

# syslog-ng: from log collection to processing and information extraction

2015. Libre Software Meeting, Beauvais

Peter Czanik / BalaBit

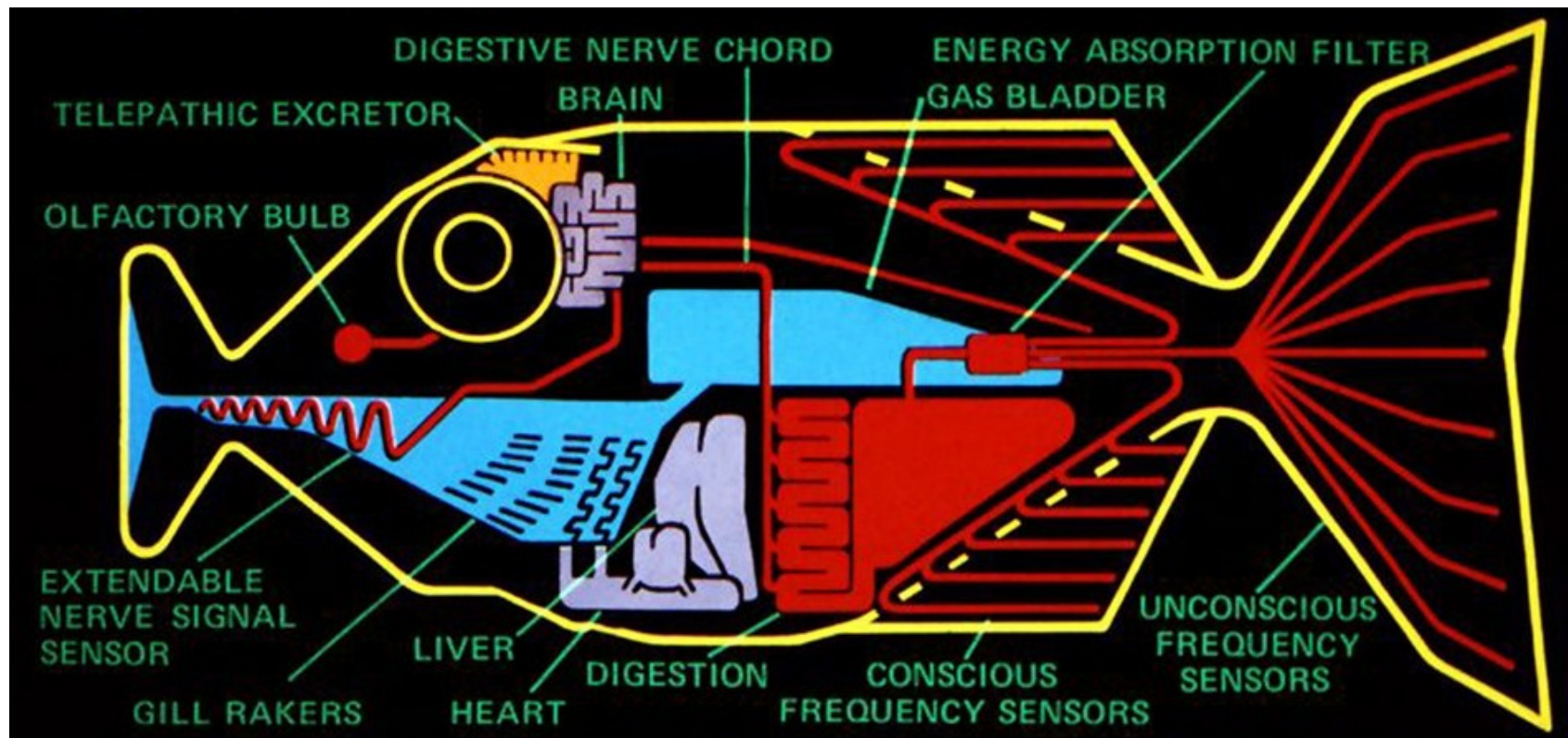
# About me

- 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 → syslog-ng

- Logging: recording events
- Jan 14 11:38:48 linux-0jbu sshd[7716]: Accepted publickey for root from 127.0.0.1 port 48806 ssh2
- syslog-ng: enhanced log daemon, with a focus on central log collection, supporting a wide range of input and output methods with a flexible configuration language

# Babel Fish (The hitchhiker's guide to the galaxy)





# syslog-ng: sources

- Receive and send RFC3164 (legacy, BSD) and RFC5424 (“new”, IETF) style syslog messages over the network
  - <34>Oct 11 22:14:15 mymachine su: 'su root' failed for lonvick on /dev/pts/8
  - <165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47 [exampleSDID@32473 iut="3" eventSource= "Application" eventID="1011"]  
BOMAn application event log entry...
- Files, sockets, pipes, etc.
- A wide variety of platform specific sources:
  - /dev/log & Co
  - Journal
  - Sun streams

# syslog-ng: processing

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

# syslog-ng: destinations

- Traditional file and UDP/TCP/TLS destinations
- SQL and NoSQL destinations (mysql, mongodb)
- Visualization (graphite)
- Alerting (riemann)
- Message queuing (RabbitMQ, ZeroMQ)
- Hadoop, Elasticsearch, Kafka and many more

# Configuration

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



# syslog-ng.conf: global options

```
@version:3.6
```

```
@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_cons { file("/dev/console"); };  
destination d_mesg { file("/var/log/messages"); };  
destination d_auth { file("/var/log/secure"); };  
destination d_mail { file("/var/log/maillog" flush_lines(10)); };  
destination d_spool { file("/var/log/spooler"); };  
destination d_boot { file("/var/log/boot.log"); };  
destination d_cron { file("/var/log/cron"); };  
destination d_kern { file("/var/log/kern"); };  
destination d_mail { usertty("*"); };
```

# syslog-ng.conf: filters

```
filter f_kernel    { facility(kern); };  
filter f_default  { level(info..emerg) and  
                  not (facility(mail)  
                      or facility(authpriv)  
                      or facility(cron)); };  
filter f_auth     { facility(authpriv); };  
filter f_mail     { facility(mail); };  
filter f_emergency { level(emerg); };  
# [...]
```

# syslog-ng.conf: logpath

```
log { source(s_sys); filter(f_kernel); destination(d_kern); };  
log { source(s_sys); filter(f_default); destination(d_mesg); };  
log { source(s_sys); filter(f_auth); destination(d_auth); };  
log { source(s_sys); filter(f_mail); destination(d_mail); };  
log { source(s_sys); filter(f_emergency); destination(d_mlal); };  
log { source(s_sys); filter(f_news); destination(d_spol); };  
log { source(s_sys); filter(f_boot); destination(d_boot); };  
log { source(s_sys); filter(f_cron); destination(d_cron); };
```



# 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

# Why it does not scale

- Information is presented differently by each application
- Few logs (workstation) → easy to find information
- Many logs (server) → difficult to find information
- 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`
- Parsers in syslog-ng can turn unstructured and some structured data (csv, JSON) into name value pairs
- syslog-ng: name-value pairs inside
  - Date, facility, priority, program name, pid, etc.
- Templates: use name-value pairs for custom file names or messages

# 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”

# 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>
```

# Creating patterns for syslog-ng: editor

- Some sample patterns available:
  - <https://github.com/balabit/syslog-ng-patterndb>
- Use an XML editor or text editor with syntax highlighting
- Use “pdbtool” to
  - test, debug
  - merge
  - convert

patterns

# Creating patterns for syslog-ng: Puppet

- More friendly format (especially if you use Puppet :-)
- <https://github.com/ccin2p3/puppet-patterndb>
- Use “pdbtool” as usual

```
patterndb::simple::ruleset { 'myruleset':  
  id => '9586b525-826e-4c2d-b74f-381039cf470c',  
  patterns => [ 'sshd' ],  
  pubdate => '2014-03-24',  
  rules => [  
    {  
      id => 'd69bd1ed-17ff-4667-8ea4-087170cbceeb',  
      patterns => ['Successful login for user @QSTRING:user:"@ using method @QSTRING:method:"@']  
    }  
  ]  
}
```

# Creating patterns for syslog-ng: GUI

PASTE MESSAGE TO CREATE PATTERN FROM

```
Accepted password for dezso from 10.50.0.247 port 42156 ssh2  
Accepted password for jozsi from 1.2.3.4 port 21 ssh2  
Accepted password for bela from 192.168.1.1 port 443 ssh2
```

Create pattern from messages

- This is a work in progress
- Finds patterns automagically from similar lines
- Fields can be edited and named
- Results can be verified



# Creating patterns for syslog-ng: GUI

## SAMPLE MESSAGE




Accepted password f **dezso** from **10.50.0.247** port **42156** ssh2

## PATTERN PREVIEW

Accepted password for @STRING:user:@ from @IPv4:src\_ip:@ port @NUMBER:port:@ ssh2

Value	Type	Name	
dezso	STRING	user	✗
10.50.0.247	IPv4	src_ip	✗
42156	NUMBER	port	✗

# Creating patterns for syslog-ng: GUI

Message with fields	Match
Accepted password for dezso from 10.50.0.247 port 42156 ssh2 user=dezso   src_ip=10.50.0.247   port=42156	 Match
Accepted password for jozsi from 1.2.3.4 port 21 ssh2 user=jozsi   src_ip=1.2.3.4   port=21	 Match
Accepted password for bela from 192.168.1.1 port 443 ssh2 user=bela   src_ip=192.168.1.1   port=443	 Match

Save pattern

# 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
  
- Lua / Perl / Python / Java destinations, Lua monitoring source
  - Embedded interpreter
  - Message or full range of name value pairs can be passed
- Java/Python moving from incubator to core in 3.7

# ElasticSearch through Java destination

- Syslog-ng 3.7 beta has Java destination (originally in the incubator)
- <https://github.com/juhaszviktor/ESDestination>

```
destination d_es {  
  java(  
    class_path("/usr/local/ESDestination.jar:/usr/share/elasticsearch/lib/*.jar")  
    class_name("org.syslog_ng.elasticsearch.ElasticSearchDestination")  
    option("index", "syslog-ng_${YEAR}.${MONTH}.${DAY}")  
    option("type", "test")  
    option("cluster", "syslog-ng")  
    option("flush_limit", "100")  
    option("custom_id", "$RCPTID")  
  );  
};
```



# Central syslog-ng management

- modules for Puppet, Salt and Ansible
- Puppet is the most tested with thousands of machines
- [https://github.com/ihrwein/puppet-syslog\\_ng](https://github.com/ihrwein/puppet-syslog_ng)
- Features:
  - Installs syslog-ng and sub-modules
  - Can configure syslog-ng with minimal limitations

# Questions? (and some answers)

- Questions?
- Some useful syslog-ng resources:
  - syslog-ng: <http://syslog-ng.org/>
  - ELSA (log analysis based on syslog-ng's patterndb):  
<http://code.google.com/p/enterprise-log-search-and-archive/>
  - Alerting: <http://devops.com/features/guide-modern-monitoring-alerting/>
  - Mailing list: <https://lists.balabit.hu/pipermail/syslog-ng/>
  - My blog: <http://czanik.blogs.balabit.com/>
  - My e-mail: [czanik@balabit.hu](mailto:czanik@balabit.hu)

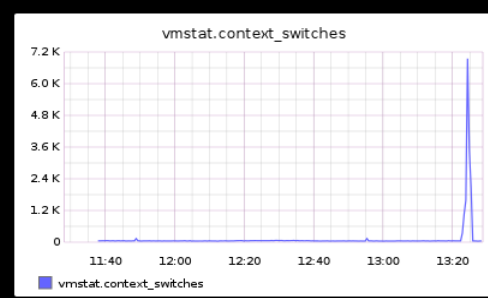
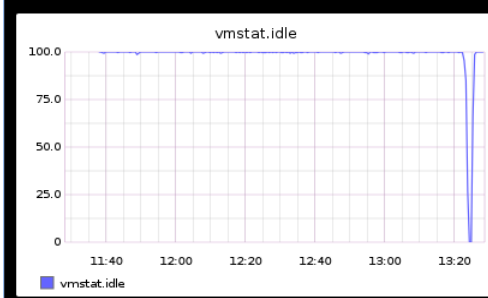
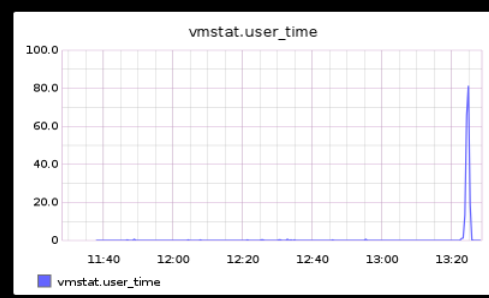
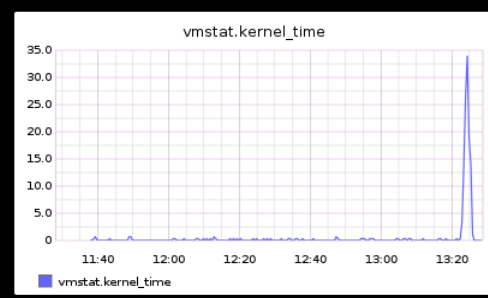
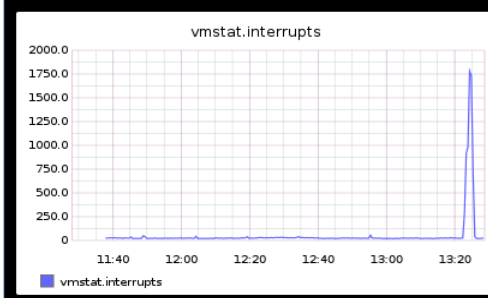
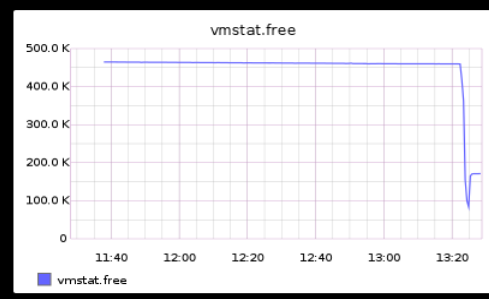
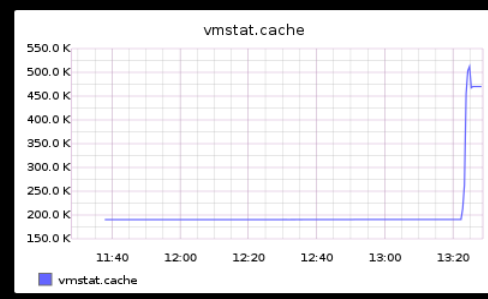
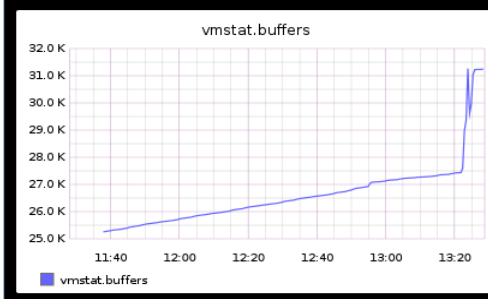
# End

# Monitoring source → Graphite

```
source s_monitor {
    monitor(monitor-freq(5) monitor-func("vmstat")
    monitor-script("/etc/syslog-ng/vmstat.lua" );
};

destination d_graphite {
    tcp( "172.16.177.139" port(2003)
    template("$$(graphite-output --key vmstat.* )" );
};

log {source(s_monitor); destination(d_graphite); };
```





# Interactive syslog-ng

- See which path a log message takes inside syslog-ng
  - Stop at break points
  - Show current state of macros
  - Built-in help and tab completion
- 
- Initial commit in syslog-ng 3.7 (beta)
  - Feedback is very welcome!