# AIL Framework for Analysis of Information Leaks

## data mining - website and darkweb correlation

CIRCL
Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy
alexandre.dulaunoy@circl.lu
Aurélien Thirion
aurelien.thirion@circl.lu

info@circl.lu

April 20, 2020

# Privacy, AIL and GDPR

- Many modules in AIL can process personal data and even special categories of data as defined in GDPR (Art. 9).
- The data controller is often the operator of the AIL framework (limited to the organisation) and has to define **legal grounds for processing personal data**.
- To help users of AIL framework, a document is available which describe points of AIL in regards to the regulation[1].

[1] https://www.circl.lu/assets/files/information-leaks-analysis-and-gdpr.pdf

# Potential legal grounds

- **Consent of the data subject** is in many cases not feasible in practice and often impossible or illogical to obtain (Art. 6(1)(a)).
- Legal obligation (Art. 6(1)(c)) - This legal ground applies mostly to CSIRTs, in accordance with the powers and responsibilities set out in CSIRTs mandate and with their constituency, as they may have the legal obligation to collect, analyse and share information leaks without having a prior consent of the data subject.
- Art. 6(1)(f) - Legitimate interest - Recital 49 explicitly refers to CSIRTs' right to process personal data provided that they have a legitimate interest but not colliding with fundamental rights and freedoms of data subject.

# Objectives

# Our objectives

- Show how to use and extend an open source tool to monitor web pages, pastes, forums and hidden services
- Explain challenges and the design of the AIL open source framework
- Learn how to create new modules
- Learn how to use, install and start AIL
- **Supporting investigation using the AIL framework**

# AIL Framework

# From a requirement to a solution: AIL Framework

History:

- AIL initially started as an **internship project** (2014) to evaluate the feasibility to automate the analysis of (un)structured information to find leaks.
- In 2019, AIL framework is an **open source software** in Python. The software is actively used (and maintained) by CIRCL and many organisations.

# AIL Framework: A framework for Analysis of Information Leaks

*"AIL is a modular framework to analyse potential information leaks from unstructured data sources."*

# Capabilities Overview

# Common usage

- **Check** if mail/password/other sensitive information (terms tracked) leaked
- **Detect** reconnaissance of your infrastructure
- **Search** for leaks inside an archive
- **Monitor** and crawl websites

# Support CERT and Law Enforcement activities

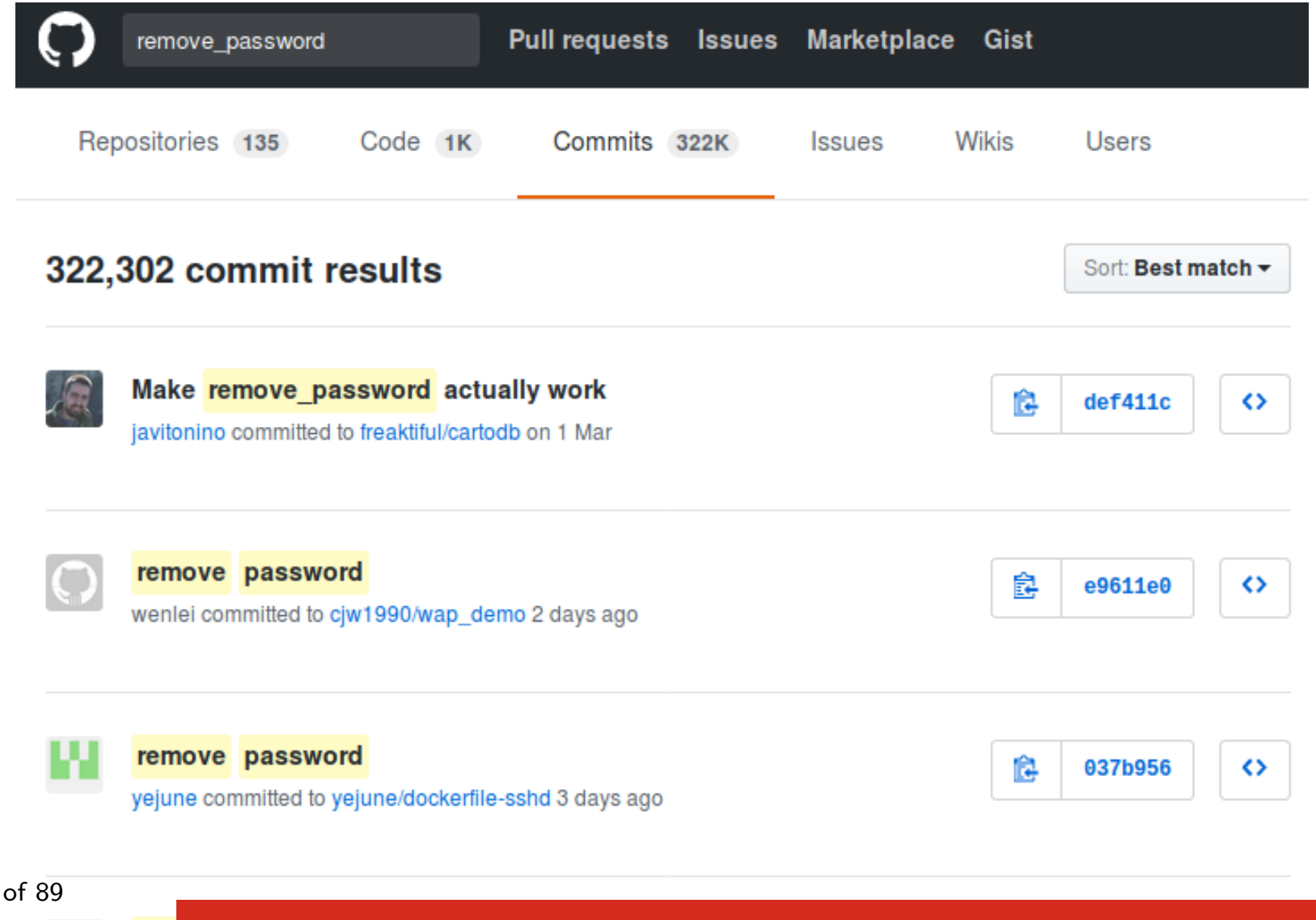
- Proactive investigation: leaks detection
  - List of emails and passwords
  - Leaked database
  - AWS Keys
  - Credit-cards
  - PGP private keys
  - Certificate private keys
- Feed Passive DNS or any passive collection system
- CVE and PoC of vulnerabilities most used by attackers

# Support CERT and Law Enforcement activities

- Website monitoring
  - monitor booters
  - Detect encoded exploits (WebShell, malware encoded in Base64, ...)
  - SQL injections
- Automatic and manual submission to threat sharing and incident response platforms
  - MISP
  - TheHive
- Term/Regex monitoring for local companies/government

# Sources of leaks

# Mistakes from users:



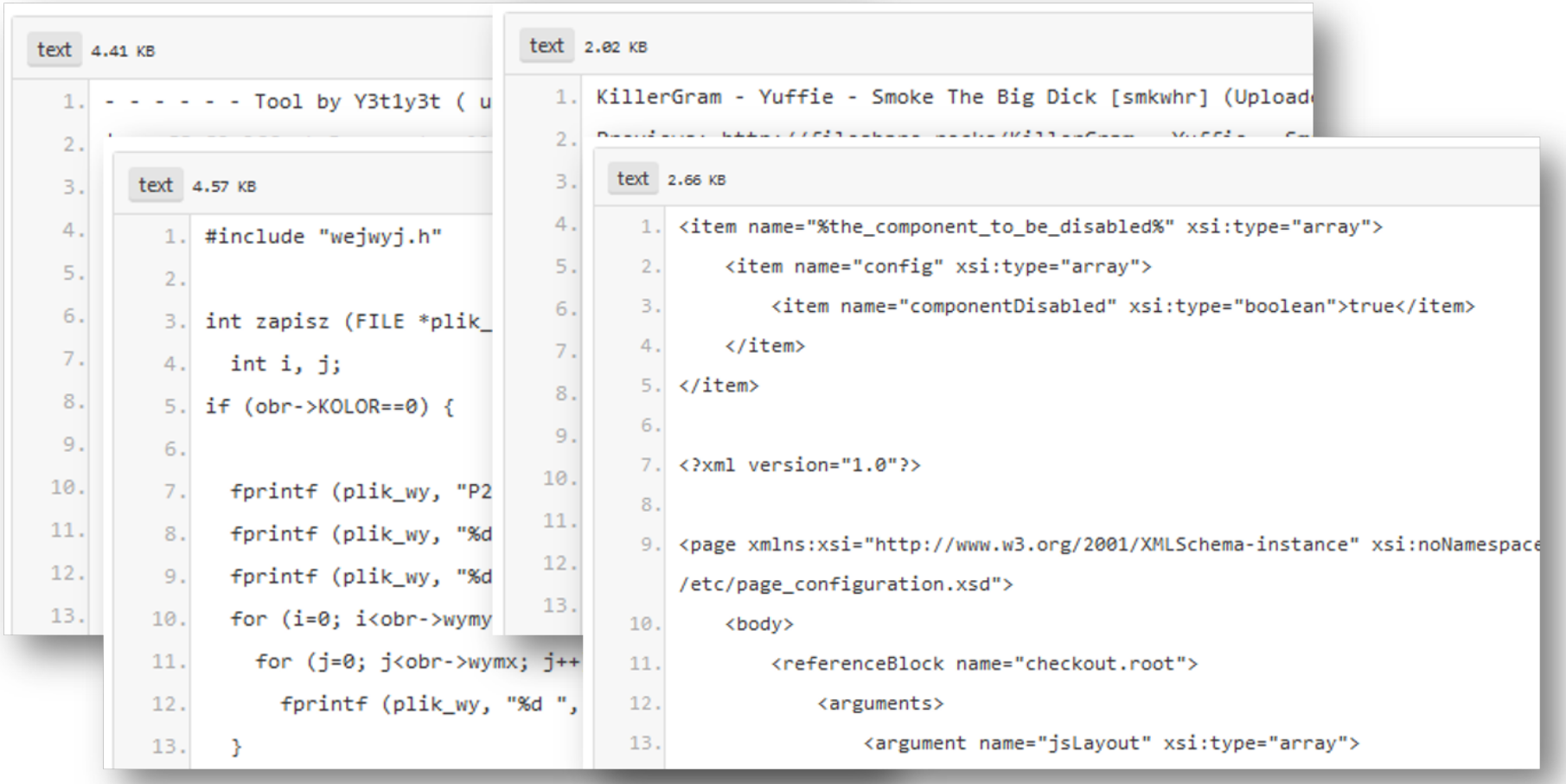
# Sources of leaks: Paste monitoring

- Example: `http://pastebin.com/`
  - Easily storing and sharing text online
  - Used by programmers and legitimate users
    $\rightarrow$ Source code & information about configurations

# Sources of leaks: Paste monitoring

- Example: `http://pastebin.com/`
  - Easily storing and sharing text online
  - Used by programmers and legitimate users
    $\rightarrow$ Source code & information about configurations
- Abused by attackers to store:
  - List of vulnerable/compromised sites
  - Software vulnerabilities (e.g. exploits)
  - Database dumps
    $\rightarrow$ User data
    $\rightarrow$ Credentials
    $\rightarrow$ Credit card details
  - More and more ...

# Examples of pastes



text 4.41 KB

```
- - - - - - Tool by Y3t1y3t ( u
```

text 2.02 KB

```
KillerGram - Yuffie - Smoke The Big Dick [smkwhr] (Upload
```

text 4.57 KB

```
#include "wejwyj.h"

int zapisz (FILE *plik_
  int i, j;
if (obr->KOLOR==0) {

  fprintf (plik_wy, "P2
  fprintf (plik_wy, "%d
  fprintf (plik_wy, "%d
  for (i=0; i<obr->wymy
    for (j=0; j<obr->wymx; j++
      fprintf (plik_wy, "%d ",
  }
```

text 2.66 KB

```
<item name="%the_component_to_be_disabled%" xsi:type="array">
    <item name="config" xsi:type="array">
        <item name="componentDisabled" xsi:type="boolean">true</item>
    </item>
</item>

<?xml version="1.0"?>

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
/etc/page_configuration.xsd">
    <body>
        <referenceBlock name="checkout.root">
            <arguments>
                <argument name="jsLayout" xsi:type="array">
```

# Why so many leaks?

- Economical interests (e.g. Adversaries promoting services)
- Political motives (e.g. Adversaries showing off)
- Collaboration (e.g. Criminals need to collaborate)
- Operational infrastructure (e.g. malware exfiltrating information on a pastie website)
- Mistakes and Errors

# Are leaks frequent?

Yes!

and we have to deal with this as a CSIRT.

- **Contacting companies or organisations** who did specific accidental leaks
- **Discussing with media** about specific case of leaks and how to make it more practical/factual for everyone
- Evaluating the economical market for cyber criminals (e.g. DDoS booters[2] or reselling personal information - reality versus media coverage)
- Analysing collateral effects of malware, software vulnerabilities or exfiltration

→ And it's important to detect them automatically.

[2] https://github.com/D4-project/

# Paste monitoring at CIRCL: Statistics

- Monitored paste sites: 27
  - *pastebin.com*
  - *ideone.com*
  - ...

| | 2016 | 2017 | 08.2018 |
|---|---|---|---|
| Collected pastes | 18,565,124 | 19,145,300 | 11,591,987 |
| Incidents | 244 | 266 | 208 |

Table: Pastes collected and incident[3] raised by CIRCL

[3] `http://www.circl.lu/pub/tr-46`

# MISP

# MISP Taxonomies

- **Tagging** is a simple way to attach a classification to an event or anattribute.
- **Classification must be globally used to be efficient.**
- Provide a set of already defined classifications modeling estimative language
- Taxonomies are implemented in a simple JSON format [4].
- Can be easily cherry-picked or extended

[4] https://github.com/MISP/misp-taxonomies

# Taxonomies useful in AIL

- **infoleak**: Information classified as being potential leak.
- **estimative-language**: Describe quality and credibility of underlying sources, data, and methodologies.
- **admiralty-scale**: Rank the reliability of a source and the credibility of an information
- **fpf**[5]: Evaluate the degree of identifiability of personal data and the types of pseudonymous data, de-identified data and anonymous data.

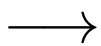
[5] Future of Privacy Forum

# Taxonomies useful in AIL

- **tor**: Describe Tor network infrastructure.
- **dark-web**: Criminal motivation on the dark web.
- **copine-scale**[6]: Categorise the severity of images of child sex abuse.

[6] Combating Paedophile Information Networks in Europe

# threat sharing and incident response platforms


**Goal:** submission to threat sharing and incident response platforms.

# threat sharing and incident response platforms





1. Use infoleak taxonomy[7]
2. Add your own tags
3. Export AIL objects to MISP core format
4. Download it or Create a MISP[8]

[7] https://www.misp-project.org/taxonomies.html

[8] https://www.misp-standard.org/rfc/misp-standard-core.txt

# MISP Export

## 1Gt545E48EPsyTC8voKQDCFfpTkwiuXduw :

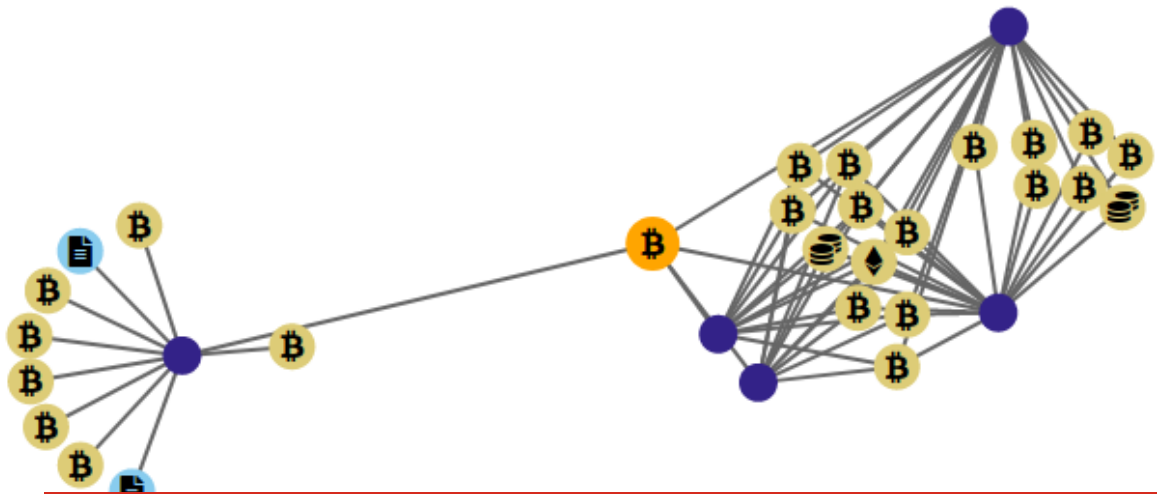
| Object type | type | First seen | Last seen | Nb seen |
|---|---|---|---|---|
| cryptocurrency | bitcoin | 2020/01/17 | 2020/02/20 | 5 |

Expand Bitcoin address

Graph

Resize Graph

Add to MISP Export



# MISP Export

UP

nttfj36sp47cw2yecop572zjvjeazgazieunllouudplzqt2m5h465yd.onion :

| First Seen | Last Check | Ports |
|---|---|---|
| 2020/02/19 | 2020/02/19 | ['80'] |

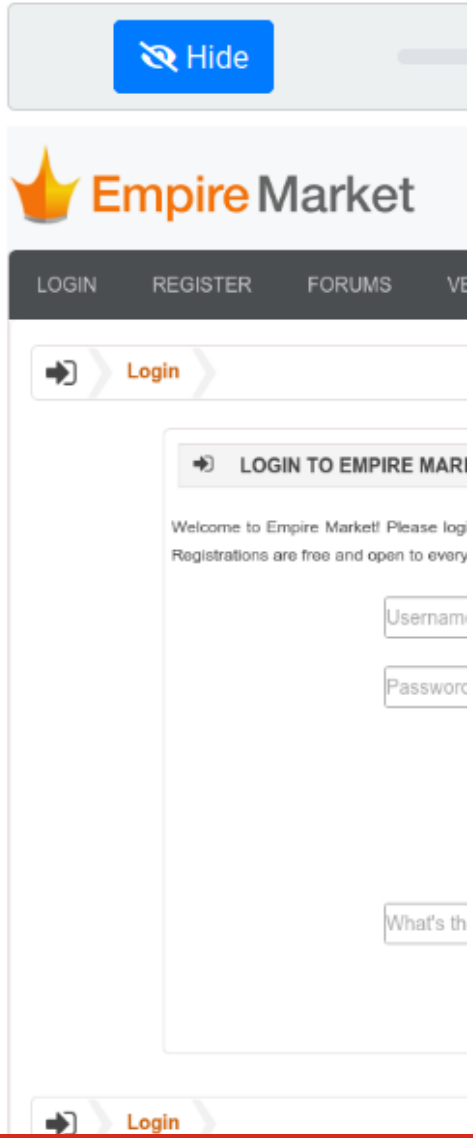
infoleak:automatic-detection="onion"

Last Origin: crawled/2020/02/19/dark.failc126d32a-3ed1-468f-ba24-f2e5956f4035

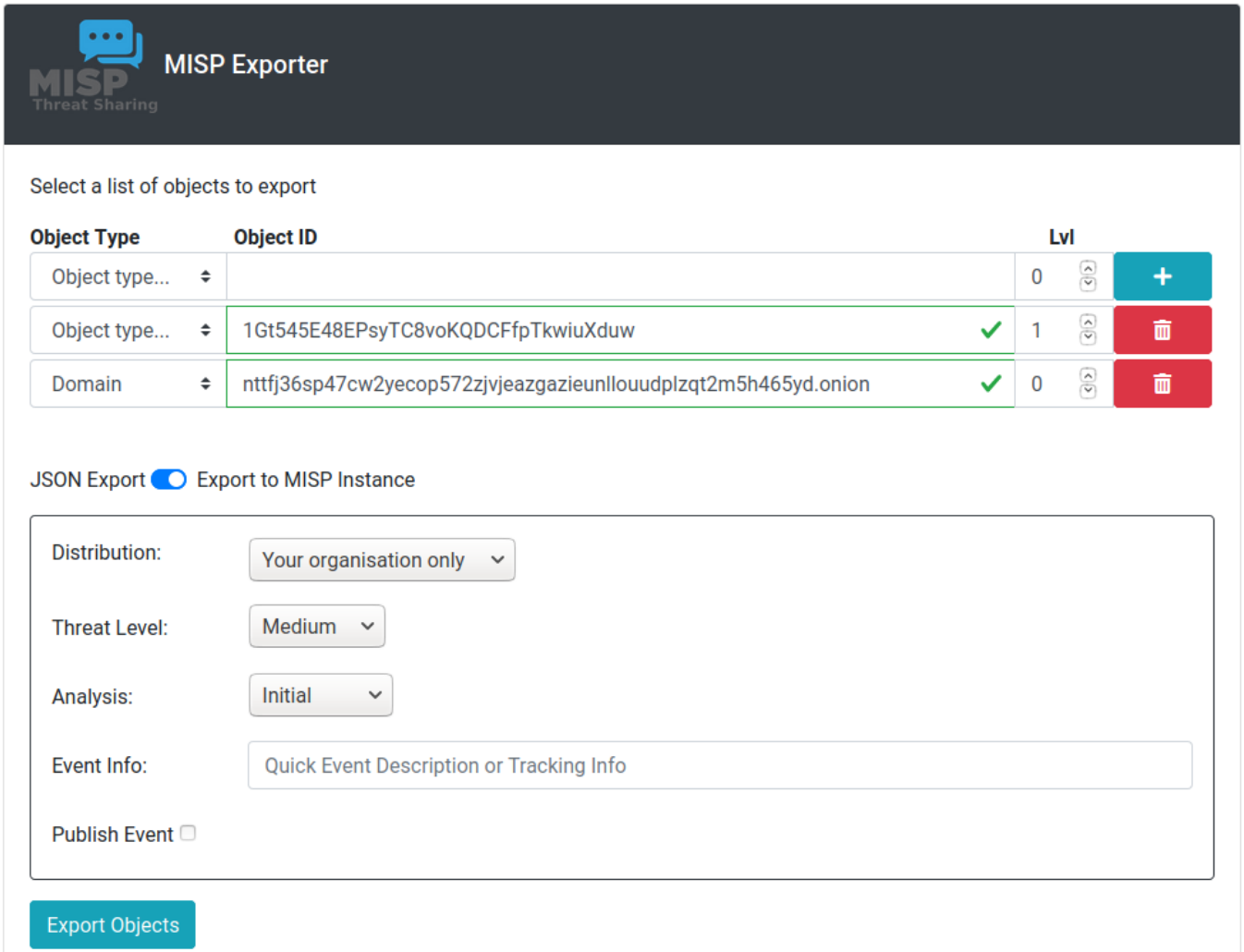
Show Domain Correlations 4

Add to MISP Threat Sharing Export

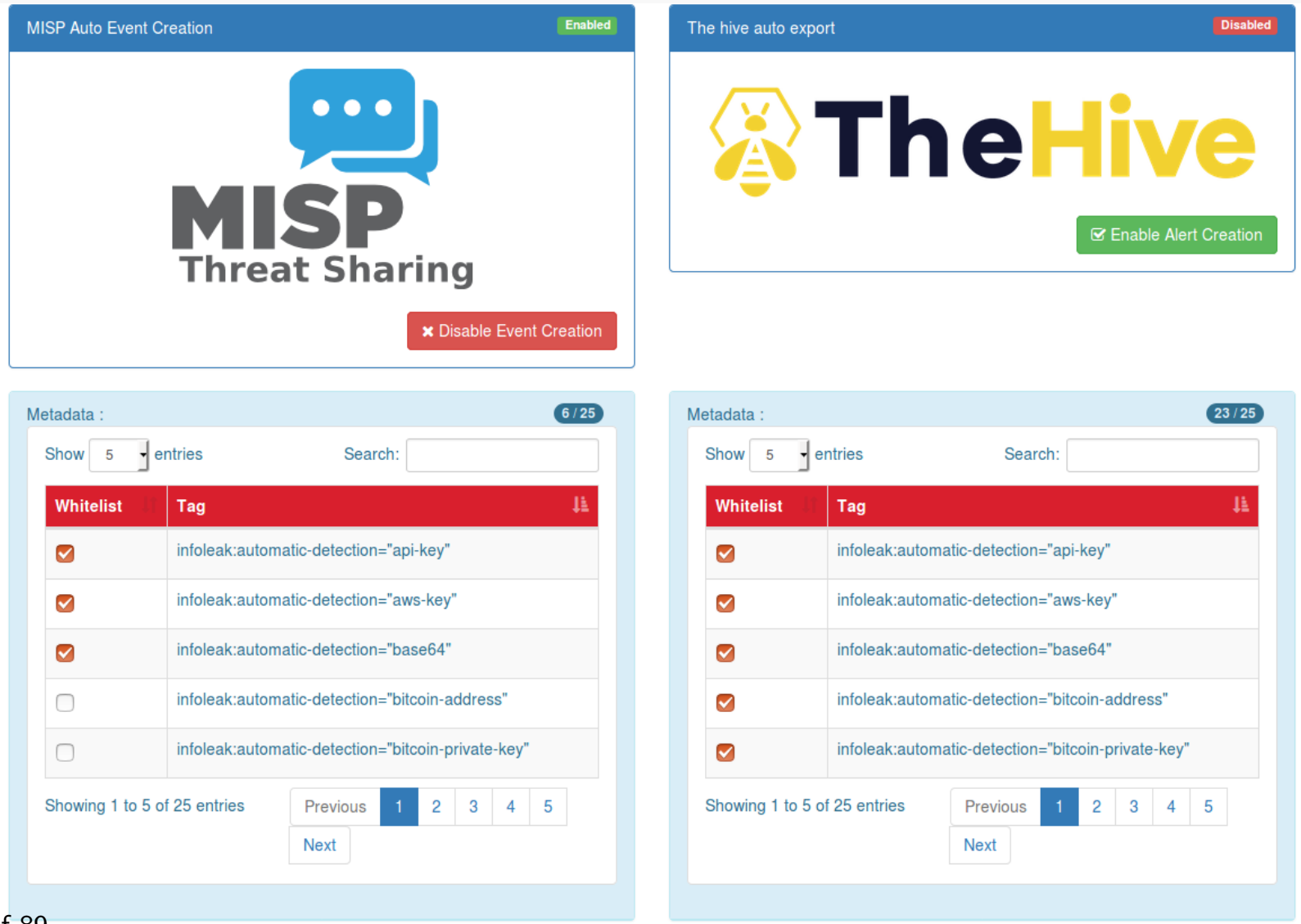
Screenshot 4



# MISP Export


MISP Exporter
Threat Sharing
Select a list of objects to export
Object Type
Object ID
Lvl
Object type...
0
Object type...
1Gt545E48EPsyTC8voKQDCFfpTkwiuXduw
1
Domain
nttfj36sp47cw2yecop572zjvjeazgazieunllouudplzqt2m5h465yd.onion
0
JSON Export
Export to MISP Instance
Distribution:
Your organisation only
Threat Level:
Medium
Analysis:
Initial
Event Info:
Quick Event Description or Tracking Info
Publish Event
Export Objects

# Automatic submission on tags



# Current capabilities

# AIL Framework: Current capabilities

- Extending AIL to add a new **analysis module** can be done in 50 lines of Python
- The framework **supports multi-processors/cores by default**. Any analysis module can be started multiple times to support faster processing during peak times or bulk import
- **Multiple** concurrent **data input**
- Tor Crawler

# AIL Framework: Current features

- Extracting **credit cards numbers, credentials, phone numbers, ...**
- Extracting and validating potential **hostnames**
- Keeps track of **duplicates**
- Submission to threat sharing and incident response platform (**MISP** and **TheHive**)
- **Full-text indexer** to index unstructured information
- **Tagging** for classification and searches
- Terms, sets and regex **tracking and occurences**
- Archives, files and raw **submission** from the UI
- PGP and Decoded (Base64, ...) Correlation
- And many more

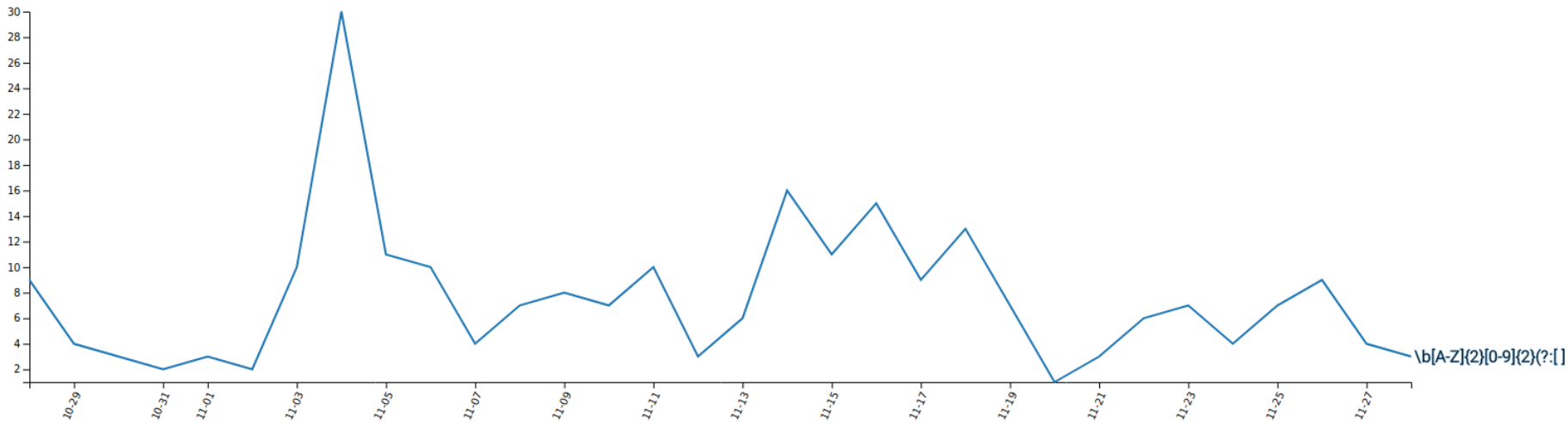
# Terms Tracker

- Search and monitor specific keywords
  - Automatic Tagging
  - Email Notifications
- Track Term
  - ddos
- Track Set
  - booter,ddos,stresser;2
- Trag Regex
  - circl\.lu

# Terms Tracker:

82a87a6a-88f1-4ab1-ba53-1bf15211b4b8

| Type | Tracker | Date added | Level | Created by | First seen | Last seen | Tags | Email |
|---|---|---|---|---|---|---|---|---|
| regex | \b[A-Z]{2}[0-9]{2}(?:[ ]?[0-9]{4}){4}(?!(?:[ ]?[0-9]){3})(?:[ ]?[0-9]{1,2})?\b | 2019/09/12 | 1 | admin@admin.test | 2018/08/31 | 2019/11/28 | | |



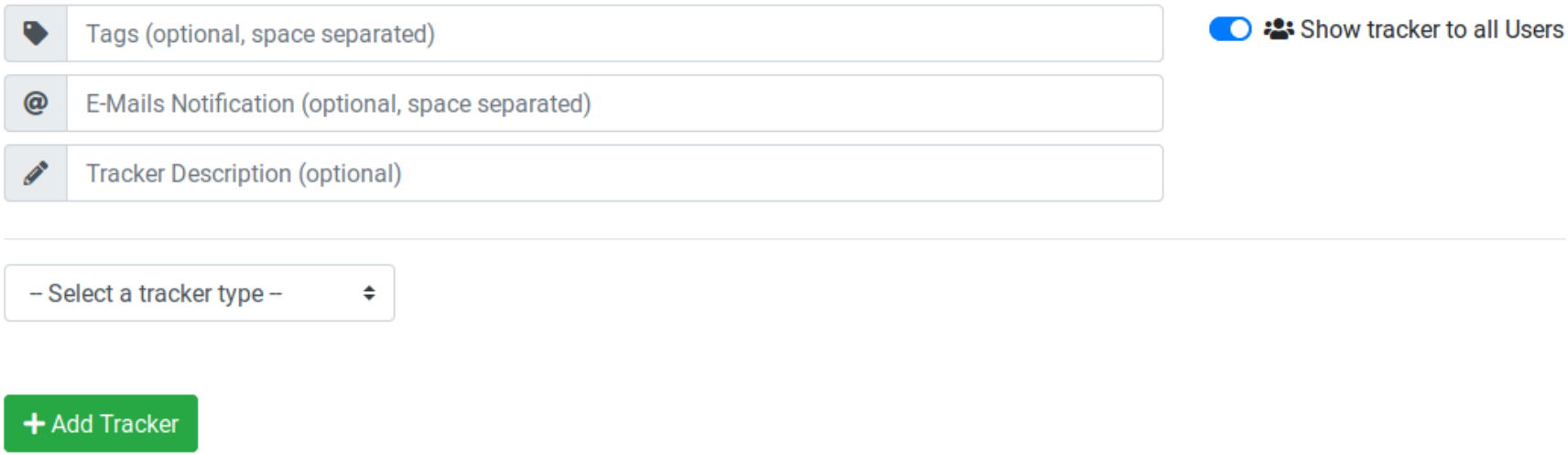

yyyy-mm-dd

yyyy-mm-dd

Search Tracked Items

# Terms Tracker - Practical part

- **Create and test** your own term tracker



# Recon and intelligence gathering tools

- **Attacker also share informations**
- Recon tools detected: 94
  - sqlmap
  - dnscan
  - whois
  - msfconsole (metasploit)
  - dnmap
  - nmap
  - ...

# Recon and intelligence gathering tools

```
###############################################################################
===============================================================================
Hostname        www.pabloquintanilla.cl                 ISP     Wix.com Ltd.
Continent       North America           Flag
US
Country         United States           Country Code    US
Region  Unknown                 Local time      19 Nov 2019 07:59 CST
City    Unknown                 Postal Code     Unknown
IP Address      185.230.60.195          Latitude        37.751
                        Longitude       -97.822
===============================================================================
###############################################################################
> www.pabloquintanilla.cl
Server:         38.132.106.139
Address:        38.132.106.139#53

Non-authoritative answer:
www.pabloquintanilla.cl canonical name = www192.wixdns.net.
www192.wixdns.net       canonical name = balancer.wixdns.net.
Name:   balancer.wixdns.net
Address: 185.230.60.211
>
###############################################################################
Domain name: pabloquintanilla.cl
Registrant name: SERGIO TORO
Registrant organisation:
Registrar
```

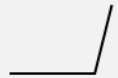
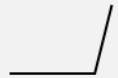
# Decoder

- Search for encoded strings
  - Base64
  - Hexadecimal
  - Binary
- Guess Mime-type
- Correlate paste with decoded items

# Decoder: Practical Part

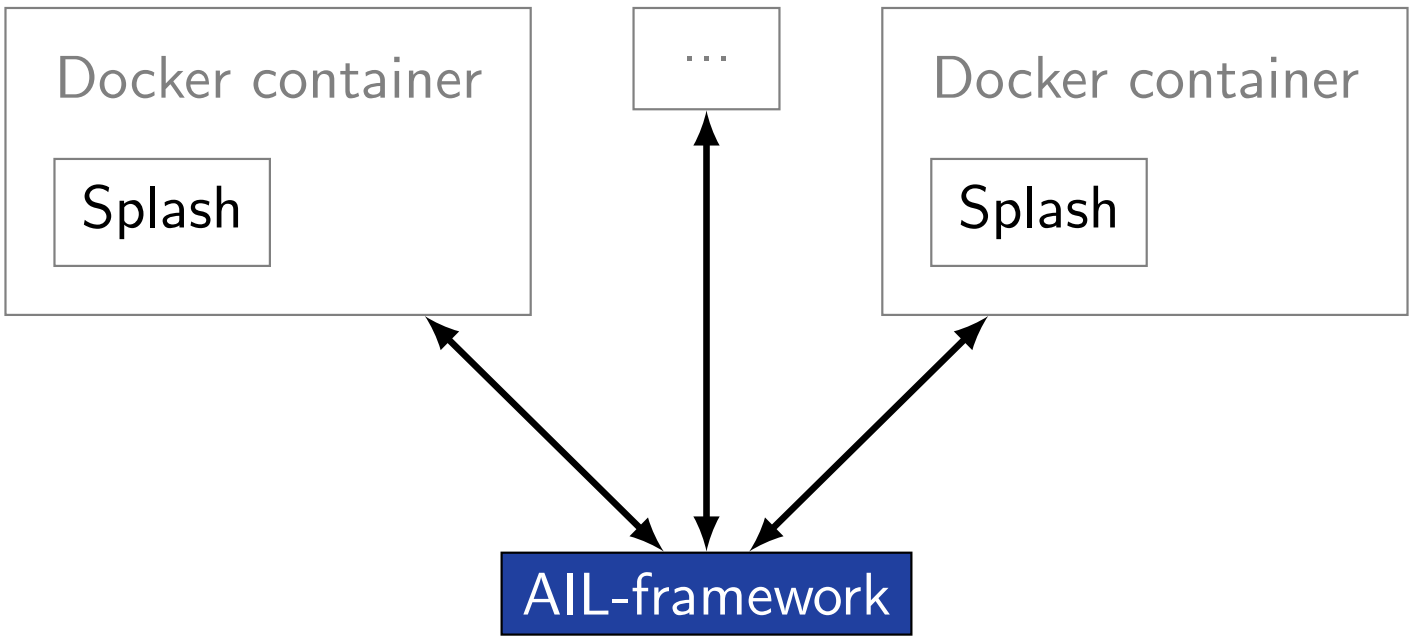
Which type of decoded file have the highest size ?

# Decoder: Practical Part

| estimated type | hash | first seen | last seen | nb item | size | Virus Total | Sparkline |
|---|---|---|---|---|---|---|---|
| application/x-dosexec | c11c2be8d9ba4e86c8effaa411aa6b867ba75abe | 2019/11/28 | 2019/11/28 | 1 | 191 | Send this file to VT |  |
| application/x-dosexec | a50cba731204ecce193b40178399a250b5ce6f67 | 2019/11/28 | 2019/11/28 | 1 | 32768 | Send this file to VT |  |
| application/x-dosexec | cc5f2f0da71f443ec12ae1b3cb6ab8bad80f22c4 | 2019/11/28 | 2019/11/28 | 1 | 203 | Send this file to VT |  |
| application/x-dosexec | eed67e8fa9cb9a43fea21ae653983a8e0a174f63 | 2019/11/26 | 2019/11/28 | 6 | 83 | Send this file to VT |  |

# Crawler

- Crawlers are used to navigate on regular website as well as .onion addresses (via automatic extraction of urls or manual submission)
- Splash ("scriptable" browser) is rending the pages (including javascript) and produce screenshots (HAR archive too)

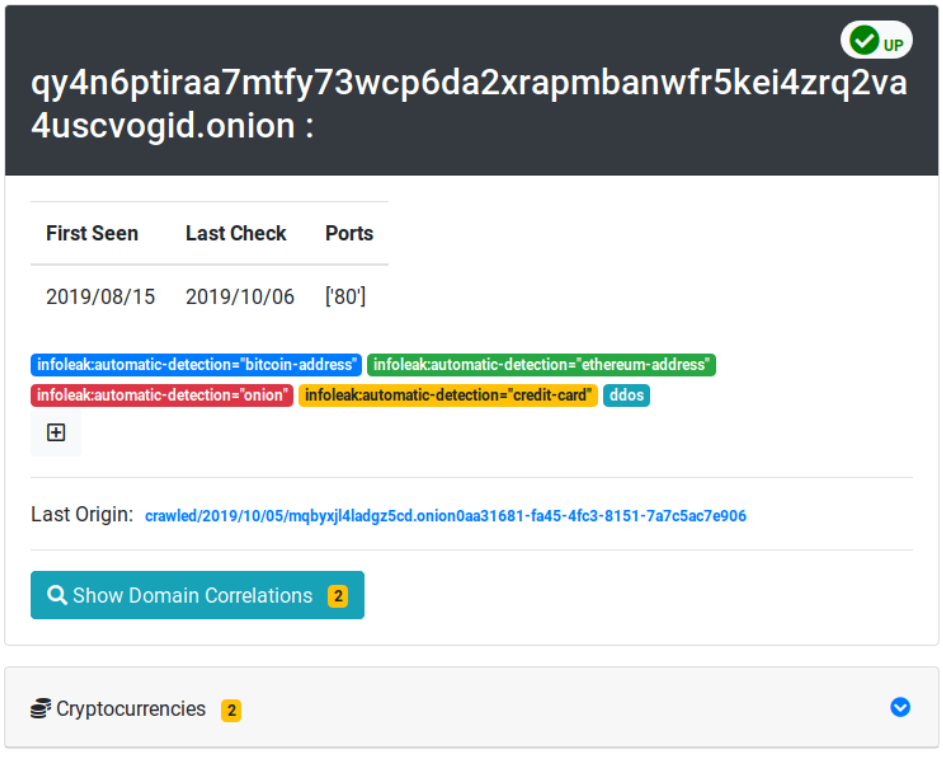


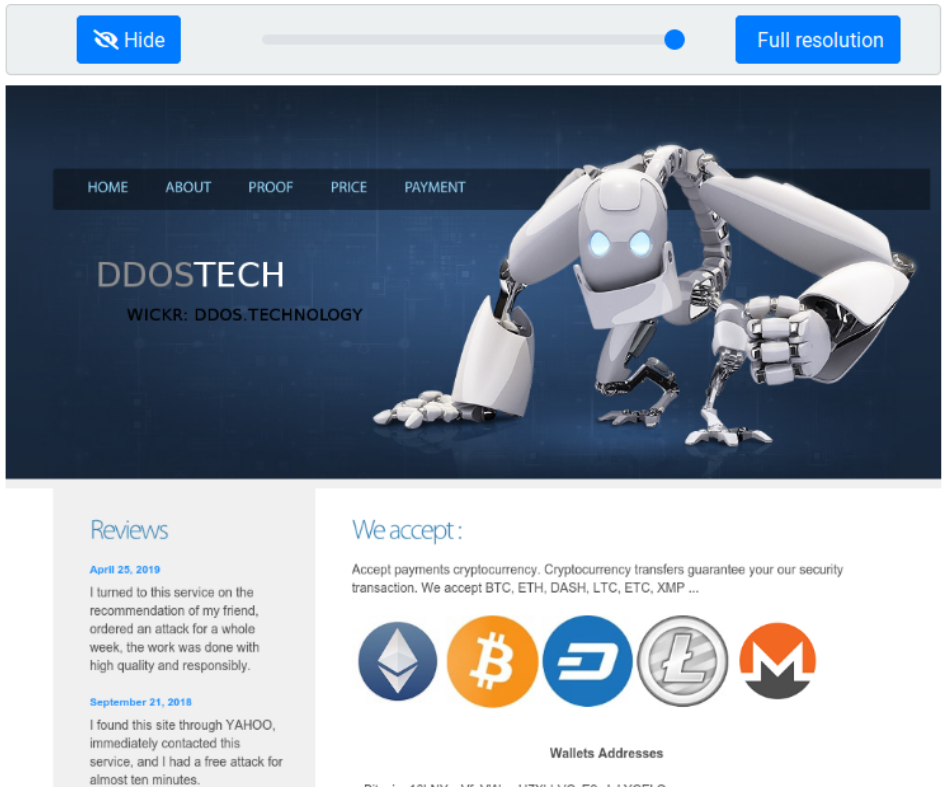
# Crawler

How a domain is crawled by default

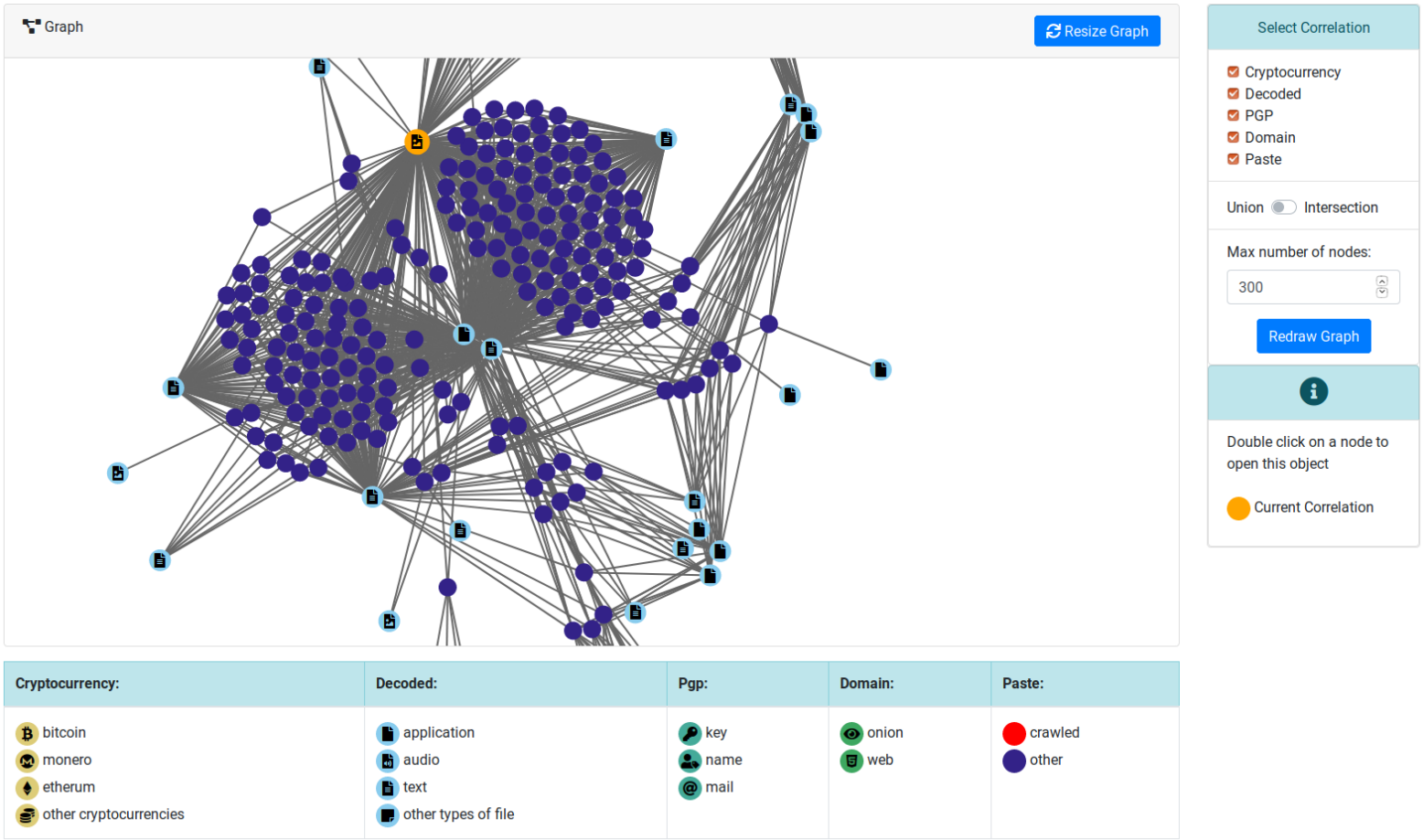
1. Fetch the first url
2. Render javascript (webkit browser)
3. Extract all urls
4. Filter url: keep all url of this domain
5. crawl next url (max depth = 1)

# Crawler: DDoS Booter



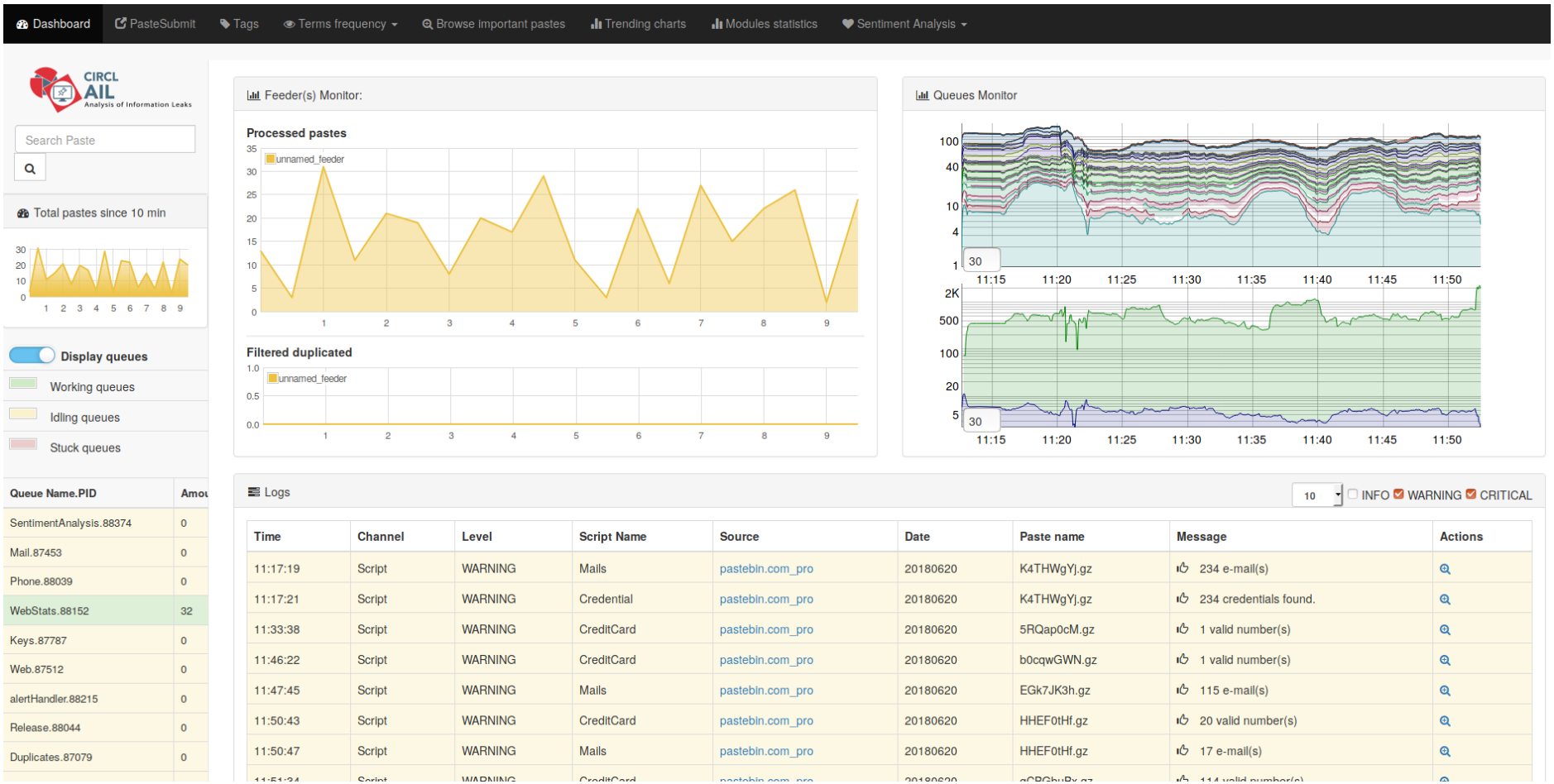



# Correlations and relationship

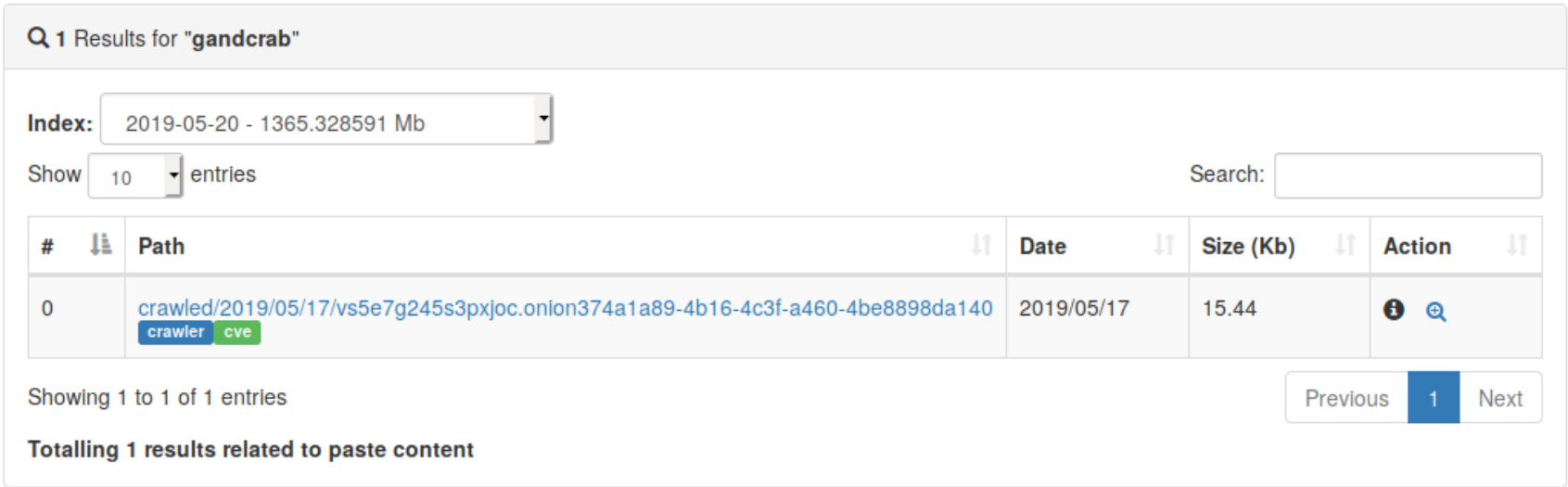


# Live demo!

# Example: Dashboard



# Example: Text search



# Example: Pastes Metadata (1)

infoleak:automatic-detection="phone-number" infoleak:automatic-detection="mail" infoleak:automatic-detection="base64" +

| Date | Source | Encoding | Language | Size (Kb) | Mime | Number of lines | Max line length |
|---|---|---|---|---|---|---|---|
| 04/05/2019 | pastebin.com_pro | text/plain | None | 6.12 | text/plain | 1650 | 100 |

Create MISP Event

Duplicate list:

Show 10 entries Search:

| Hash type | Paste info | Date | Path | Action |
|---|---|---|---|---|
| ['tlsh'] | Similarity: [19]% | 2019-04-13 | archive/pastebin.com_pro/2019/04/13/EbMVR87S.gz | |
| ['tlsh'] | Similarity: [10]% | 2019-04-11 | archive/pastebin.com_pro/2019/04/11/2X5HRVnX.gz | |
| ['tlsh'] | Similarity: [23]% | 2019-04-25 | archive/pastebin.com_pro/2019/04/25/TS2b6M4c.gz | |
| ['tlsh'] | Similarity: [14]% | 2019-04-17 | archive/pastebin.com_pro/2019/04/17/CuS93H7K.gz | |
| ['tlsh'] | Similarity: [23]% | 2019-04-20 | archive/pastebin.com_pro/2019/04/20/AQd0qGVQ.gz | |
| ['tlsh'] | Similarity: [20]% | 2019-04-20 | archive/pastebin.com_pro/2019/04/20/6DDc13b8.gz | |
| ['tlsh'] | Similarity: [21]% | 2019-05-05 | alerts/pastebin.com_pro/2019/05/05/X8nJLzda.gz | |
| ['tlsh'] | Similarity: [7]% | 2019-04-13 | archive/pastebin.com_pro/2019/04/13/Lyp4FVWW.gz | |

Showing 1 to 8 of 8 entries Previous 1 Next

# Example: Pastes Metadata (2)

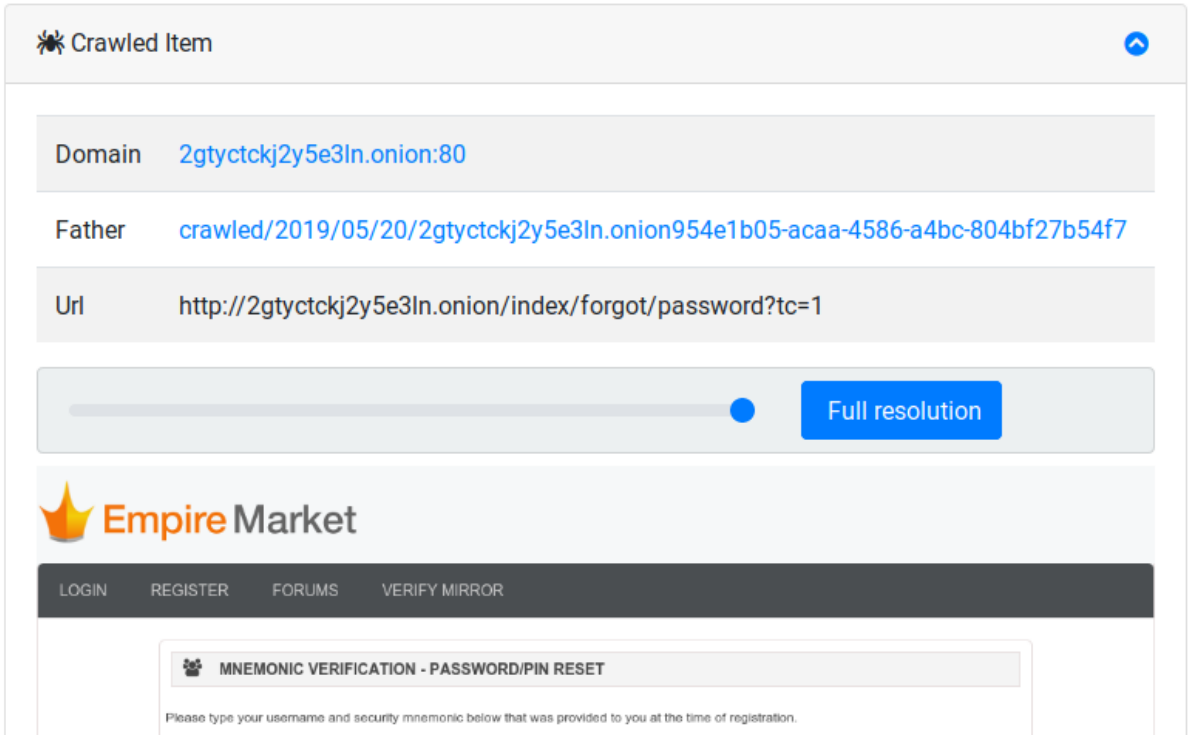
Hash files:

Show 5 entries Search:

| estimated type | hash | saved_path | Virus Total |
|---|---|---|---|
| application/octet-stream | 3975f058bb0d445b60c10a11f1a5d88e19e4fa84 (1) | HASHS/application/octet-stream /39/3975f058bb0d445b60c10a11f1a5d88e19e4fa84 | Send this file to VT |
| application/octet-stream | fed93c1753270fc849a4db37027b569cdd9a6108 (1) | HASHS/application/octet-stream /fe/fed93c1753270fc849a4db37027b569cdd9a6108 | Send this file to VT |

Showing 1 to 2 of 2 entries Previous 1 Next

# Example: Pastes Metadata (3)



# Example: Browsing content

Content:

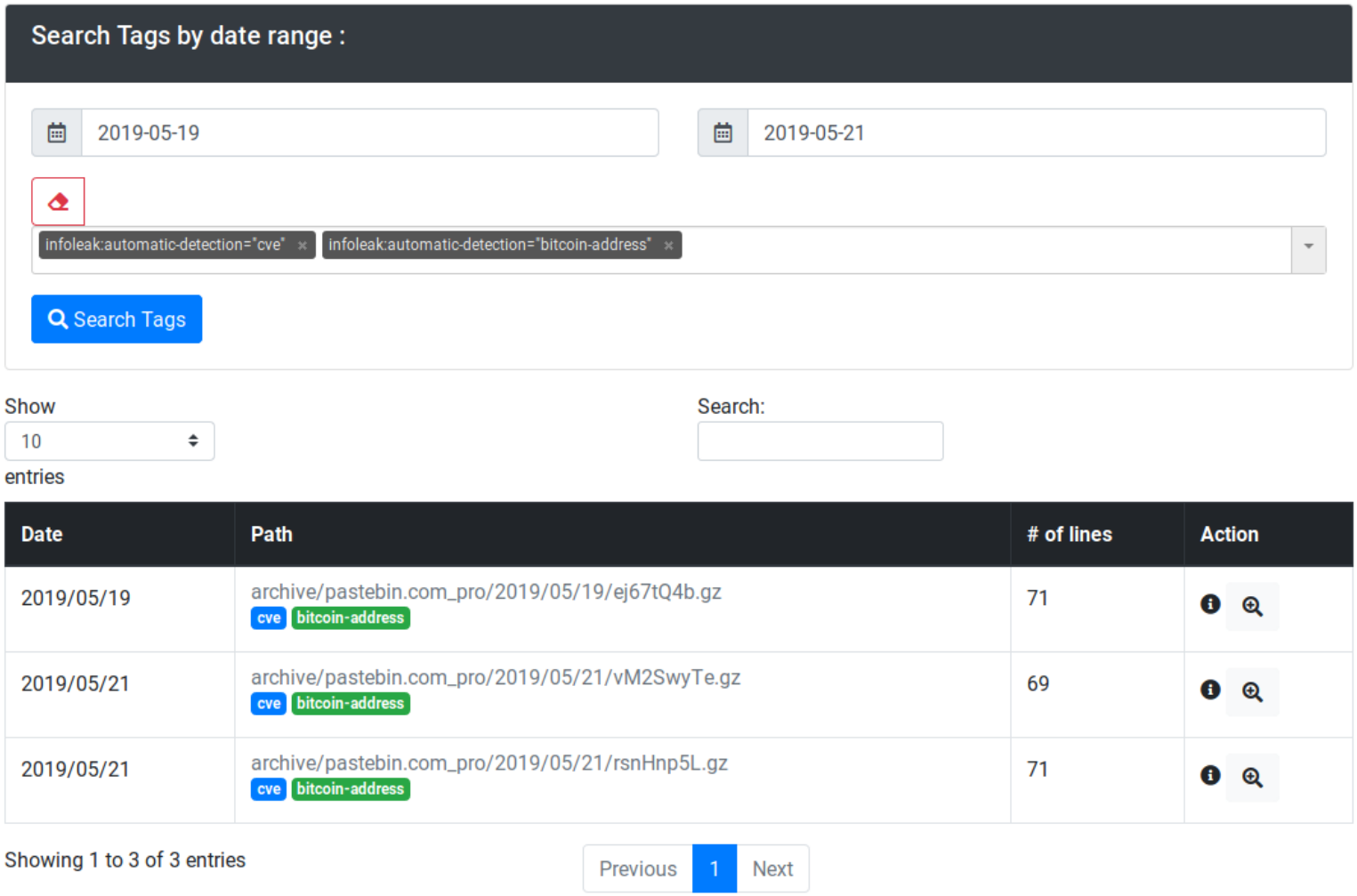
```
http://members2.mofosnetwork.com/access/login/
somosextremos:buddy1990
brazzers_glenn:cocklick
brazzers61:braves01

http://members.naughtyamerica.com/index.php?m=login
gernblanston:3unc2352
Janhuss141200:310575
igetalliwant:1377zeph
pwilks89:mon22key
Bman1551:hockey

MoFos IKnowThatGirl PublicPickUps
http://members2.mofos.com
Chrismagg40884:loganm40
brando1:zzbrando1
aacoen:1q2w3e4r
1rstunkle23:my8self

BraZZers
http://ma.brazzers.com
gcjensen:gcj21pva
skycsc17:rbcdnd

                                ############################################################
                                    >| Get Daily Update Fresh Porn Password Here |<

                                          =>   http://www.erq.io/4mF1
```

# Example: Browsing content

Content:

```
Over 50000+ custom hacked xxx passwords by us! Thousands of free xxx passwords to the hottest paysites!

##########################################################
 >| Get Fresh New Premium XXX Site Password Here |<

  =>   http://www.erq.io/4mF1


##########################################################



http://ddfnetwork.com/home.html
eu172936:hCSBgKh
UecwB6zs:159X0$!r#6K78FuU

http://pornxn.stiffia.com/user/login
feldwWek8939:RObluJ8XtB
dabudka:17891789
brajits:brajits1

http://members.pornstarplatinum.com/sblogin/login.php/
gigiriveracom:xxxjay
jayx123:xxxjay69

http://members.vividceleb.com/
Rufio99:fairhaven
ScHiFRvi:102091
Chaos84:HOLE5244
Riptor795:blade7
Domi80:harkonnen
GaggedUK:a1k0chan

http:
```

# Example: Search by tags



# API

# Setting up the framework

# Setting up AIL-Framework from source or virtual machine

## Setting up AIL-Framework from source

```
1 git clone
      https://github.com/ail-project/ail-framework.git
2 cd AIL-framework
3 ./installing_deps.sh
```

# AIL ecosystem - Challenges and design

# AIL ecosystem: Technologies used

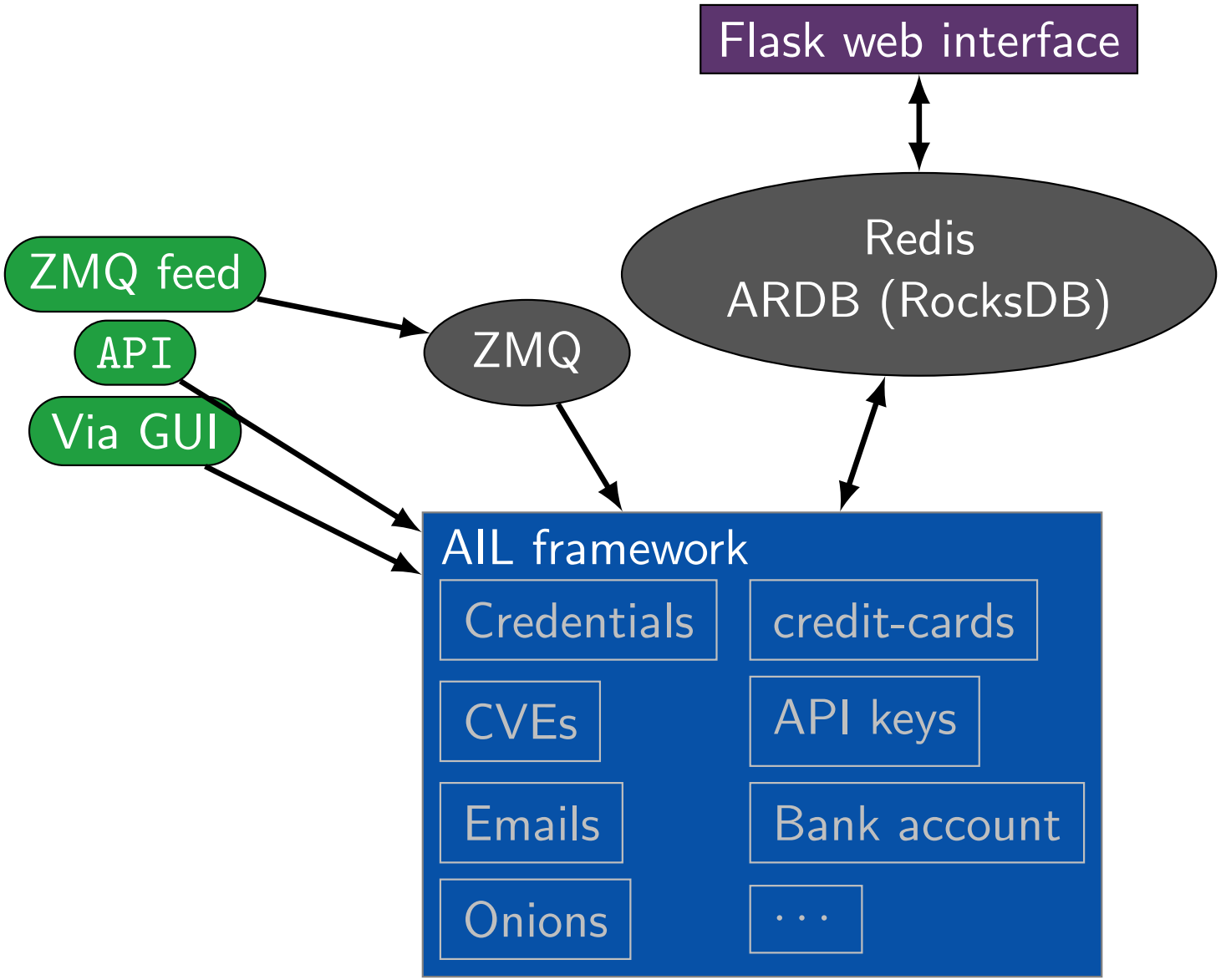
**Programming language:** Full python3

**Databases:** Redis and ARDB

**Server:** Flask

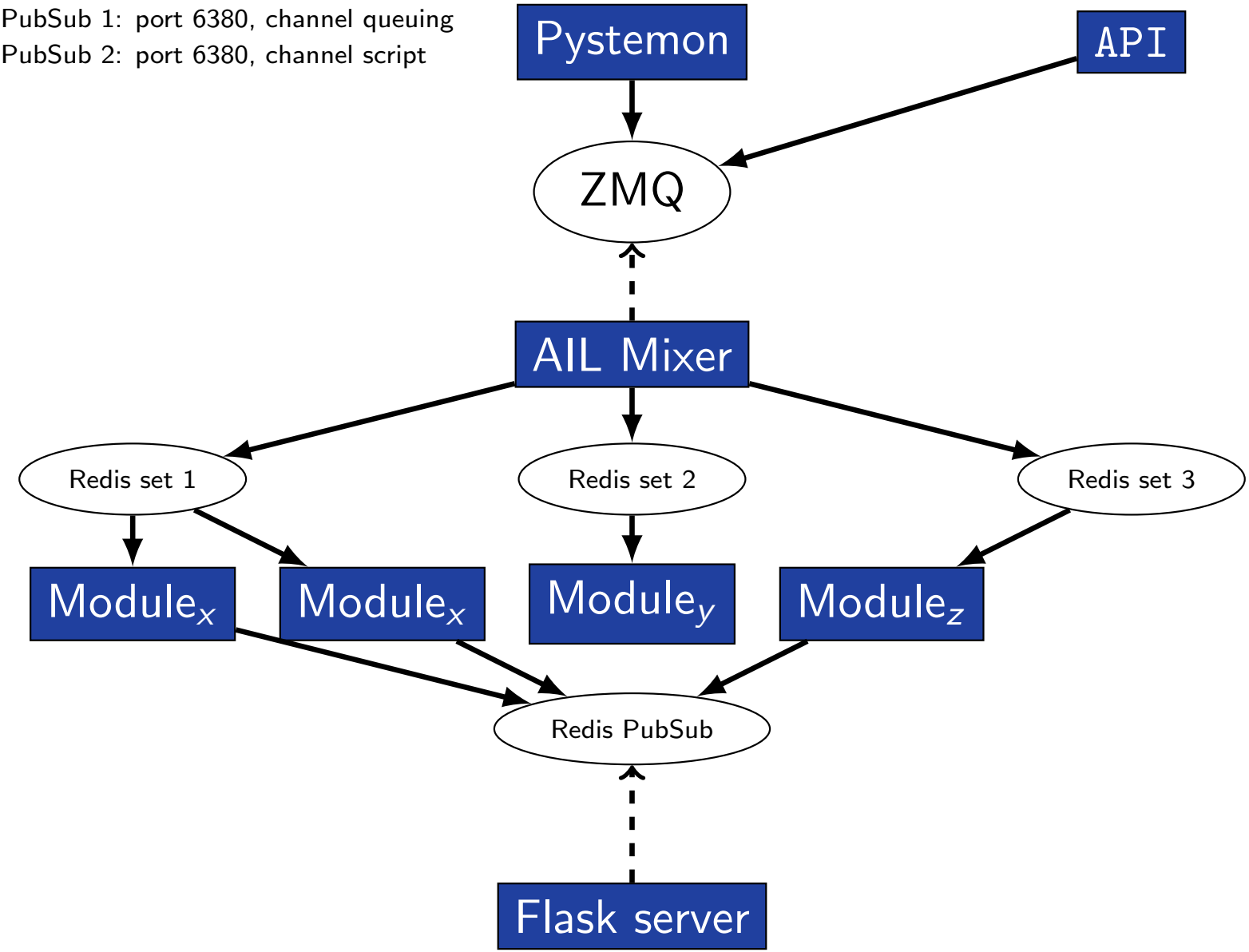
**Data message passing:** ZMQ, Redis list and Redis Publisher/Subscriber

# AIL global architecture 1/2



# AIL global architecture 2/2

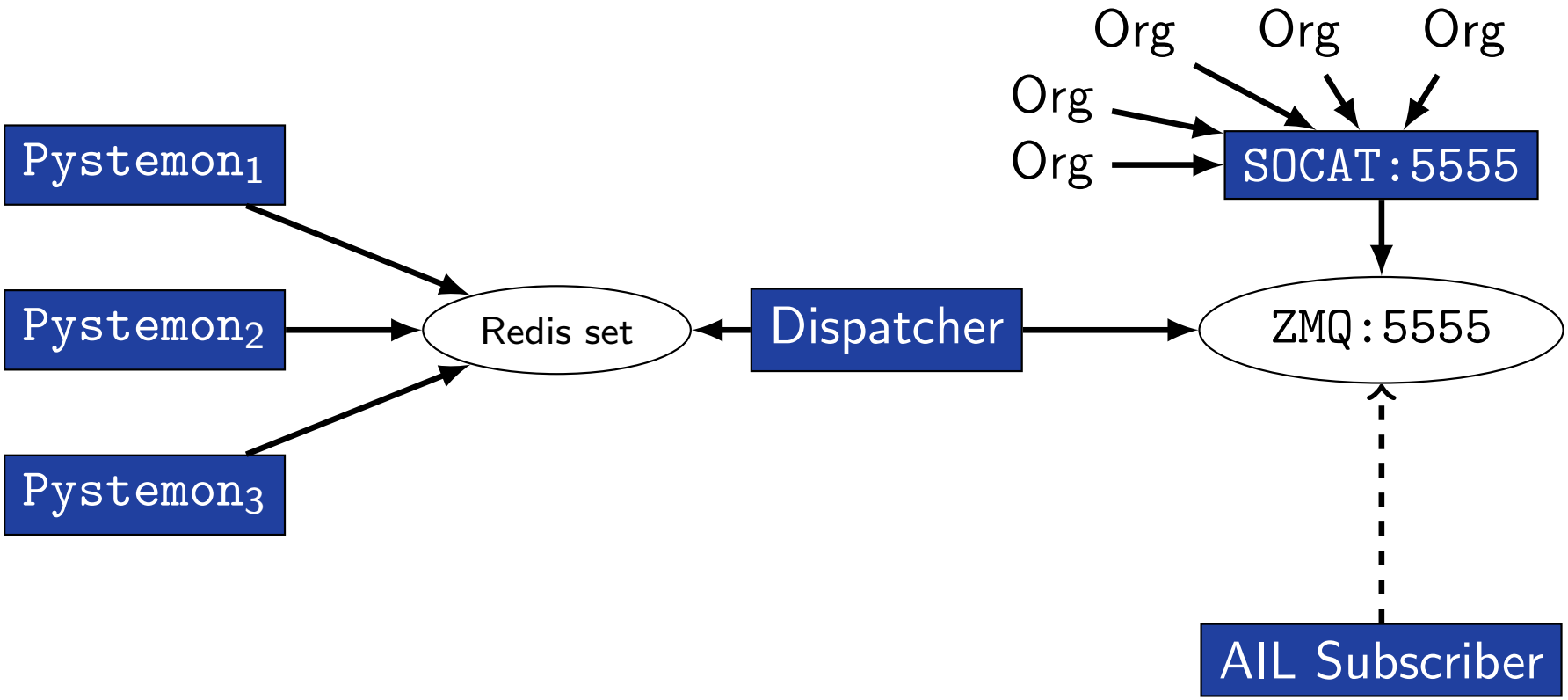
Redis PubSub 1: port 6380, channel queuing
Redis PubSub 2: port 6380, channel script



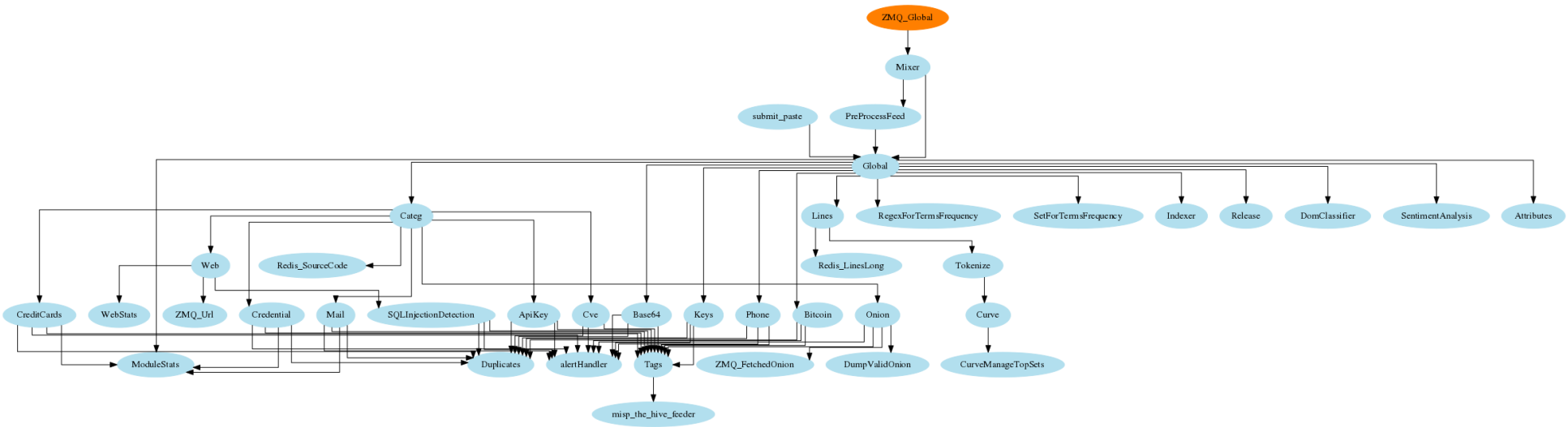
# Data feeder: Gathering pastes with pystemon

## Pystemon global architecture

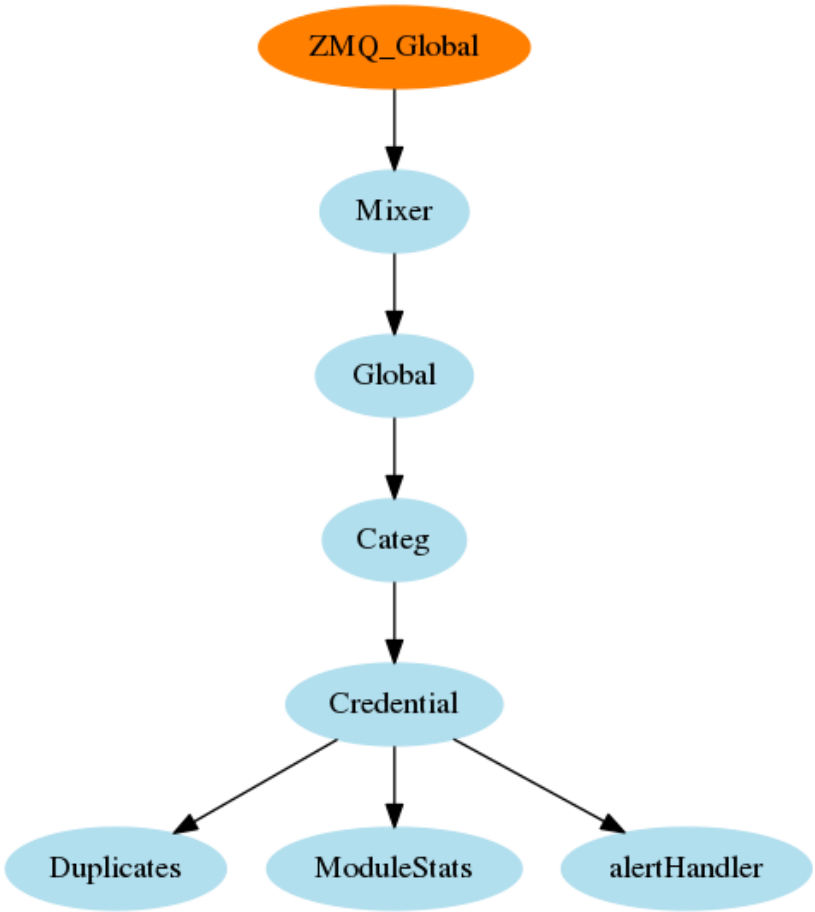
Redis PubSub 1: port 6380, channel queuing
Redis PubSub 2: port 6380, channel script



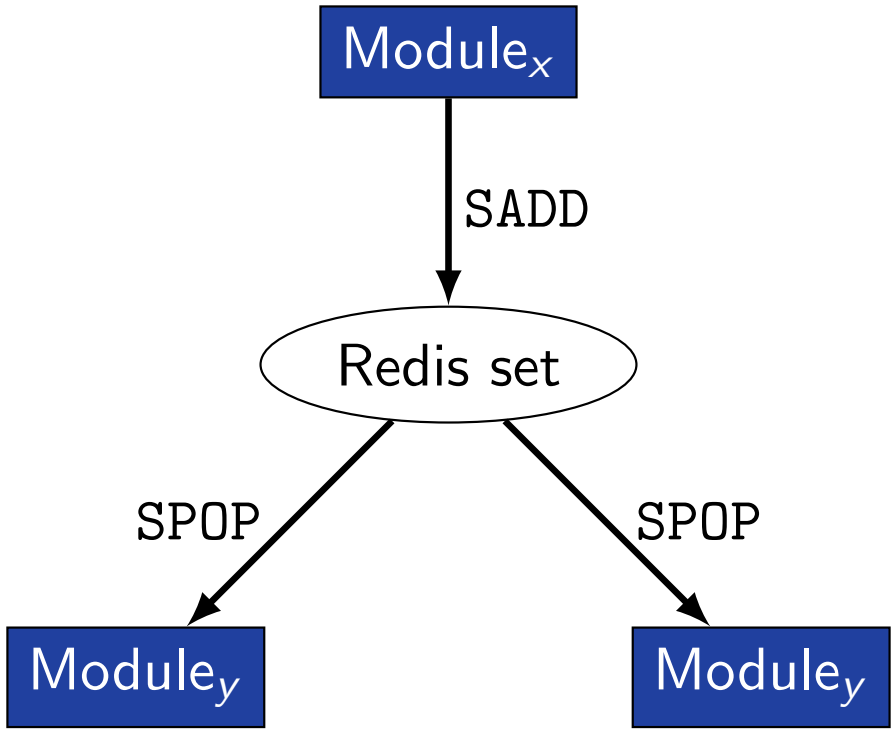
# AIL global architecture: Data streaming between module



# AIL global architecture: Data streaming between module (Credential example)



# Message consuming




→ No message lost nor double processing

→ Multiprocessing!

# Starting the framework

# Running your own instance from source

Make sure that ZMQ_Global→address = `tcp://crf.circl.lu:5556,tcp://127.0.0.1:5556` in configs/core.cfg

**Accessing the environment and starting AIL**

```
1
2 # Launch the system and the web interface
3 cd bin/
4 ./LAUNCH -l
```

# Running your own instance using the virtual machine

**Login and passwords:**

```
# Web interface (default network settings)
    https://127.0.0.1:7000/
# Web interface:
    admin@admin.test
    Password1234
# SSH:
    ail
    Password1234
```

# Feeding the framework

# Feeding AIL

There are different way to feed AIL with data:

1. Be a trusted partner with CIRCL and ask to get access to our feed

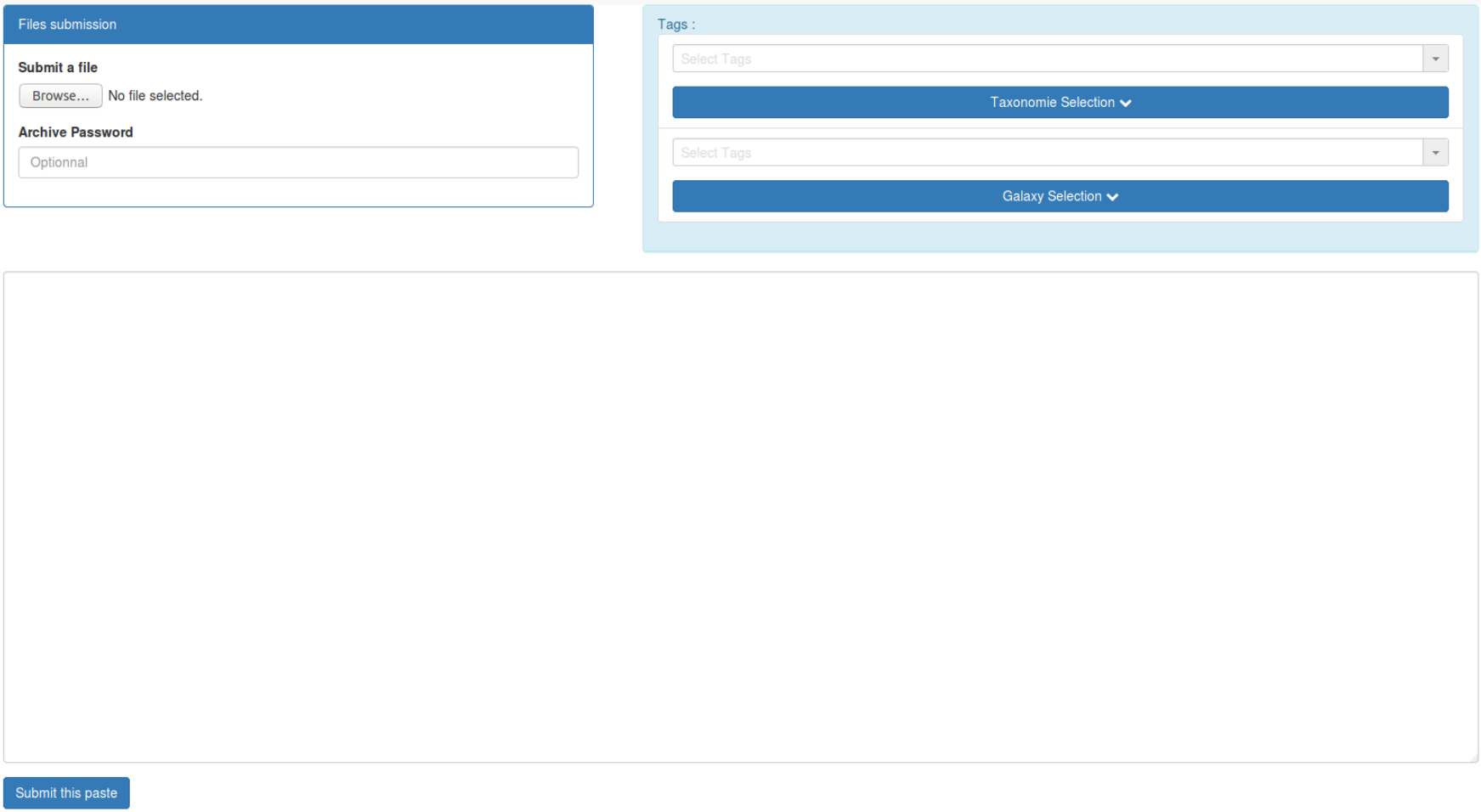
   info@circl.lu
2. Setup *pystemon* and use the custom feeder
   - *pystemon* will collect pastes for you
3. Feed your own data using the API or the `import_dir.py` script
4. Feed your own file/text using the UI (`Submit section`)

# Feeding AIL

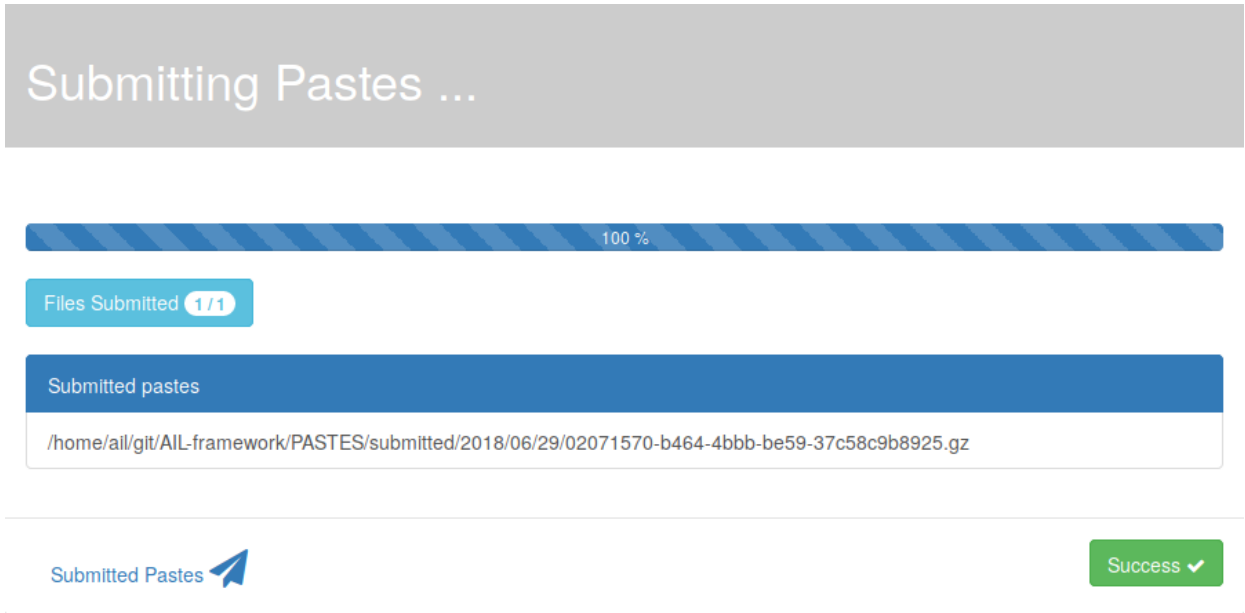
There are different way to feed AIL with data:

1. CIRCL trusted partners can ask to access our feed info@circl.lu
   - ▷ You already have access
2. ~~Setup *pystemon* and use the custom feeder~~
   - ~~*pystemon* will collect pastes for you~~
3. Feed your own data using the API or `import_dir.py` script
4. Feed your own file/text using the UI (`Submit section`)

# Via the UI (1)


Files submission
Submit a file
Browse...
No file selected.
Archive Password
Optionnal
Tags :
Select Tags
Taxonomie Selection
Select Tags
Galaxy Selection
Submit this paste

# Via the UI (2)


Submitting Pastes ...
100 %
Files Submitted 1 / 1
Submitted pastes
/home/ail/git/AIL-framework/PASTES/submitted/2018/06/29/02071570-b464-4bbb-be59-37c58c9b8925.gz
Submitted Pastes
Success

# Feeding AIL with your own data - API

**api/v1/import/item**

```
{
  "type": "text",
  "tags": [
    "infoleak:analyst-detection=\"private-key\""
  ],
  "text": "text to import"
}
```

# Feeding AIL with your own data - `import_dir.py` (1)

/!\ requirements:

- Each file to be fed must be of a reasonable size:
  - $\sim$ `3 Mb` / file is already large
  - This is because some modules are doing regex matching
  - If you want to feed a large file, better split it in multiple ones

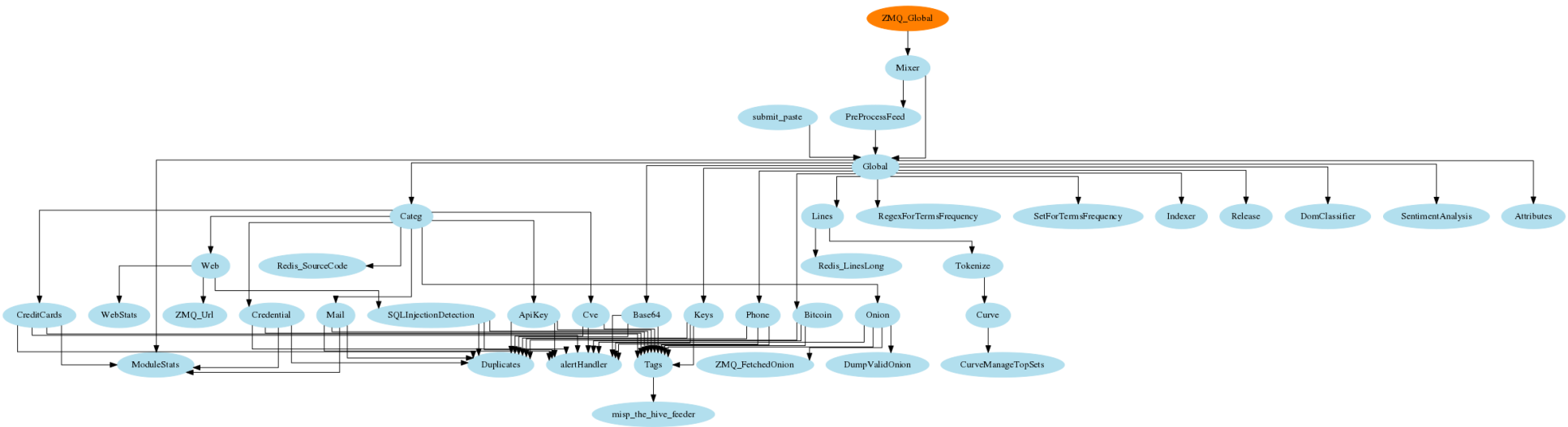
# Feeding AIL with your own data - `import_dir.py` (2)

1. Check your local configuration `bin/package/config.cfg`
   - In the file `bin/package/config.cfg`,
   - Add `127.0.0.1:5556` in `ZMQ_Global`
   - (should already be set by default)
2. Launch `import_dir.py` with de directory you want to import
   - `import_dir.py -d dir_path`

# Creating new features

# Developing new features: Plug-in a module in the system

Choose where to put your module in the data flow:




Then, modify `bin/package/modules.cfg` accordingly

# Writing your own modules - /bin/template.py

```
1  import time
2  from pubsublogger import publisher
3  from Helper import Process
4  if __name__ == '__main__':
5      # logger setup
6      publisher.port = 6380
7      publisher.channel = 'Script'
8      # Section name in configs/core.cfg
9      config_section = '<section name>'
10     # Setup the I/O queues
11     p = Process(config_section)
12     # Endless loop getting messages from the input queue
13     while True:
14         # Get one message from the input queue
15         message = p.get_from_set()
16         if message is None:
17             publisher.debug("{} queue is empty, waiting".format(config_section))
18             time.sleep(1)
19             continue
20         # Do something with the message from the queue
21         something_has_been_done = do_something(message)
22
```

# Practical part

# Practical part: Pick your choice

1. Update support of docker/ansible
2. Graph database on `Credential.py`
   - Top used passwords, most compromised user, ...
3. Webpage scrapper
   - Download html from URL found in pastes
   - Re-inject html as paste in AIL
4. Improvement of `Phone.py`
   - Way to much false positive as of now. Exploring new ways to validate phone numbers could be interesting
5. **Your custom feature**

# Contribution rules

# How to contribute

# Glimpse of contributed features

- Docker
- Ansible
- Email alerting
- SQL injection detection
- Phone number detection

# How to contribute

- Feel free to fork the code, play with it, make some patches or add additional analysis modules.

# How to contribute

- Feel free to fork the code, play with it, make some patches or add additional analysis modules.
- Feel free to make a pull request for your contribution

# How to contribute

- Feel free to fork the code, play with it, make some patches or add additional analysis modules.
- Feel free to make a pull request for your contribution
- That's it!

# Final words

- Building AIL helped us to find additional leaks which cannot be found using manual analysis and **improve the time to detect duplicate/recycled leaks**.

  $\rightarrow$ Therefore quicker response time to assist and/or inform proactively affected constituents.

# Ongoing developments

- Python API wrapper
- **Data retention (export/import)**
- MISP format support (MISP modules expansion)
- auto Classify content by set of terms
  - CE contents
  - DDOS booters
  - ...
- Crawled items
  - duplicate crawled domains
  - tor indexer
  - crawler cookie authentication

# Annexes

# Managing AIL: Old fashion way

**Access the script screen**

```
screen -r Script
```

Table: GNU screen shortcuts

| Shortcut | Action |
|---|---|
| C-a d | detach screen |
| C-a c | Create new window |
| C-a n | next window screen |
| C-a p | previous window screen |

# Managing your modules: Using the helper