

AIL Framework for Analysis of Information Leaks

Workshop - A generic analysis information leak open source software



CIRCL

Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy

alexandre.dulaunoy@circl.lu

Sami Mokaddem

sami.mokaddem@circl.lu

info@circl.lu

November 28, 2017

Objectives of the workshop

Our objectives of the workshop

- Demonstrate why data-analysis is critical in information security
- Explain challenges and the design of the AIL framework
- Learn how to install and start AIL
- Learn how to properly feed AIL with custom data
- Learn how to manage current modules
- Learn how to create new modules
- Practical part

Your objectives of the workshop

What are your expectations?

Sources of leaks

Sources of leaks: Paste monitoring

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

Sources of leaks: Paste monitoring

- Example: <http://pastebin.com/>
 - Easily storing and sharing text online
 - Used by programmers and legitimate users
 - 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

The image displays three overlapping text pastes from a code editor:

- Top-left paste (4.41 KB):** A C program snippet starting with a comment: `1. - - - - - Tool by Y3t1y3t (u`
- Top-right paste (2.02 KB):** A text file snippet starting with: `1. KillerGram - Yuffie - Smoke The Big Dick [smkwhr] (Upload`
- Bottom-left paste (4.57 KB):** A C program snippet starting with: `1. #include "wejwyj.h"`
- Bottom-right paste (2.66 KB):** An XML snippet starting with: `1. <item name="%the_component_to_be_disabled%" xsi:type="array">`

Sources of leaks: Others

- Mistakes from users

- https://github.com/search?q=remove_password&type=Commits&ref=searchresults

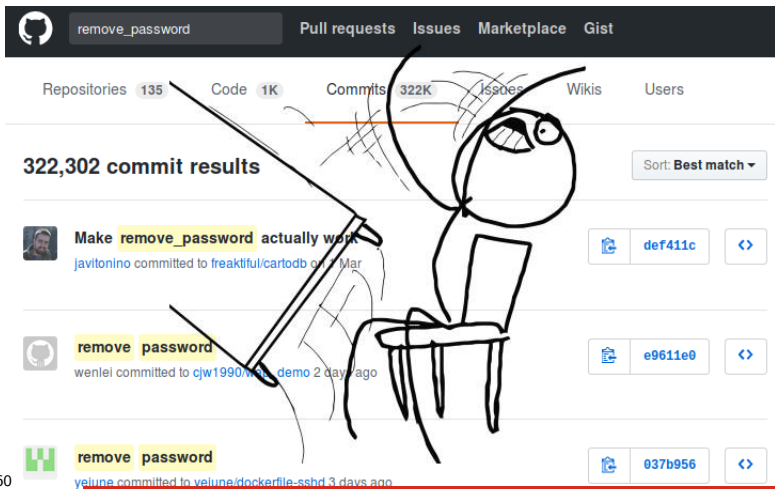
The screenshot shows the GitHub search interface for the query 'remove_password'. The search bar at the top contains the text 'remove_password'. Below the search bar, navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Gist' are visible. The search results are filtered to 'Commits', showing 322,302 results. The results are sorted by 'Best match'. Three commit entries are visible:

- Make remove_password actually work** by javitonino, committed to freakiful/cartodb on 1 Mar. Commit hash: def411c.
- remove password** by wenlei, committed to cjlw1990/wap_demo 2 days ago. Commit hash: e9611e0.
- remove password** by yelune, committed to yelune/dockerfile-sshd 3 days ago. Commit hash: 037b956.

Sources of leaks: Others

- Mistakes from users

- https://github.com/search?q=remove_password&type=Commits&ref=searchresults



The screenshot shows the GitHub search interface for the query "remove_password". The search bar at the top contains the text "remove_password". Below the search bar, navigation tabs include "Pull requests", "Issues", "Marketplace", and "Gist". The search results are categorized by "Repositories" (135), "Code" (1K), "Commits" (322K), "Issues", "Wikis", and "Users". The "Commits" category is highlighted with an orange line, and a hand-drawn cartoon character is pointing at it with a pencil. The character is a stick figure with a large head, wearing glasses, and sitting on a chair. The search results list three commits:

- Make remove_password actually work** by javitonino, committed to freaktiful/cartodb on 7 Mar. Commit ID: def411c.
- remove password** by wenlei, committed to cju1990/w... demo 2 day / ago. Commit ID: e9611e0.
- remove password** by yelune, committed to yelune/dockerfile-ssh3 3 days ago. Commit ID: 037b956.

Are leaks frequent?

Yes!

And it's important to detect them.

Paste monitoring at CIRCL: Statistics

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

Table: Statistics for 2016

Pastes 2016	Monthly average	Total
Fetches pastes	1 547 094	18 565 124
Security related (TR-46)	21	252
Incidents & investigations	54	649

AIL Framework

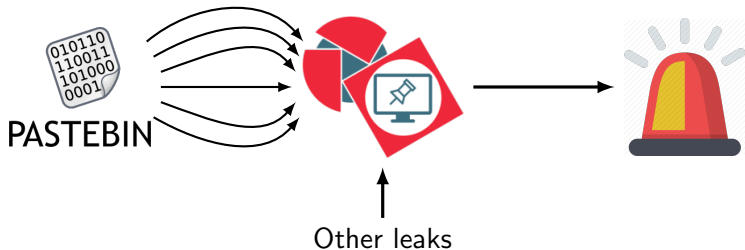
From a requirement to a solution: AIL Framework

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 2017, AIL framework is an open source software in Python. The software is actively used (and maintained) by CIRCL.

AIL Framework: A framework for Analysis of Information Leaks

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



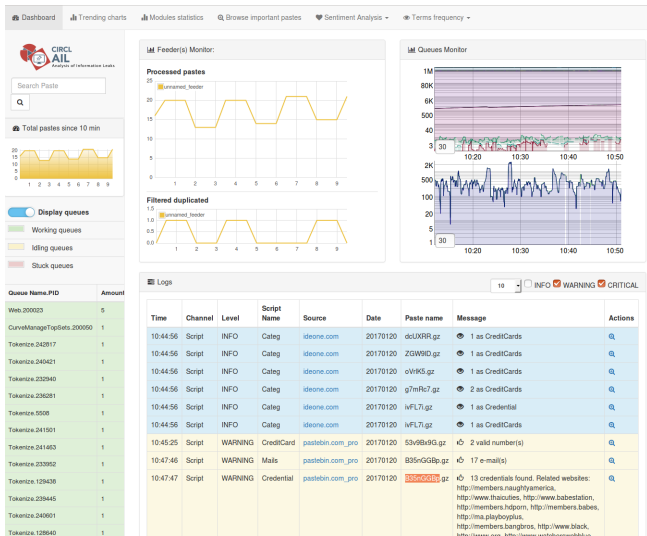
AIL Framework: Current capabilities

- 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**

AIL Framework: Current features

- Extracting **credit cards numbers, credentials, phone numbers, ...**
- Extracting and validating potential **hostnames**
- Keeps track of **duplicates**
- **Full-text indexer** to index unstructured information
- Terms, sets and regex **tracking and occurrences**
- **Sentiment/Mood analyser** for incoming data
- Modules manager
- And many more



Example: Following a notification (0) - Dashboard



Example: Following a notification (1) - Searching

Q 1 Results for "B35nGGBp"

Show entries Search:

#	Path	Date	Size (Kb)	Action
1	/home/adulau/git/AIL-framework/PASTES/archive/pastebin.com_pro/2017/01/20/B35nGGBp.gz	2017/01/20	5.8	 

Showing 1 to 1 of 1 entries Previous **1** Next

Totalling 0 results related to paste content

Example: Following a notification (2) - Metadata

Date	Source	Encoding	Language	Size (Kb)	Mime	Number of lines	Max line length
20/01/2017	pastebin.com_pro	text/plain	('en', 1.0)	5.8	text/plain	510	336

Duplicate list:

Show entries Search:

Hash type	Paste info	Date	Path
tlsh	Similarity: 93%	2017-01-12	/home/adulau/git/AIL-framework/PASTES/archive/pastebin.com_pro/2017/01/12/WeizLQUx.gz
tlsh	Similarity: 93%	2017-01-17	/home/adulau/git/AIL-framework/PASTES/archive/pastebin.com_pro/2017/01/17/Xqbx62vU.gz
tlsh	Similarity: 93%	2017-01-10	/home/adulau/git/AIL-framework/PASTES/archive/pastebin.com_pro/2017/01/10/iyfet4UM.gz
tlsh	Similarity: 92%	2017-01-14	/home/adulau/git/AIL-framework/PASTES/archive/pastebin.com_pro/2017/01/14/G7AB7q1m.gz
tlsh	Similarity: 92%	No date available	/home/adulau/git/AIL-framework/PASTES/archive/pastebin.com_pro/2016/12/31/CpDdkKbU.gz

Example: Following a notification (3) - Browsing content

Content:

```
http://members2.mofosnetwork.com/access/login/  
somoextremos: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:rbcndnd
```

```
#####
```

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

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

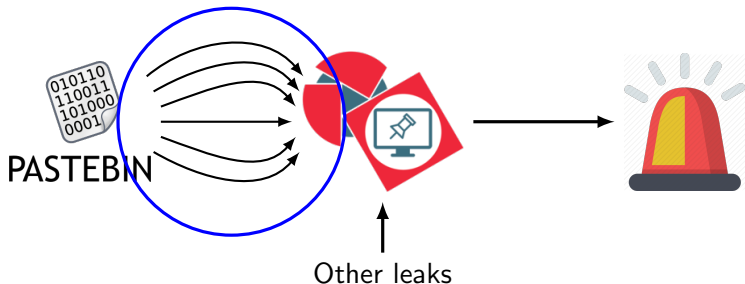
Example: Following a notification (3) - Browsing content

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  
ScHiFRv1:102091  
Chaos84:HOLE5244  
Riptor705:blade7  
Dnm18
```

Pystemon

Pystemon: A monitoring tool for PasteBin-alike sites



Pystemon: Current capabilities

- Flexible design, minimal effort to add another paste* site
- Use custom download functions for complex pastie sites
- Uses multiple threads per unique site to download the pastes
- (optional) Uses random User-Agents
- (optional) Uses random proxies
- Removes a proxy if it is unreliable (fails 5 times)
- (optional) Compress saved files with Gzip. (no zip to limit external dependencies)
- And more...

Setting up the framework

Setting up AIL-Framework from source or virtual machine

Setting up AIL-Framework from source

```
1 git clone https://github.com/CIRCL/AIL-framework.git
2 cd AIL-framework
3 ./installing_deps.sh
4 cd var/www/
5 ./update_thirdparty.sh
```

Using the virtual machine:

1. Download https://www.circl.lu/assets/files/ail-training/AIL_v@4986352.ova
2. Start virtualbox
3. File → import appliance → select AIL_v@4986352.ova
4. (for now) Prevent the automatic launch and `git pull` the changes

AIL ecosystem - Challenges and design

ALL ecosystem: Technologies used

Programming language: Essentially python2 (slowly migrating to python3)

Databases: Redis and Redis-levelDB

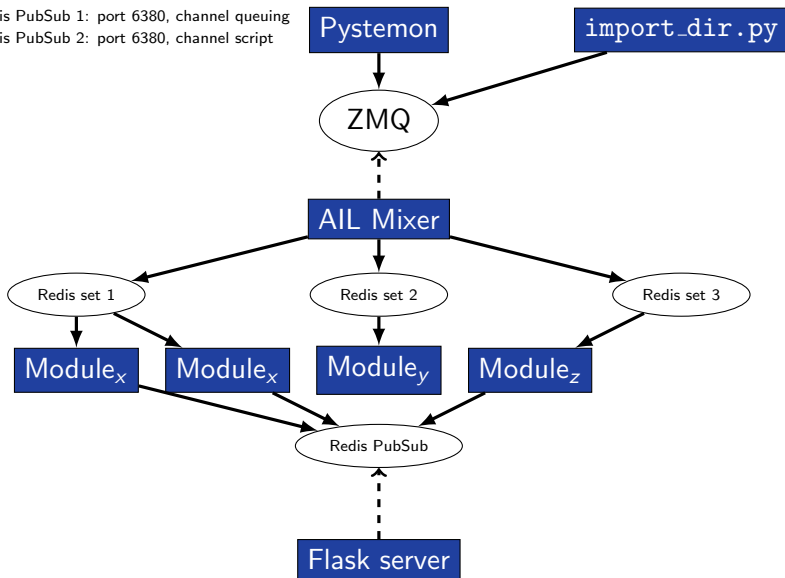
Server: Flask

Data message passing: ZMQ and Redis Publisher/Subscriber

AIL global architecture

Redis PubSub 1: port 6380, channel queuing

Redis PubSub 2: port 6380, channel script

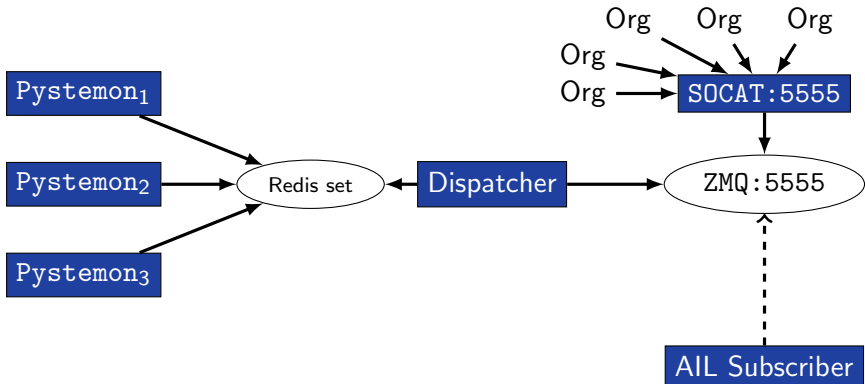


Data feeder: Gathering pastes with pystemon

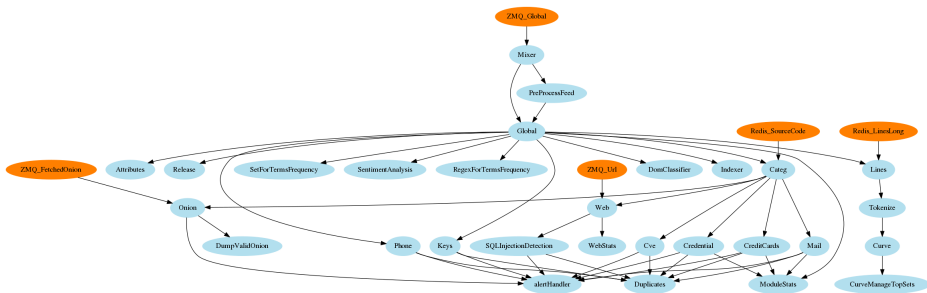
Pystemon global architecture

Redis PubSub 1: port 6380, channel queuing

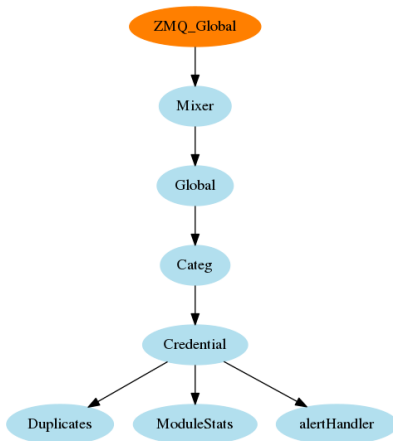
Redis PubSub 2: port 6380, channel script



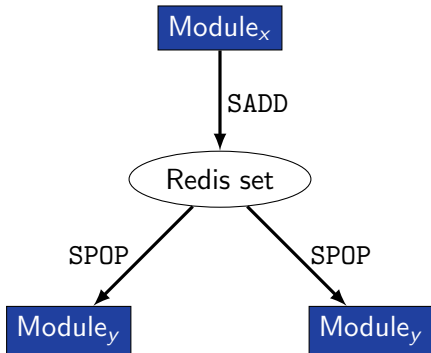
AIL global architecture: Data streaming between module



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



Message consuming



- No message lost nor double processing
- Multiprocessing!

Starting the framework

Running your own instance from source

Make sure that ZMQ_Global→address =
tcp://crf.circl.lu:5556,tcp://127.0.0.1:5556

Accessing the environment and starting AIL

```
1 # Activate the virtualenv
2 . ./AILENV/bin/activate
3
4 # Launch the system
5 cd bin/
6 ./LAUNCH
7     # check options 1->5
8
9 # Start web interface
10 cd var/www/
11 ./Flask_server.py
12     # -> Browse http://localhost:7000/
```

Running your own instance using the virtual machine

Login and passwords:

```
1 Web interface (default network settings):  
2   http://192.168.56.51:7000/  
3 Shell/SSH:  
4   ail/Password1234  
5
```

Managing the framework

Managing AIL: Old fashion way

Access the script screen

```
1 screen -r Script
```

Table: GNU screen shortcuts

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

Managing your modules: Using the helper

```
screen(1: ModuleInformation)
Running Queues
Action Queue name PID # S Time R Time Processed element CPU % Mem % Avg CPU%
<K> Attributes 31731 5 2017-08-03 00:24:03 0:00:01 G3rBPVqV 3.10% 1.56% 3.60%
<K> BrowseWarningPaste 31952 2 2017-08-03 00:23:55 0:00:09 yP3DaL03 0.00% 0.43% 0.00%
<K> Categ 31766 30 2017-08-03 00:23:58 0:00:06 Hs13zr6Y 6.70% 1.64% 17.40%
<K> Credential 31822 7 2017-08-03 00:24:04 0:00:00 yP3DaL03 3.50% 1.63% 3.50%
<K> CreditCards 31783 11 2017-08-03 00:24:04 0:00:00 q9qssLnd 4.80% 1.66% 4.80%
<K> DomClassifier 31755 71 2017-08-03 00:23:52 0:00:12 WzDFFBX 1.70% 1.64% 5.73%
<K> Indexer 31870 10 2017-08-03 00:24:03 0:00:01 0255zMLU 67.60% 1.93% 61.47%
<K> Lines 31744 5 2017-08-03 00:24:03 0:00:01 zLEpJfB 5.20% 1.57% 3.37%
<K> Mlxer 31704 2 2017-08-03 00:24:03 0:00:01 6GzeZ7zx 0.30% 0.43% 0.40%
<K> ModuleStats 31932 33 2017-08-03 00:23:57 0:00:07 7QCEJHTV 0.00% 1.64% 0.00%
<K> Phone 31888 2 2017-08-03 00:24:04 0:00:00 ghqFECHA 3.40% 1.59% 3.85%
<K> Release 31899 30 2017-08-03 00:23:57 0:00:07 3PwHXVtJ 1.80% 1.64% 0.55%
<K> SQLInjectionDetection 31941 1 2017-08-03 00:23:55 0:00:09 JNP00wmj 0.00% 1.49% 0.10%
<K> Tokenize 31775 42 2017-08-03 00:24:03 0:00:01 WTSF5hgL 6.60% 1.57% 6.60%
<K> Web 31818 17 2017-08-03 00:23:45 0:00:19 JNP00wmj 0.00% 1.74% 0.00%
<K> WebStats 31922 2 2017-08-03 00:23:14 0:00:50 JNP00wmj 0.00% 0.51% 0.00%

Idle Queues
Action Queue Idle Time Last paste hash
<K> Global 31717 0:00:00 nD0wHkX
<K> Keys 31880 0:00:00 yCWJXRlp
<K> Mail 31805 0:00:01 rhn2F3Yt

Queues not running
Action Queue State
<S> Curve Stuck or idle, restarting disabled
<S> CurveManagementTopSets Not running by default
<S> Cve Stuck or idle, restarting disabled
<S> DumpValidOntion Not running by default
<S> Duplicates Stuck or idle, restarting disabled
<S> Ontion Stuck or idle, restarting disabled
<S> PreProcessFeed Not running by default
<S> RegexForTermsFrequency Stuck or idle, restarting disabled
<S> SentimentAnalysis Stuck or idle, restarting disabled
<S> SetForTermsFrequency Stuck or idle, restarting disabled

Logs
TTime Module PID Info
00:23:29 Duplicates 31725 Cleared invalid pid in MODULE_TYPE_Duplicates
00:23:29 SentimentAnalysis 31961 *invalid pid in MODULE_TYPE_SentimentAnalysis
00:23:29 RegexForTermsFrequency 31852 *id pid in MODULE_TYPE_RegexForTermsFrequency
00:23:29 Curve 31837 Cleared invalid pid in MODULE_TYPE_Curve
00:23:29 SetForTermsFrequency 31864 *alid pid in MODULE_TYPE_SetForTermsFrequency
00:23:11 * cleared redis module info

0:24 0$ bash [1 ModuleInformation] 2$ Mlxer 3$ Global 4$ Duplicates 5$ Attributes 6$ Lines 7$ DomClassifier 8$ Categ 9$ Tokenize 10$ CreditCards 11$ Ontion 12$ Mail 13$ Web 14$ Creden
```


Feeding the framework

Feeding AIL

There are different ways to feed AIL with data:

1. Be a partner with CIRCL and ask to get access to our feed info@circl.lu
2. Setup *pystemon* and use the custom feeder
 - *pystemon* will collect pastes for you
3. Feed your own data using the `import_dir.py` script

Feeding AIL

There are different ways to feed AIL with data:

1. CIRCL partners and ask to access our feed info@circl.lu
 - ▷ You already have access
2. ~~Setup *pystemon* and use the custom feeder~~
 - ~~*pystemon* will collect pastes for you~~
3. Feed your own data using the `import_dir.py` script

Plug-in AIL to the CIRCL feed

You can freely access the CIRCL feed during this workshop!

- In the file `bin/package/config.cfg`,
- Set `ZMQ_Global->address` to `tcp://crf.circl.lu:5556`

Feeding ALL with your own data - import_dir.py (1)

/!\ 2 requirements:

1. Data to be fed must have the path hierarchy as the following:
 - 1.1 year/month/day/(textfile/gzfile)
 - 1.2 This is due to the inner representation of paste in ALL
2. Each file to be fed must be of a reasonable size:
 - 2.1 ~ 3 Mb is already large
 - 2.2 This is because some modules are doing regex matching
 - 2.3 If you want to feed a large file, better split it in multiple ones

Feeding ALL with your own data - import_dir.py (2)

1. Change your local configuration `bin/package/config.cfg`
 - In the file `bin/package/config.cfg`,
 - Add `127.0.0.1:5556` in `ZMQ_Global`
 - (should already be set by default)

Feeding ALL with your own data - import_dir.py (2)

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

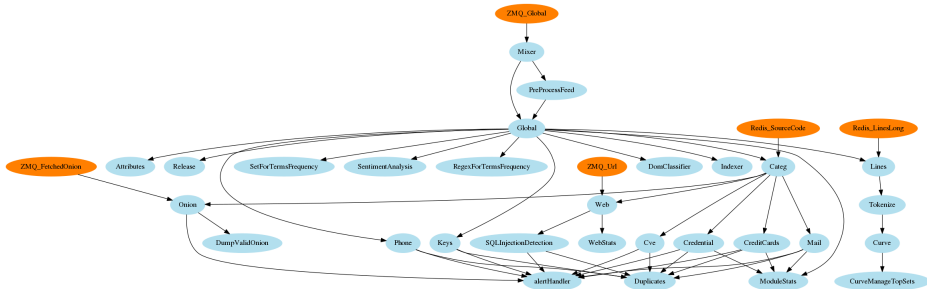
Feeding ALL with your own data - import_dir.py (2)

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

Creating new features

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

Choose where to locate your module in the data flow:



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

Writing your own modules - /bin/template.py

```
1 import time
2 from pubsublogger import publisher
3 from Helper import Process
4 if __name__ == '__main__':
5     # Port of the redis instance used by pubsublogger
6     publisher.port = 6380
7     # Script is the default channel used for the modules.
8     publisher.channel = 'Script'
9     # Section name in bin/packages/modules.cfg
10    config_section = '<section name>'
11    # Setup the I/O queues
12    p = Process(config_section)
13    # Sent to the logging a description of the module
14    publisher.info("<description of the module>")
15    # Endless loop getting messages from the input queue
16    while True:
17        # Get one message from the input queue
18        message = p.get_from_set()
19        if message is None:
20            publisher.debug("{} queue is empty, waiting".format(config_section))
21            time.sleep(1)
22            continue
23        # Do something with the message from the queue
24        something_has_been_done = do_something(message)
```

ALL - Add your own web interface

1. Launch `var/www/create_new_web_module.py`
2. Enter the module's name
3. A template and flask skeleton has been created for your new webpage in `var/www/modules/`
4. You can start **coding** server-side in:

```
var/www/modules/your_module_name/Flask_your_module_name.py
```

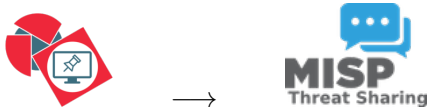
5. You can start **coding** client-side in:

```
var/www/modules/your_module_name/templates/your_module_name.html
```

```
var/www/modules/your_module_name/templates/header_your_module_name.html
```

Case study: Push alert to MISP

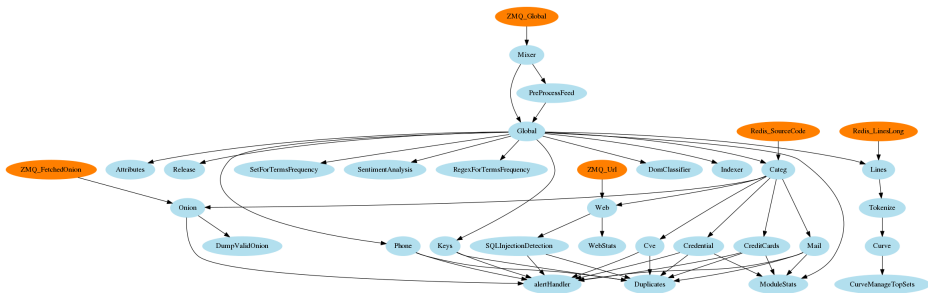
Push alert to MISP



Goal: Every alert concerning `Credential.py` and `CreditCards.py` are pushed to MISP

Case study: Finding the best place in the system

Best place to put it?



Case study: Updating alertHandler.py

alertHandler.py - commit 83e082e62a20a49e1ca2d546e4f19209135ac59d

```
1 [...]
2 if message is not None:
3     message = message.decode('utf8') #decode because of python3
4     module_name, p_path = message.split(';')
5 [...]
6
7 # Create MISP AIL-leak object and push it
8 if flag_misp: # MISP is connected
9     allowed_modules = ['credential', 'creditcards']
10    if module_name in allowed_modules:
11        # create and setup the MISP object
12        wrapper.add_new_object(module_name, p_path)
13        wrapper.pushToMISP()
14    else:
15        print('not pushing to MISP:', module_name, p_path)
16
```

Practical part

Practical part: Pick your choice

1. Improve module `keys.py` to support other type of keys (ssh, ...)
 - <https://github.com/veorq/blueflower/blob/master/blueflower/constants.py>
2. Graph database on `Credential.py`
 - Top used passwords, most compromised user, ...
3. Webpage scrapper
 - Download html from URL found in pastes
 - Re-inject html as paste in AIL
4. Integration of *truffleHog*
 - Searches through git repositories for high entropy strings and secrets, digging deep into commit history
 - <https://github.com/dxa4481/truffleHog>
5. Your custom feature

Contribution rules

How to contribute



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!

< (^ . ^) >

Conclusion

- 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.