C.A.A.S

Content

Purpose

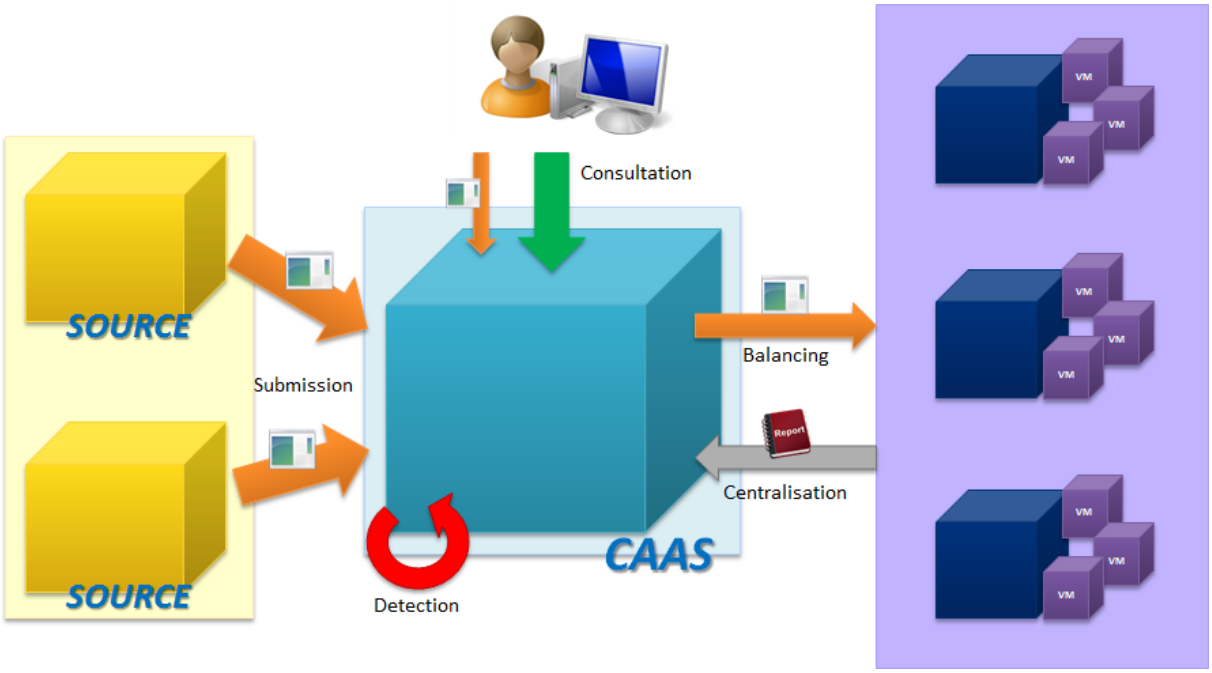
CAAS is a tool which allows automatizing cuckoo analyses from different sources, and dispatching them on different cuckoo servers. For now, this is just a really basic tool, this PDF file is here to provide minimal information so anyone can test/use it.

Any source can submit file-related metadata: for example, when using suricata file capture to provide files, suricata metadata (MD5, URL, hostname, src/dst IP, port numbers, protocol, etc.) will be stored into database. Any metadata can be handled.

Each file can be analyzed twice (or once, depending on the configuration): using the standard “cuckoomon” dll to trace activity, or using the “zer0m0n” driver. Analysis timeout can also be customized.

The web interface allows:

- Submitting manual analyses
- Specifying a sampling rate (N% files will be analyzed, their metadata will always be stored)
- Searching results on metadata values, signatures, etc.
- Scoring warning/alerts by specifying score limits (warning/alert) on total matched signatures score
- Writing signatures on metadata values or signatures text
- System configuration (sources, scoring, cuckoo servers, etc.)



Install/How to use

Base

To install CAAS on a debian/Ubuntu host system, first install dependencies:

```
#apt-get install php5 apache2 php5-sqlite sqlite3 python2.7 libapache2-mod-php5 php5-json python-pip
#pip install paramiko argparse
```

Clone the github repository:

```
$git clone https://github.com/conix-security/CAAS/
```

Edit the install.sh shell script, which will take care of the installation (folders creation, access rights, etc.) for you, using your favorite text editor (vim). If you do not have any cuckoo server or sources to add at this time, just remove the *echo "EDIT ME" / exit 0* lines. Then:

```
#chmod +x install.sh
#./install.sh
```

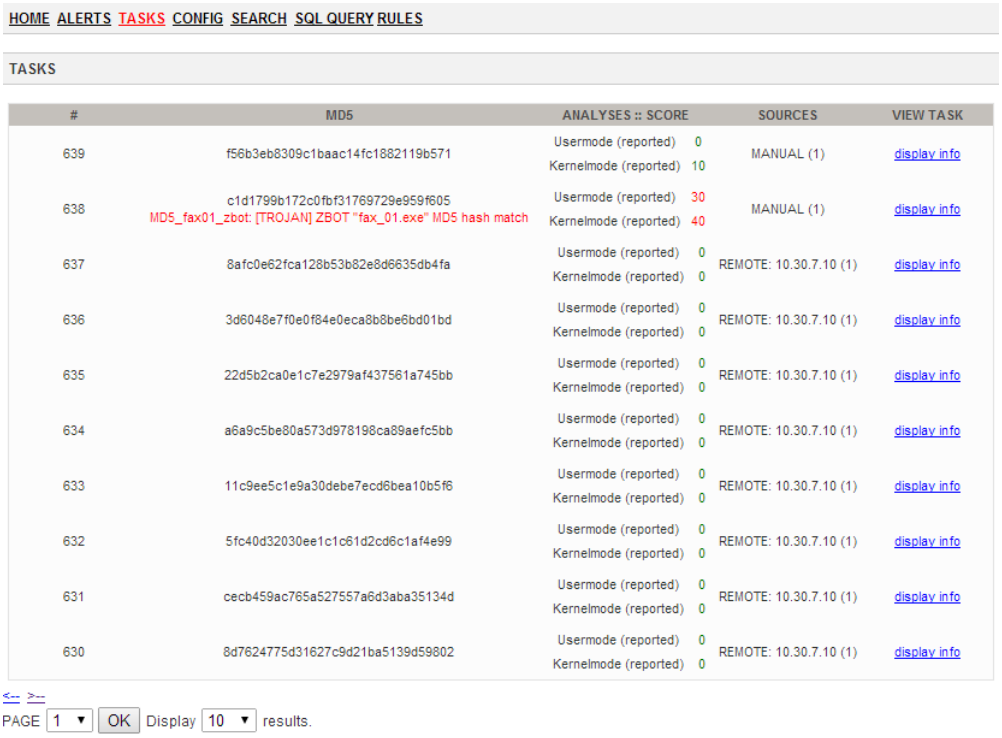
To enable the web interface through apache, modify the default (or default-SSL) website according to the web/README.php indications, then reload it (and enable the new site if you created one). If your CAAS install path is "/caas/", you just have to add:

```
    ScriptAlias /CAAS/ "/caas/web/"
    <Directory "/caas/web/">
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
```

Then, restart apache2.

```
#service apache2 restart
```

The Web interface should be accessible through http://localhost/CAAS/.



HOME ALERTS TASKS CONFIG SEARCH SQL QUERY RULES

TASKS

#	MD5	ANALYSES :: SCORE	SOURCES	VIEW TASK
639	f56b3eb8309c1baac14fc1882119b571	Usermode (reported) 0 Kernelmode (reported) 10	MANUAL (1)	display info
638	c1d1799b172c0fbf31769729e959f605 MD5_fax01_zbot: [TROJAN] ZBOT "fax_01.exe" MD5 hash match	Usermode (reported) 30 Kernelmode (reported) 40	MANUAL (1)	display info
637	8afc0e62fca128b53b82e8d6635db4fa	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
636	3d6048e7f0e0f84e0eca8b8be6bd01bd	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
635	22d5b2ca0e1c7e2979af437561a745bb	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
634	a6a9c5be80a573d978198ca89aefc5bb	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
633	11c9ee5c1e9a30debe7ecd6bea10b5f6	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
632	5fc40d32030ee1c1c61d2cd6c1af4e99	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
631	cecb459ac765a527557a6d3aba35134d	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info
630	8d7624775d31627c9d21ba5139d59802	Usermode (reported) 0 Kernelmode (reported) 0	REMOTE: 10.30.7.10 (1)	display info

<-- >--

PAGE 1 OK Display 10 results.

Configuration

The CAAS configuration can be performed using the query.py script, or using the web interface. Here are the web interface settings explained:

Main config

The “usermode analysis” and “kernelmode analysis” allows you to enable/disable respectively the “cuckoomon” and “zer0m0n” analyses. When enabled, analysis timeout and warning/alert scores limits can be specified. When a total signatures score reaches the “warning” limit, it’s displayed in orange, red if reaches the “alert” limit, otherwise, green.

The “sampling rate” setting indicates which amount of files will be analyzed. When a file is not analyzed, only its metadata will be stored.

The “report autodownload“ setting allows auto downloading JSON analysys reports. When disabled, they must be downloaded using the “query.py” script.

The “metadata” setting indicates if metadata must be processed: if no, nothing will be stored into database.

Cuckoo servers

To add a cuckoo server, you must indicate its IP address, SSH credentials and remote SSH listening port. If the server is a local, just enter 127.0.0.1 and nothing else is required.

Cuckoo servers may not be deleted, only disabled.

Local/remote sources

To add a local source, just indicate which folder must be processed. For remote sources, just indicate its IP address (required for IP address whitelisting). Again, sources may not be deleted, only disabled.

Custom metadata

If you are planning to provide custom metadata to CAAS, the formalism is really simple. The metadata must be contained into a separate file which has the same name with the ".meta" suffix (ex: toto.exe and toto.exe.meta). Meta files contents must follow the "field: value" schema. Suricata filecapture metadata files already follow this scheme.

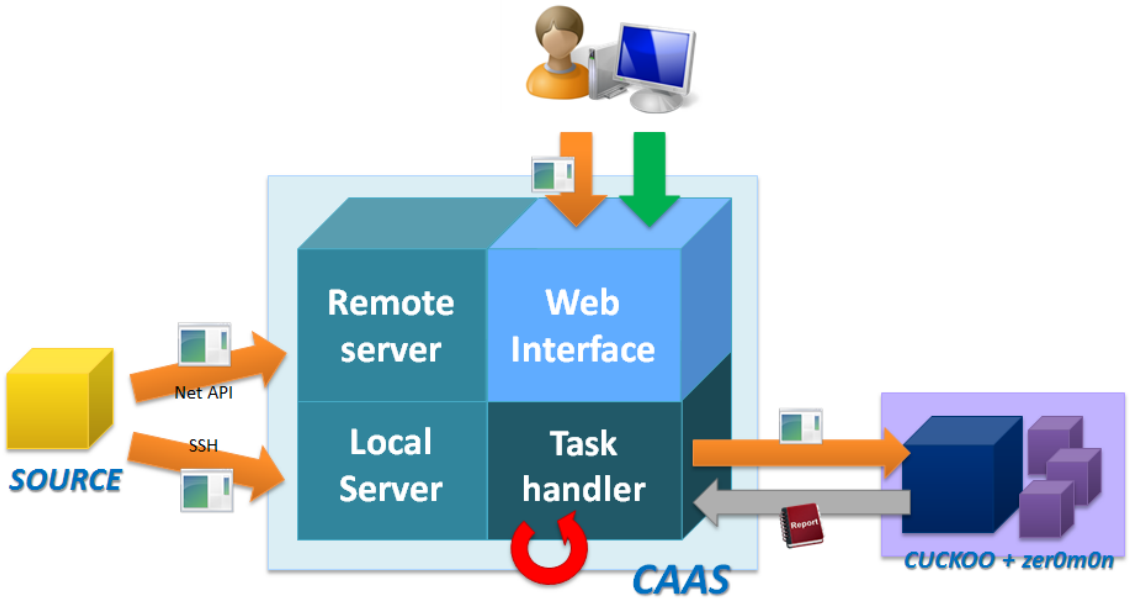
Alerts

Alerting rules work with SQL triggers after database inserts. A label (trigger name), description (alert description), criticity (low/medium/high) are mandatory. The alert can consider:

- MD5 file hash
- Signature keyword
- Metadata field (given regarding the existing metadata stored)

Once created, the SQL trigger can be viewed, and removed.

Generic structure



The main structure is quite simple: a remote and local "server" waits for new analyses which are stored into a sqlite3 database. The task handler waits for new database entries, dispatches analyses and gathers results after completions, which are stored into the database. SQL triggers are created to perform alerting on metadata/signatures. The Web interface is just a PHP application which can be integrated to Apache2 web server. All of these services can be run using python, and a management script can also be used to perform command-line actions.

Remote service

The remote server listens on a dedicated port. Any authorized remote source (IP whitelist filtering) can connect and submit a new candidate. A specific protocol (not encrypted) is used. First, the source sends md5, then the remote server asks for the file (if not already known) and/or metadata (if any). A new database analysis query entry is created along with the supplied metadata.

To start the remote service, run the “run_remote_service.py” python script.

To submit files using the python client script, run one of the “remote_submit” folder scripts. The “rem_sub.py” allows submitting a single file, and the “rem_local_sub.py” allows submitting any new file created in a specific folder. For instance, this can be used along with suricata file capture feature.

To add (or disable) a new remote source, configure it through the Web interface, or using the “query.py” python script.

Local service

The local server just lookup a local folder and picks up any new file. If a “.meta” file is present along with a file (i.e “malware.exe” and “malware.exe.meta”), its content will be parsed and stored into database. This allows submitting files using file sharing features (ftp, scp, etc.).

To start the local service, run the “run_local_service.py” python script.

To add (or disable) a new local source, configure it through the Web interface, or using the “query.py” python script.

Task Handler

The task handler just waits for new database entries. If a new analysis entry is detected, it dispatches to a cuckoo server using SSH connection. The service also monitors for finished analyses and gathers JSON reports. These JSON reports are parsed and signature-related information (scores, descriptions, names, and total score) are stored into the database. These reports are saved into a dedicated folder.

To run the service, just run the “run_service.py” python script.

To configure the service (sampling rate, cuckoo servers SSH parameters, etc.), do it through the Web interface, or using the “query.py” python script.