Reliable log data transfer

About (r)syslog, logstash, and log data signing

A field report

pascal.buchbinder@adnovum.ch
• Why we need log data transfer
• Syslog
  – UDP vs TCP
  – Necessary tools (for Apache httpd)
  – Reliability
• Logstash
• Log data signing (Apache httpd, Logstash)
• Many distributed systems
• Need to collect log data centralized
  – Prevention from data loss / manipulation
  – Archiving (transaction audit, PCIDSS)
  – Alerting / monitoring
  – Viewing / troubleshooting
  – Statistics (planning, trends, anomaly)
  – Reporting / DWH
  – Accounting
• Reliability: Are we losing any messages?
• **Syslog**
  – Embedded into your software: direct data transfer (via local syslogd or direct connection) to loghost.
    • Software usually available on every (Unix) host.
  – Standardized protocol, format, levels, facilities, etc.

• **Proprietary**
  – External daemon: usually appending to files and forwarding the data to the loghost.
    • Software needs to be installed on every host.
  – Vendor specific software, configuration, and protocol (server and client side).
Setup syslog

Apache httpd → qslogger → rotatelogs → file

syslogd

file → syslogd

logstash → Elasticsearch
• Piped logging

• qslogger:
  – Writes data to syslogd and stdout (local file)
  – Filtering by severity (don't forward debug messages to loghost)
  – Severity detection: set message's level at syslog protocol
• UDP has less overhead (faster)
• Non-blocking
• No flow control

• Plaintext:
  – No confidentiality
  – No key management

• TCP has more overhead (slower)
• Blocking
• Flow control and error handling

• Encrypted:
  – Confidentiality
  – Key management
• **UDP: Potential data loss.**
  – If your server (receiver) becomes too busy (thousands of log messages per second from many clients).
  – When your server (receiver) is down.

• **TCP: No data loss.**
  – As long as your server (receiver) is available.
  – You may configure buffers (memory or file) to store messages temporarily if receiver is not available.
  – Syslog is still non-blocking, even we lose messages due to full buffers or an unavailable receiver.
• Priority: rsyslogd may drop low priority (level) messages preferring high priority ones.

```bash
$SystemLogRateLimitInterval 2
$SystemLogRateLimitBurst 5000
$SystemLogRateLimitSeverity 6
```

– **Know your infrastructure's limitation!**
– Decide whether messages with low severity shall have lower priority.
– Mainly for UDP setup.
• Cluster: setup a primary and secondary log host.

  local3.* @@logmaster.adnovum.ch
  $ActionExecOnlyWhenPreviousIsSuspended on
  @@logslave.adnovum.ch
  & /var/log/syslogbuffer
  $ActionExecOnlyWhenPreviousIsSuspended off

  – Allows you to maintain your log host (receiver), e.g. reboot the server.
  – Makes only sense within a setup using TCP.
• Open Source (Apache license)
• Locally installed daemon
• Appends to files
  – Free buffers: buffering works event when files are rotated while reading (no message loss until a rotated file gets deleted before the data has sent)
• Forward data using the lumberjack protocol
• Supports 2-way SSL (mutual authentication)
Logstash forwarder setup

OWASP
The Open Web Application Security Project

Apache httpd ➔ rotatelogs ➔ file ➔ logstash forwarder

file ➔ logstash ➔ Elasticsearch
• How do you know not loosing log data?
  – Easy in lab while testing (by counting messages).
  – Difficult in real production environment.

• Signing log data solves this problem:
  – Each message is signed (corrupt messages).
  – Sequence number shows data loss (or injected messages).
• Sign each message using a dedicated tool (qssign, piped logging, PSK)
  – Adds sequence number and signature.
• Verify signature by using a logstash filter plug-in
  – Verifies each message at server (receiver): signature and sequence number.
  – Potential problem: checking sequence within a cluster setup (switching multiple times).
<table>
<thead>
<tr>
<th>method</th>
<th>request</th>
<th>status</th>
<th>duration</th>
<th>sequence</th>
<th>signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/a/8.jpg</td>
<td>200</td>
<td>258</td>
<td>0000000000043</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/5.jpg</td>
<td>200</td>
<td>323</td>
<td>0000000000042</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/9.jpg</td>
<td>200</td>
<td>77</td>
<td>0000000000041</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/6.jpg</td>
<td>200</td>
<td>161</td>
<td>0000000000040</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/2.jpg</td>
<td>200</td>
<td>161</td>
<td>0000000000039</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/4.jpg</td>
<td>200</td>
<td>156</td>
<td>0000000000038</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/3.jpg</td>
<td>200</td>
<td>145</td>
<td>0000000000037</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/1.jpg</td>
<td>200</td>
<td>132</td>
<td>0000000000036</td>
<td>valid</td>
</tr>
<tr>
<td>GET</td>
<td>/a/7.jpg</td>
<td>200</td>
<td>74</td>
<td>0000000000035</td>
<td>valid</td>
</tr>
</tbody>
</table>
filter {
grok {
match => [ "message", "%{GREEDYDATA:data} %{INT:sequence}#{%{NOTSPACE:hmac}}" ]
tag_on_failure => [ ]
}
if [data] {
qssign {
message => "data"
source => "path"
sequence => "sequence"
hmac => "hmac"
secret => "/var/opt/keys/keypass.sh"
}
mutate {
replace => [ "message", "%{data}" ]
remove_field => [ "data" ]
}
} else {
mutate {
add_field => [ "signature", "missing" ]
}}}}
• Syslog
  – Usually works “out of the box”
  – UDP works well under “normal” conditions
  – No data loss when using a TCP and a cluster/buffers

• Logstash forwarder
  – Additional software, configuration, certificates
  – No data loss even your log host is temporary down

• Signatures
  – Know when loosing data
  – No message injection or manipulation
- http://www.rsyslog.com/
- https://www.elastic.co/products/logstash