AIL Project

Open source framework to efficiently collect, crawl, dig, and analyze unstructured data

CIRCL
Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy
alexandre.dulaunoy@circl.lu

Aurelien Thirion
aurelien.thirion@circl.lu

Jean-Louis Huynen
jean-louis.huynen@circl.lu

Sami Mokaddem
sami.mokaddem@circl.lu

info@circl.lu

Links

- AIL project `https://github.com/ail-project`
- AIL framework
  `https://github.com/ail-project/ail-framework`
- Training materials
  `https://github.com/ail-project/ail-training`
- Online chat `https://gitter.im/ail-project/community`

Legal and Ethics

Ethics in Information Security and Cybersecurity

- The materials and tools presented can open a significant numbers of questions regarding ethics;
- Our researches and tools are there for education, supporting the public good and improve incident response;
- We ask all users and participants to **follow ethical principles and act professionaly**[1].

[1] `https://www.acm.org/code-of-ethics`
`https://www.first.org/global/sigs/ethics/ethics-first`

Collecting, processing and analysing content - web pages

- Building a search engine on the web is a challenging task because:
  - it has to crawl webpages,
  - it has to to make sense of **unstructured data**,
  - it has to **index** these data,
  - it has to provide a way to retrieve data and structure data (e.g. correlation).
- Doing so on Tor is even more challenging because:
  - services don't always want to be found,
  - parts of the dataset have to be discarded.
- in each case, it requires a lot of bandwidth, storage and computing power.

Collecting, processing and analysing content - structured data

- Some data are structured and are easy to process:
  - metadata!
  - API responses.
- Some even provide cryptographic evidences:
  - authentication mechanisms between peers,
  - OpenGPG can leak a lot of metadata
    - key ids,
    - subject of email in thunderbird,
  - Bitcoin's Blockchain is public,
  - pivoting on these data with external sources yields interesting results.

AIL Design Objectives

Session Objectives

- Demonstrate the practical usage and extensibility of an open source tool for monitoring web pages, pastes, forums, and hidden services
- Discuss the challenges involved and delve into the design principles of the AIL open source framework
- Explore various **collection mechanisms and sources utilized** by the AIL framework
- Gain knowledge on creating new modules within the AIL framework
- Acquire proficiency in using, installing, and initializing AIL
- Understand the significance of integrating the AIL framework into the cyber threat intelligence life cycle, with notable tools such as MISP

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.
- In 2020, AIL framework became a complete project called **ail project**[2].
- In 2023, AIL framework version 5.0 released with a new datastorage back-end.

[2] `https://github.com/ail-project/`

Capabilities Overview

Common usage

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

Support CERT and Law Enforcement activities

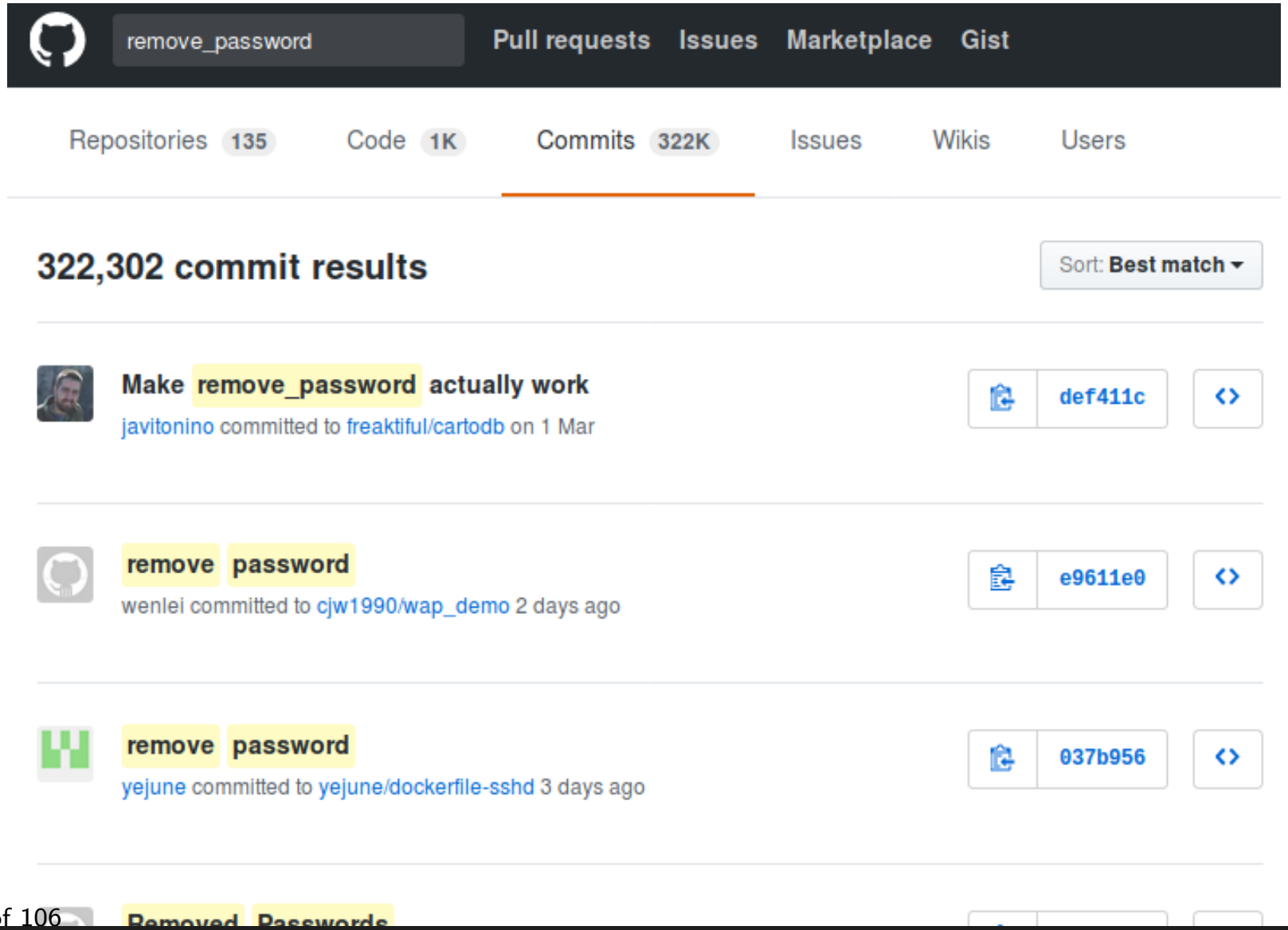
- Proactive investigation: leaks detection
  - List of emails and passwords
  - Leaked database
  - AWS Keys
  - Credit-cards
  - PGP private keys
  - Certificate private keys
- Feed Passive DNS or any passive collection system
- CVE and PoC of vulnerabilities most used by attackers

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/YARA monitoring for local companies/government

Sources of leaks

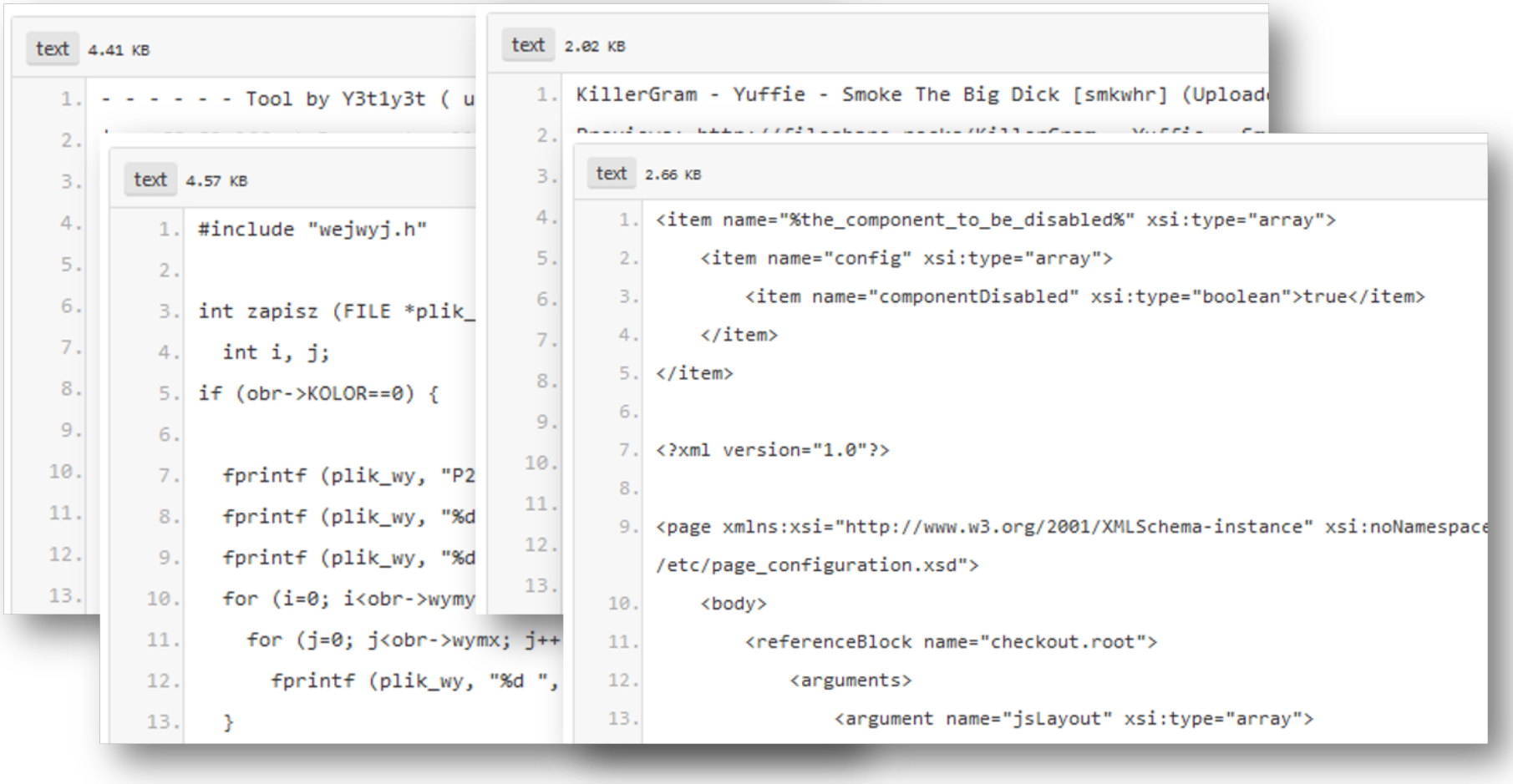
Mistakes from users:



Sources of leaks: Paste monitoring

- Example: `https://gist.github.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 (items)



text 4.41 KB

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

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 2.02 KB

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

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)
- Ransom model (e.g. To publicly pressure the victims)
- 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[3] or reselling personal information - reality versus media coverage)
- Analysing collateral effects of malware, software vulnerabilities or exfiltration

$\rightarrow$ And it's important to detect them automatically.

[3] `https://github.com/D4-project/`

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 (handle cookies authentication)

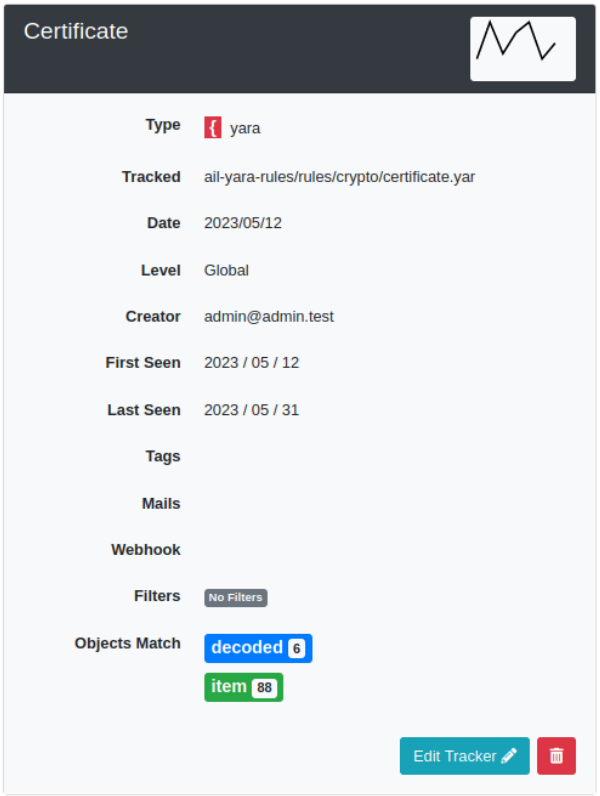
AIL Framework: Current features

- 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, regex and YARA **tracking and occurrences**
- Archives, files and raw **submission** from the UI
- PGP, Cryptocurrency, Decoded (Base64, ...) and username Correlation
- And many more

Trackers - Retro Hunt

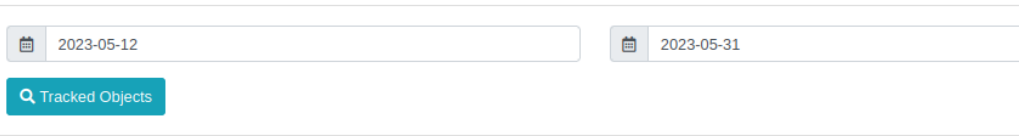
- Search and monitor specific keywords/patterns
  - Automatic Tagging
  - Email Notifications
- Track Word
  - ddos
- Track Set
  - booter,ddos,stresser;2
- Track Regex
  - circl\.lu
- YARA rules
  - https://github.com/ail-project/ail-yara-rules

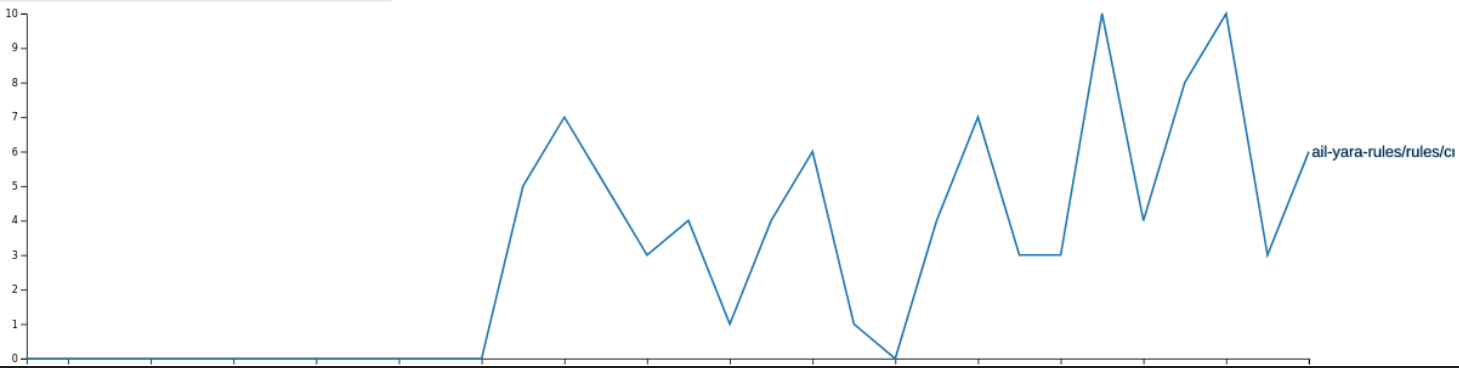
YARA Tracker



Yara Rule:

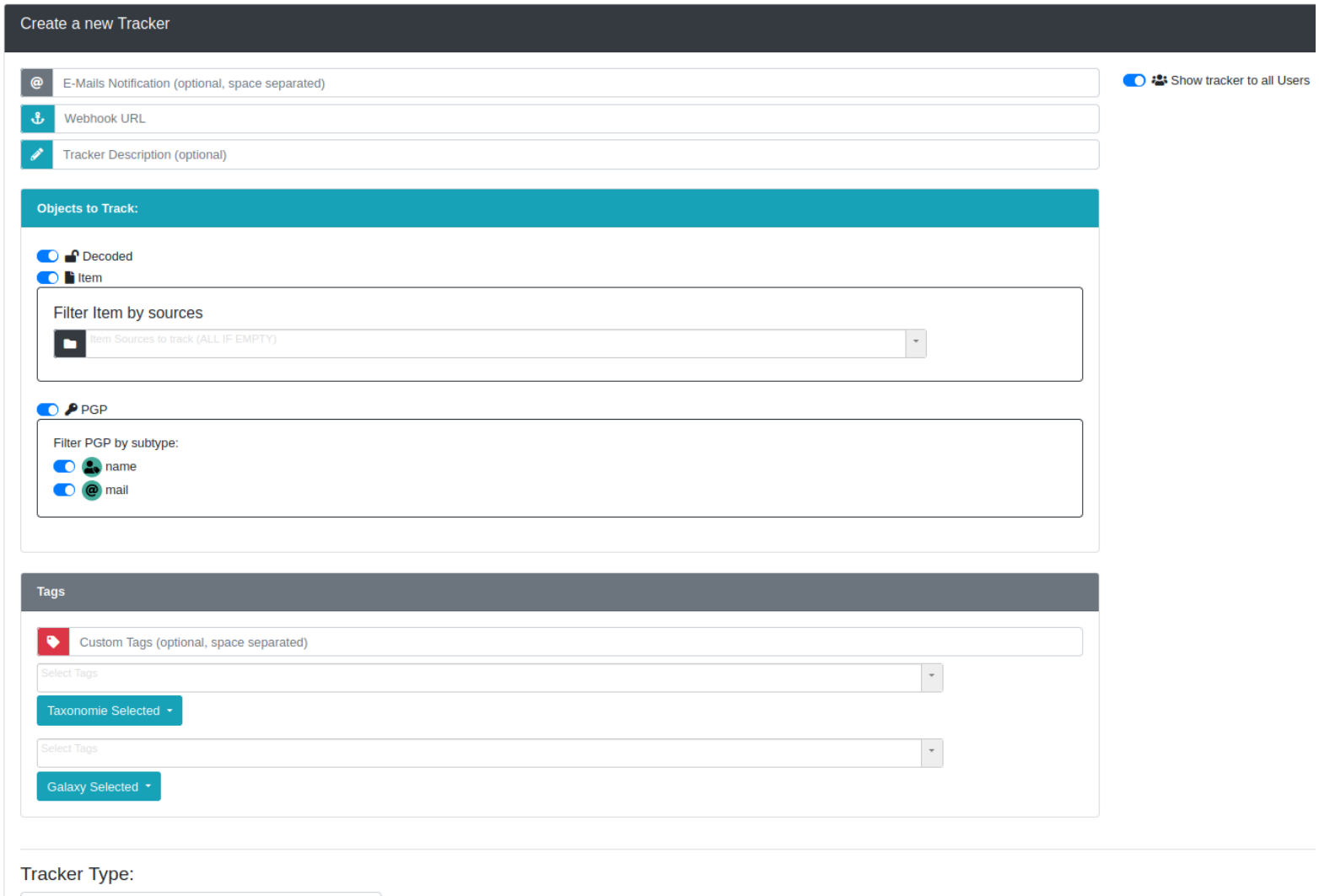
```
rule certificates
{
    meta:
        author = "@KevTheHermit"
        info = "Part of PasteHunter"
        reference = "https://github.com/kevthehermit/PasteHunter"

    strings:
        $ssh_priv = "BEGIN RSA PRIVATE KEY" wide ascii nocase
        $openssh_priv = "BEGIN OPENSSH PRIVATE KEY" wide ascii nocase
        $dsa_priv = "BEGIN DSA PRIVATE KEY" wide ascii nocase
        $ec_priv = "BEGIN EC PRIVATE KEY" wide ascii nocase
        $pgp_priv = "BEGIN PGP PRIVATE KEY" wide ascii nocase
        $pem_cert = "BEGIN CERTIFICATE" wide ascii nocase
        $pkcs7 = "BEGIN PKCS7"

    condition:
        any of them

}
```





Trackers - Practical part

- **Create and test** your own tracker



Retro Hunt

test

completed

Date 2023/05/10

Description None

Tags

Creator admin@admin.test

Filters

```
{
    "item": {
        "date_from": "20230304",
        "date_to": "20230601"
    }
}
```

Objects Match item 3

Show Objects

```
rule certificates
{
    meta:
        author = "@KevTheHermit"
        info = "Part of PasteHunter"
        reference = "https://github.com/kevthehermit/PasteHunter"

    strings:
        $ssh_priv = "BEGIN RSA PRIVATE KEY" wide ascii nocase
        $openssh_priv = "BEGIN OPENSSH PRIVATE KEY" wide ascii nocase
        $dsa_priv = "BEGIN DSA PRIVATE KEY" wide ascii nocase
        $ec_priv = "BEGIN EC PRIVATE KEY" wide ascii nocase
        $pgp_priv = "BEGIN PGP PRIVATE KEY" wide ascii nocase
        $pem_cert = "BEGIN CERTIFICATE" wide ascii nocase
        $pkcs7 = "BEGIN PKCS7"

    condition:
        any of them

}
```

Show 10 entries

Search:

Type	Id	Tags
	archive/gist.github.com/2023/04/14/luizmiranda7_3b3d1133a3d3842092c5fc5fb39e84f2.gz	infoleak:automatic-detection="private-key" test23 test12 infoleak:automatic-detection="certificate"
	submitted/2023/04/20/submitted_cc9190ab-80d2-4d2b-9c9e-97c51e69a855.gz	infoleak:submission="manual" test12 infoleak:automatic-detection="rsa-private-key" infoleak:automatic-detection="vpn-static-key" test23 infoleak:automatic-detection="certificate" infoleak:automatic-detection="onion"
	archive/gist.github.com/2023/04/13/chipzoller_d8d6d2d737d02ad4fe9d30a897170761.gz	test12 test23 infoleak:automatic-detection="certificate"

Showing 1 to 3 of 3 entries

Previous 1 Next

Recon and intelligence gathering tools

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

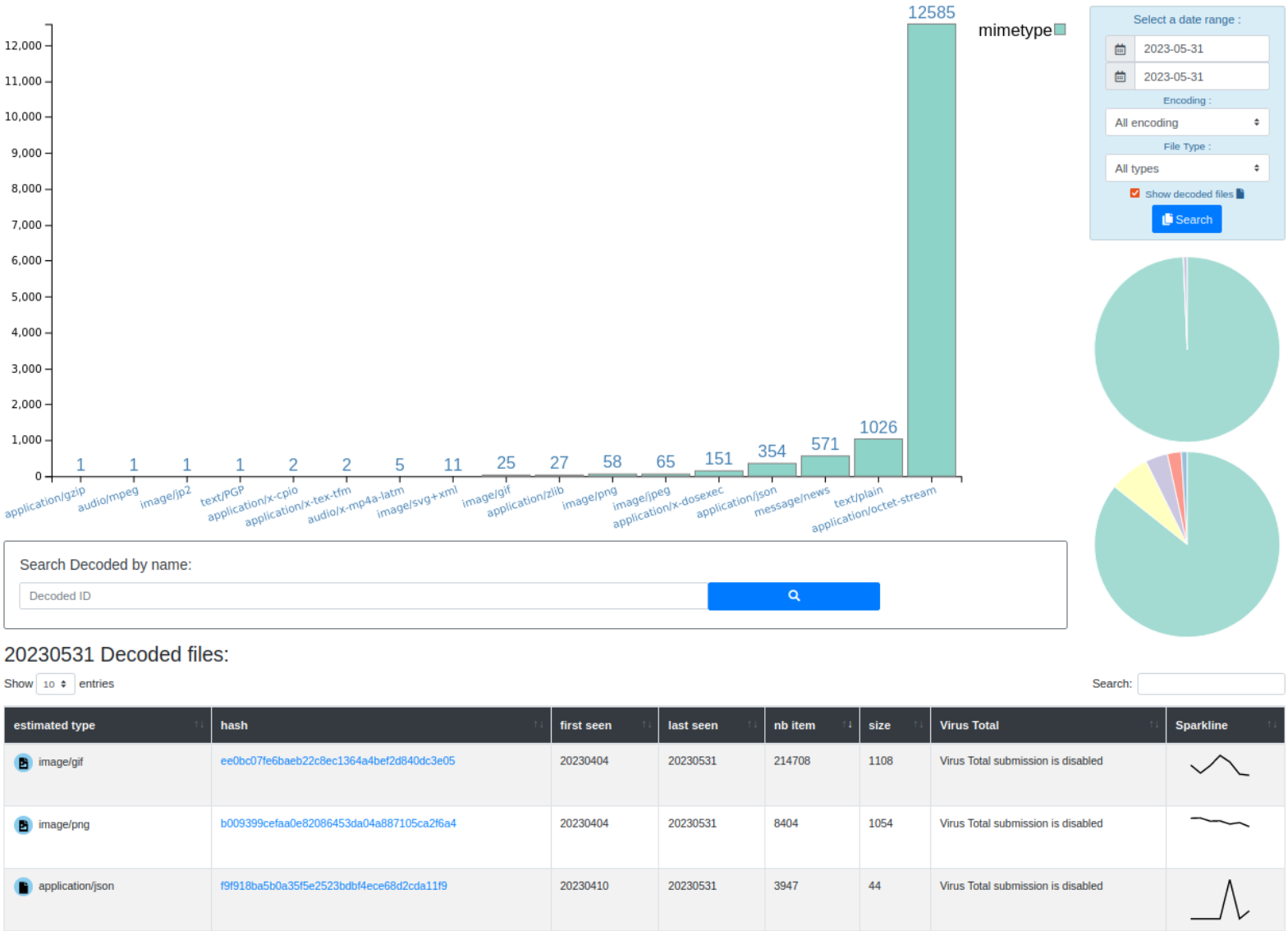
Recon and intelligence gathering tools

```
################################################################################
================================================================================
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 name: NIC Chile
Registrar URL: https://www.nic.cl
```

Decoder

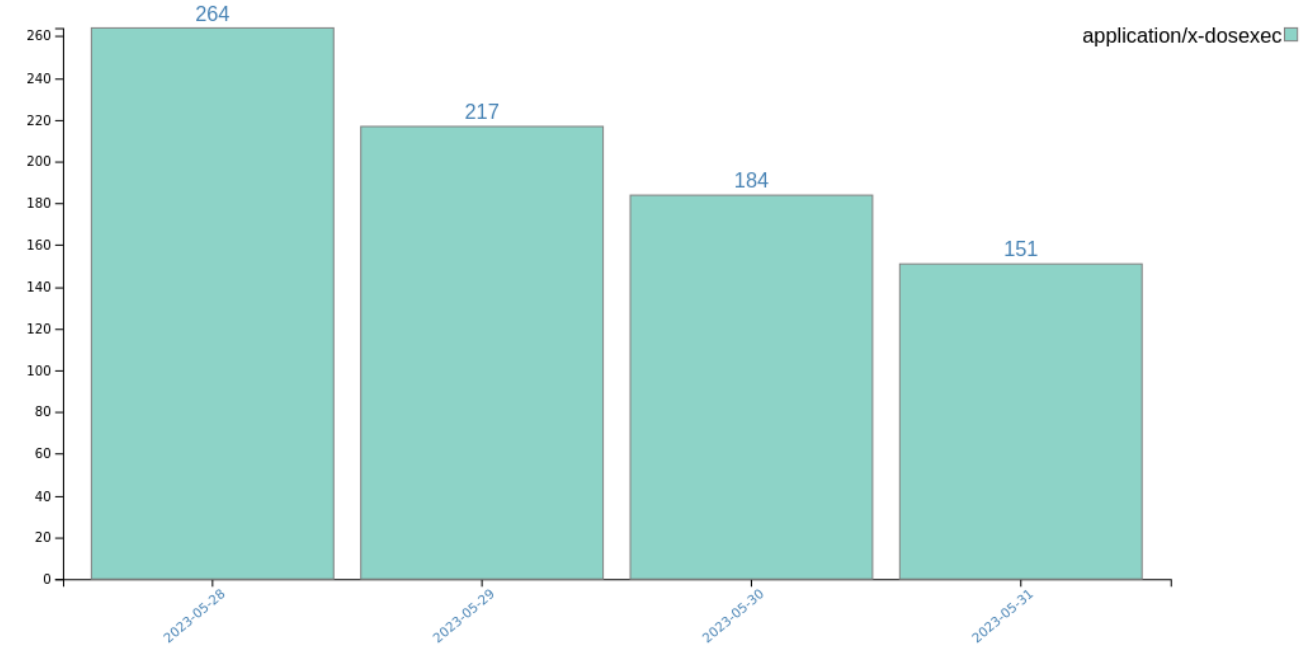
- Search for encoded strings
  - Base64
  - Hexadecimal
  - Binary
- Guess Mime-type
- Items/Domains Correlation

Decoder:



estimated type	hash	first seen	last seen	nb item	size	Virus Total	Sparkline
image/gif	ee0bc07fe6baeb22c8ec1364a4bef2d840dc3e05	20230404	20230531	214708	1108	Virus Total submission is disabled	
image/png	b009399cefaa0e82086453da04a887105ca2f6a4	20230404	20230531	8404	1054	Virus Total submission is disabled	
application/json	f9f918ba5b0a35f5e2523bdbf4ece68d2cda11f9	20230410	20230531	3947	44	Virus Total submission is disabled	

Decoder:



Select a date range :

2023-05-28

2023-05-31

Encoding :

All encoding

File Type :

application/x-dosexec

Show decoded files

Search

Search Decoded by name:

Decoded ID

20230528 to 20230531 Decoded files:

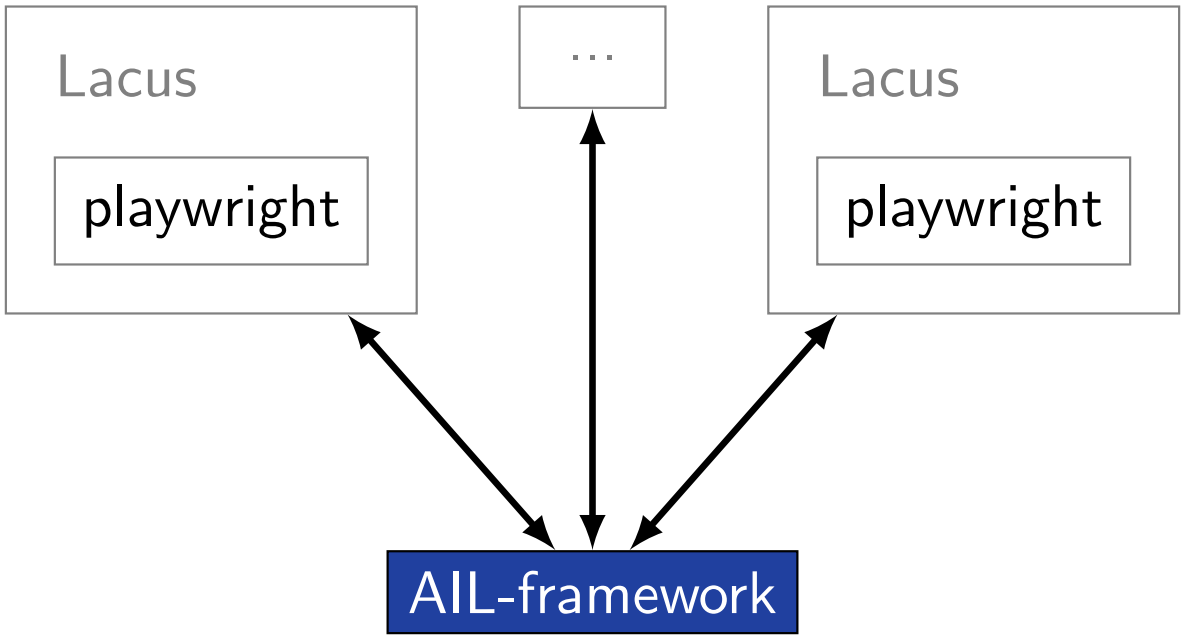
Show 10 entries

Search:

estimated type	hash	first seen	last seen	nb item	size	Virus Total	Sparkline
application/x-dosexec	c408501f7f2bd8279704c380dd61d329c6119fc2	20230421	20230529	76	64	Virus Total submission is disabled	
application/x-dosexec	a9ecbb74ce7d22b70f0dcf0f57299f31ce570161	20230405	20230530	56	55666	Virus Total submission is disabled	
application/x-dosexec	e5805aa6a66c6e013d5ebddbab4fbc45b4c84127	20230529	20230531	4	32	Virus Total submission is disabled	

Crawler

- Crawlers are used to navigate on regular website as well as .onion addresses (via automatic extraction of urls or manual submission)
- Lacus[4] ("scriptable" browser) is rending the pages (including javascript) and produce screenshots (HAR archive too)



[4] https://github.com/ail-project/lacus

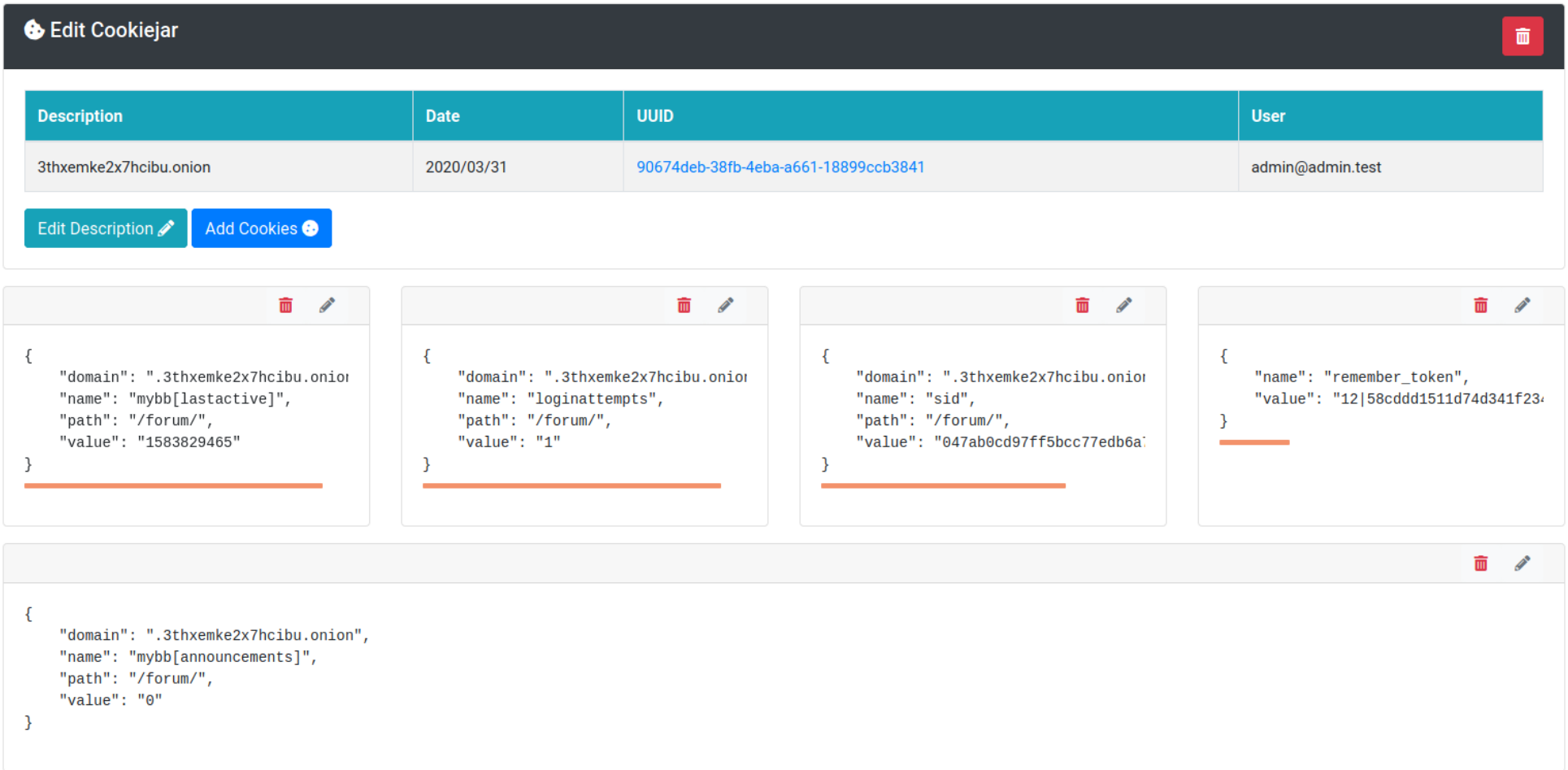
Crawler

How a domain is crawled by default

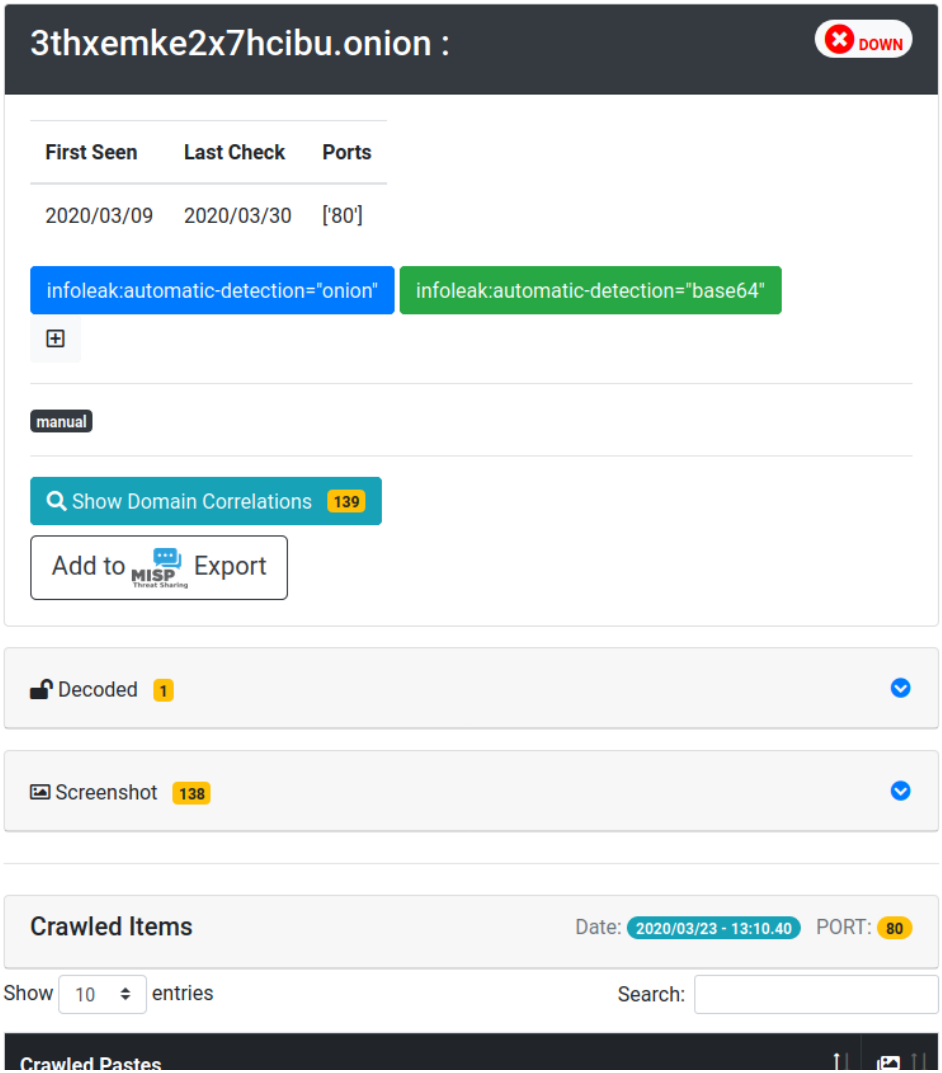
1. Fetch the first url
2. Render javascript (webkit browser)
3. Extract all urls
4. Filter url: keep all url of this domain
5. crawl next url (max depth = 1)

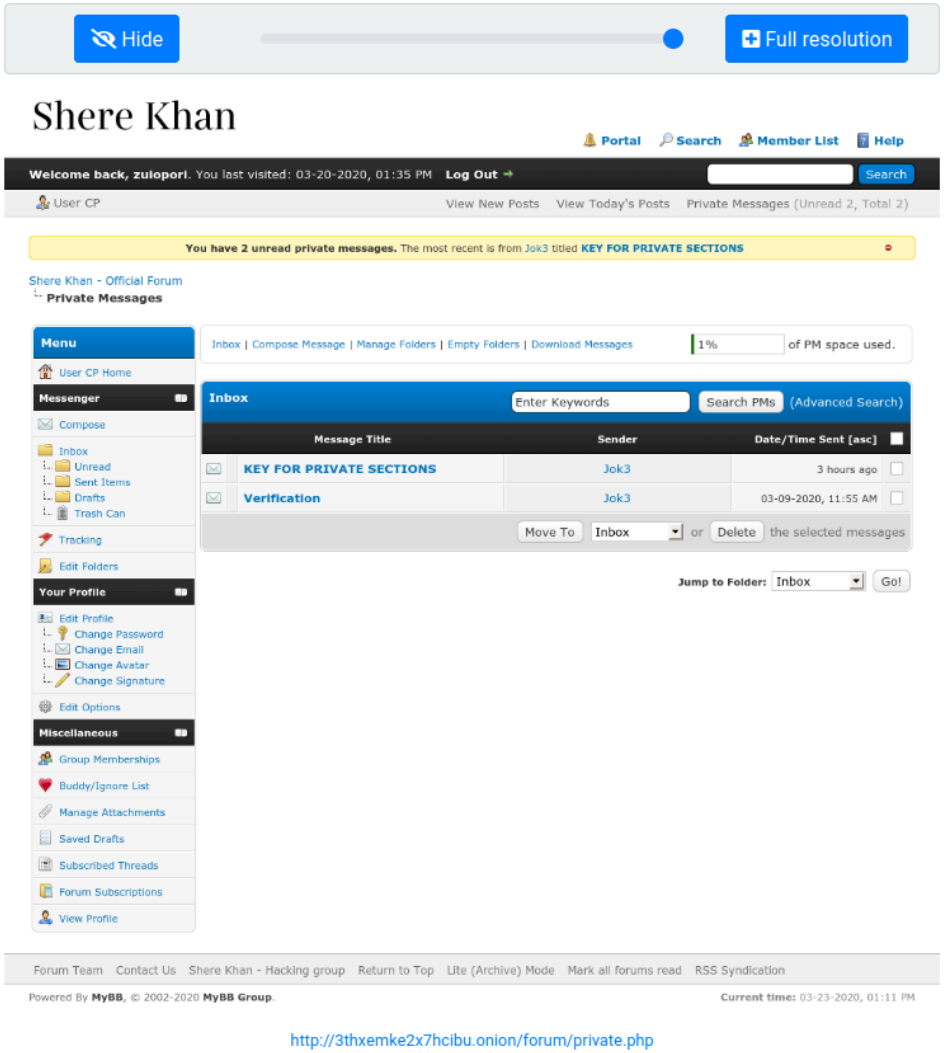
Crawler: Cookiejar

Use your cookies to login and bypass captcha



Crawler: Cookiejar





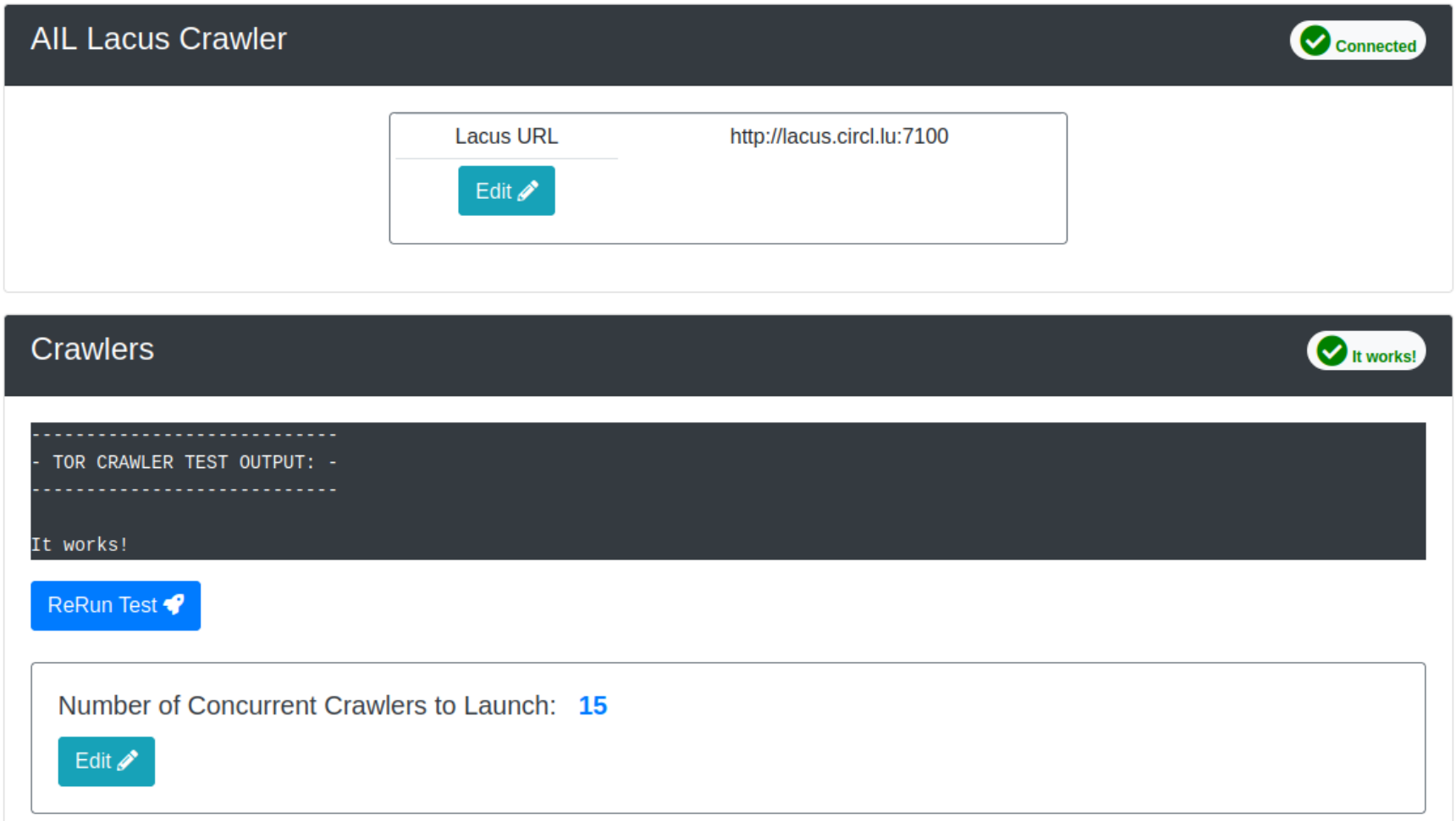
http://3thxemke2x7hcibu.onion/forum/private.php

Lacus

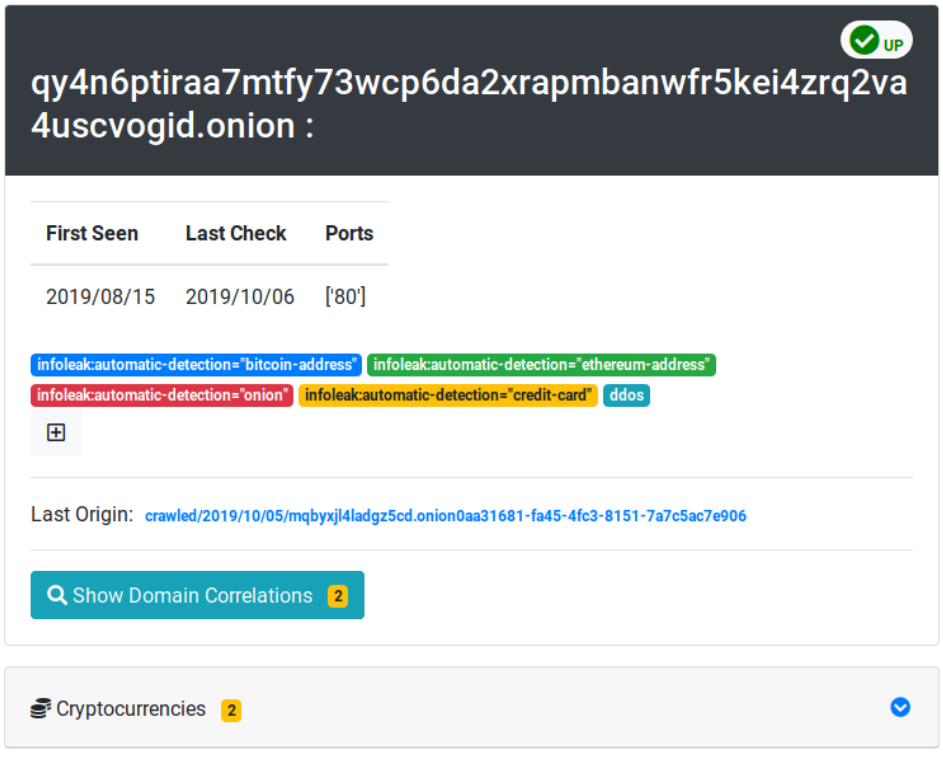
- Lacus[5] is a capturing system using playwright, as a web service
- AIL utilizes Lacus for fetching and rendering domains.
  - Lacus can be installed and executed outside of AIL,
  - Enqueue what you want to capture,
  - Trigger the capture,
  - Get the capture result,

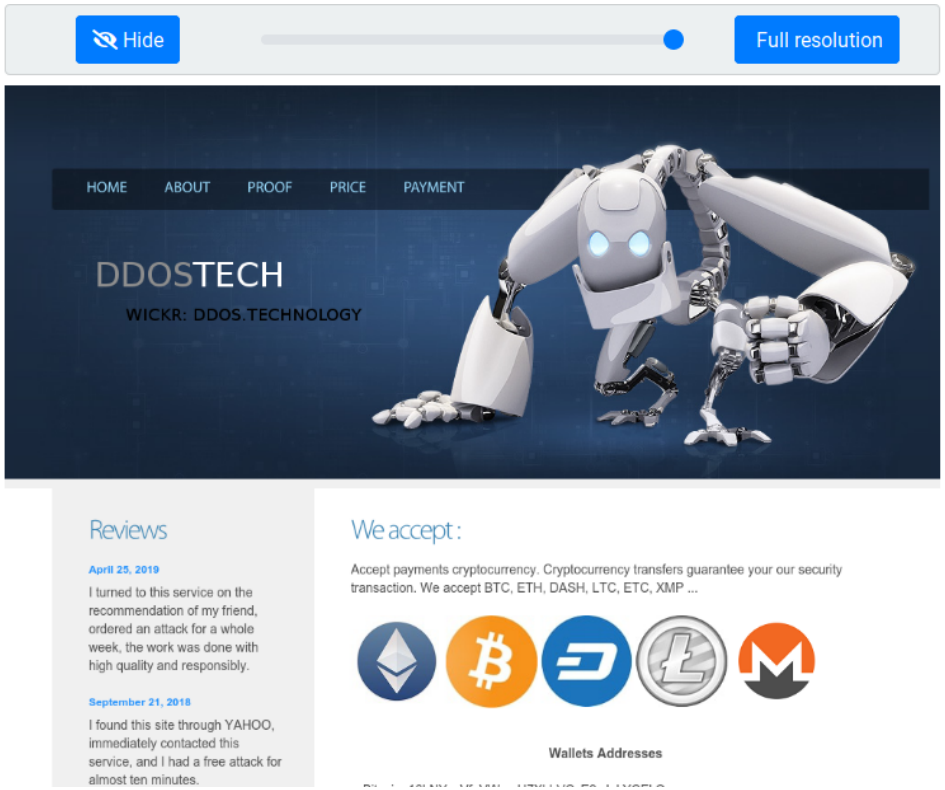
[5] `https://github.com/ail-project/lacus`

Crawler Settings - Lacus

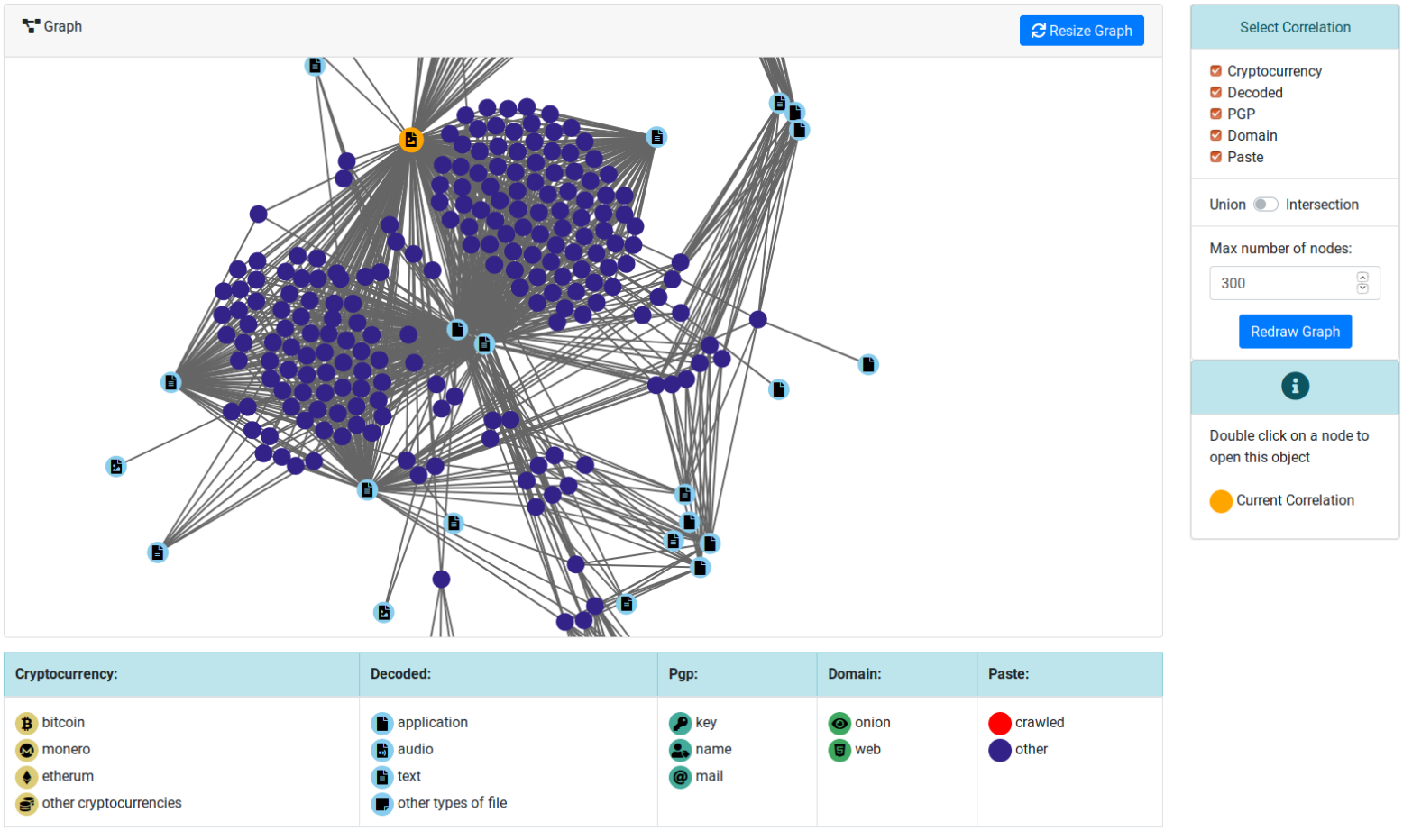


Crawler: DDoS Booter

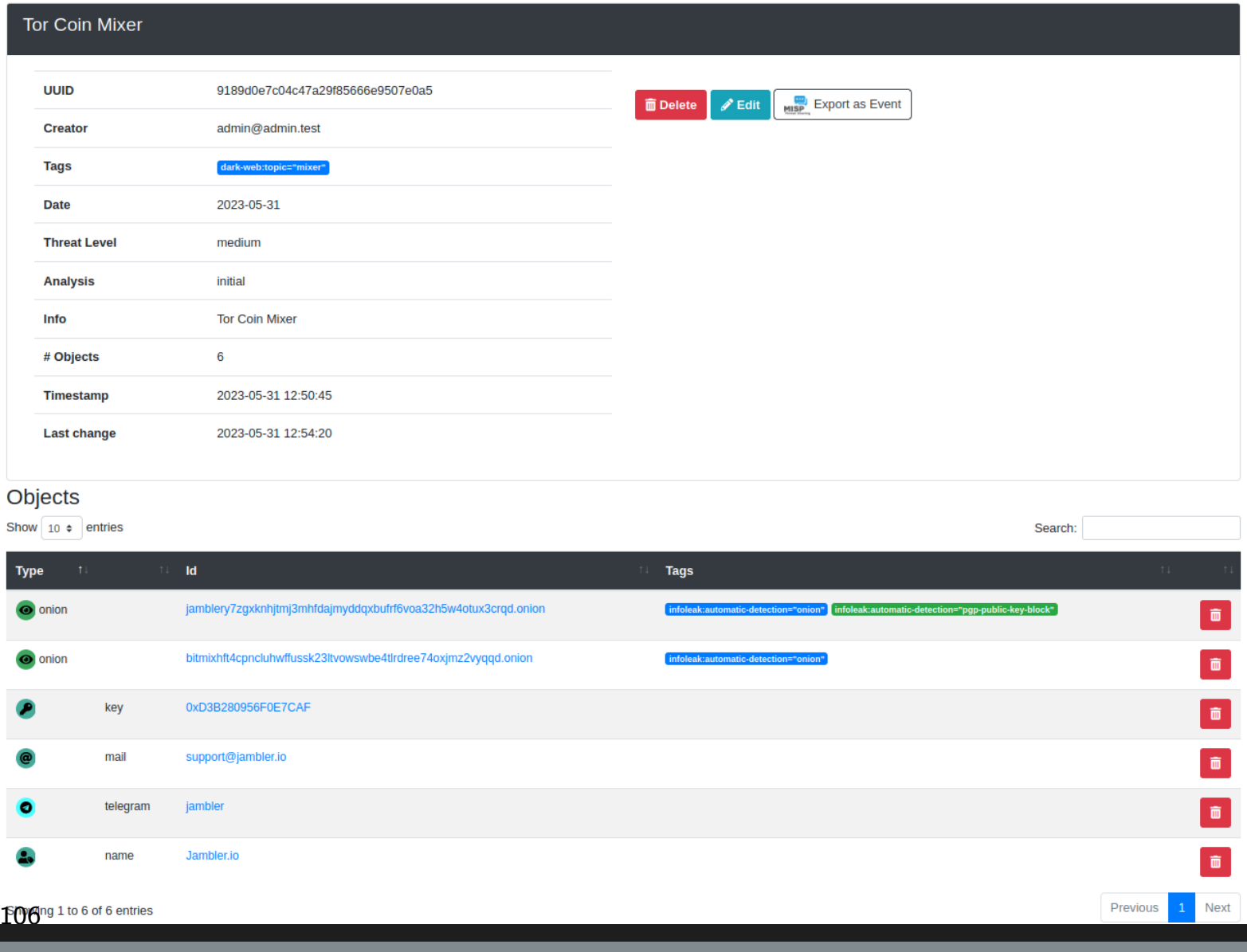




Correlations and relationship

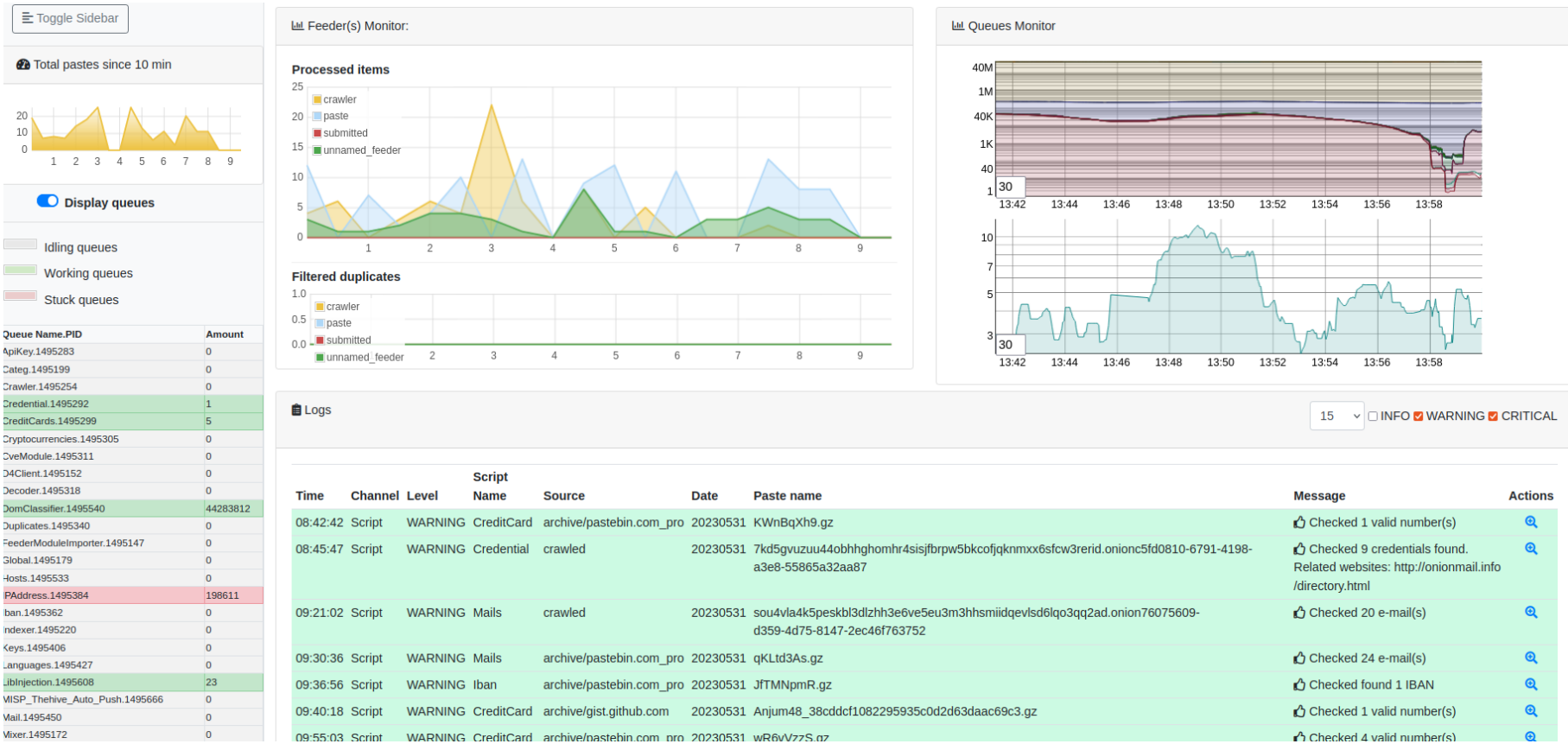


Investigations



Live demo!

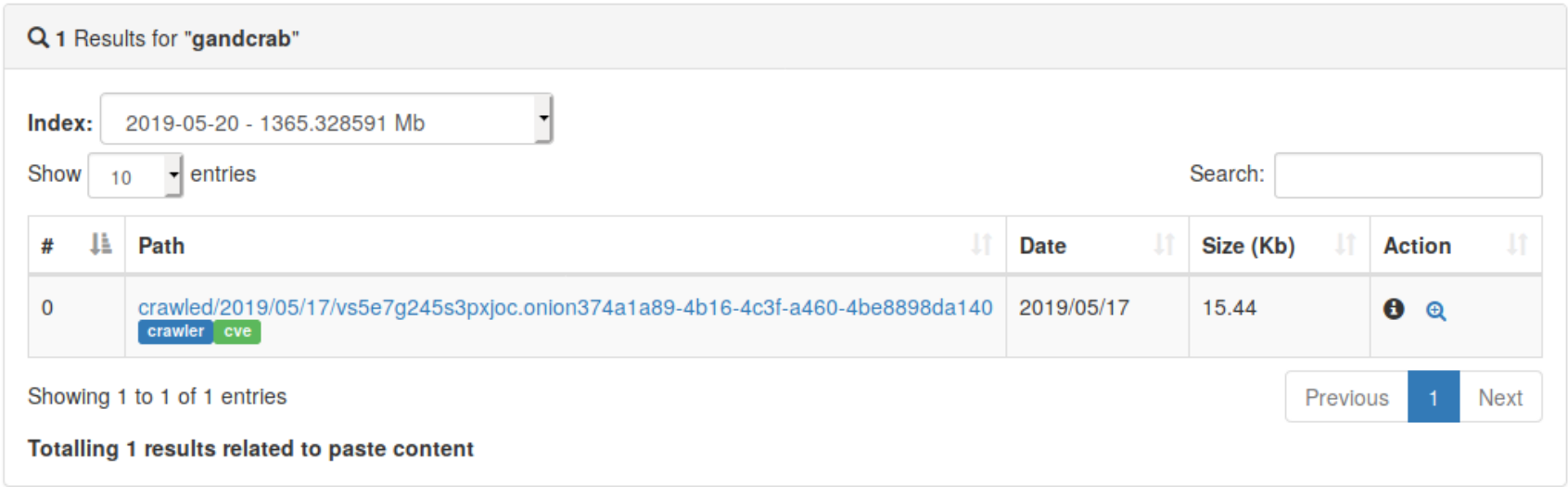
Example: Dashboard



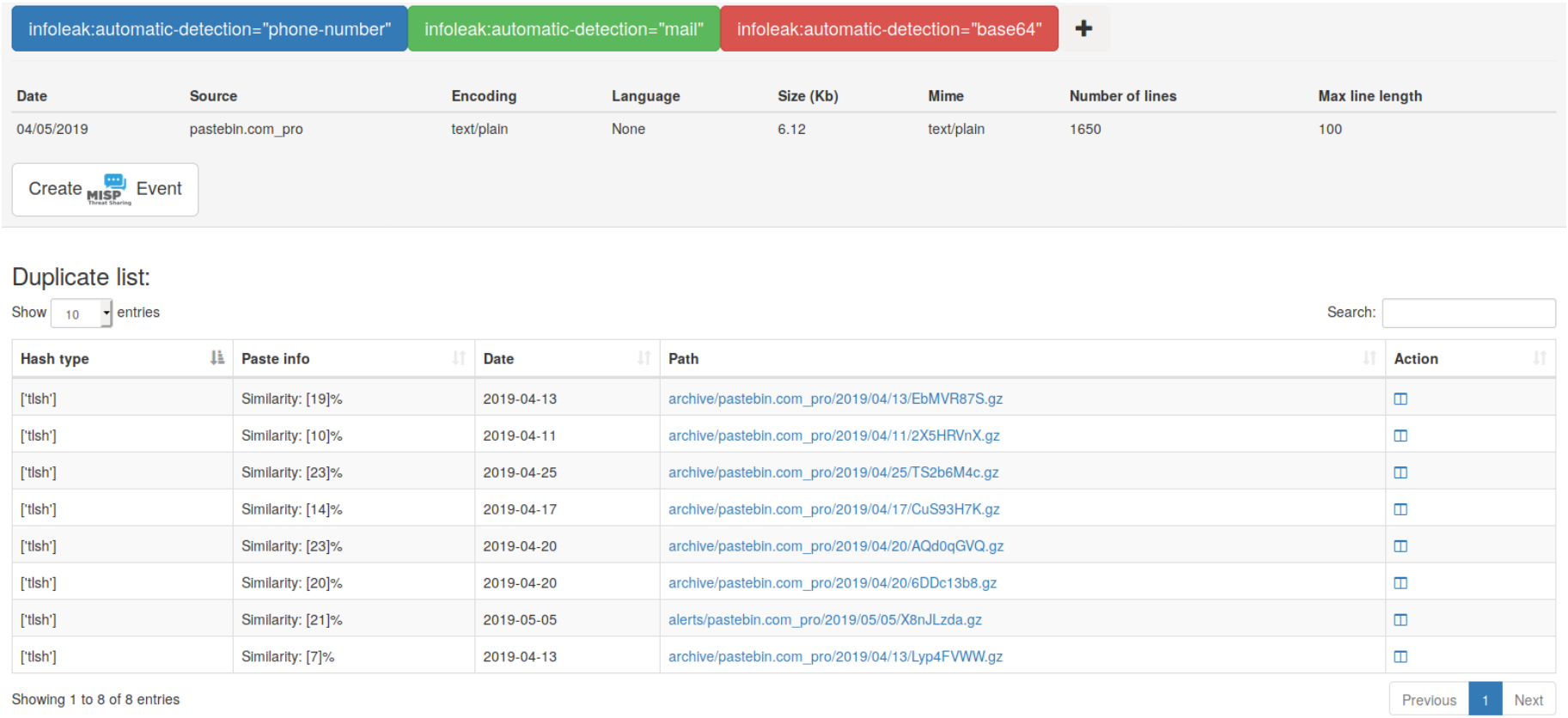
Queue Name.PID	Amount
ApiKey.1495283	0
Categ.1495199	0
Crawler.1495254	0
Credential.1495292	1
CreditCards.1495299	5
Cryptocurrencies.1495305	0
CveModule.1495311	0
D4Client.1495152	0
Decoder.1495318	0
DomClassifier.1495540	44283812
Duplicates.1495340	0
FeederModuleImporter.1495147	0
Global.1495179	0
Hosts.1495533	0
IPAddress.1495384	198611
Iban.1495362	0
Indexer.1495220	0
Keys.1495406	0
Languages.1495427	0
LibInjection.1495608	23
MISP_Thehive_Auto_Push.1495666	0
Mail.1495450	0
Mixer.1495172	0

Time	Channel	Level	Script Name	Source	Date	Paste name	Message	Actions
08:42:42	Script	WARNING	CreditCard	archive/pastebin.com_pro	20230531	KWnBqXh9.gz	Checked 1 valid number(s)	
08:45:47	Script	WARNING	Credential	crawled	20230531	7kd5gvuzuu44obhhghomhr4sisjfbrpw5bkcofjqknmxx6sfcw3rerid.onionc5fd0810-6791-4198-a3e8-55865a32aa87	Checked 9 credentials found. Related websites: http://onionmail.info/directory.html	
09:21:02	Script	WARNING	Mails	crawled	20230531	sou4vla4k5peskbl3dlzhh3e6ve5eu3m3hhsmiidqevlsd6lqo3qq2ad.onion76075609-d359-4d75-8147-2ec46f763752	Checked 20 e-mail(s)	
09:30:36	Script	WARNING	Mails	archive/pastebin.com_pro	20230531	qKLtd3As.gz	Checked 24 e-mail(s)	
09:36:56	Script	WARNING	Iban	archive/pastebin.com_pro	20230531	JfTMNpmR.gz	Checked found 1 IBAN	
09:40:18	Script	WARNING	CreditCard	archive/gist.github.com	20230531	Anjum48_38cddcf1082295935c0d2d63daac69c3.gz	Checked 1 valid number(s)	
09:55:03	Script	WARNING	CreditCard	archive/pastebin.com_pro	20230531	wR6vVzzS.gz	Checked 4 valid number(s)	

Example: Text search



Example: Items Metadata (1)



Example: Items Metadata (2)

Hash files:

Show 5 entries

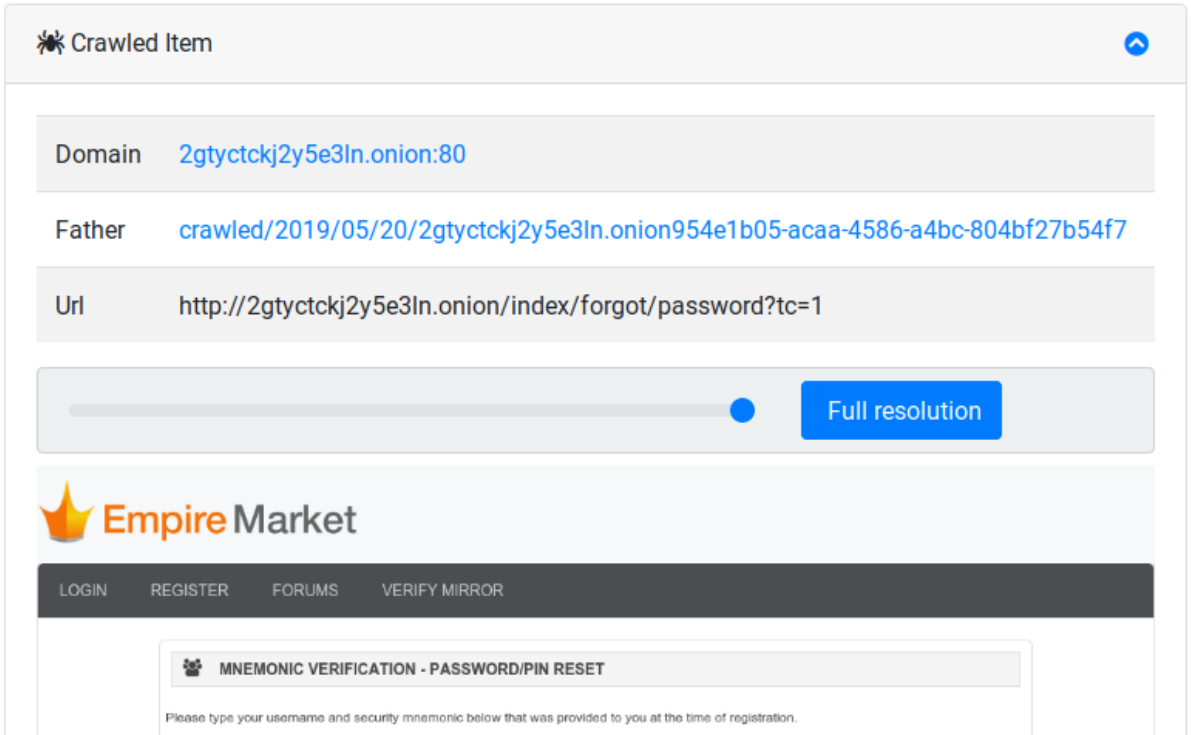
Search:

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: Items Metadata (3)



Example: Browsing content

Content:

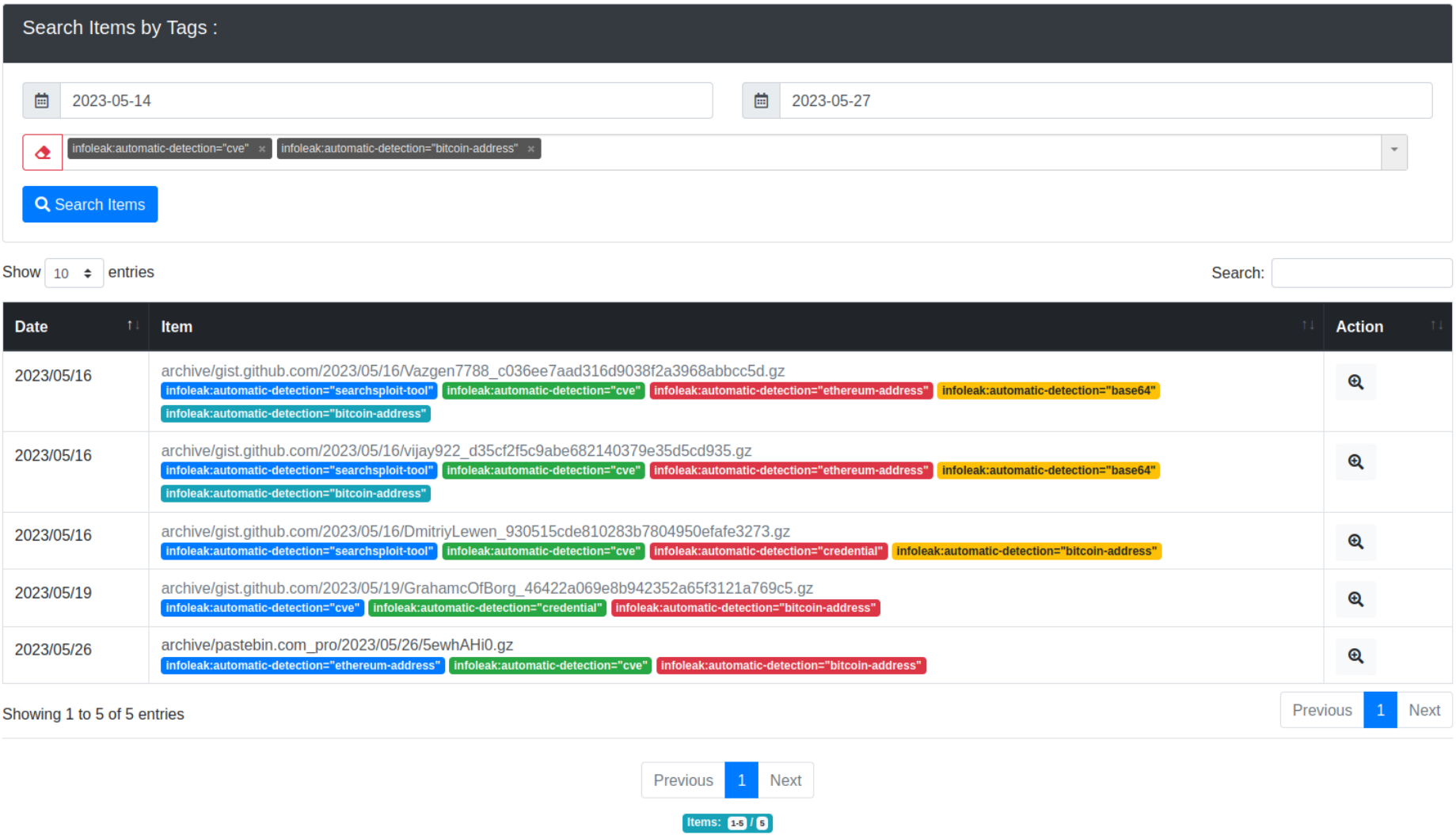
```
http://members2.mofosnetwork.com/access/login/
somosextremos:buddy1990
brazzers_glenn:cocklick
brazzers61:braves01

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://www.ariellaferrera.com/
```

Example: Search by tags



MISP

MISP Taxonomies

- **Tagging** is a simple way to attach a classification to an event or attribute.
- **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 [6].
- Can be easily cherry-picked or extended

[6] `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**[7]: Evaluate the degree of identifiability of personal data and the types of pseudonymous data, de-identified data and anonymous data.

[7] Future of Privacy Forum

Taxonomies useful in AIL

- **tor**: Describe Tor network infrastructure.
- **dark-web**: Criminal motivation on the dark web.
- **copine-scale**[8]: Categorise the severity of images of child sex abuse.

[8] Combating Paedophile Information Networks in Europe

threat sharing and incident response platforms

$\longrightarrow$

Goal: submission to threat sharing and incident response platforms.

threat sharing and incident response platforms

1. Use infoleak taxonomy[9]
2. Add your own tags
3. Export AIL objects to MISP core format
4. Download it or Create a MISP Event[10]

[9] https://www.misp-project.org/taxonomies.html
[10] https://www.misp-standard.org/rfc/misp-standard-core.txt

MISP Export

1Gt545E48EPsyTC8voKQDCFfpTkwiuXduw :

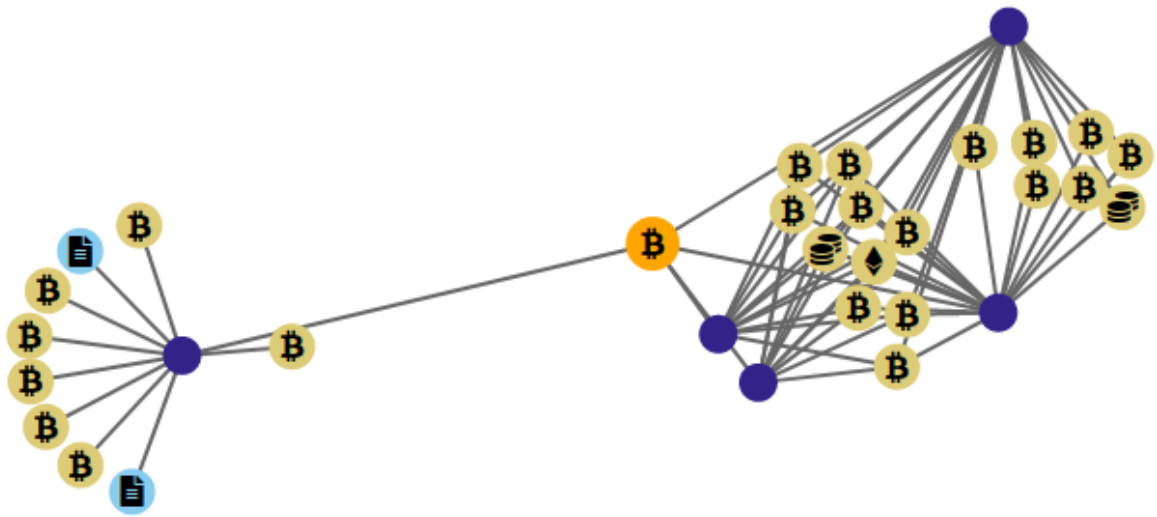
Object type	type	First seen	Last seen	Nb seen
cryptocurrency	bitcoin	2020/01/17	2020/02/20	5

Expand Bitcoin address

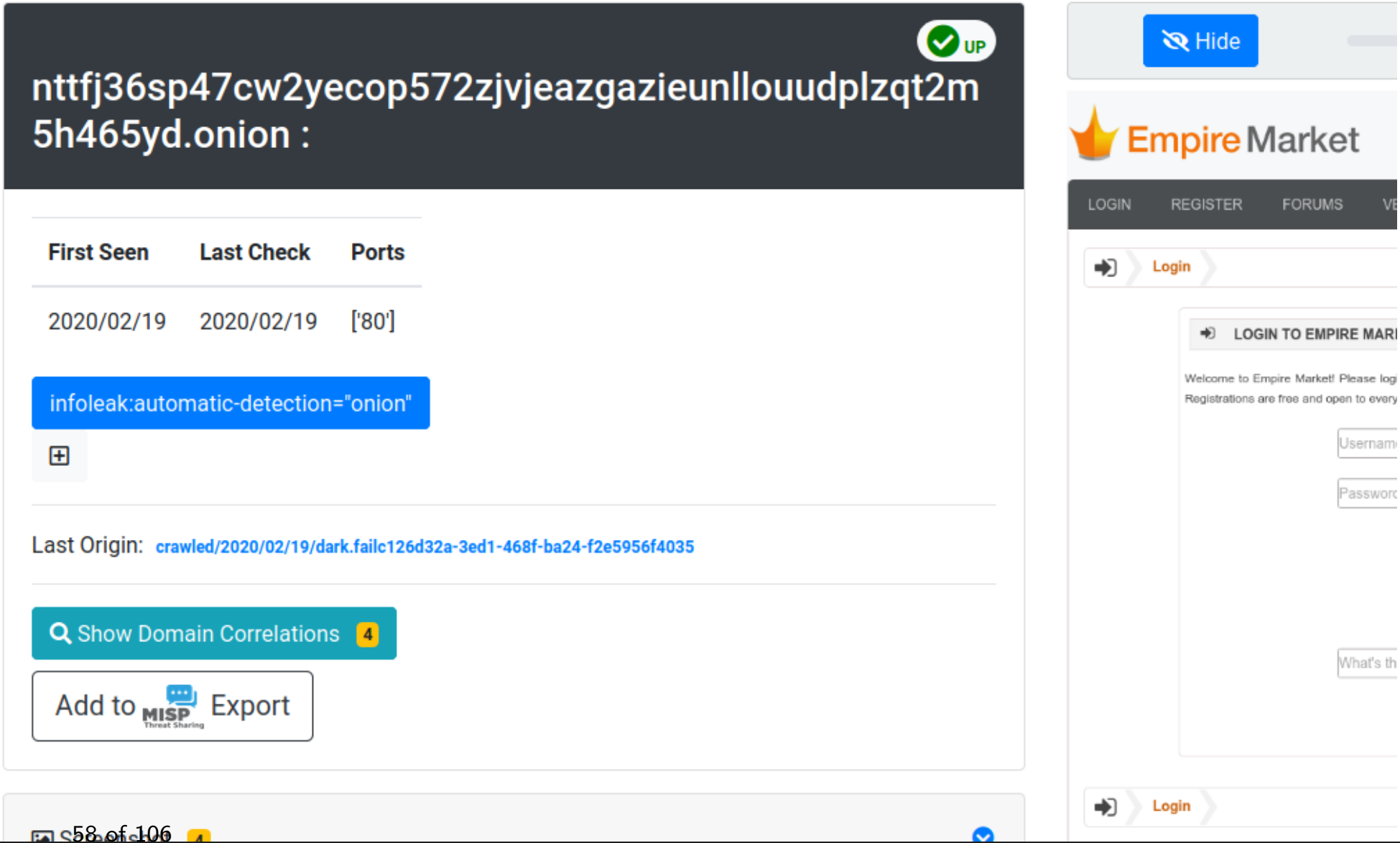
Graph

Resize Graph

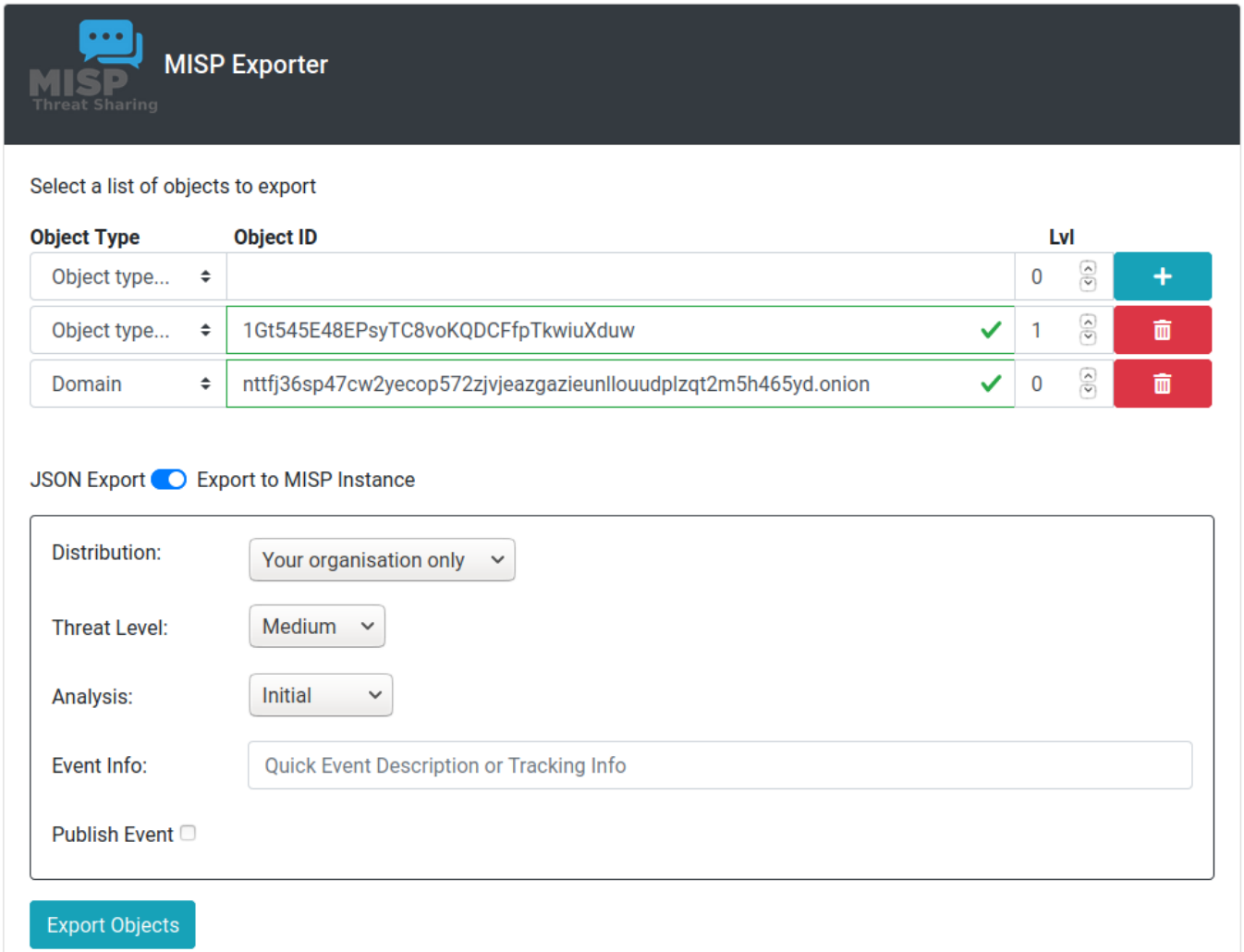
Add to MISP Export



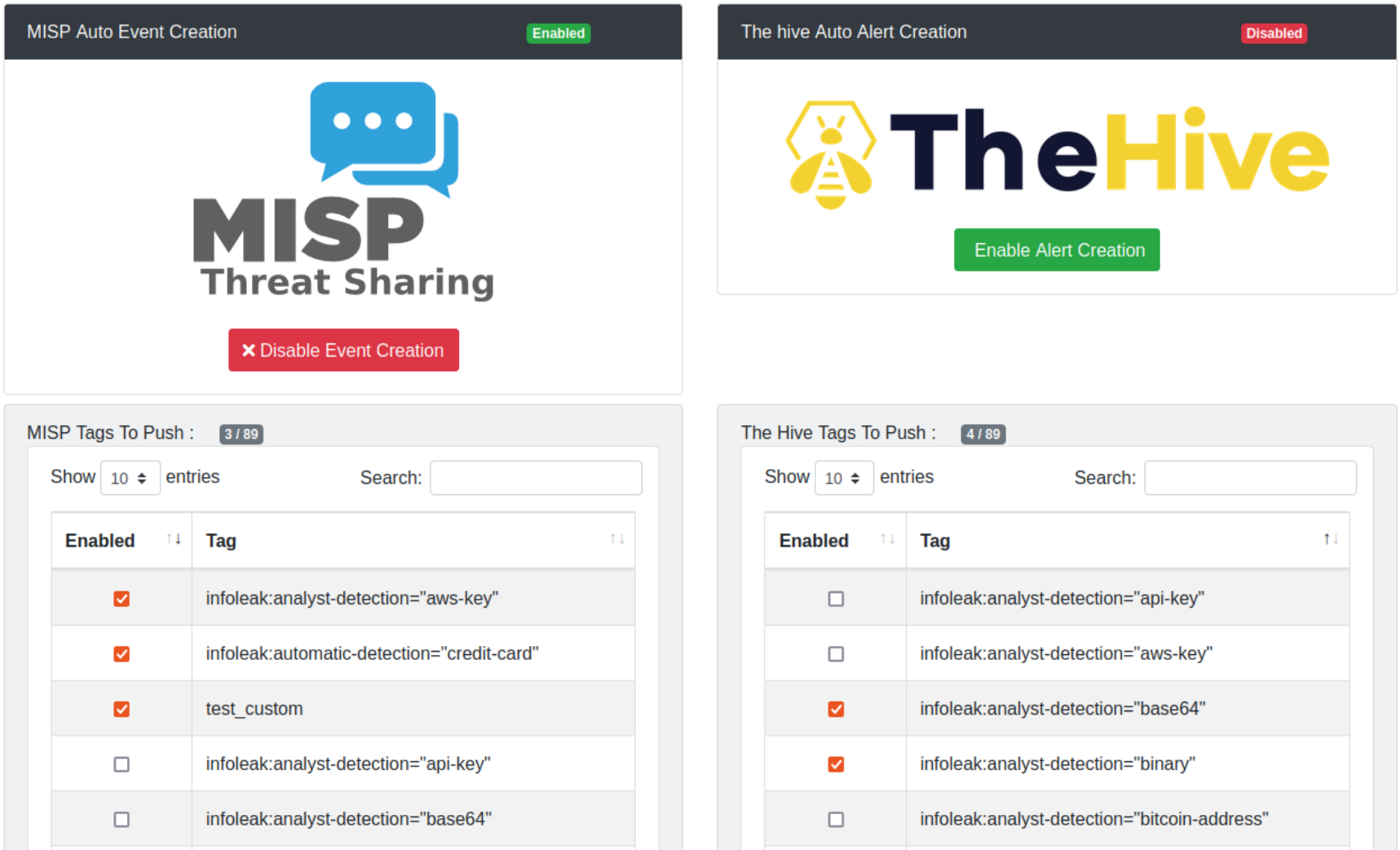
MISP Export


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 Export
Hide
Empire Market
LOGIN
REGISTER
FORUMS
Login
LOGIN TO EMPIRE MARI
Welcome to Empire Market! Please log
Registrations are free and open to every
Login

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 MISP Export on tags



API

AIL exposes a ReST API which can be used to interact with the back-end[11].

```
curl https://127.0.0.1:7000/api/v1/get/item/default
        --header "Authorization:
    iHc1_ChZxj1aXmiFiF1mkxxQkzawwriEaZpPqyTQj "
        -H "Content-Type: application/json"
        --data @input.json -X POST

```

[11] https://github.com/ail-project/ail-framework/blob/master/doc/README.md

Setting up the framework

Setting up AIL-Framework from source

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
```

Feeding the framework

Feeding Data to AIL

There are different ways to feed data into AIL:

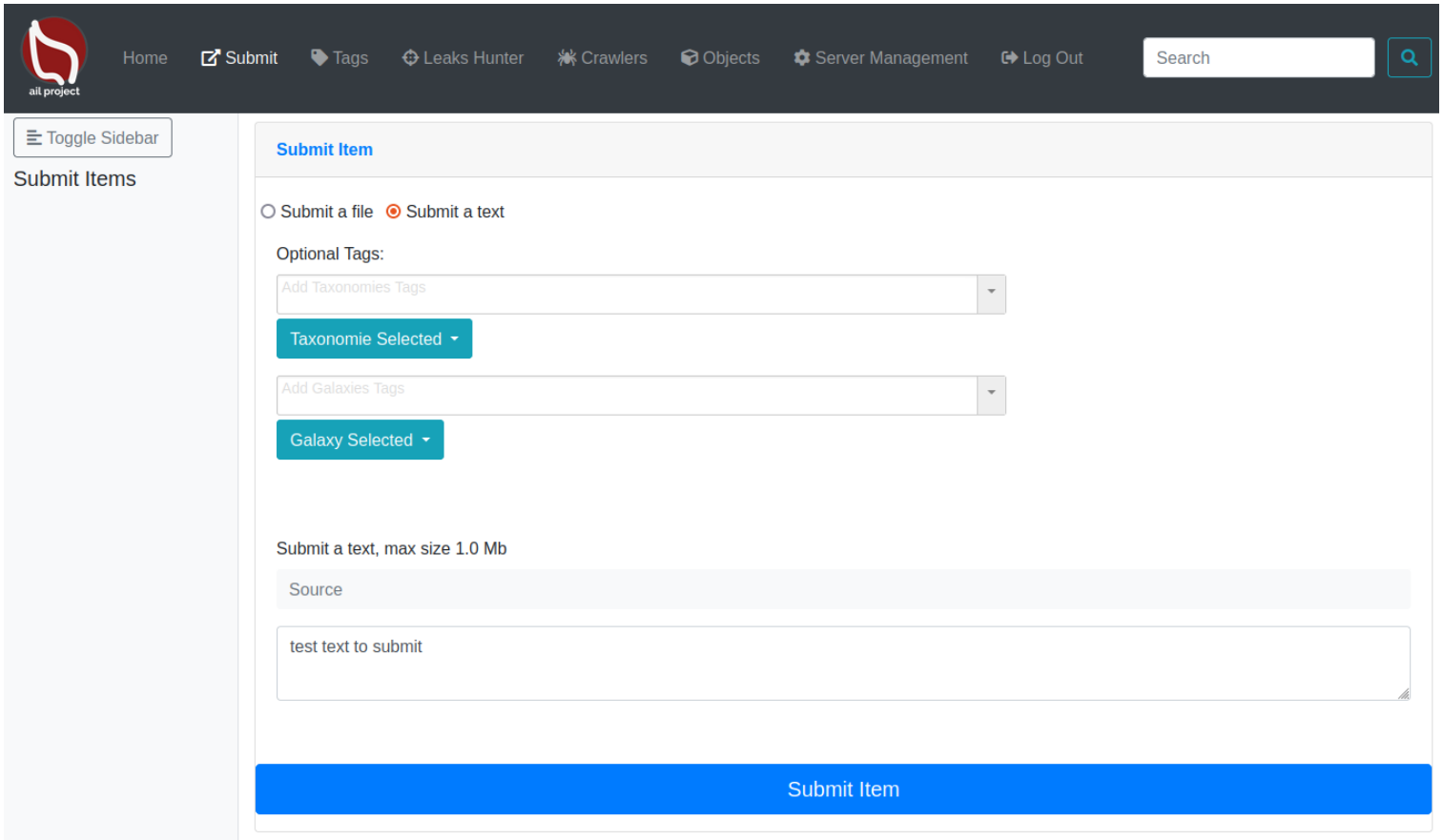
1. AIL Importers:
   - Dir / Files
   - ZMQ
   - *pystemon*
2. AIL Feeders (discord, telegram, ...)
3. Feed your own data using the API
4. Feed your own file/text using the UI (`Submit section`)

Feeding Data to AIL - Limitation

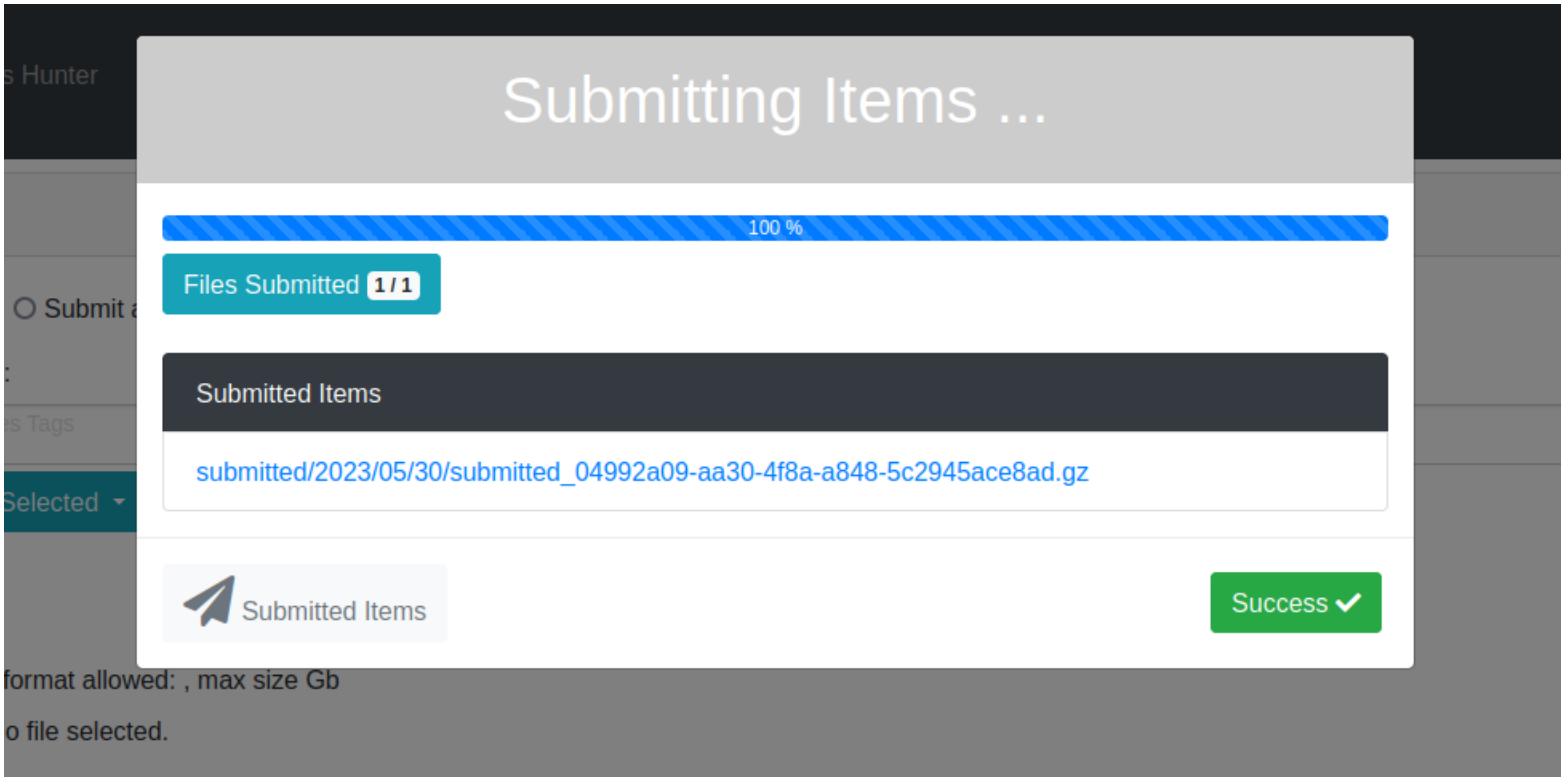
/!\ Limitation:

- 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 (default timeout of 30 seconds)
  - If you want to feed a large file, better split it in multiple ones

Via the UI (1)


Home
Submit
Tags
Leaks Hunter
Crawlers
Objects
Server Management
Log Out
Search
Toggle Sidebar
Submit Items
Submit Item
Submit a file
Submit a text
Optional Tags:
Add Taxonomies Tags
Taxonomie Selected
Add Galaxies Tags
Galaxy Selected
Submit a text, max size 1.0 Mb
Source
test text to submit
Submit Item

Via the UI (2)


Submitting Items ...
100 %
Files Submitted 1 / 1
Submitted Items
submitted/2023/05/30/submitted_04992a09-aa30-4f8a-a848-5c2945ace8ad.gz
Submitted Items
Success
format allowed: , max size Gb
o file selected.

API - Feeding AIL with your own data

api/v1/import/item

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

Importers

- Importers are located in the `/bin/importer` directory
- They are used to import different types of data into AIL
- Adding new Importers is straightforward.
- Available Importers:
  - AIL Feeders
  - ZMQ
  - pystemon
  - Files

File Importer

- `importer/FileImporter.py`

Import File

```
. ./AILENV/bin/activate
cd tools/
./file_dir_importer.py -f MY_FILE_PATH
```

Import Dir

```
. ./AILENV/bin/activate
cd tools/
./file_dir_importer.py -d MY_DIR_PATH
```

AIL feeders Importers

- **12+ feeders are available** for all AIL users to feed from external sources
- External feeders can run anywhere and are completely separated from AIL framework
- The feeder can use their **own internal logic** and even push JSON metadata
- Feeder are then pushing the generated JSON to AIL API

Certificate transparency feeder for AIL

- ail-feeder-cti[12] is a generic software to extract information from a certstream server (certificate transparency)
- All metadata extracted will be processed by AIL
- Onion addresses crawled automatically by AIL if seen in a certificate

[12] `https://github.com/ail-project/ail-feeder-ct`

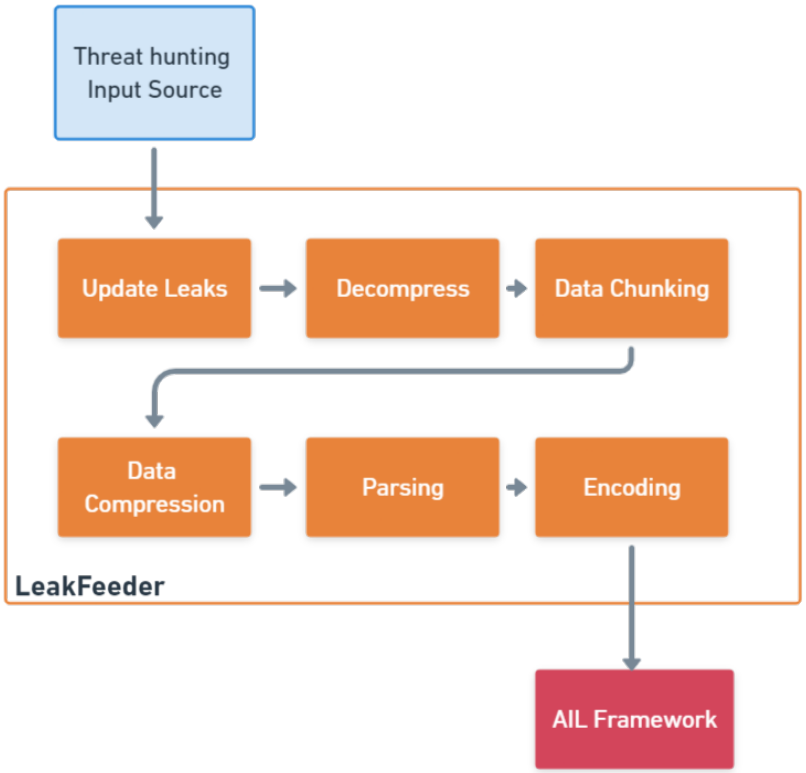
GitHub archive and GitHub repository

- ail-feeder-gharchive[13] is a generic software to extract informations from **GHArchive**, collect and feed AIL via AIL ReST API
- ail-feeder-github-repo[14] is collecting from a GitHub repository and push everything to AIL
- For monitoring a set of **suspicious git repositories** or finding leaks on existing or managed git repositories, it's a simple way to feed AIL with such source.

[13] `https://github.com/ail-project/ail-feeder-gharchive`
[14] `https://github.com/ail-project/ail-feeder-github-repo`

AIL LeakFeeder

- ail-feeder-leak[15] automates the process to feed leaked large files automatically to AIL



[15] https://github.com/ail-project/ail-feeder-leak

AIL feeder ActivityPub

- ail-feeder-activity-pub[16] is feeder for the ActivityPub standard used in distributed social networks
- Accounts are required on the ActivityPub instance to get the stream

[16] `https://github.com/ail-project/ail-feeder-activity-pub`

AIL feeder telegram

- ail-feeder-telegram[17] is a **Telegram feeder**
- An API ID/hash for Telegram is required and linked to your Telegram phone number

[17] `https://github.com/ail-project/ail-feeder-telegram`

More feeders

- ail-feeder-discord[18] is a generic **Discord** feeder for AIL
- ail-feeder-atom-rss[19] is an **Atom and RSS reader** and feeder for AIL
- ail-feeder-jsonlogs[20] is a **JSON aggregator** to submit generic JSON input into AIL

[18] `https://github.com/ail-project/ail-feeder-discord`
[19] `https://github.com/ail-project/ail-feeder-atom-rss`
[20] `https://github.com/ail-project/ail-feeder-jsonlogs`

Feeding AIL with custom JSON

conti leaks
@ContiLeaks

conti jabber leaks anonfiles.com/VeP6K6K5xc/1_t...

9:22 PM · 27 févr. 2022 · Twitter Web App

123 Retweets **23** Tweets cités **297** J'aime

```
{
  "ts": "2020-09-08T00:28:49.471678",
  "from": "ceram@q3mcco35auwcstmt.onion",
  "to": "stern@q3mcco35auwcstmt.onion",
  "body": "Проинструктируйте меня. Что делать?"
}
```

Feeding AIL with Conti leaks

- Conti jabber leaks are a good candidate for AIL analysis:
  - PGP keys
  - Bitcoin addresses, maybe others,
  - onion hidden services
- first we translated the files on english using deepl.com
- then we created a feeder to import json data in AIL
- Support added in AIL to correlate jabber usernames

Feeding AIL with Conti leaks

```
from pyail import PyAIL
#... imports
#...  setup code
for content in sys.stdin:
    elm = json.loads(content)
    tmp = elm['body']
    tmpmt = {}
    tmpmt['jabber:to'] = elm['to']
    tmpmt['jabber:from'] = elm['from']
    tmpmt['jabber:ts'] = elm['ts']
    tmpmt['jabber:id'] = "{}".format(uuid.uuid4())
    pyail.feed_json_item(tmp, tmpmt, ailfeedertype,
    source_uuid)
```

feeder.py

```
$ cat ~/conti/* | jq . -c | python ./feeder.py
```

Feeding AIL with Conti leaks

- use grep to limit the noise on an instance by only sending interesting bits:
  - PGP keys

```
$ cat ~/conti/* | jq . -c | grep PGP | python ./
    feeder.py
```

  - onion hidden services | `grep http://` |
  - telegram addresses | `grep tg://` |
  - bitcoins addresses | `egrep --regexp="[13][a-km-zA-HJ-NP-Z1-9]25,34" |`

Starting the framework

Running your own instance from source

Accessing the environment and starting AIL

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

Updating AIL

Launch the updater:

```
cd bin/
# git pull and launch all updates:
./LAUNCH -u


# PS:
# The Updater is launched by default each time
# you start the framework with
# ./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:
   test
   Password1234
```

AIL ecosystem - Challenges and design

AIL ecosystem: Technologies used

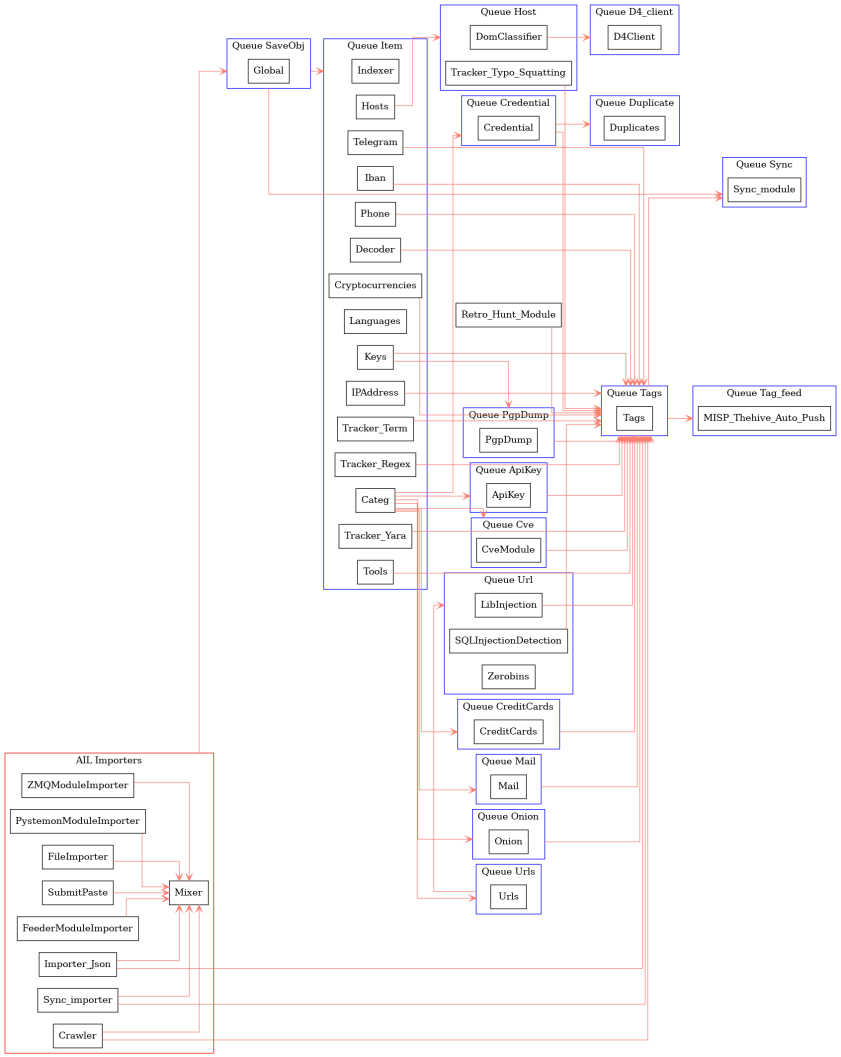
Programming language: Full python3

Databases: Redis and Kvrocks

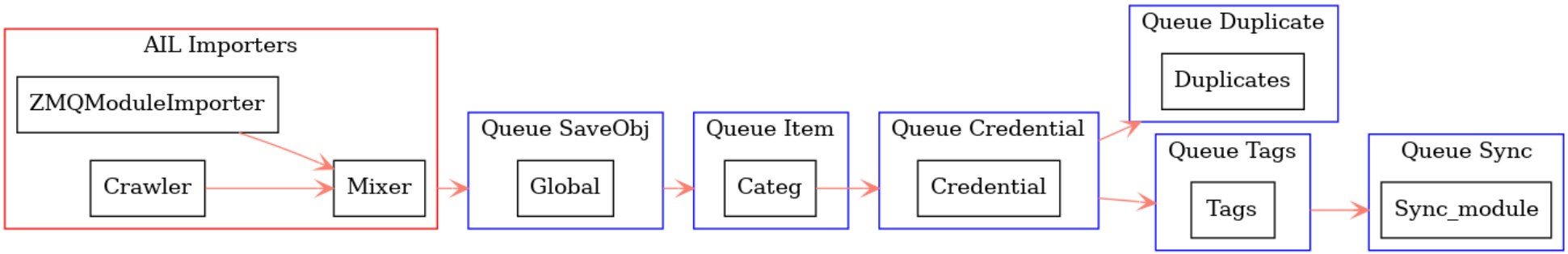
Server: Flask

Data message passing: Redis Set

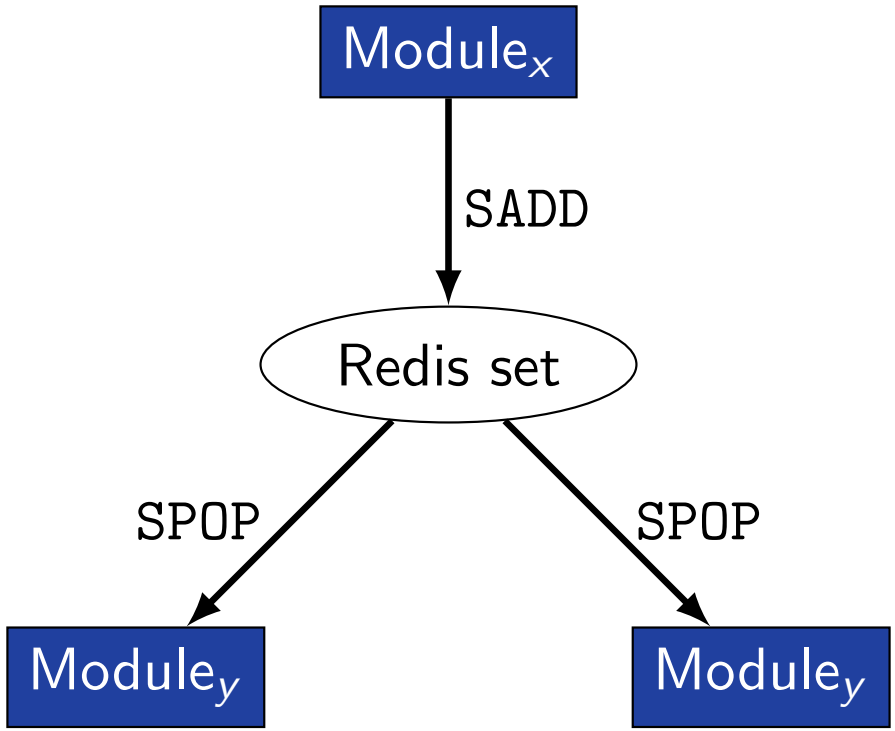
AIL global architecture: Data streaming between module



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



Message consuming



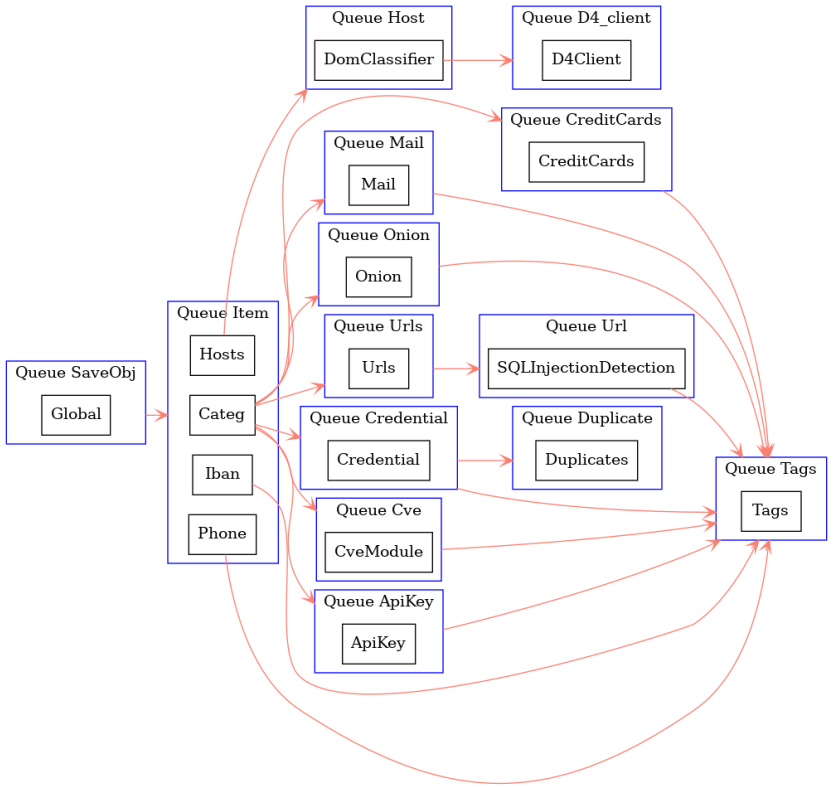
→ No message lost nor double processing

→ Multiprocessing!

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 `configs/modules.cfg` accordingly

Writing your own modules - /bin/modules/TemplateModule.py

```
1  from modules.abstract_module import AbstractModule
2
3  class NewModule(AbstractModule):
4
5    def __init__(self):
6          super().__init__()
7      self.logger.info(f'Module {self.module_name} initialized')
8
9    # Do something with the message from the queue
10   def compute(self, message):
11     # Process Message
12
13 # LAUNCH MODULE
14 if __name__ == '__main__':
15     module = NewModule()
16     module.run()
17
18
```

Writing your own Importer - /bin/importer/

```python
from importer.abstract_importer import AbstractImporter
from modules.abstract_module import AbstractModule

class MyNewImporter(AbstractImporter):

    def __init__(self):
        super().__init__()
        # super().__init__(queue=True)   # if it's an one-time run importer
        self.logger.info(f'Importer {self.name} initialized')

    def importer(self, my_var): # import function
        # Process my_var and get content to import
        content = GET_MY_CONTENT_TO_IMPORT
        # if content is not gzipped and/or not b64 encoded,
        # set gzipped and/or b64 to False
        message = self.create_message(item_id, content)
        return message
        # if it's an one-time run, otherwise create an AIL Module
        # self.add_message_to_queue(message)

class MyNewModuleImporter(AbstractModule):
    def __init__(self):
        super().__init__() # init module ...
        # init module ...
        self.importer = MyNewImporter()
```

Writing your own Importer - /bin/importer/

```
1
2       def get_message(self):
3           return self.importer.importer()
4
5       def compute(self, message):
6           self.add_message_to_queue(message)
7
8   if __name__ == '__main__':
9       module = MyNewModuleImporter()
10      module.run()
11
12      # if it's an one-time run:
13      # importer = MyImporter()
14      # importer.importer(my_var)
15
16
```

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.
- Feel free to make a pull request for your contribution
- That's it!

〈(^.^)ノ

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.

Implementation Steps in AIL project

- **Gradual changes** in AIL to add required functionalities to support the objectives.
- **Time-memory trade-off** can be challenging to ensure a functional framework.
- Evaluation and integration of new modules in AIL based on time-memory comparisons.
- Semantic aspects are challenging due to the diverse data sources, unstructured data and languages seen.

Ongoing developments

- MISP Importer
- Bloom filter filtering
- Data retention and lifetime management of objects
- MISP modules expansion
- Auto classification of content by set of terms (semantic analysis)
- Improved export stream to third parties software
- Improved indexing relying on Solr, Lucene or other components

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