Flexible, scalable and secure logging using syslog-ng

HEPIX 2017
Peter Czanik / Balabit
ABOUT ME

- Peter Czanik from Hungary
- Community Manager at Balabit: syslog-ng upstream
- syslog-ng packaging, support, advocacy

Balabit is an IT security company with development HQ in Budapest, Hungary

Over 200 employees: the majority are engineers
OVERVIEW

- What is syslog-ng
- The four roles of syslog-ng
- Message parsing
- Enriching messages
- Blacklist filtering
- Configuring syslog-ng
- Scaling and reliability
- Analyzing logs: heat map, anonymization and more
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 high-performance 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
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 data from applications:

- Through files, sockets, pipes, etc.
- Application output
ROLE: PROCESSING

Classify, normalize and structure logs with built-in parsers:
- CSV-parser, DB-parser (PatternDB), JSON parser, key=value 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
“TRADITIONAL”
• File, network, TLS, SQL, etc.

“BIG DATA”
• Distributed file systems:
  • Hadoop
• NoSQL databases:
  • MongoDB
  • Elasticsearch
• Messaging systems:
  • Kafka
WHICH SYSLOG-NG VERSION IS THE MOST USED?

- Project started in 1998
- RHEL EPEL has version 3.5
- Latest stable version is 3.9, released three months ago
FREE-FORM 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 process them with scripts
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 into name-value pairs
  
  - CSV-parser, JSON parser, key=value parser
  - DB-parser (PatternDB),
  - Rust and Python parsers
PATTERNDB PARSER

Extracts information from unstructured messages into name-value pairs

- Add status fields based on message text
- Message classification (like LogCheck)

Needs XML describing log messages

Example: an ssh login failure:
- Parsed: app=sshd, user=root, source_ip=192.168.123.45
- Added: action=login, status=failure
- Classified as “violation”

```xml
<pattern>Failed @ESTRING:usracct.authmethod: @for @ESTRING:usracct.username: @from @ESTRING:usracct.device: @port @ESTRING:: @@ANYSTRING:usracct.service@</pattern>
```
ENRICHING LOG MESSAGES

Additional name-value pairs based on message content

PatternDB
- Add status fields based on message text
- Message classification (like LogCheck)

GeoIP: find the geo-location of an IP address
- Country name or longitude/latitude
- Detect anomalies or 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 text file
- Case sensitive

Use cases
- Poor mans 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: global options

@version:3.7
@include "scl.conf"

# this is a comment :)

options {
    flush_lines (0);
    keep_hostname (yes);
};
syslog-ng.conf: sources

source s_sys {
    system();
    internal();
};

source s_net {
    udp(ip(0.0.0.0) port(514));
};
syslog-ng.conf: destinations

destination d_mesg { file("/var/log/messages"); };

destination d_es {
    elasticsearch(
        index("syslog-ng_${YEAR}.${MONTH}.${DAY}")
        type("test")
        cluster("syslog-ng")
        template("$(format-json --scope rfc3164 --scope nv-pairs --exclude R_DATE --key ISODATE)\n"));
    );
};
syslog-ng.conf: filters, parsers

filter f_nodebug { level(info..emerg); };
filter f_messages { level(info..emerg) and
    not (facility(mail)
    or facility(authpriv)
    or facility(cron)); };

parser pattern_db {
    db-parser(file("/opt/syslog-ng/etc/patterndb.xml") );
};
syslog-ng.conf: logpath

log { source(s_sys); filter(f_messages); destination(d_mesg); };
log {
    source(s_net);
    source(s_sys);
    source(s_sys);
    filter(f_nodebug);
    parser(pattern_db);
    destination(d_es);
    flags(flow-control);
};
Patterndb & ElasticSearch & Kibana
SCALING SYSLOG-NG

- Traditionally: Client > Server
  - Too much processing
  - UDP is problematic
- Client > Relay > Server
  - Distribute some of the processing to Client/Relay
  - Collect UDP as close to the source as possible
  - Adds reliability: logs are collected even if central server inaccessible
SCALING SYSLOG-NG

Data Center (New York) <-> syslog-ng Central Server <-> Company HQ (Singapore) <-> Data Center (Chicago)

TCP / ENCRYPTED LOG TRAFFIC

syslog-ng Relay

syslog-ng Architecture Management

Servers
SCALING WITH LOG ROUTING

- Based on filtering
- Send the right logs to the right places
- Message parsing can increase accuracy
  - E-mail on root logins

- Can optimize SIEM / log analyzer tools
  - Only relevant messages: cheaper licensing
  - Throttling: evening out peaks
ANONYMIZING MESSAGES

Many regulations about what can be logged

- PCI-DSS: credit card numbers
- Europe: IP addresses, user names

Locating sensitive information:

- Regular expression: slow, works also in unknown logs
- Patterndb, CSV parser: fast, works only in known log messages

Anonymizing:

- Overwrite it with a constant
- Overwrite it with a hash of the original
GeoIP

- parser p_kv{ kv-parser(prefix("kv."))); }

- parser p_geoip { geoip("${kv.SRC}"), prefix( "geoip." ) database( "/usr/share/GeoIP/GeoLiteCity.dat" ) ); }

- rewrite r_geoip {
  set(
    "${geoip.latitude},${geoip.longitude}",
    value( "geoip.location" ),
    condition(not "${geoip.latitude}" == "")
  );
}

- log {
  source(s_tcp);
  parser(p_kv);
  parser(p_geoip);
  rewrite(r_geoip);
  destination(d_elastic);
};
WHAT IS NEW IN SYSLOG-NG 3.8 AND 3.9

- Disk-based buffering
- Grouping-by(): correlation independent of patterndb
- Elasticsearch 2.x & 5.0 support
- HTTP destination
- Performance improvements
- Coming up: parsers written in Python
SYSLOG-NG BENEFITS FOR LARGE ENVIRONMENTS

- **High-performance reliable log collection**: Single application for both syslog and application data.
- **Simplified architecture**: Parsed and presented in a ready-to-use format.
- **Easier-to-use data**: Efficient message filtering and routing.
- **Lower load on destinations**.
JOINING THE COMMUNITY

- syslog-ng information: http://syslog-ng.org/
- Binaries: https://syslog-ng.org/3rd-party-binaries/
- Source on GitHub: https://github.com/balabit/syslog-ng
- Mailing list: https://lists.balabit.hu/pipermail/syslog-ng/
- IRC: #syslog-ng on freenode
QUESTIONS?

My blog: http://czanik.blogs.balabit.com/
My e-mail: peter.czanik@balabit.com
Twitter: https://twitter.com/PCzanik
<?xml version='1.0' encoding='UTF-8'?>
<patterndb version='3' pub_date='2010-07-13'>
<ruleset name='opensshd' id='2448293e-6d1c-412c-a418-a80025639511'>
<pattern>sshd</pattern>
<rules>
<rule provider="patterndb" id="4dd5a329-da83-4876-a431-ddcb59c2858c" class="system">
<patterns>
<pattern>Accepted @ESTRING:usracct.authmethod: @for @ESTRING:usracct.username: @from @ESTRING:usracct.device: @port @ESTRING:: @@ANYSTRING:usracct.service@</pattern>
</patterns>
<examples>
<example>
<test_message program="sshd">Accepted password for bazsi from 127.0.0.1 port 48650 ssh2</test_message>
<test_values>
<test_value name="usracct.username">bazsi</test_value>
<test_value name="usracct.authmethod">password</test_value>
<test_value name="usracct.device">127.0.0.1</test_value>
<test_value name="usracct.service">ssh2</test_value>
</test_values>
</example>
</examples>
<values>
<value name="usracct.type">login</value>
<value name="usracct.sessionid">$PID</value>
<value name="usracct.application">$PROGRAM</value>
<value name="secevt.verdict">ACCEPT</value>
</values>
</rule>
</ruleset>
</patterndb>