IntelMQ hands-on workshop

TF-CSIRT/FIRST meeting Malaga 2020/1/31

Aaron Kaplan Kaplan@cert.at
Overview of today

- Introduction presenters
- Check: are we ready (VMs)?
- Lesson 1: Theory section (1h)
- Hands-on Lesson 2
- Break (10:30 – 11:00)
- (Cont.) Lesson 2
- IntelMQ future
- (in parallel for the fast ones Lesson 3)
Introduction presenters
Are we ready?

- VirtualBox + VM
- Network connectivity
Theory part
History & background

• In the beginning, there was Abusehelper (~ 2013). Still around for some teams
  • Too complex (at that time) for us, too expensive, semi-open source, hard to get PRs upstream, etc.
  • CERT.pt (Tomas Lima) created initial IntelMQ version in ~ 2014
  • “works”
  • Aaron Kaplan brings in IntelMQ to CERT.at, CERT.at takes over maintainership
  • Many steps to change the PoC to production ready code
• Emphasis on open source (via github.com)
• Emphasis on KISS
The IntelMQ promise

IntelMQ follows the following basic meta-guidelines:

- Don't break simplicity – **KISS (think Lego blocks for IT security automation)**
- Keep it open source - forever
- Strive for perfection while keeping a deadline
- Reduce complexity/avoid feature bloat
- Embrace unit testing
- Code readability: test with unexperienced programmers
- Communicate clearly
The community project

• At the heart of all of this is: IntelMQ is a community project
• Open source for ever (AGPL v3)
• Many contributions (thanks to CSIRT.cz, .SK, .PT, BSI/CERT-Bund, etc. etc.)
• Maybe one day, you will join us?
• It’s easy if you try
The extended community project - IHAP

- **Incident Handling Automation Projects**
- Regular meetings of tool developers for incident handling automation
- Next meeting in Vienna ~ April 2020 (TBA)
- **Mailing-list:** ihap@lists.trusted-introducer.org
- Subscription? → ping Kaplan@cert.at
CERT.at’s role in IntelMQ

- Maintainer / steward of the project
- Contribute code
- Code submission QA review
- Release mgmt
- Coordination
- Architecture design of future versions (with the community)
Okay, so what is IntelMQ?
TL;DR Version (1)

• A framework (python) of “lego blocks” of IT security CERT automation.
• Lego blocks are simple (KISS principle).
• Lego blocks can be re-combined as you need.
• You are missing a block? -> Look, if someone already wrote it. If not, write it and share with the community ➔ synergy effects
• Every CERT has its own workflow.
• Blocks can be connected with each other to create “flow” via a MQ
• Data-flow oriented architecture
• The stuff that “flows” is log lines (“events”)
• Similar to unix pipelines
Use-cases: what do we need to solve?

• A CERT receives tons of data feeds (shadowserver, etc.)
• There is no single format for all of that → write parsers for everything(?)
• Need to process, filter, verify, ...
• send out to constituency and/or
• React on the data feed (firewall blocks, etc.)

• → IntelMQ is the glue for this streaming data. It connects.
Differentiation from MISP

• MISP was meant to
  • share IoCs amongst analysts (especially APTs etc)
  • **Correlation** is a key feature
  • Lower volume

• IntelMQ was made to as an ETL (Extract Transform Load) framework for
  • High throughput / high volume
  • No Correlation
  • For parsing all the feeds, a CERT might receive, process it a bit and send it on.
Terminology

• “feed”
  • Streaming or
  • Download
• “Bot”
• “Botnet”
• ”Parser”
• “Collector”
• “Expert”
• “Output”
• “Report”
• “Event”
• “DHO” – Data Harmonization Ontology (== internal format for events)
Terminology in Detail

• Bot = Small python script which
  • Inherits from the Bot class
  • Implements an init() method
  • Implements a process() method

• Botnet = a collection of bots. A set of DAGs of bots.
• Pipeline = the structure of the MQs which connect the bots
Example bot

```python
class MyBot(Bot):
    def init(self):
        # optional initialization
    def process(self):
        event = self.receive_message()  # dict
        # process event
        self.send_message(event)
```
Pipelines
Terminology: types of Bots

- Collector → emits Report
- Parser → emits Events
- Expert → enriches / filters
- Output → sends it somewhere

- Examples for Experts:
  - filter by country code (expert)
  - Add ASN by IP address

- Examples for Output:
  - Write to MongoDB, PostgreSQL DB
  - Write to Elastic Search
  - Send to Ticket System
  - Send via email
Types of bots
Types of Messages

• Report:
  • Data (base64 combined) + metadata
  • Example: the whole blocklist from spamhaus

• Event:
  • A report gets split up into individual log lines (== “Events”)
  • An event is in the DHO format
  • Following bots in the pipeline can rely on the format
<table>
<thead>
<tr>
<th>Source</th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>source.abuse_contact</td>
<td>LowercaseString</td>
<td>Abuse contact for source address. A comma separated list.</td>
</tr>
<tr>
<td>Source</td>
<td>source.account</td>
<td>String</td>
<td>An account name or email address, which has been identified to relate to the source of an abuse event.</td>
</tr>
<tr>
<td>Source</td>
<td>source.allocated</td>
<td>DateTime</td>
<td>Allocation date corresponding to BGP prefix.</td>
</tr>
<tr>
<td>Source</td>
<td>source.as_name</td>
<td>String</td>
<td>The autonomous system name from which the connection originated.</td>
</tr>
<tr>
<td>Source</td>
<td>source.asn</td>
<td>ASN</td>
<td>The autonomous system number from which originated the connection.</td>
</tr>
<tr>
<td>Source</td>
<td>source.domain_suffix</td>
<td>FQDN</td>
<td>The suffix of the domain from the public suffix list.</td>
</tr>
<tr>
<td>Source</td>
<td>source.fqdn</td>
<td>FQDN</td>
<td>A DNS name related to the host from which the connection originated. DNS allows even binary data in DNS, so we have to allow everything. A final point is stripped, string is converted to lower case characters.</td>
</tr>
</tbody>
</table>

Configuration parameters

```
user@malaga:~$
user@malaga:~$
user@malaga:~$ cd /opt/intelmq/etc/
user@malaga:/opt/intelmq/etc$ ls -al
```
```
Configuration parameters

• JSON files
  • Runtime.conf – bots parameters
  • Pipeline.conf – the pipeline setup
Questions on the theory?