BALABIT

CONTEXTUAL SECURITY INTELLIGENCE

GET THE MOST OUT OF YOUR SECURITY LOGS USING SYSLOG-NG

Peter Höltzl / Balabit

Bsides Budapest 2017

WHO AM I?

Peter Höltzl

Senior trainer of Balabit

Former integration engineer

On the deffensive side

Linux fun for more than 20 years

peter.holtzl@balabit.com

http://www.balabit.com

MY TOPIC TODAY

What is logging? Why logging?

Problems of logging

How to process your log messages with syslog-ng

How to use it

WHY LOGGING?

Logging is to see what happened in the past in order to:

- Find errors
- Fine-tune the system
- Find anomalies and irregularities
- Find malicious activity

Internal need

Regulation / compliance

WHY CENTRAL LOGGING?

Ease of use

- Single place insted of many
- Better reports, better better correlation
- Log consumers remain interchangable

Availability

- Event the sender is down

Security

- Syslog is a one-way communication
- Cannot modify if sender is compromised (log poisoning still possible)

PROBLEMS OF LOGGING

The protocols:

- Large variety of transport: RFC3164, RFC3194, RFC5424 family etc.
- Other transports: EVTX (over RPC), SQL, REST, SNMP and vendor specific

The message format:

- Free text --> Human readable
- Structured text --> Machine readable

The content:

- What shall be logged? (No best practice, no standard)

STRUCTURED MESSAGE FORMATS

RFC5424 structured data

Webtrends Enhanced Logfile Format (WELF)

EVT(X)

ArcSight Common Event Format (CEF)

JSON

Extended Logfile Format by W3C

Common Event Expression (CEE) by MITRE

...and many other

STRUCTURED MESSAGE EXAMPLES

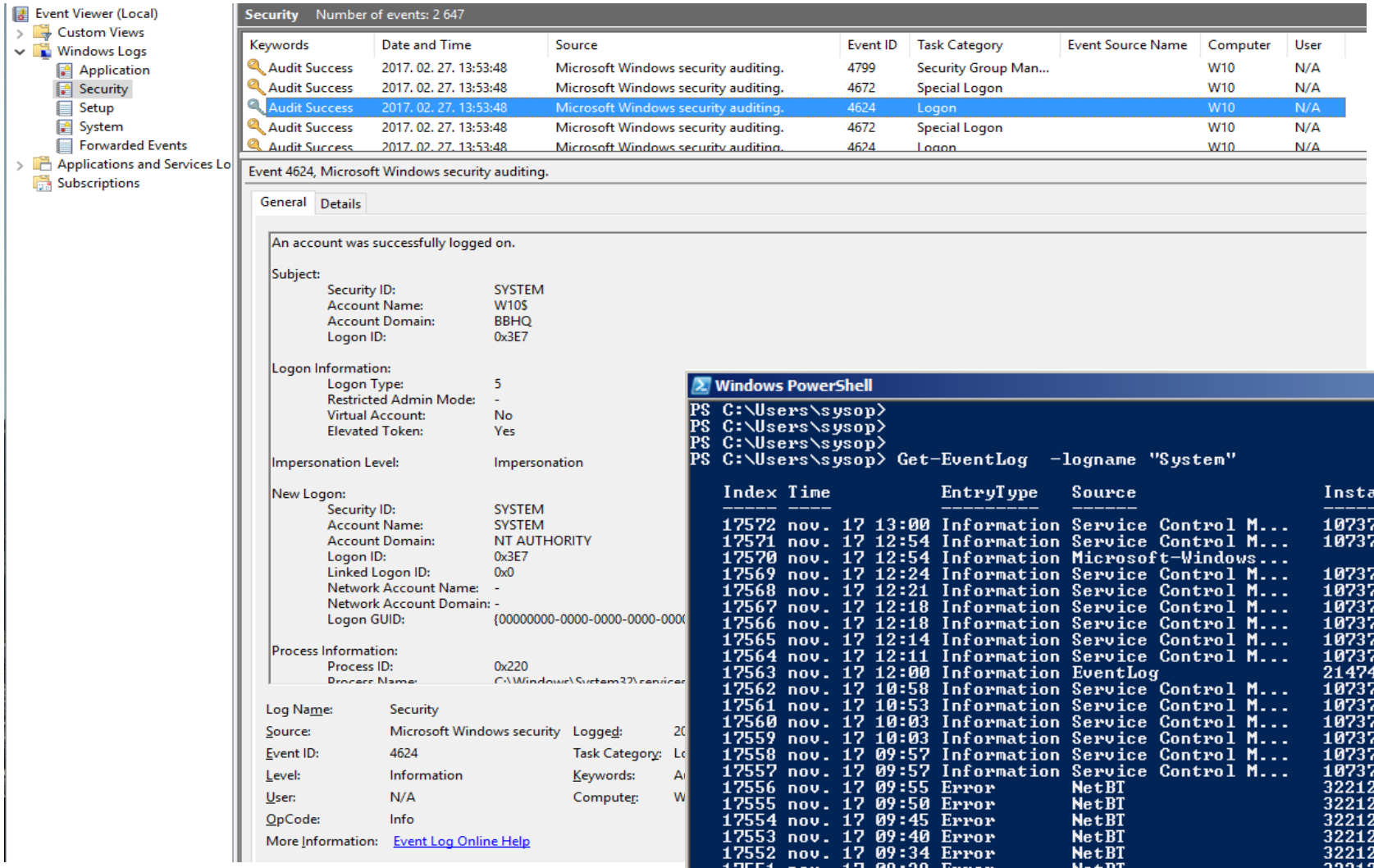
- RFC5425 SDATA field

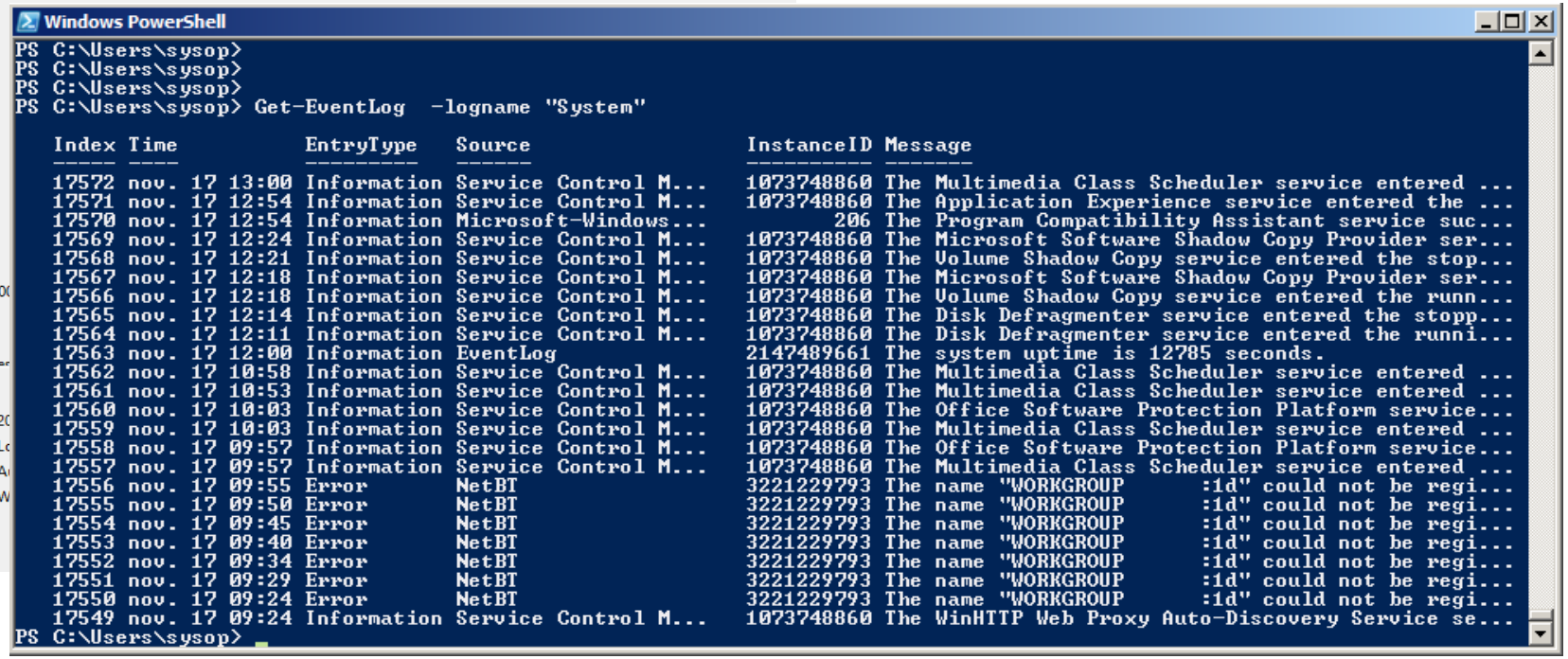
```
[win@18372.4 EVENT_CATEGORY="Logon" EVENT_FACILITY="16"
EVENT_ID="4624" EVENT_LEVEL="0" EVENT_NAME="Security"
EVENT_REC_NUM="278198" EVENT_SID="N/A"
EVENT_SOURCE="Microsoft Windows security auditing."
EVENT_TASK="Logon" EVENT_TYPE="Success Audit"
EVENT_USERNAME="DEMO\\user"][meta sequenceId="4027"
sysUpTime="670"]
```

- Webtrends Enhanced Logfile Format (WELF)

```
id=firewall time="2017-03-02 12:01:01" fw=192.168.0.238
pri=6 rule=3 proto=http src=192.168.0.23 dst=192.168.1.12
```

STRUCTURED MESSAGE EXAMPLES





BALABIT

STRUCTURED MESSAGE EXAMPLES

- ArcSight Commin Event Format (CEF)

```
CEF:0|ArcSight|ArcSight|6.0.3.6664.0|agent:030|Agent [test] type [testalertng]
started|Low|eventId=1 mrt=1396328238973 categorySignificance=/Normal
categoryBehavior=/Execute/Start categoryDeviceGroup=/Application catdt=Security
Mangement categoryOutcome=/Success categoryObject=/Host/Application/Service
art=1396328241038 cat=/Agent/Started eviceSeverity=Warning rt=1396328238937
fileType=Agent cs2=<Resource D\="3DxKlG0UBABCAA0cXXAZIwA\=\="/>
c6a4=fe80:0:0:0:495d:cc3c:db1a:de71 cs2Label=Configuration Resource c6a4Label=Agent
IPv6 Address ahost=SKEELES10 agt=888.99.100.1 agentZoneURI=/All Zones/ArcSight
System/Private Address Space Zones/RFC1918: 888.99.0.0-888.200.255.255 av=6.0.3.6664.0
atz=Australia/Sydney aid=3DxKlG0UBABCAA0cXXAZIwA\=\= at=testalertng dvchost=SKEELES10
dvc=888.99.100.1 deviceZoneURI=/All Zones/ArcSight System/Private Address Space
Zones/RFC1918:  888.99.0.0-888.200.255.255 dtz=Australia/Sydney _cefVer=0.1
```

- JSON

```
{"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"}
```

FREE TEXT MESSAGE FORMATS

- RFC3164:

```
<123>Mar 2 10:10:10 server sshd[123]: Accepted
password for peter from 192.168.56.1 port 36858 ssh2
```

- RFC5424 with an almost empty SDATA:

```
<123>1 2017-03-02T10:10:10.01+01:00
server.mycompany.lan sshd 123 ID321 [file@18372.4
.classifier.class=unknown]Accepted password for peter
from 192.168.56.1 port 36858 ssh2
```

CONCLUSION

The future is Structured logging, but:

- Too many format
- Project Lumberjack (R.I.P. 2014)
- MITRE CEE (R.I.P. 2014)
- RFC5424 (Born 2009, penetration <1%)
- Free text (unstructured still widespread)

We need hacking!

WHAT IS SYSLOG-NG?

First release 1997 (yes! It was 20 years ago!)

Dual licensed (have a free version)

Supported platforms:

Linux, HP-UX, BSD, AIX, Solaris, Microsoft Windows, As/400

Used in many embedded system (Amazon Kindle, BMW i3)

HOW SYSLOG-NG CAN HELP?

Collect

Process

Filter

Store (&Forward)

COLLECTION

All RFC protocols are supported

Large variety of non-RFC, or RFC variants

EVT(X) API (commercial only)

SQL fetching

ROLES OF SYSLOG-NG

Client/Agent:

- Collect local and forward

Relay:

- Collect and forward (with disk based buffer)
- Protocol conversion
- Format conversion
- Filter and pre-process

Server:

- Collect and store

PROCESSING

Parse, classify and restructure

- Built in parsers (protocol parsers)
- Custom parsers: CSV, JSON, KEY-VALUE, PATTERN-DB (Radix tree)

Reformat

- Reorder, normalize (date to ISODATE), WELF formatter

Rewrite and normalize

- Anonymize/Pseudonimize
- Remove message parts (MASK credit card info)

Enrich data

- Add GeoIP or other from CSV files

THE CSV PARSER

Input:

```
127.0.0.1 - frank [02/Mar/2017:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
"http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"
```

The parser:

```
csv-parser(columns(".apache.clientip", ".apache.identname", ".apache.user", ".apache.ts",
".apache.url", ".apache.status", ".apache.contentlength", ".apache.referer", ".apache.useragent")
flags(escape-double-char,strip-whitespace) delimiters(" ") quote-pairs('""[]'));
```

The output:

```
[.apache.clientip=127.0.0.1 .apache.identname=- .apache.user=frank
.apache.ts=[02/Mar/2017:13:55:36 -0700] .apache.url="GET /apache_pb.gif HTTP/1.0"
.apache.status=200 .apache.contentlength=2326 .apache.referer=
"http://www.example.com/start.html" .apache.useragent=" Mozilla/4.08 [en] (Win98; I ;Nav)"]
```

Usage:

```
${apache.user}
```

THE K-V PARSER

Input:

```
<123>Mar  2 10:10:10 myhost myapp[123]: a=12 b=15 c=22 d=fixme
```

The parser:

```
parser p_kv { kv-parser(value_separator("=") prefix(".kv.")
template("${MESSAGE}"));};
```

The output:

```
[.kv.a=12 .kv.b=15 .kv.c=22 .kv.d=fixme]
```

Usage:

```
${.kv.b}
```

THE JSON PARSER

Input:

```
{"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"}
```

The parser:

```
Parser p_json {json-parser(prefix(".json."));};
```

The Output:

```
[.json.PROGRAM=prg00000 .json.PRIORITY=info .json.PID=1234 ...
```

Usage:

```
${.json.PROGRAM}
```

THE DB-PARSER

Input:

```
Accepted password for peter from 1.2.3.4 port 567 ssh2
```

The parser:

```
Accepted @STRING:.ssh.auth:@ for @STRING:.ssh.uid:@ from @IPv4:.ssh.ip:@ port @NUMBER:.ssh.port:@
ssh2
```

Output:

```
[.classifier.class=system .ssh.auth=password .ssh.uid=peter .ssh.ip=1.2.3.4 .ssh.port=567]
```

Usage:

```
${.ssh.uid}
```

ENRICH BY GEO-IP

The parser:

```
parser p_ssh_geoip { geoip("${.ssh.ip}", prefix(".geoip.")
database("/var/lib/geoip-database-contrib/GeoLiteCity.dat"));
};
```

The output:

```
[.classifier.class=system .ssh.auth=password .ssh.uid=peter
.ssh.ip=207.46.13.167 .ssh.port=567 .geoip=47.680099,-
122.120598]
```

ENRICH BY GROUP INFO

The parser:

```
parser p_add_context_data { add-contextual-data(selector("${.ssh.uid}"),
database("context-info-db.csv"), prefix(".metadata") default-selector("no-uid"));};
```

The database:

```
peter;serveradmin

no-uid;no-group
```

The Output:

```
[.classifier.class=system .ssh.auth=password .ssh.uid=peter .ssh.ip=207.46.13.167
.ssh.port=567 .geoip=47.680099,-122.120598 .metadata=serveradmin]
```

REWRITE MESSAGE CLASS

Input:

```
Accepted password for peter from 1.2.3.4 port 567 ssh2

Accepted password for root from 4.3.2.1 port 765 ssh2
```

The rewrite:

```
rewrite r_violation {

set("violation" value(".classifier.class") condition(match("root" value(".classifier.uid"))));

};
```

Output:

```
[.classifier.class=system .ssh.auth=password .ssh.uid=peter .ssh.ip=1.2.3.4 .ssh.port=567
.geoip=47.680099,-122.120598 .metadata=serveradmin]

[.classifier.class=violation .ssh.auth=password .ssh.uid=root .ssh.ip=4.3.2.1 .ssh.port=765
.geoip=47.680099,-122.120598 .metadata=goduser]
```

ANONYMIZE MESSAGES

The rewrite:

```
rewrite r_anon { set("HIDDEN") template("${.ssh.uid}");
set("HIDDEN") template("${.ssh.ip}"); };
```

The output:

```
[.classifier.class=system .ssh.auth=password .ssh.uid=HIDDEN
.ssh.ip=HIDDEN .ssh.port=567 .geoip=47.680099,-122.120598
.metadata=serveradmin]
```

Problems:

- Kills log analysis (no more correlation)

PSEUDONIMIZE MESSAGES

The rewrite:

```
rewrite r_uid { subst(".*", "$(sha1 $0)", value(".ssh.uid"));
subst(".*", "$(sha1 $0)", value(".ssh.ip"));};
```

The ouput:

```
[.classifier.class=system .ssh.auth=password
.ssh.uid=5d2c6a9b917d0dce3cbd4dc4c0626c56f6cf9298
.ssh.ip=bd4dc4c0626c56f6cf9295d2c6a9b917d0dce3c8 .ssh.port=567
.geoip=47.680099,-122.120598 .metadata=serveradmin]
```

REWRITE FREE TEXT MESSAGES

Input:

```
Accepted password for peter from 1.2.3.4 port 567 ssh2
```

The rewrite:

```
rewrite pseudonymize_ip_addresses_in_message {

subst("((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5]))", "$(sha1 $0)", value("MSG"), flags(global)

  );};
```

Output:

```
Accepted password for peter from 5d2c6a9b917d0dce3cbd4dc4c0626c56f6cf9298 port 567 ssh2
```

REWRITE FREE TEXT MESSAGES

Input:

```
<123>Mar  2 10:10:10 www myapp[123]: Payment done by
378282246310005 (AMEX).
```

The rewrite:

```
rewrite { credit-card-mask(value("MSG")); };
```

The output:

```
<123>Mar  2 10:10:10 www myapp[123]: Payment done by
378282******005 (AMEX).
```

REWRITE FREE TEXT MESSAGES

Input:

```
<123>Mar  2 10:10:10 www myapp[123]: Payment done by
378282246310005 (AMEX).
```

The rewrite:

```
rewrite { credit-card-hash(value("MSG")); };
```

The output:

```
<123>Mar  2 10:10:10 www myapp[123]: Payment done by
5d2c6a9b917d0dce3cbd4dc4c0626c56f6cf9298 (AMEX).
```

FILTER

Discard noise (SIEM prefiltering)

Filter based on

- Message content (any parsed parts)
- Parameters, macros or name-value pairs
- Any SDATA part
- Comparsions
- Can use regular expression
- Combinations of the above

STORE AND FORWARD

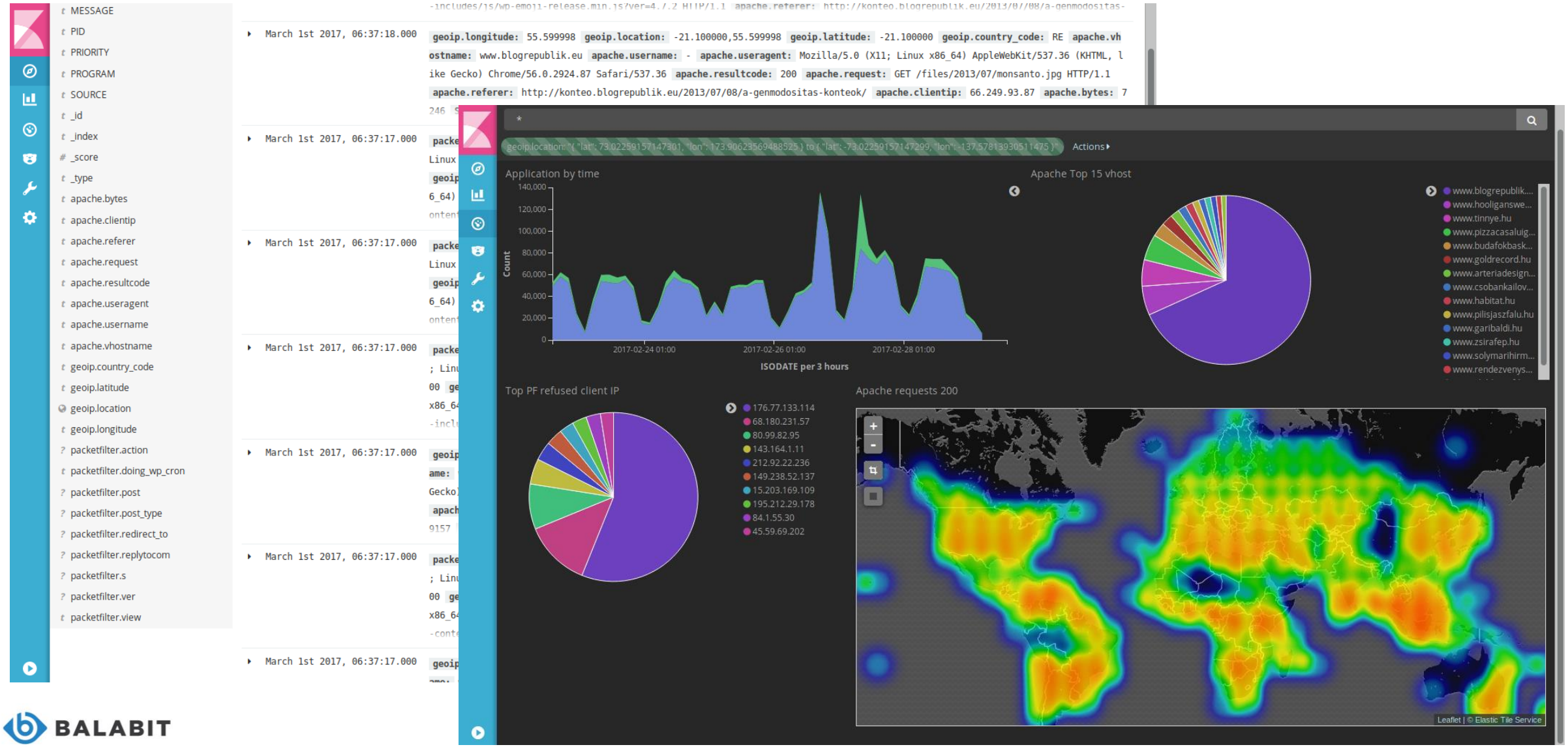
"Small" data and traditional:

- File, syslog (RFC3164, RFC5424) or SQL INSERT

"Big data":

- Hadoop
- MongoDB
- Elasticsearch
- Kafka messaging system

HOW TO USE SDATA?



BALABIT

SUMMARY AND Q&A

Syslog is a versatily data

- Many formats (free text or structured data)
- Syslog-ng is a tool that help
  - collect, parse
  - rewrite/reformat
  - store and forward in a good way;-)