balabit.com

syslog-ng: from raw data to Big Data

Scale 14x, Los Angles

Peter Czanik / Balabit

About me

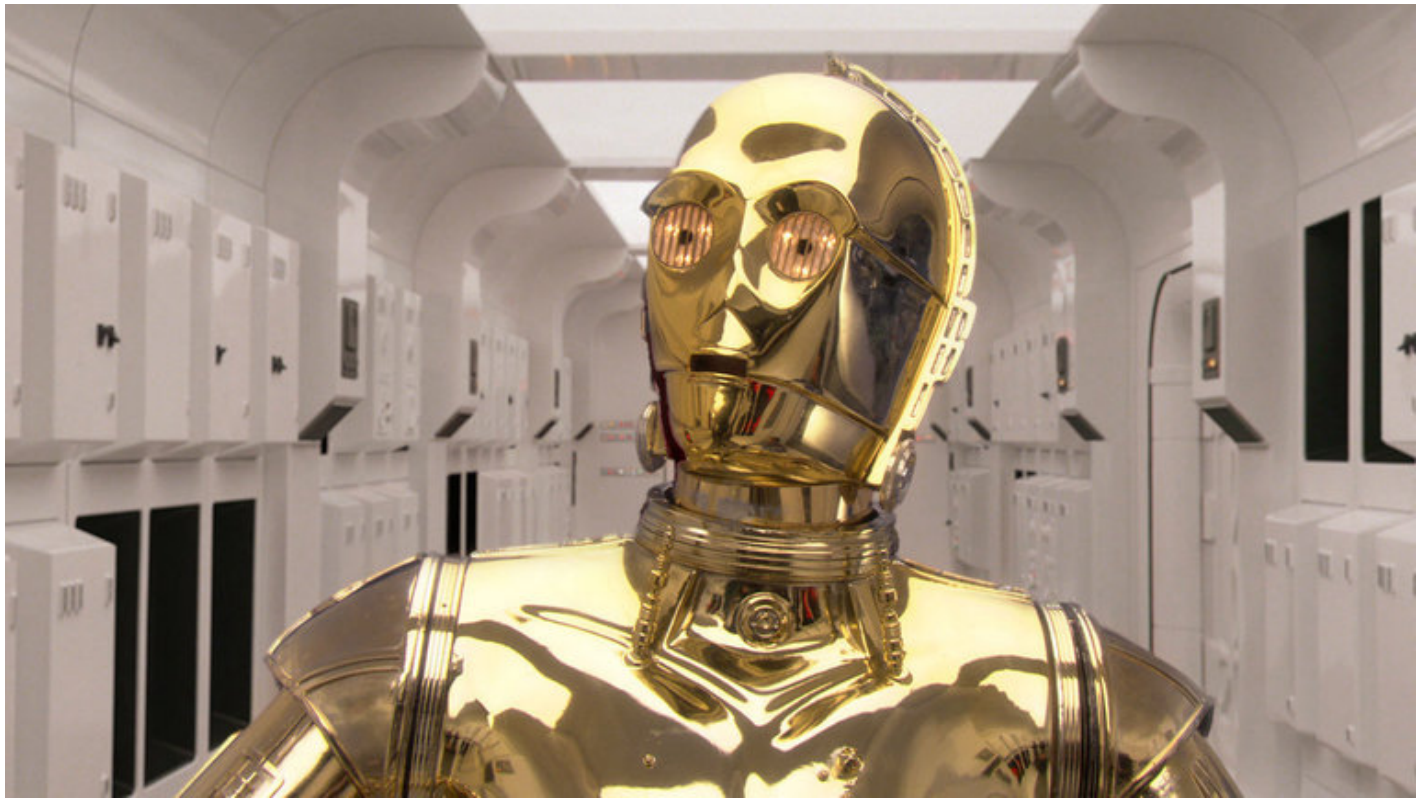
- Peter Czanik from Hungary
- Community manager at BalaBit: syslog-ng upstream
- Doing syslog-ng packaging, support, advocating

- BalaBit is an IT security company with development HQ in Budapest, Hungary
- Over 200 employees: the majority are engineers

syslog-ng

- Logging: recording events, like this one:
  - 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 central log collection.
  - Not only syslog
  - Processing and filtering messages
  - Storing to a central location or forwarding to a wide variety of destinations

C-3PO (Star Wars)



syslog-ng and Big Data

syslog-ng can facilitate the data pipeline to Big Data in many ways:

- Data collector
- Data processor
- Data filtering

syslog-ng: 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

syslog-ng: processing

Process messages close to the source: easier filtering, lower load on the consumer side

- classify, normalize and structure logs with built-in parsers:
  - CSV-parser, DB-parser (PatternDB), JSON 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

syslog-ng: data filtering

Main uses:

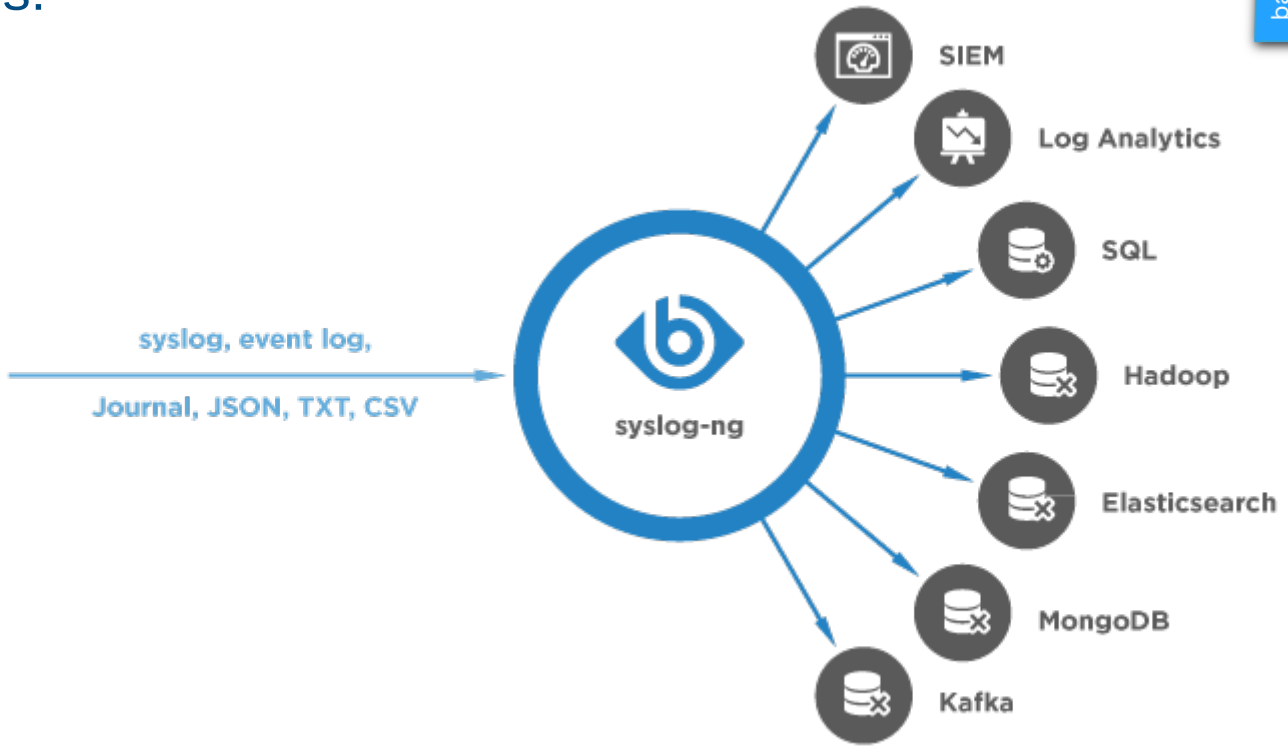
- Message routing (login events to SIEM, smtp logs to separate file, etc.)
- Throw away surplus logs (don't store debug level messages to SQL)

Many possibilities:

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

syslog-ng “Big Data” destinations

- Distributed file systems:
  - Hadoop
- NoSQL databases:
  - MongoDB
  - Elasticsearch
- Messaging systems:
  - Kafka



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:
  - source_ip=192.168.123.45
  - app=sshd
  - user=root
- 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

JSON parser

- Turns JSON based log messages into name-value pairs

- {"PROGRAM":"prg00000","PRIORITY":"info","PID":"1234","MESSAGE":"seq: 0000000000, thread: 0000, runid: 1374490607, stamp: 2013-07-22T12:56:47 MESSAGE...
","HOST":"localhost","FACILITY":"auth","DATE":"Jul 22 12:56:47"}

csv parser

- csv-parser: parses columnar data into fields

```
parser p_apache {
    csv-parser(columns("APACHE.CLIENT_IP", "APACHE.IDENT_NAME", "APACHE.USER_NAME",
        "APACHE.TIMESTAMP", "APACHE.REQUEST_URL", "APACHE.REQUEST_STATUS",
        "APACHE.CONTENT_LENGTH", "APACHE.REFERER", "APACHE.USER_AGENT",
        "APACHE.PROCESS_TIME", "APACHE.SERVER_NAME")
         flags(escape-double-char,strip-whitespace) delimiters(" ") quote-pairs("""[]')
         );
};
destination d_file { file("/var/log/messages-${APACHE.USER_NAME:-nouser}"); };
log { source(s_local); parser(p_apache); destination(d_file);};
```

PatternDB parser

- PatternDB message parser:
  - Can extract useful 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:
  - user=root, source_ip=192.168.123.45, action=login, status=failure
  - classified as “violation”

Anonymizing messages

- Many regulations about what can be logged
  - PCI-DSS: credit card numbers
  - Europe: IP addresses, user names
- Locating sensitive information:
  - Regular expressions: slow, works also in unknown logs
  - Patterndb: fast, only in known log messages
- Anonymizing:
  - Overwrite it with constant
  - Overwrite it with a hash of the original

Language bindings in syslog-ng

- The primary language of syslog-ng is C:
  - High performance: processes a lot more EPS than interpreted languages
- Not everything is implemented in C
- Rapid prototyping is easier in interpreted languages
- Python & Java destinations in syslog-ng, Lua & Perl in incubator
  - Embedded interpreter
  - Message or full range of name value pairs can be passed
  - Proper error handling

Java based “Big Data” destinations

- Most of “Big Data” is written in Java
- C and Python clients exist, but Java is official and maintained together with the server component

- More effort to get started:
  - Due to missing JARs and build tools (gradle) not yet in distributions
  - libjvm.so needs to be added to LD_LIBRARY_PATH
- https://czanik.blogs.balabit.com/2015/08/getting-started-with-syslog-ng-3-7-1-and-elasticsearch-hadoop-kafka/

Configuration

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

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

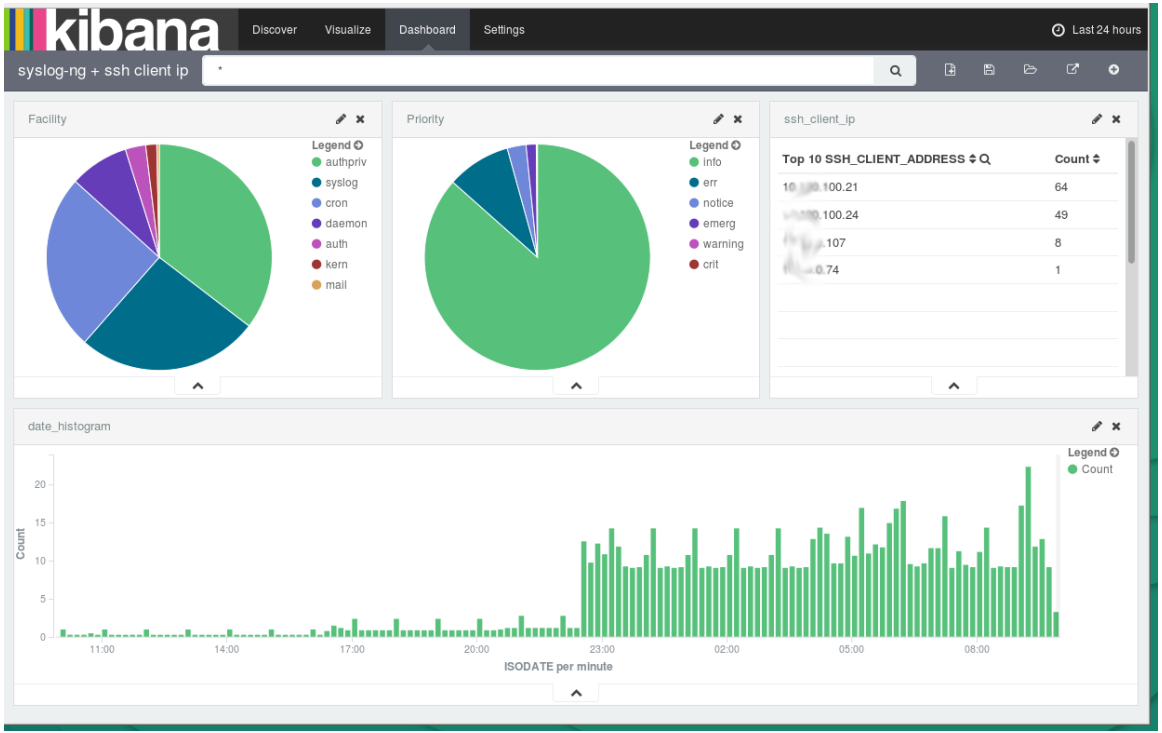
```
filter f_nodebug    { level(info..emerg); };
filter f_messages    { level(info..emerg) and
                       not (facility(mail)
                       or facility(authpriv)
                       or facility(cron)); };

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);
  filter(f_nodebug);
  parser(pattern_db);
  destination(d_es);
  flags(flow-control);
};
```

Patterndb & ElasticSearch & Kibana



Kafka

- Publish – subscribe messaging
- Data backbone for data driven organizations
  - LinkedIn
  - Spotify
- Kafka destination is already in syslog-ng
  - Source is planned

syslog-ng benefits for Big Data

- 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

Joining 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/
- IRC: #syslog-ng on freenode

- University students:
  - open trainee positions!
  - syslog-ng universe

Questions?

- Questions?

  - My blog: http://czanik.blogs.balabit.com/
  - My e-mail: peter.czanik@balabit.com

End

Sample XML

```
<?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>
```