AIL Framework for Analysis of Information Leaks
Practical and Efficient Data-Mining of Suspicious Websites, Forums and Tor Hidden-Services

Alexandre Dulaunoy
alexandre.dulaunoy@circl.lu
Aurelien Thirion
aurelien.thirion@circl.lu
Jean-Louis Huynen
jean-louis.huynen@circl.lu
info@circl.lu

April 1, 2021
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
• 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.

• In 2020, AIL framework is now a complete project called ail project\(^2\).

\(^2\)https://github.com/ail-project/
AIL Framework: A framework for Analysis of Information Leaks

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

ail project

Other leaks
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/YARA monitoring for local companies/government
Sources of leaks
Mistakes from users:

322,302 commit results

Make `remove_password` actually work
javitonino committed to freakiful/cartodb on 1 Mar

`remove password`
wenlei committed to cjl990/wap_demo 2 days ago

`remove password`
yejune committed to yejune/dockerfile-sshd 3 days ago
Sources of leaks: Paste monitoring

- Example: https://gist.github.com/
  - Easily storing and sharing text online
  - Used by programmers and legitimate users
    → Source code & information about configurations
Sources of leaks: Paste monitoring

• Example: https://gist.github.com/
  ○ Easily storing and sharing text online
  ○ Used by programmers and legitimate users
    → Source code & information about configurations

• Abused by attackers to store:
  ○ List of vulnerable/compromised sites
  ○ Software vulnerabilities (e.g. exploits)
  ○ Database dumps
    → User data
    → Credentials
    → Credit card details
  ○ More and more ...
Examples of pastes (items)
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

→ And it’s important to detect them automatically.

\(^3\) [https://github.com/D4-project/](https://github.com/D4-project/)
Paste monitoring at CIRCL: Statistics

- Monitored paste sites: 27
  - gist.github.com
  - ideone.com
  - ...

<table>
<thead>
<tr>
<th></th>
<th>2016</th>
<th>2017</th>
<th>08.2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collected pastes</td>
<td>18,565,124</td>
<td>19,145,300</td>
<td>11,591,987</td>
</tr>
<tr>
<td>Incidents</td>
<td>244</td>
<td>266</td>
<td>208</td>
</tr>
</tbody>
</table>

**Table:** Pastes collected and incident\(^4\) raised by CIRCL

\(^4\)http://www.circl.lu/pub/tr-46
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
Terms Tracker

• Search and monitor specific keywords/patterns
  ◦ Automatic Tagging
  ◦ Email Notifications

• Track Term
  ◦ ddos

• Track Set
  ◦ booter,ddos,stresser;2

• Track Regex
  ◦ circl\lu

• YARA rules
  ◦ https://github.com/ail-project/ail-yara-rules
Terms Tracker:

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Tracker</th>
<th>Date added</th>
<th>Level</th>
<th>Created by</th>
<th>First seen</th>
<th>Last seen</th>
</tr>
</thead>
<tbody>
<tr>
<td>regex</td>
<td><code>\\b[A-Z]{2}[0-9]{2}[?][?][0-9]{4}(?!\d)(?&lt;!\d)\d{2}(?!\d)</code></td>
<td>2019/09/12</td>
<td>1</td>
<td><a href="mailto:admin@admin.test">admin@admin.test</a></td>
<td>2018/08/31</td>
<td>2019/11/28</td>
</tr>
</tbody>
</table>
YARA Tracker:

<table>
<thead>
<tr>
<th>Type</th>
<th>Tracker</th>
<th>Date added</th>
<th>Level</th>
<th>Created by</th>
<th>First seen</th>
<th>Last seen</th>
<th>Tags</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>yara</td>
<td>all-yara-rules/rules/code/vbscript.yar</td>
<td>2020/09/17</td>
<td>1</td>
<td><a href="mailto:admin@admin.test">admin@admin.test</a></td>
<td>2020/09/17</td>
<td>2021/04/01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
rule test_vbscript
{
    meta:
        author = "kevthehermit"
        info = "Part of Pastemancer"
        reference = "https://github.com/kevthehermit/Pastemancer"
        strings:
            $a = "Function" nocase wide ascii fullword
            $b = "CreateObject" nocase wide ascii fullword
            $c = "vbscript" nocase wide ascii fullword
            $d = "As Long" nocase wide ascii fullword
            $e = "run" nocase wide ascii fullword
            $f = "for each" nocase wide ascii fullword
            $g = "end function" nocase wide ascii fullword
            $h = "mallocatevirtualMemory" nocase wide ascii fullword
            $i = "mallocatevirtualMemory" nocase wide ascii fullword
        condition:
            $a or $b
    }
```
Terms Tracker - Practical part

- **Create and test** your own tracker

Tags (optional, space separated)

E-Mails Notification (optional, space separated)

Tracker Description (optional)

- Select a tracker type –

Add 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
Decoder

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

<table>
<thead>
<tr>
<th>estimated type</th>
<th>hash</th>
<th>first seen</th>
<th>last seen</th>
<th>nb item</th>
<th>size</th>
<th>Virus Total</th>
<th>Sparkline</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/x-dosexec</td>
<td>c11c2be8d9ba4e86c8e3faa411aa6b867ba75abe</td>
<td>2019/11/28</td>
<td>2019/11/28</td>
<td>1</td>
<td>191</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application/x-dosexec</td>
<td>a50c6a73204ecce193b4017b398a250b5ce6f67</td>
<td>2019/11/28</td>
<td>2019/11/28</td>
<td>1</td>
<td>32768</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application/x-dosexec</td>
<td>cc5f200da711443ec12ae1b3cb6ab8bed80f22c4</td>
<td>2019/11/28</td>
<td>2019/11/28</td>
<td>1</td>
<td>203</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application/x-dosexec</td>
<td>eed67e8fa9cb9a43fe21ae653983a8e0a174f63</td>
<td>2019/11/26</td>
<td>2019/11/28</td>
<td>6</td>
<td>83</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Crawler

- Crawlers are used to navigate on regular websites as well as .onion addresses (via automatic extraction of URLs or manual submission).
- Splash ("scriptable" browser) is rendering 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: Cookiejar

Use your cookies to login and bypass captcha

<table>
<thead>
<tr>
<th>Description</th>
<th>Date</th>
<th>UUID</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>3thxemke2x7hcibu.onion</td>
<td>2020/03/31</td>
<td>90674deb-38fb-4eba-a661-18899ccb3841</td>
<td><a href="mailto:admin@admin.test">admin@admin.test</a></td>
</tr>
</tbody>
</table>

```json
{ "domain": ".3thxemke2x7hcibu.onion", "name": "mybb[lastactive]", "path": "/forum/", "value": "1583829465" }
```

```json
{ "domain": ".3thxemke2x7hcibu.onion", "name": "loginattempts", "path": "/forum/", "value": "1" }
```

```json
{ "domain": ".3thxemke2x7hcibu.onion", "name": "sid", "path": "/forum/", "value": "847ab8c97ff5bccc77ed8a" }
```

```json
{ "name": "remember_token", "value": "12\58cdd1511d74d341f23e" }
```

```json
{ "domain": ".3thxemke2x7hcibu.onion", "name": "mybb[announcements]", "path": "/forum/", "value": "9" }
```
Crawler: DDoS Booter
Correlations and relationship
Live demo!
Example: Dashboard
Example: Text search

Q: 1 Results for “gandcrab”

<table>
<thead>
<tr>
<th>#</th>
<th>Path</th>
<th>Date</th>
<th>Size (Kb)</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>crawled/2019/05/17/vs5e7g245s3pxjoc.onion374a1a89-4b16-4c3f-a460-4be8898da140</td>
<td>2019/05/17</td>
<td>15.44</td>
<td></td>
</tr>
</tbody>
</table>

Showing 1 to 1 of 1 entries

Totalling 1 results related to paste content
### Example: Items Metadata (1)

<table>
<thead>
<tr>
<th>Date</th>
<th>Source</th>
<th>Encoding</th>
<th>Language</th>
<th>Size (Kb)</th>
<th>Mime</th>
<th>Number of lines</th>
<th>Max line length</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/05/2019</td>
<td>pastebin.com_pro</td>
<td>text/plain</td>
<td>None</td>
<td>6.12</td>
<td>text/plain</td>
<td>1650</td>
<td>100</td>
</tr>
</tbody>
</table>

[Create MISP Event]

### Duplicate list:

<table>
<thead>
<tr>
<th>Hash type</th>
<th>Paste info</th>
<th>Date</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>['tsh']</td>
<td>Similarity: 19%</td>
<td>2019-04-13</td>
<td>archive/pastebin.com_pro/2019/04/13/EbMVR87S.gz</td>
</tr>
<tr>
<td>['tsh']</td>
<td>Similarity: 10%</td>
<td>2019-04-11</td>
<td>archive/pastebin.com_pro/2019/04/11/2X5HRVnX.gz</td>
</tr>
<tr>
<td>['tsh']</td>
<td>Similarity: 14%</td>
<td>2019-04-17</td>
<td>archive/pastebin.com_pro/2019/04/17/CuS90H7K.gz</td>
</tr>
<tr>
<td>['tsh']</td>
<td>Similarity: 23%</td>
<td>2019-04-20</td>
<td>archive/pastebin.com_pro/2019/04/20/AGQdToQVQ.gz</td>
</tr>
<tr>
<td>['tsh']</td>
<td>Similarity: 20%</td>
<td>2019-04-20</td>
<td>archive/pastebin.com_pro/2019/04/20/EDDq3b8.gz</td>
</tr>
<tr>
<td>['tsh']</td>
<td>Similarity: 21%</td>
<td>2019-05-05</td>
<td>archive/pastebin.com_pro/2019/05/05/X8nLZ6a.gz</td>
</tr>
<tr>
<td>['tsh']</td>
<td>Similarity: 7%</td>
<td>2019-04-13</td>
<td>archive/pastebin.com_pro/2019/04/13/Lyp4FVWW.gz</td>
</tr>
</tbody>
</table>

Showing 1 to 8 of 8 entries
Example: Items Metadata (2)

Hash files:

<table>
<thead>
<tr>
<th>estimated type</th>
<th>hash</th>
<th>saved_path</th>
<th>Virus Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/octet-stream</td>
<td>3975f058bb0d445b60c10a11f1a5d88e19e4fa84 (1)</td>
<td>HASHS/application/octet-stream /39/3975f058bb0d445b60c10a11f1a5d88e19e4fa84</td>
<td><a href="#">Send this file to VT</a></td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>fed93c1753270fc849a4db37027b569cdd9a6108 (1)</td>
<td>HASHS/application/octet-stream /te/fed93c1753270fc849a4db37027b569cdd9a6108</td>
<td><a href="#">Send this file to VT</a></td>
</tr>
</tbody>
</table>

Showing 1 to 2 of 2 entries
Example: Items Metadata (3)
Example: Browsing content

Content:

http://members2.mofosnetwork.com/access/login/
somosexual:6000
brazzers_glen:cocklick
brazzers61:braves61

http://members.naughtyamerica.com/index.php?m=login
gernblanston:unc2352
Janhuss141290:3199575
1getalliwant1:1377zeph
pwilks89:mon22key
Bman1551:hockey

MoFos IKnowThatGirl PublicPickUps
http://members2.mofos.com
Chrismagg40884:logannmd0
brand01:zzbrando1
aacoen:1q2w3e4r
1rstunkle23:my8self

BrazZers
http://ma.brazzers.com
gcjensen:gcj2ipva
skycscl7:rbcndn

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

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

*******************************************************************************

http://ddtnetwork.com/home.html
eu172930:hCSBgKh
UecwB6zs:15OXB$r#6K78FvU

http://pornxn.stiFFia.com/user/login
ReldWWek8939:Ro8LuJXxtB
dabudka:a7891789
brajits:brajits1

http://members.pornstarplatinum.com/sblogin/login.php/
gigiriveracom:xxxjay
jay123:xxxjay69

http://members.vividceleb.com/
Rufio99:fairhaven
ScHiFRvi:192091
Chaos84:HOLE5244
Riptor795:blade7
Dom188:harkonnen
GaggedUK:aik8chan

http:
Example: Search by tags

<table>
<thead>
<tr>
<th>Date</th>
<th>Path</th>
<th># of lines</th>
<th>Action</th>
</tr>
</thead>
</table>
| 2019/05/19| `archive/pastebin.com_pro/2019/05/19/ej67tQ4b.gz`
            |            | 71      |        |
|           | `cve` `bitcoin-address`                                              |            |        |
| 2019/05/21| `archive/pastebin.com_pro/2019/05/21/vM2SwyTe.gz`
            |            | 69      |        |
|           | `cve` `bitcoin-address`                                              |            |        |
| 2019/05/21| `archive/pastebin.com_pro/2019/05/21/rsnHnp5L.gz`
            |            | 71      |        |
|           | `cve` `bitcoin-address`                                              |            |        |
MISP
MISP Taxonomies

- **Tagging** is a simple way to attach a classification to an event or an 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 \(^5\).
- Can be easily cherry-picked or extended

\(^5\)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**: Evaluate the degree of identifiability of personal data and the types of pseudonymous data, de-identified data and anonymous data.

---

\(^6\)Future of Privacy Forum
Taxonomies useful in AIL

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

\(^7\)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
2. Add your own tags
3. Export AIL objects to MISP core format
4. Download it or Create a MISP Event

---

8 https://www.misp-project.org/taxonomies.html
MISP Export

1Gt545E48EPsyTC8voKQDCFfpTkwiuXduw:

<table>
<thead>
<tr>
<th>Object type</th>
<th>type</th>
<th>First seen</th>
<th>Last seen</th>
<th>Nb seen</th>
</tr>
</thead>
<tbody>
<tr>
<td>cryptocurrency</td>
<td>bitcoin</td>
<td>2020/01/17</td>
<td>2020/02/20</td>
<td>5</td>
</tr>
</tbody>
</table>

Expand Bitcoin address

Graph

Resize Graph

Add to MISP Export

53 of 88
MISP Export

nttfj36sp47cw2yecop572zvjjeazgazieunllouudplzqt2m5h465yd.onion :

<table>
<thead>
<tr>
<th>First Seen</th>
<th>Last Check</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020/02/19</td>
<td>2020/02/19</td>
<td>[80]</td>
</tr>
</tbody>
</table>

infolink:automatic-detection="onion"

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

Show Domain Correlations 4

Add to MISP Export
Automatic submission on tags

- MISP Auto Event Creation: Enabled
- The hive auto export: Disabled

[Images of MISP and TheHive interfaces showing tags and metadata]
API
AIL exposes a ReST API which can be used to interact with the back-end\textsuperscript{10}.

\begin{verbatim}
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
\end{verbatim}

- AIL API is currently covering 60\% of the functionality of back-end.

\textsuperscript{10}\url{https://github.com/ail-project/ail-framework/blob/master/doc/README.md}
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`
Feeding the framework
Feeding AIL

There are different ways to feed AIL with data:
1. Setup *pystemon* and use the custom feeder
   ○ *pystemon* will collect items for you
2. Use the new JSON Feeder (twitter)
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)
Via the UI (1)
Via the UI (2)

Submitting Pastes ...

Files Submitted: 1/1

Submitted pastes:
/home/all/git/AIFramework/PASTES/submitted/2018/06/29/02071576-b464-4bb-be59-37c58c9b8925.gz

Submitted Pastes

Success
Feeding All with your own data - API

```json
api/v1/import/item

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

"text": "text to import"
```
Feeding AIL with Twitter posts and associated urls

- AIL - feeder from Twitter\textsuperscript{11}
- The AIL-feeder-twitter search in Twitter using Twint (without API), crawls the urls and pushes the results in AIL
- The JSON format format can be extended via meta fields

\textsuperscript{11}https://github.com/ail-project/ail-feeder-twitter
Feeding AI with your own data - import_dir.py (1)

// requirements:

• Each file to be fed must be of a reasonable size:
  ○ ~ 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 configs/core.cfg
   ○ In the file configs/core.cfg,
   ○ Add 127.0.0.1:5556 in ZMQ_Global
   ○ (should already be set by default)

2. Launch import_dir.py with the directory you want to import
   ○ import_dir.py -d dir_path
Starting the framework
Running your own instance from source

Accessing the environment and starting AIL

# Launch the system and the web interface

cd bin/

./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
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 ARDB

**Server:** Flask

**Data message passing:** ZMQ, Redis list and Redis Publisher/Subscriber
AIL global architecture: Data streaming between module
AIL global architecture: Data streaming between module (Credential example)
Message consuming

\[\text{Module}_x \rightarrow \text{Redis set} \leftarrow \text{Module}_y \rightarrow \text{Module}_y\]

→ 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 bin/package/modules.cfg accordingly
import time
from pubsublogger import publisher
from Helper import Process

if __name__ == '__main__':
    # logger setup
    publisher.port = 6380
    publisher.channel = 'Script'
    # Section name in configs/core.cfg
    config_section = '<section name>'
    # Setup the I/O queues
    p = Process(config_section)
    # Endless loop getting messages from the input queue
    while True:
        # Get one message from the input queue
        message = p.get_from_set()
        if message is None:
            publisher.debug("{} queue is empty, waiting".format(config_section))
            time.sleep(1)
            continue
        # Do something with the message from the queue
        something_has Been Done = do_something(message)
Contribution rules
How to contribute

ONE DOES NOT SIMPLY

MAKE NEW CONTENT WITHOUT CONTRIBUTING
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.
Ongoing developments

- New JSON feeders
- Python API wrapper
- Data retention (export/import)
- MISP modules expansion
- auto Classify content by set of terms
  - CE contents
  - DDOS booters
  - ...
- Crawled items
  - duplicate crawled domains
  - tor indexer
Annexes
Managing AIL: Old fashion way

Access the script screen

```
1 screen -r Script
```

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-a d</td>
<td>detach screen</td>
</tr>
<tr>
<td>C-a c</td>
<td>Create new window</td>
</tr>
<tr>
<td>C-a n</td>
<td>next window screen</td>
</tr>
<tr>
<td>C-a p</td>
<td>previous window screen</td>
</tr>
</tbody>
</table>