CIRCL
Computer Incident
Response Center
Luxembourg

# AIL Framework for Analysis of Information Leaks

## Practical and Efficient Data-Mining of Suspicious Websites, Forums, Chats and Tor Hidden-Services

https://ail-project.org/

Aurelien Thirion
Alexandre Dulaunoy - info@circl.lu

July 16, 2025

CIRCL https://www.circl.lu

# Links

- AIL project website `https://ail-project.org`
- AIL project open source framework `https://github.com/ail-project`
- AIL framework `https://github.com/ail-project/ail-framework`
- Training materials `https://github.com/ail-project/ail-training`
- Mastodon `https://infosec.exchange/@ail_project`
- Online chat `https://gitter.im/ail-project/community`

# Legal and Ethics

# Privacy, AIL and GDPR (PII)

- 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.

# 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**[2].

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

# Introduction

# Concepts - Deep Web

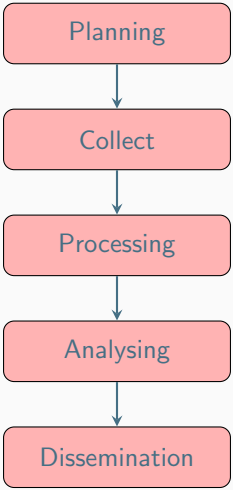
- **Deep Web** is the part of World Wide Web not indexed or directly accessible by standard web search-engines;
- This can be content hidden from **crawlers** by requiring a specific access and this can includes private social media, password-protected forums or content protected by different measures such as paywalls or specific security interface to access the information;
- A large portion of content accessible via Internet is part of the deep web[3].

[3] also called invisible web, hidden web or non-indexed web

# Concepts - darknet

- **Darknet** is an overlay network running on top of Internet requiring specific software to access the network and its services;
- Tor, I2P and Freenet are the most commonly used ones. Many are used for hidden services access and some for proxy access to the Internet;
- There are **legitimate use-cases** for such network but also many **illegal or criminal usage**.

# Lifecycle of collection and analysis



# 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

# Objectives of the session

- Show how to use and extend an open source tool to monitor web pages, pastes, chats, forums and hidden services
- Explain challenges and the design of the AIL open source framework
- Review different **collection mechanisms** and **sources**
- Learn how to create new modules
- Learn how to use, install and start AIL
- **Supporting investigation using the AIL framework** and including it in cyber threat intelligence life cycle

# 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 is now a complete project called **ail project**[4].
- In 2023, AIL framework version 5.0 released with new datastorage back-end.

[4] 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 an archive
- **Monitor** and crawl chats/websites

# Support CERT and Law Enforcement activities

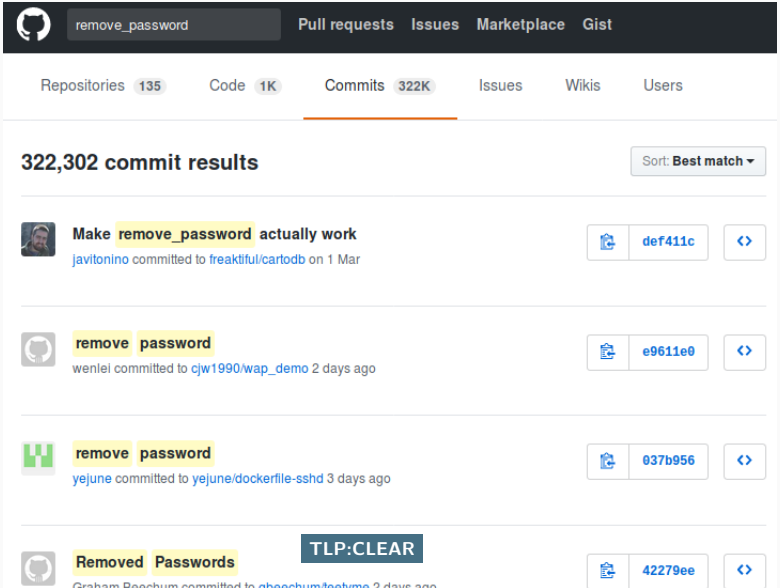
- Proactive investigation: leaks detection
  - List of emails and passwords
  - Leaked database
  - AWS Keys
  - Credit-cards
  - PGP private keys
  - Certificate private keys
- Feed Passive DNS or any passive collection system
- CVE and PoC of vulnerabilities most used by attackers

# Support CERT and Law Enforcement activities

- Website monitoring
  - monitor booters
  - Detect encoded exploits (WebShell, malware encoded in Base64, ...)
  - SQL injections
- Chat/Channel monitoring
  - Monitor Threat Actor Chat and Community Activities
- Automatic and manual submission to threat sharing and incident response platforms
  - MISP
  - *flowIntel (Coming soon)*
- Term/Regex/YARA monitoring for local companies/government
- YARA Retro Hunt

# Sources of leaks

# Mistakes from users:

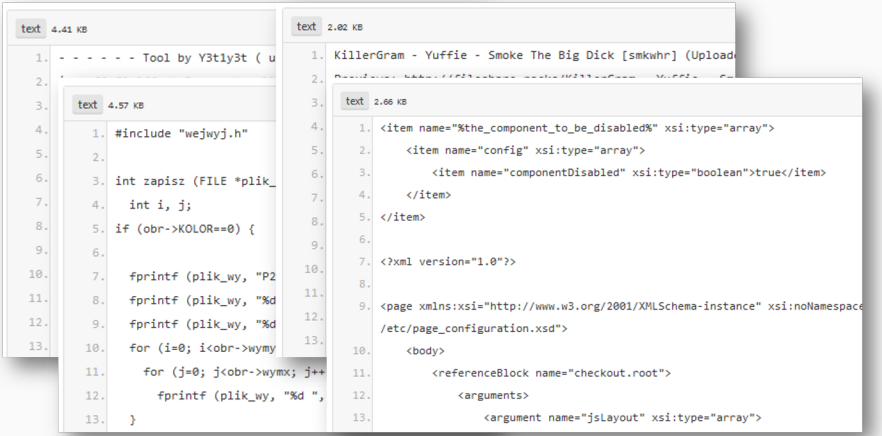


# Sources of Leaks: Paste Monitoring

- **Example:** `https://gist.github.com/`
  - Easily stores and shares text online
  - Used by programmers and legitimate users
    → Source code & configuration information

- **Abused by attackers to store:**
  - Lists of vulnerable/compromised sites
  - Software vulnerabilities (e.g., exploits)
  - Database dumps
    → User data
    → Credentials
    → Credit card details
  - Increasingly more sensitive information

# Examples of pastes (items)



# Why So Many Leaks?

- **Economic Interests:**
  - Adversaries promoting services
- **Ransom Model:**
  - To publicly pressure the victims
- **Political Motives:**
  - Adversaries showing off
- **Collaboration:**
  - Criminals need to collaborate
- **Operational Infrastructure:**
  - Malware exfiltrating information on a paste website
- **Mistakes and Errors from Users**

# What's your role while discovering such leak?

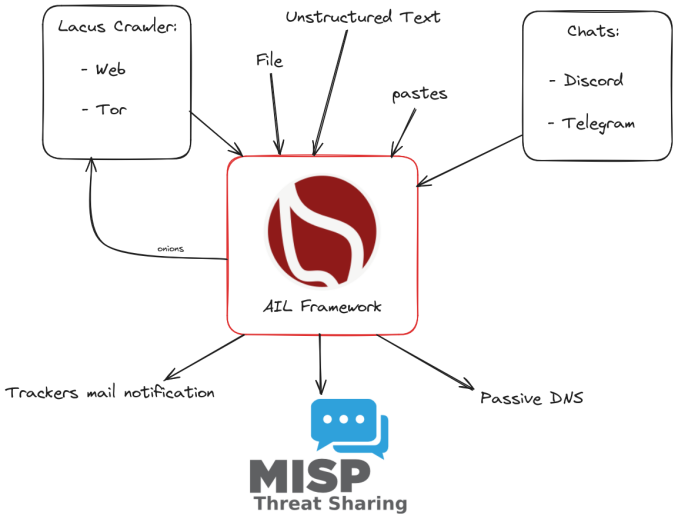
As a CSIRT, we must address this issue.

- **Contacting companies or organizations** responsible for accidental leaks
- **Engaging with the media** about specific leak cases to ensure practical and factual reporting
- **Evaluating the economic landscape** for cyber criminals (e.g., DDoS booters[5] or the resale of personal information - comparing reality versus media coverage)
- **Analyzing collateral effects** of malware, software vulnerabilities, or data exfiltration

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

# Current capabilities

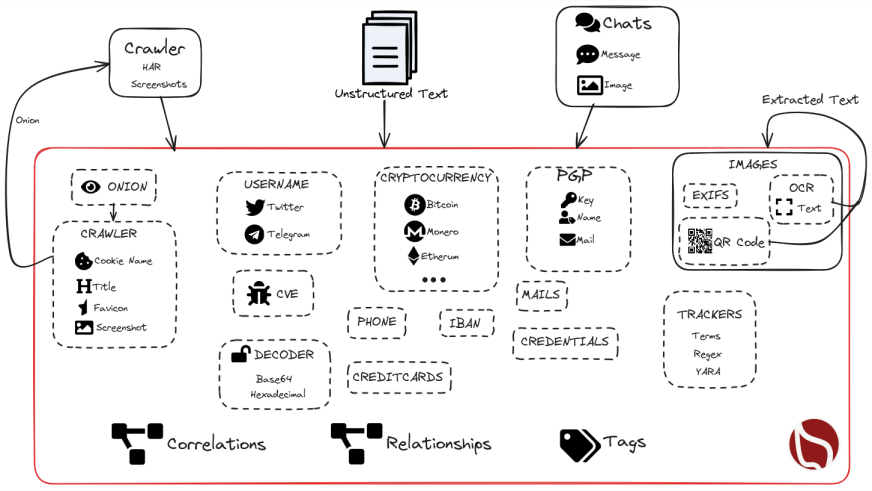
# High Level Overview



# AIL Framework: Extensible 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 inputs**
- Tor Crawler (handle cookies authentication)
- Feeders: Discord, Telegram, ...

# Analysis of unstructured information



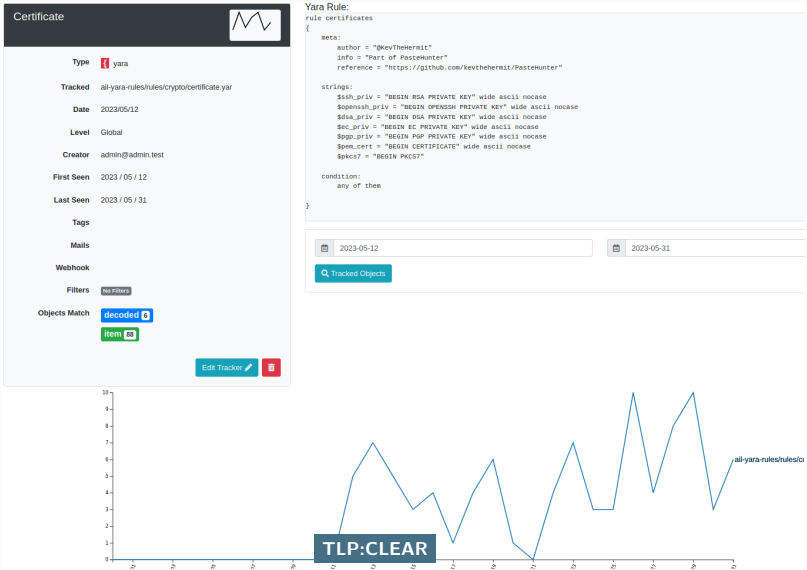
# AIL Framework: Features

- Extracting **credit card numbers, credentials, phone numbers, ...**
- Extracting and validating potential **hostnames**
- Keeps track of **duplicates**
- Submission to threat sharing and incident response platforms (**MISP**)
- **Full-text indexer** to index unstructured information
- **Tagging** for classification and searches
- Terms, sets, regex and YARA **tracking, occurrences, and history**
- Archives, files, and raw **submission** from the UI
- Correlation engine based on PGP ID, cryptocurrencies, decoded (Base64, ...), usernames, cookie names, and many selectors to find relationships
- And many more

# Trackers - Retro Hunt

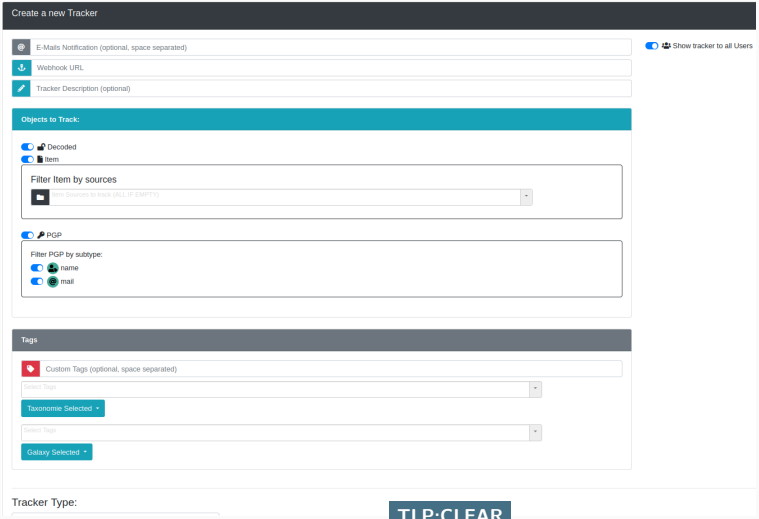
- Search and monitor specific keywords/patterns
  - Automatic Tagging
  - Email Notifications
- Track Word
  - ddos
- Track Set
  - booter,ddos,stresser;2
- Track Regex
  - circl\.lu
- YARA rules
  - https://github.com/ail-project/ail-yara-rules

# YARA Tracker

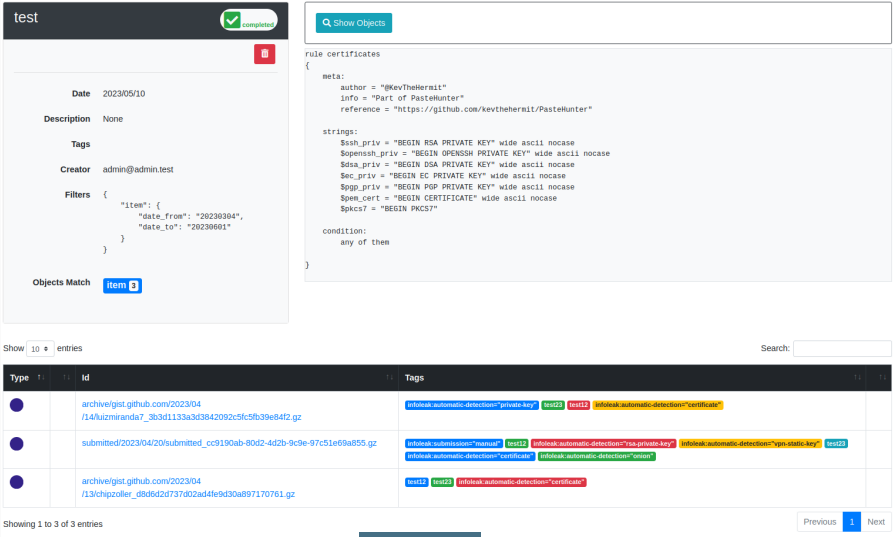


# Trackers - Practical part

- **Create and test** your own tracker



# Retro Hunt




TLP:CLEAR

# 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

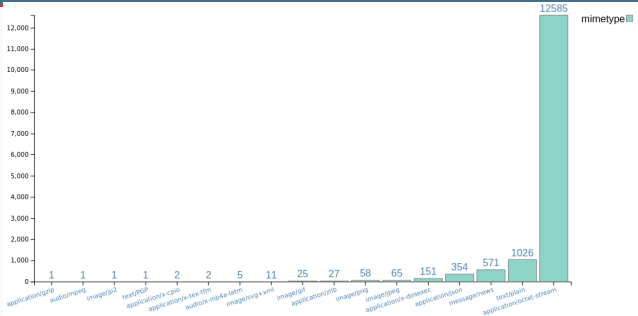
```
################################################################################
================================================================================
Hostname        www.pabloquintanilla.cl                 ISP     Wix.com Ltd.
Continent       North America           Flag
US
Country         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
Creation date: 2018-11-21 14:34:34 CLST
```

TLP:CLEAR

# Decoder

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

# Decoder:



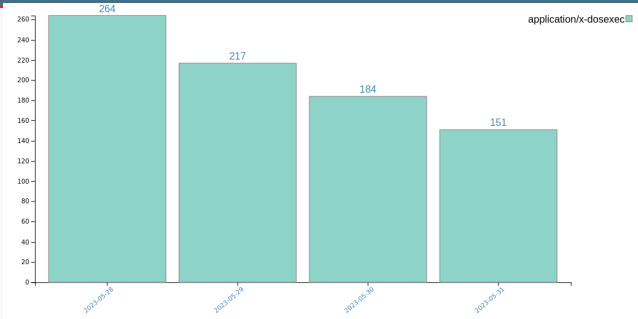

20230531 Decoded files:

Show 10 entries

Search:

| 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 | f9f918ba5b0a35f5e2523bd0f4ece68d2cda11f9 | 20230410 | 20230531 | 3947 | 44 | Virus Total submission is disabled | |

# Decoder:



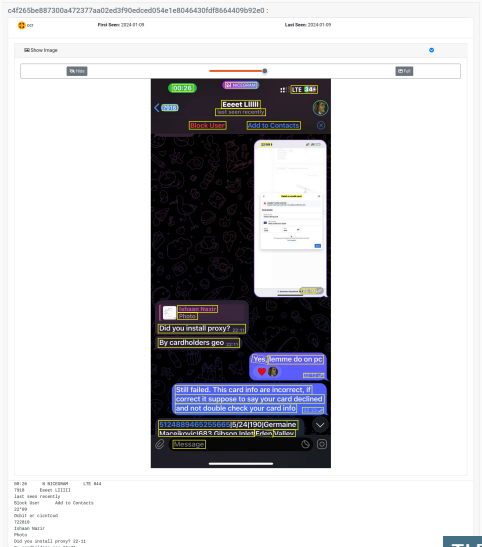

20230528 to 20230531 Decoded files:

Show 10 entries Search:

| estimated type | hash | first seen | last seen | nb item | size | Virus Total | Sparkline |
|---|---|---|---|---|---|---|---|
| application/x-dosexec | c408501f7f2bd8279704c380da61d329c6119fc2 | 20230421 | 20230529 | 76 | 64 | Virus Total submission is disabled | |
| application/x-dosexec | a9ecdb74ce7d22b70f0dcf0f57299f31ce570161 | 20230405 | 20230530 | 56 | 55666 | Virus Total submission is disabled | |
| application/x-dosexec | e5805aa6a66c6e013d5ebddbab4fbc45b4c84127 | 20230529 | 20230531 | 4 | 32 | Virus Total submission is disabled | |

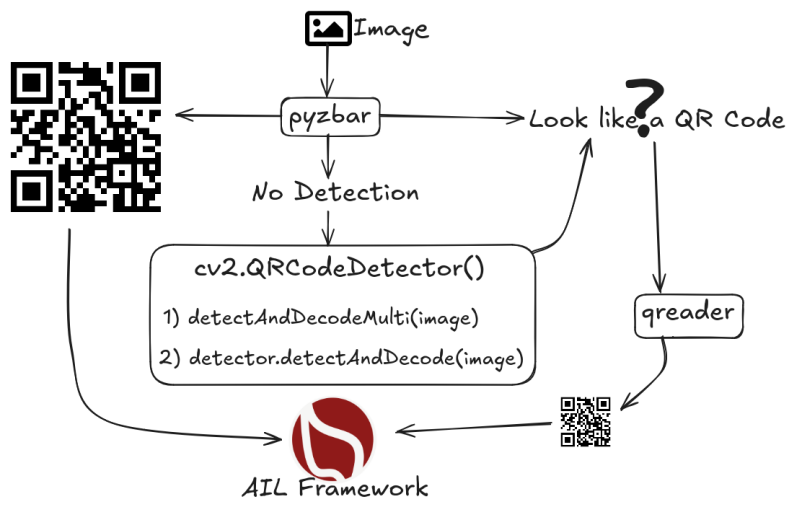
TLP:CLEAR

# OCR: Optical Character Recognition

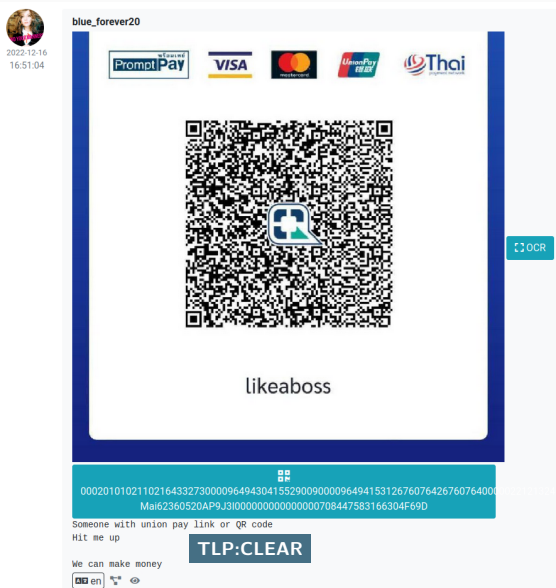



- Threat actors are often verbose and frequently **share extensive details in private channels**.
- Many messages contain screenshots and images.
- Text detection and extraction are performed across 80+ languages using a CRNN (Convolutional Recurrent Neural Network).
- **Enables keyword-based matching and detection**.

# QR Code Extractor



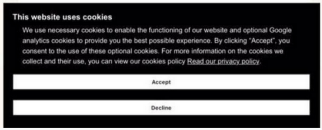
# QR Codes




TLP:CLEAR

# Bar Codes

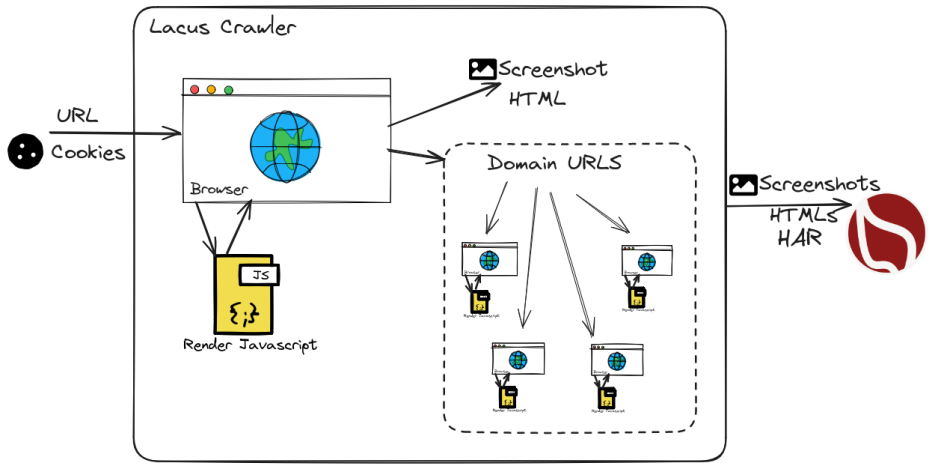

OCR 8745860537300061927671300649 13




Cool man its done as always , I appreciate for all you do beloved❤️ .
You have brought a lot of ppl to riches and I will forever be loyal, God bless you ..I wish I knew this earlier...🏦

en

# Collection - Lacus Crawler



# Crawler

- Crawlers are used to navigate on regular website as well as .onion addresses (via automatic extraction of urls or manual submission)
- Lacus[6] ("scriptable" browser) is rending the pages (including javascript) and produce screenshots (HAR archive too)
- 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)

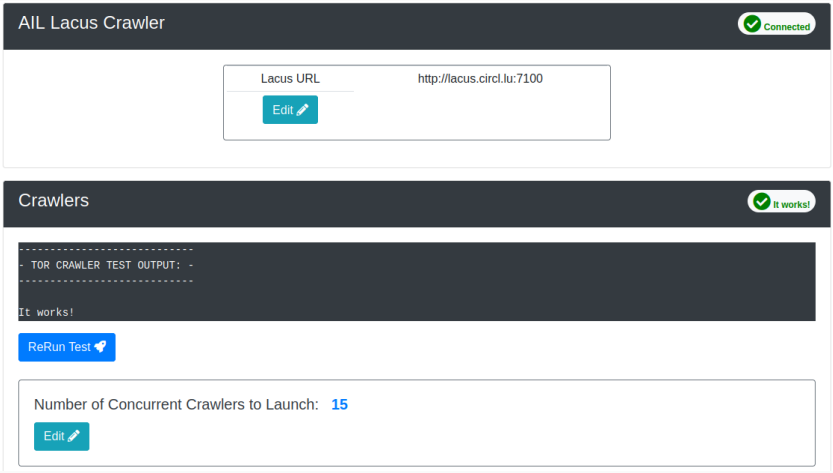
[6] https://github.com/ail-project/lacus

# Lacus

- Lacus[7] 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,

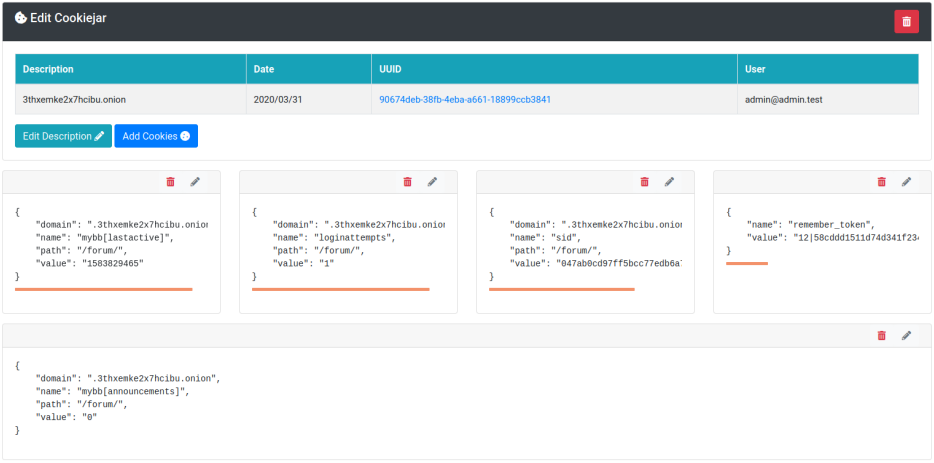
[7] https://github.com/ail-project/lacus

# Crawler Settings - Lacus

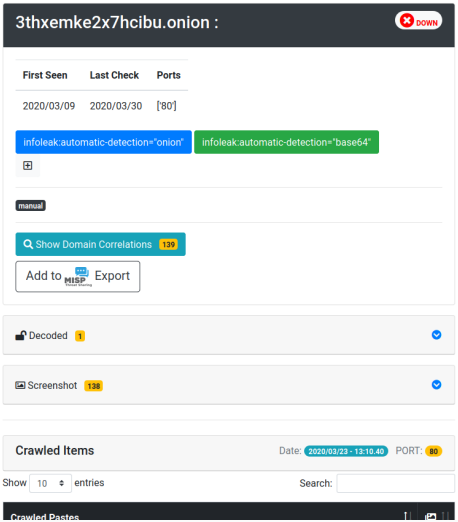


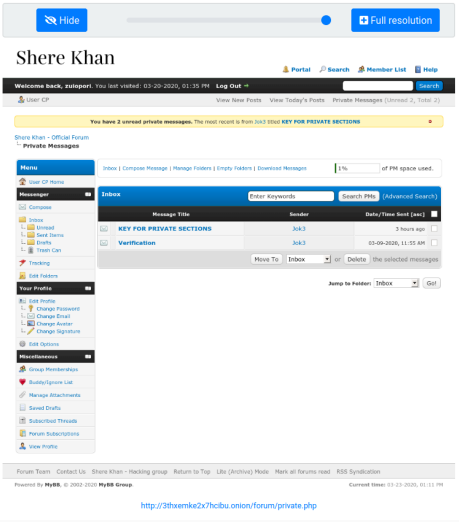
# Crawler: Cookiejar

Use your cookies to login and bypass captcha



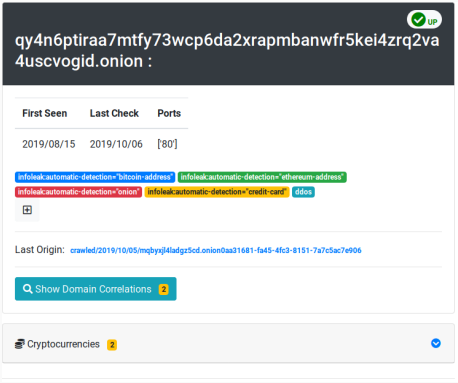
# Crawler: Cookiejar

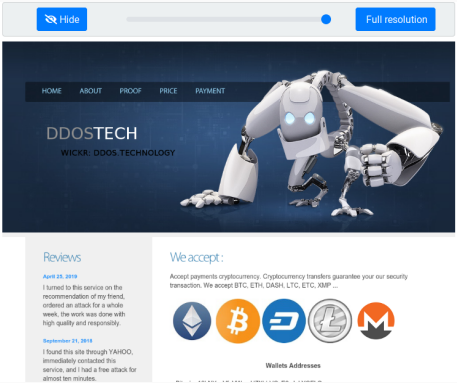


# Live demo!

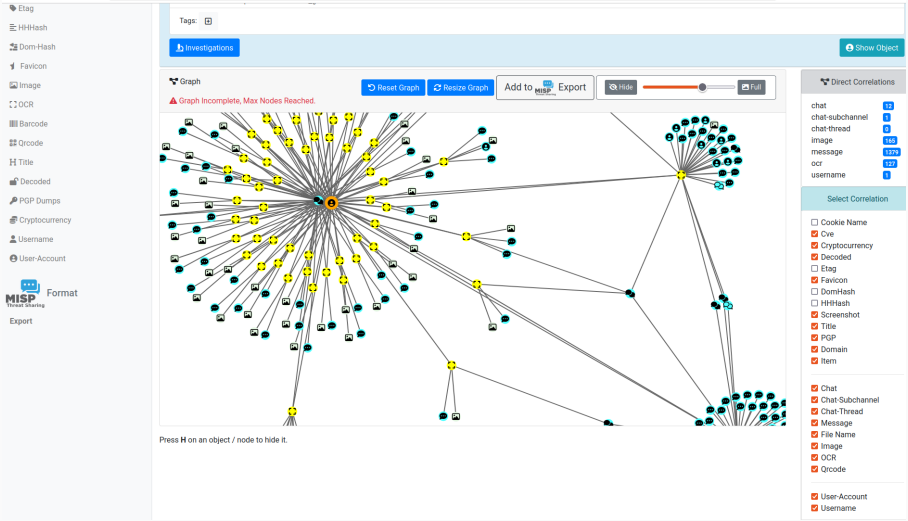
# Crawler: DDoS Booter



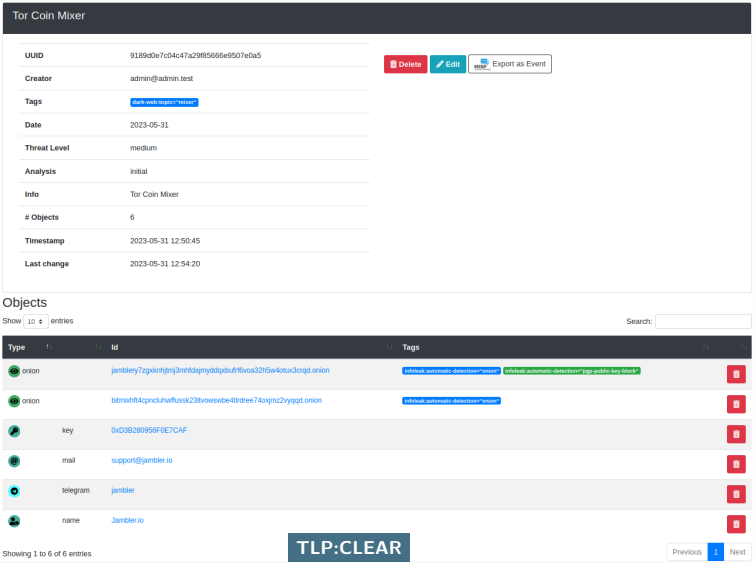




TLP:CLEAR

# Correlations and relationship

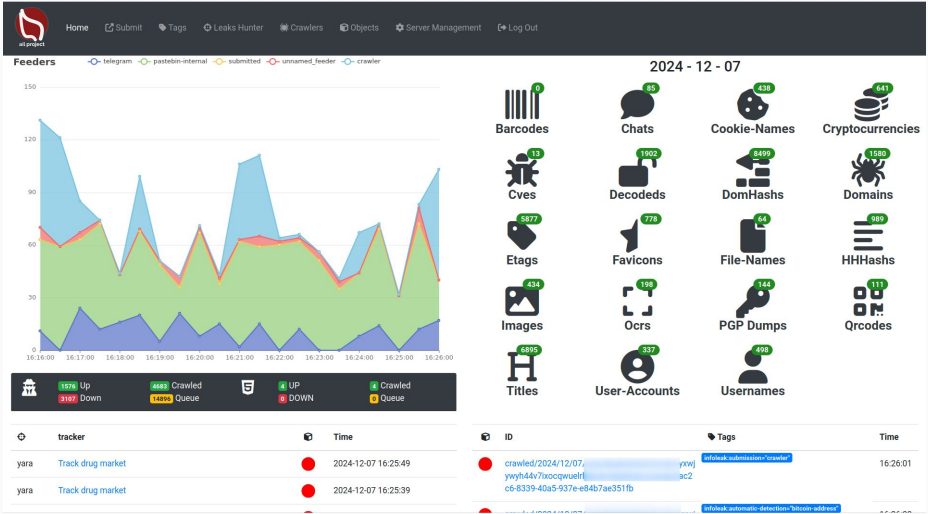


# Investigations




TLP:CLEAR

# Dashboard


Home
Submit
Tags
Leaks Hunter
Crawlers
Objects
Server Management
Log Out
Feeders
telegram
pastebin-internal
submitted
unnamed_feeder
crawler
2024 - 12 - 07
Barcodes
Chats
Cookie-Names
Cryptocurrencies
Cves
Decodeds
DomHashs
Domains
Etags
Favicons
File-Names
HHHashs
Images
Ocrs
PGP Dumps
Qrcodes
Titles
User-Accounts
Usernames
1576 Up
3107 Down
4683 Crawled
1489 Queue
4 UP
0 DOWN
4 Crawled
0 Queue
tracker
Time
yara
Track drug market
2024-12-07 16:25:49
yara
Track drug market
2024-12-07 16:25:39
ID
Tags
Time
infoleak:submission="crawler"
16:26:01
infoleak:automatic-detection="bitcoin-address"

# Example: Items Extracted

| Date | Source | Size (Kb) | Number of lines | Max line length |
|---|---|---|---|---|
| 2025/02/01 | telegram | 180.34 | 6033 | 27975 |

infoleak:automatic-detection="base64" infoleak:automatic-detection="phone-number" infoleak:automatic-detection="credit-card" infoleak:automatic-detection="credential" infoleak:automatic-detection="iban"

Add to MISP Export | Investigations | Correlations Graph

Extracted 3

Show 10 entries Search:

| Type | ID | Extracted |
|---|---|---|
| tag | infoleak:automatic-detection="phone-number" | +97695152402<br>+972535242150<br>+919247146104<br>+212 608-950286<br>+212 629-117431<br>+212630864857<br>+84708794730<br>+96894427669<br>+44 7934 122985<br>+971501997173<br>+971-501027162<br>+971569883905<br>+971-585856200 |
| tag | infoleak:automatic-detection="iban" | AE980260001015865178901<br>AE980260001015865178901 |
| tag | infoleak:automatic-detection="credit-card" | 373778604875195<br>4439137249559009 |

Showing 1 to 3 of 3 entries Previous 1 Next

TLP:CLEAR

# Example: Items Domain

Crawler

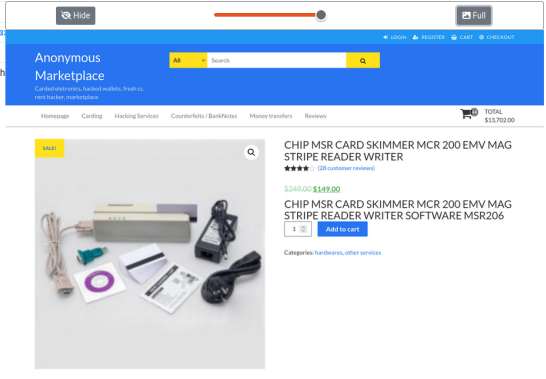
Last Origin:

crawled/2025/02/03/drugsxeuauw4xt6eoqwztfkpa2ul5453q4ao4kqrtedhfoy6jmui5ryd.onionae52293f-ffda-4bd9-b81

drugsxeuauw4xt6eoqwztfkpa2ul5453q4ao4kqrtedhfoy6jmui5ryd.onion

url http://drugsxeuauw4xt6eoqwztfkpa2ul5453q4ao4kqrtedhfoy6jmui5ryd.onion/?product=ch

skimmer




TLP:CLEAR

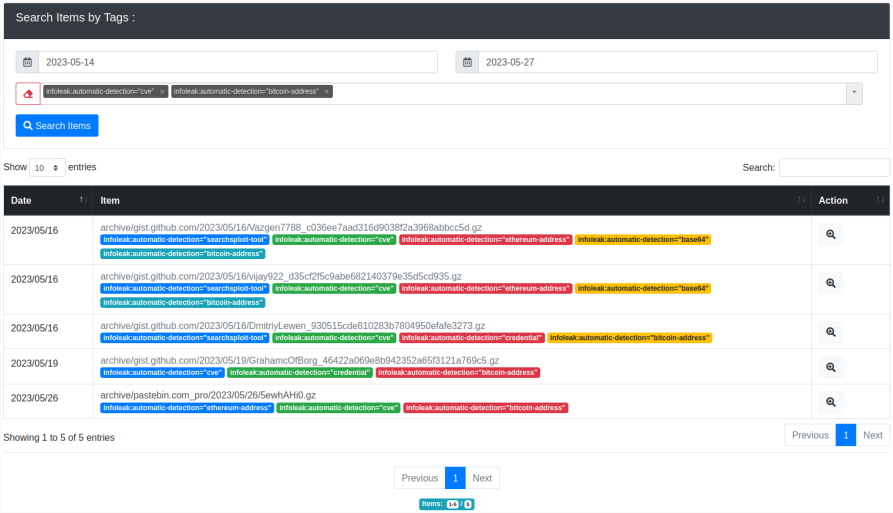
# Example: Browsing content

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/
```

TLP:CLEAR

# 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 [8].
- Can be easily cherry-picked or extended

[8] 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**[9]: Evaluate the degree of identifiability of personal data and the types of pseudonymous data, de-identified data and anonymous data.

[9] Future of Privacy Forum

# Taxonomies useful in AIL

- **tor**: Describe Tor network infrastructure.
- **dark-web**: Criminal motivation on the dark web.
- **copine-scale**[10]: Categorise the severity of images of child sex abuse.

[10] Combating Paedophile Information Networks in Europe



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



1. Use infoleak taxonomy[11]
2. Add your own tags
3. Export AIL objects to MISP core format
4. Download it or Create a MISP Event[12]

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

# MISP Export

## 1Gt545E48EPsyTC8voKQDCFfpTkwiuXduw :

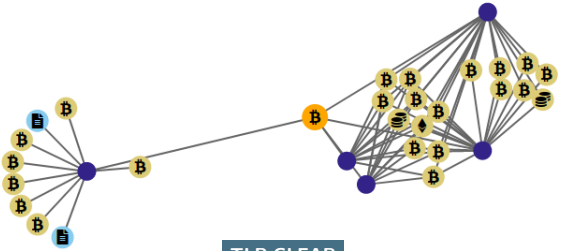
| Object type | type | First seen | Last seen | Nb seen |
|---|---|---|---|---|
| cryptocurrency | bitcoin | 2020/01/17 | 2020/02/20 | 5 |

Expand Bitcoin address

Graph

Resize Graph

Add to MISP Export



TLP:CLEAR

# MISP Export

UP

nttfj36sp47cw2yecop572zjvjeazgazieunllouudplzqt2m5h465yd.onion :

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

infoleak:automatic-detection="onion"

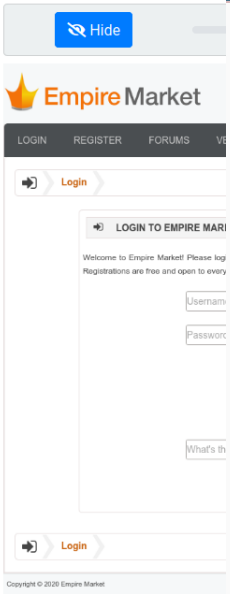
Last Origin: crawled/2020/02/19/dark.failc126d32a-3ed1-468f-ba24-f2e5956f4035

Show Domain Correlations 4

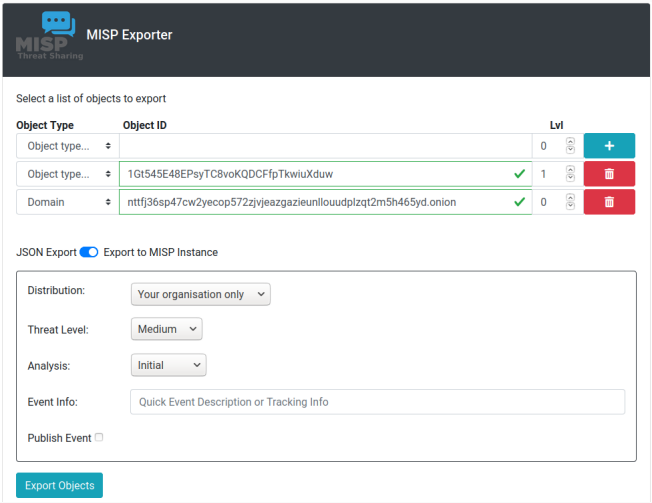
Add to MISP Export

Screenshot 4

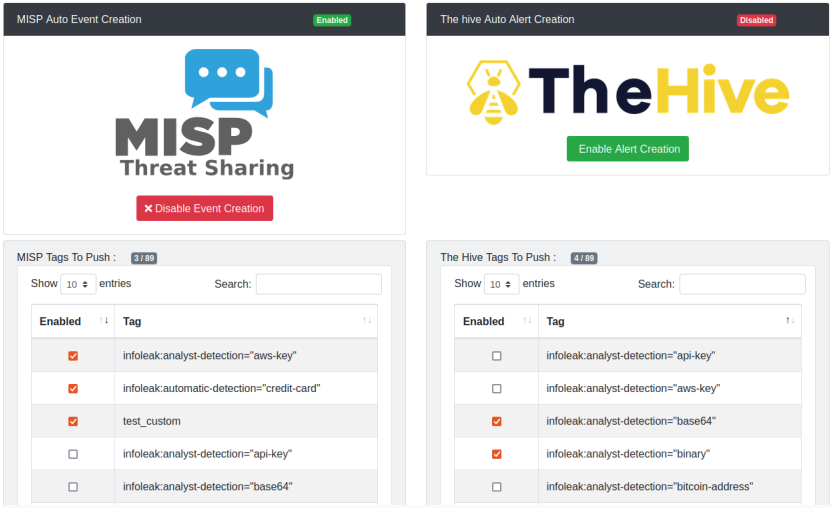
TLP:CLEAR



# MISP Export


MISP Exporter
Select a list of objects to export
Object Type
Object ID
Lvl
Object type...
0
Object type...
1GtS45E48EPsyTC8voKQDCFfpTkwiuXduw
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

# ReST API Access

AIL exposes a ReST API which can be used to interact with the back-end[13].

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

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

# Setting up the framework

# Setting up AIL-Framework from Source

**Installation Steps**

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

# Feeding the framework

# Feeding Data to AIL

There are different ways to feed data into AIL:

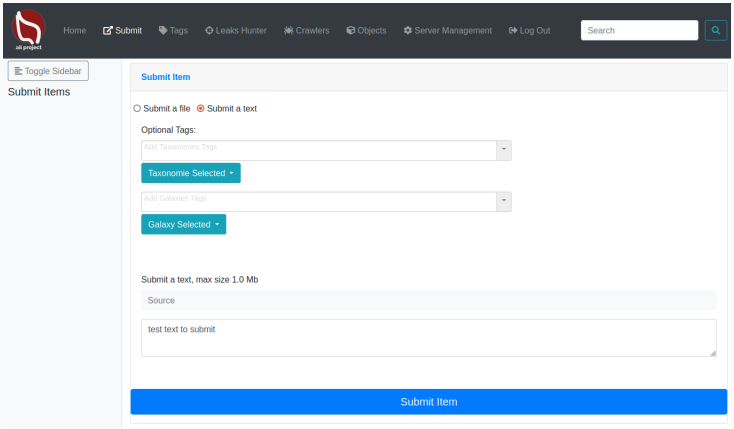
1. AIL Importers:
   - Dir / Files
   - ZMQ
   - *pystemon*
2. AIL Feeders (discord, telegram, ...)
3. Feed your own data using the API
4. Feed your own file/text using the UI (`Submit section`)

# Feeding Data to AIL - Limitation

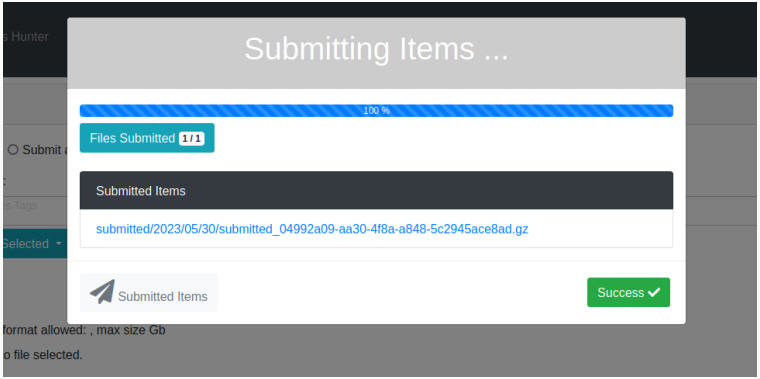
/!\ Limitation:

- 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

**Endpoint:** `api/v1/import/item`

**Example JSON Payload**

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

# AIL Feeders and Importers

- **More than 12 feeders** are available for AIL users to import data from external sources.
- External feeders can run anywhere and are completely independent of the AIL framework.
- Feeders can use their **own internal logic** and even generate custom JSON metadata.
- The resulting JSON is then pushed to the AIL API.

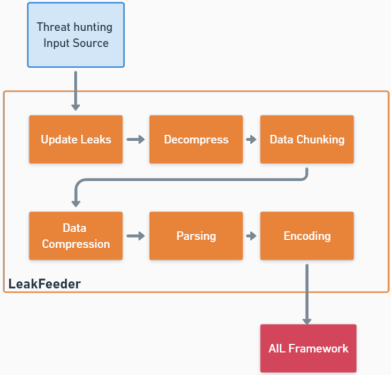
# GitHub Archive and GitHub Repository Feeders

- `ail-feeder-gharchive`[14] is a generic tool to extract information from **GHArchive**, collect data, and feed AIL via the AIL ReST API.
- `ail-feeder-github-repo`[15] collects data from GitHub repositories and pushes it to AIL.
- This is a simple way to feed AIL with content from a set of **suspicious Git repositories** or to find leaks in existing or monitored repositories.

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

# AIL LeakFeeder

- `ail-feeder-leak`[16] automates the process of feeding large leaked files directly into AIL.




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

# AIL Feeder – Telegram

- `ail-feeder-telegram`[17] is a **Telegram data feeder** for AIL.
- It requires a Telegram API ID and hash, which are linked to your Telegram phone number.

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

# Importers

- Importers are located in the `/bin/importer` directory.
- They are used to import various types of data into AIL.
- Adding new importers is straightforward and modular.
- Available importers include:
  - AIL Feeders
  - ZMQ
  - pystemon
  - File-based importers

# File Importer

- importer/FileImporter.py

## Import File:

**Import a Single File**

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

## Import Directory:

**Import a Directory**

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

# Feeding AIL with custom JSON


```
{
  "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 – Python Feeder

**feeder.py**

```
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'] = str(uuid.uuid4())
    pyail.feed_json_item(tmp, tmpmt, ailfeedertype, source_uuid)
```

# Feeding AIL with Conti Leaks – Running the Feeder

**Run the feeder with Conti leaks**

```
$ 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:

**Filter for PGP keys**

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

**Filter for .onion URLs**

```
$ cat ~/conti/* | jq . -c | grep "http://"
```

**Filter for Telegram links**

```
$ cat ~/conti/* | jq . -c | grep "tg://"
```

**Filter for Bitcoin addresses**

```
$ cat ~/conti/* | jq . -c | 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:

**Start AIL from source:**

```
# Launch the system and the web interface
cd bin/
./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
```

# AIL ecosystem - Challenges and design

# AIL ecosystem: Technologies used

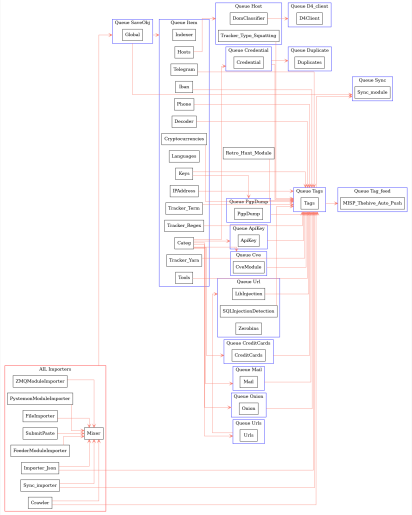
**Programming language:** Full python3

**Databases:** Redis and Kvrocks

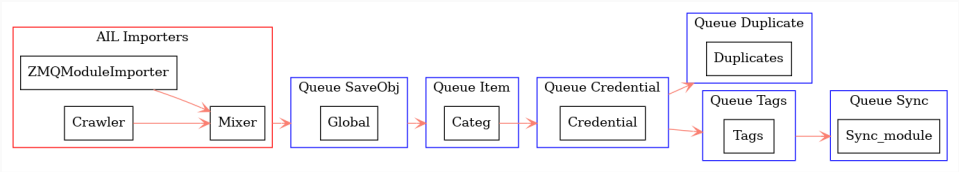
**Server:** Flask

**Data message passing:** Redis Set

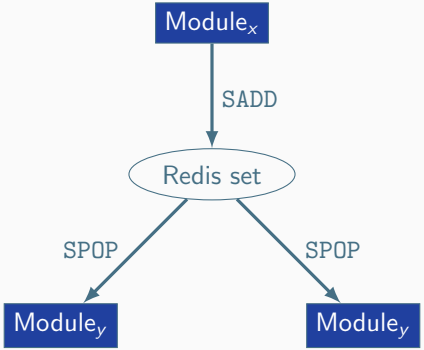
# AIL global architecture: Data streaming between module



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



# Message consuming



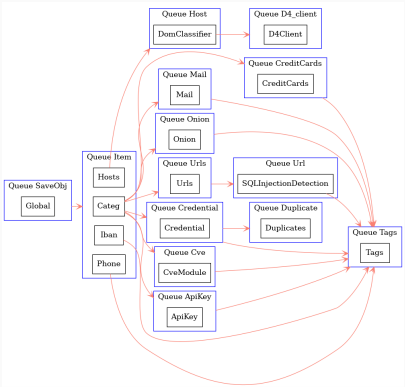

→ No message lost nor double processing

→ 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

**Python Module Example:**

```
from modules.abstract_module import AbstractModule

class NewModule(AbstractModule):

    def __init__(self):
        super().__init__()
        self.logger.info(f'Module {self.module_name} initialized')

    # Do something with the message from the queue
    def compute(self, message):
        # Process Message

# LAUNCH MODULE
if __name__ == '__main__':
    module = NewModule()
```

TLP:CLEAR

# Writing your own Importer - /bin/importer/

**Importer Example:**

```
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 a 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 g[TLP:CLEAR]nd/or not b64 encoded,
        # set gzipped and/or b64 to False
```

# Writing your own Importer - /bin/importer/

**Module Runner Example:**

```
    def get_message(self):
        return self.importer.importer()

    def compute(self, message):
        self.add_message_to_queue(message)

if __name__ == '__main__':
    module = MyNewModuleImporter()
    module.run()

    # if it's a one-time run:
    # importer = MyImporter()
    # importer.importer(my_var)
```

# 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**.

  → 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

- Text Geolocation
- Improved MISP Export
- Bloom filter filtering - PSS
- Analysis of relationships and activity between chats, including message forwards, replies, timelines, etc.
- Improved indexing relying on Solr, Lucene or other components
- Integration with FlowIntel

# Annexes

# Managing AIL: Old fashion way

**Access the script screen**

```
screen -r Script
```

**Table 1:** 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 |