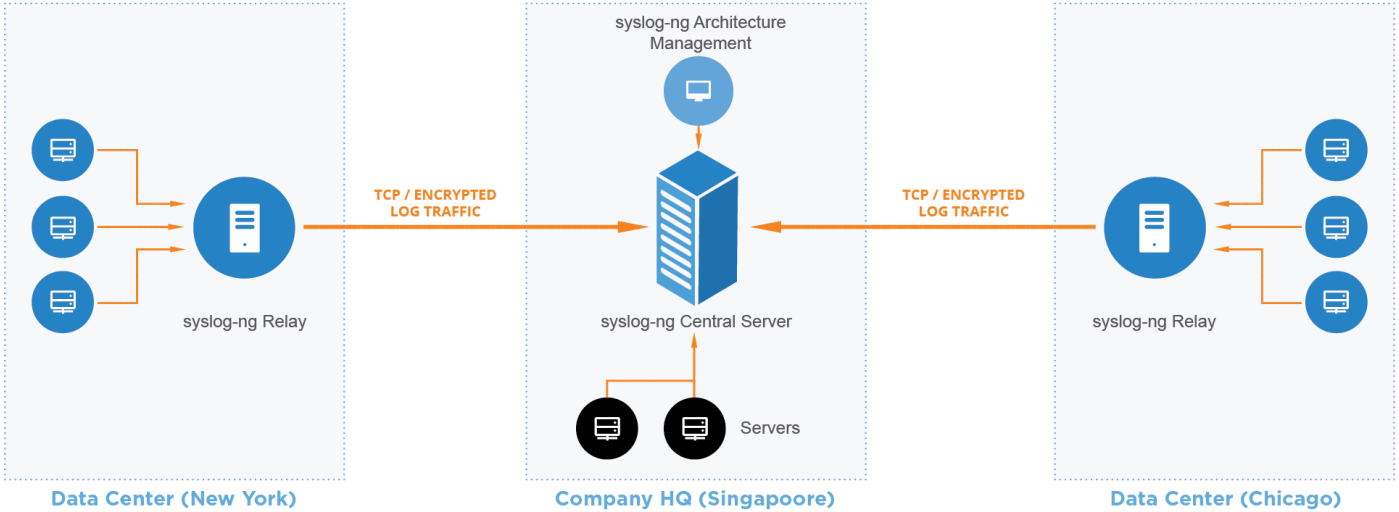
- Client mode: collecting logs from the client and sending them to the remote server (directly or through a relay)
- Relay mode: collecting logs from the clients (through the network) and sending them to the remote server (directly or through another relay)
- Server mode: collecting logs from the clients and storing them locally or in a database

Why relays?

UDP source
Collect as close as possible

Scalability
Distributing processing

Structure
A relay for each site or department



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. For 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

Name-value pairs make filtering more precise

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

```
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 {
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
  };
```

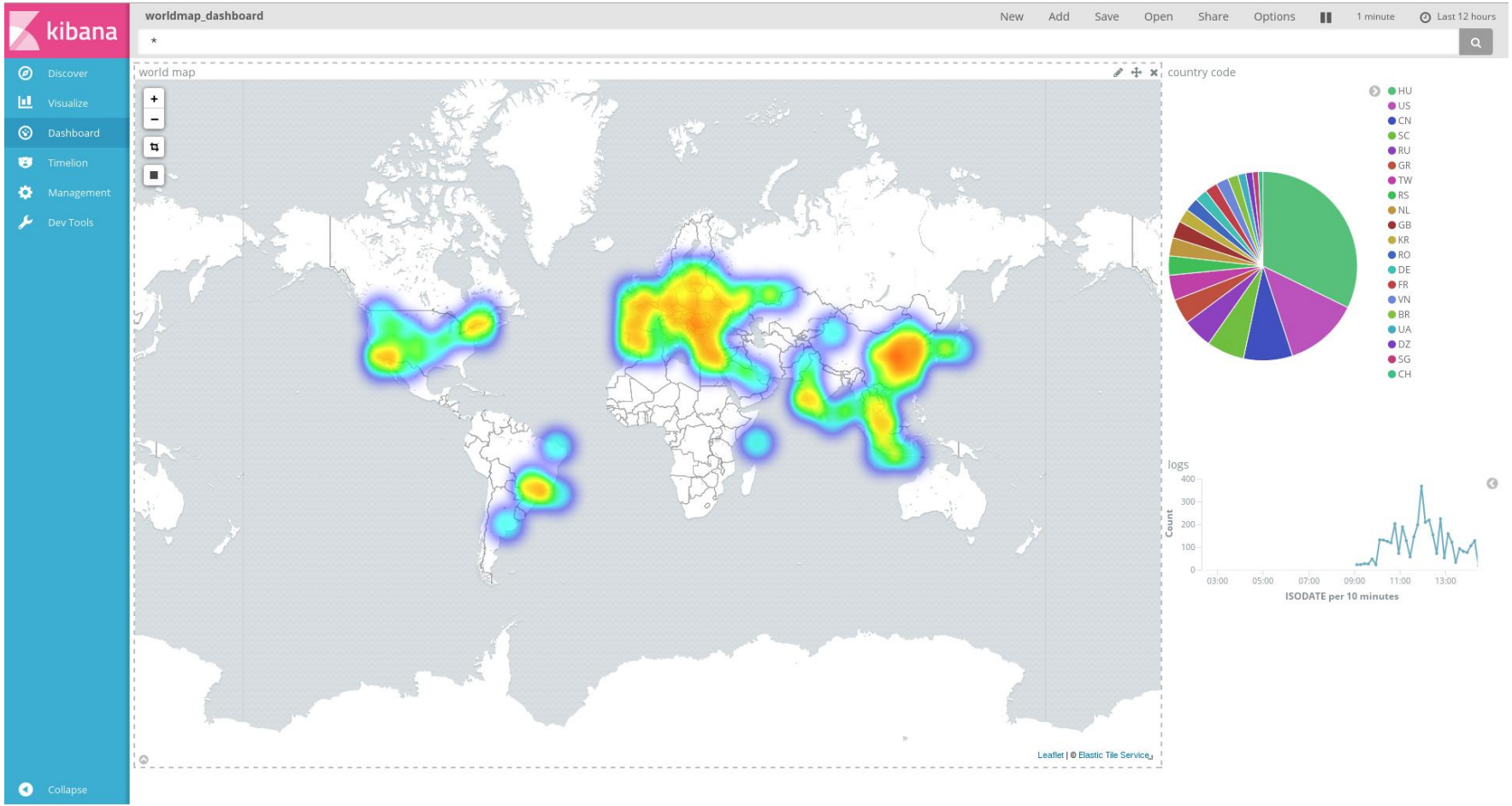
Suricata.conf: log statement 3.

```
 # 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);
```

Suricata.conf: log statement 4.

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

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



syslog-ng benefits

High-performance reliable log collection

Simplified architecture

One software instead of many

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

Questions?

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

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

Twitter: https://twitter.com/PCzanik