

# IntelMQ - a KISS incident handling automation project (IHAP)

L. Aaron Kaplan kaplan@cert.at    **Sebastian Wagner**  
**wagner@cert.at**    Tomás Lima tomas.lima@cert.pt

2015-11-21

# Overview

- 1 cert.at
- 2 Motivation
- 3 Intro to IntelMQ
- 4 History
- 5 Architecture and data flow
- 6 Installation
- 7 Writing a bot
- 8 Next steps/future

- national cert and government cert (govcert)
- project of nic.at
- awareness and warnings
- incident response
- responsible for austria
- no obligation to inform us
- not an authority
- coordination, contacts, knowledge, trust

# Motivation

- Handling automated collected incident intelligence
- We receive filtered data directly (by country) via mail
  - botnet drones, vulnerable servers (open resolvers, ntp) etc.
- We collect non-public and public data
  - c&c servers, spam, brute-force, etc.

## Motivation (2)

- Process data *mostly* automatically!
- Ensure accuracy
- Enrich data (AS, geolocation)
- Filter data (for AT, don't complain too often)
- Find responsible contacts
- notify responsible persons

# Intro to IntelMQ

- IHAP = **I**ncident **H**andling **A**utomation **P**roject. Our overall project name.
  - A project of multiple national CERTs (Trusted Introducer):  
<https://www.trusted-introducer.org/>
- IntelMQ = Threat **I**ntel feeds + **M**essage **Q**ueueing system. A concrete tool.
- Idea and architecture inspired by Abusehelper
- Data flow oriented toolkit to:
  - Automatically collect & handle events/incidents
  - Process and enrich these events
  - Send them to some output, automatic actions

## Intro to IntelMQ (2)

- Based on message queues (“MQ”) – *redis*, RabbitMQ, zmq
- Fast
- Very easy to extend
- GUI interface to create pipelines / modify dataflow (“intelmq-manager”)
- configuration
- management
- monitoring

# History

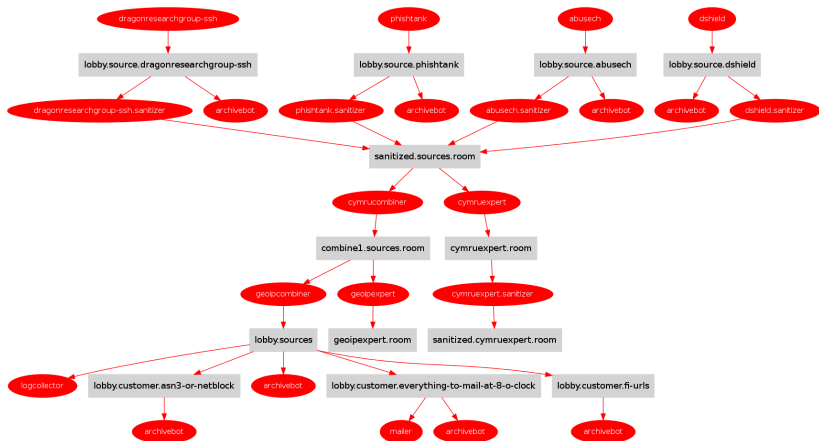
- CERT.at started with Abusehelper (open source)
- Our problem with AH: everything is co-routine orientated. That's hard to debug. Many CERTs either:
  - 1 give up or
  - 2 if they have the money buy Abuse-SA (commercial, closed source).
- For CERT.at it was too expensive so we needed to stay with the standard open source version.



# The Abusehelper Framework

- Strengths:
  - nice flow-oriented architecture
  - lots of existing bots to fetch data
  - loosely de-coupled: in **theory** easy to write new “bots” and extend Abusehelper
- open source
- Issues/Weaknesses:
  - code complexity. Are you a python guru?
  - Getting code upstream to maintainer is hard
  - hard to understand the dataflow
  - resource-hog => how to improve on this?
  - no standard way to include into ticket systems like RTIR/OTRS
  - data loss when message queue crashes

# The Abusehelper Framework



# Alternatives to Abusehelper?

- Megatron: open source, Java. Aware of two CERTs using it <https://github.com/cert-se/megatron-java>
- n6: CERT.pl <http://n6.cert.pl/>
- CIF: USA <http://csirtgadgets.org/>
- Warden: <https://wardenw.cesnet.cz/>
- overview: <https://www.cert.pl/PDF/MP-IST-111-18.pdf>

# Requirements analysis after the Heraklion meeting 5/2014

- Reduce the complexity of system administration
- Reduce the complexity of writing new bots for new data feeds
- Reduce the probability of events lost in all process with persistence functionality (even system crash)
- Use and improve the existing “Data Harmonization Ontology“ (= Abusehelper internal key-value standard)
- Use JSON format for all messages
- Integration of the existing tools (n6, AbuseHelper, CIF)
- Provide easy way to store data into Log Collectors like ElasticSearch, Splunk and DBs

# Summer sprint 2014

- IntelMQ beta 1 is the result of a sprint July-~Oct 2014.  
Persons: Tomás, Mauro, Aaron, Cosmin, ...
- Ideas:
  - KISS! (Keep it simple stupid)
  - Very similar architecture as AH, just more modern tools
  - Message Queues (redis, amq, zmq)
  - Goal: it takes 15 minutes till 1d to create a new bot (without prior knowledge!)
  - Open Source for ever – no separate commercial version
  - Python != config language! We want a simple config (GUI!)
  - Connectable with n6, AH, CIF, syslog, Elastic Search, Splunk,  
..

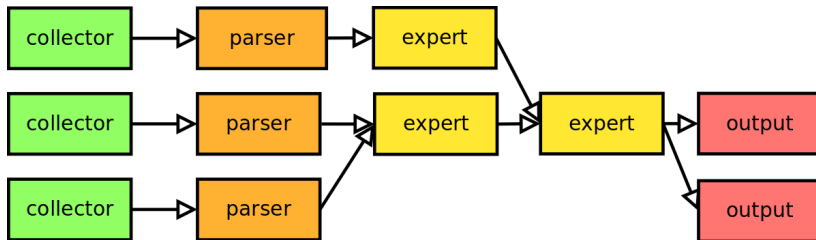
# IntelMQ @ hack.lu 2014

- Very first public presentation and open source version
- Test with Fyodor (Taiwan Uni): 15 minutes explanation of code + the next morning he had a hfeeds bot. It is simple.

# IntelMQ components

- individual and specialized bots
- Config files: JSON:
  - runtime.conf - runtime parameters of bots
  - startup.conf - which bots to start
  - BOTS = templates for all bots
  - pipeline.conf - describes how bots are connected
  - harmonization.conf - data description (field names and types)
- *Redis*, zmq, RabbitMQ or \*-MQ as message queue
- Lib/{bot.py, pipeline.py, message.py}
- Web-GUI: IntelMQ-Manager: JS + CSS + AJAX
- Outputs: Elastic Search or Postgresql or iptables ... \$foo

# IntelMQ dataflow: bots





## data examples: raw report

```
# Example feed as report
# List of malware infections
# ...
# address , time in utc , malware
192.0.2.45 , 2015-11-15 15:54:41 , feodo
203.0.113.89 , 2015-11-20 02:51:14 , zeus_p2p
```

We have an address, source timestamp, and the incident type

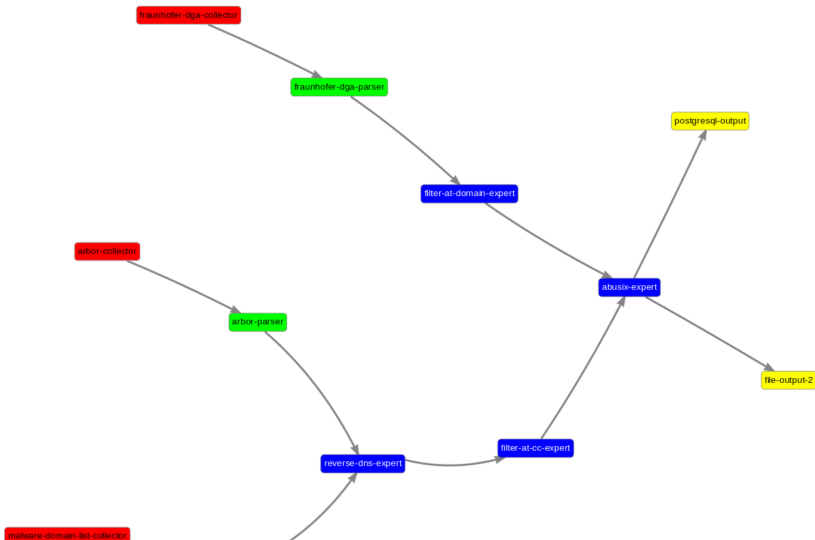
## data examples: parsed

```
{ '__type': u'Event',  
  'classification.type': 'malware',  
  'malware.name': 'zeus_p2p',  
  'feed.name': 'Example feed',  
  'raw': 'MTkyLjAuMi40NSwgMjAxNS0xMS0xNSAxNTo1NDo0MSw',  
  'source.ip': '192.0.2.45',  
  'time.observation': '2015-11-19T13:56:05+02:00',  
  'time.source': '2015-11-15T15:54:41+00:00' }
```





## data examples: cymru lookups

```
{ '__type': 'Event',  
  'classification.type': 'malware',  
  'malware.name': 'zeus_p2p',  
  'feed.name': 'Example feed',  
  'raw': 'MTkyLjAuMi40NSwgMjAxNS0xMS0xNSAxNT01ND00MSw',  
  'source.as_name': 'ATT-INTERNET4 - AT&T Services',  
  'source.asn': 7018,  
  'source.ip': '192.0.2.45',  
  'source.network': '192.0.0.0/16',  
  'source.registry': 'other',  
  'time.observation': '2015-11-19T13:56:05+02:00',  
  'time.source': '2015-11-15T15:54:41+00:00' }
```

# IntelMQ manager





# IntelMQ manager

INTELMQ  Configuration  Management  Monitor  About

























Botnet Status:

Status: stopped

Individual Bot Status:

All records per page Search:

Bot ID	Status	Actions
abusix-expert	stopped	 
arbor-collector	stopped	 
arbor-parser	stopped	 
file-output	stopped	 
file-output-2	stopped	 
filter-at-cc-expert	stopped	 
filter-at-domain-expert	stopped	 
fraunhofer-dga-collector	stopped	 
fraunhofer-dga-parser	stopped	 
malware-domain-list-collector	stopped	 
malware-domain-list-parser	stopped	 
postgresql-output	stopped	 

# IntelMQ manager

Monitoring: All Bots

Queue	Count
abusix-expert-queue	0
arbor-parser-queue	0
file-output-2-queue	0
file-output-queue	0
filter-at-cc-expert-queue	0
filter-at-domain-expert-queue	0
fraunhofer-dga-parser-queue	0
malware-domain-list-parser-queue	63
postgresql-output-queue	0
reverse-dns-expert-queue	0



# Examples of expert bots

- ASN lookup
- abuse contact
- whois
- deduplication and filtering
- geographic data
- DNS lookups (A, PTR Records)

# Installation

```
apt-get install python3
apt-get install git build-essential \
    libcurl4-gnutls-dev libffi-dev
apt-get install python-dev python-pip python-zmq \
    python-pycurl python-openssl python-pyasn1
apt-get install redis-server
```



## Installation (2)

```
git clone https://github.com/certtools/intelmq.git
cd intelmq
pip3 install -r REQUIREMENTS
python3 setup.py install
useradd -d /opt/intelmq -U -s /bin/bash intelmq
echo 'export PATH="$PATH:$HOME/bin" ' > \
    /opt/intelmq/.profile
chmod -R 0770 /opt/intelmq
chown -R intelmq.intelmq /opt/intelmq
```

# Writing a bot

```
def process(self):  
    event = self.receive_message()  
    if event is None:  
        self.acknowledge_message()  
        return  
  
    if event.contains('source.ip'):  
        if event.value('source.ip') in self.database:  
            event.add('source.tor_node', True)  
    self.send_message(event)  
    self.acknowledge_message()
```

# Next developments and project goals

- more feeds/sources
- reliability: more unittests, coverage, integration tests
- bots working in parallel
- adaptations of data harmonization
- even more simpler configuration
- more users
- stable version 1.0 this year

# Links

- <https://github.com/certtools/intelmq>
- <https://github.com/certtools/intelmq-manager>
- <https://www.enisa.europa.eu/activities/cert/support/incident-handling-automation>
- Mailing list for developers: <https://tiss.trusted-introducer.org/mailman/listinfo/ihap> (for TI members)  
or ask [kaplan@cert.at](mailto:kaplan@cert.at) for subscription

# Questions?

- <https://github.com/certtools/intelmq>
- <https://github.com/certtools/intelmq-manager>
- <https://www.enisa.europa.eu/activities/cert/support/incident-handling-automation>
- Mailing list for developers: <https://tiss.trusted-introducer.org/mailman/listinfo/ihap> (for TI members)  
or ask [kaplan@cert.at](mailto:kaplan@cert.at) for subscription