Darknet Network Analysis System (DNAS)

User guide and developer guide (ITR-UsgDNAS)

General information

Type	User guide (USG)
Reference	R005 (ATENA)
Version	1.0
State	Final
Owner	A. Deudon
Application date	29/04/2019
Classification	Public

Distribution list

Recipient	Channel	Reason
itrust staff	Internal repository	Validation

Document history

Version	Date	Author	Modifications
1.0	29/04/2019	J. Lancrenon	Finalization

Working group

Name	Organization
Alexis Deudon	itrust consulting

Approval

Name	Role	Responsibility	Date	Signature
Alexis Deudon	Tester	Consistency with product	26/04/2019	
Jean Lancrenon	QM	Validation	29/04/2019	Email to A. Deudon, email to B. Jager
Benoit Jager	Product owner	Applicability	29/04/2019	Confirmation email

Table of content

List of figures

1 Introduction

1.1 Context

The Darknet ANalysis System (DNAS) is part of the ATENA project which aims to offer measures of protection against attacks on critical infrastructures.

1.2 Objectives

There are two main objectives to this document, the first one is to provide a clear explanation to users on how the DNAS works and the second is to explain the development methodology used to develop the DNAS.

1.3 Document structure

The structure of the document is the following:

- Chapter 2 gives an explanation on the DNAS utilization.
- Chapter 3 describes the methodology that was applied to develop the system.

1.4 References

[1] Pocoo, Flask, http://flask.pocoo.org/.

[2] Scrapy, Scrapy architecture,https://docs.scrapy.org/en/latest/topics/architecture.html.

[3] ATENA Deliverable, D4.7 Design of detection agents and security components final report

1.5 Acronyms

DNAS	Darknet Network Analysis System
DC	Dark Collector
DB	Dark Brain
AI	Artificial Intelligence
VMS	Vulnerability Management System
KPI	Key Performance Indicator
PID	process ID

2 User guide

2.1 Installation

The DNAS Is composed of several python scripts. A Python3 and a Linux OS are required to start any installation process. Every module has an installation script.

- For DarkCollector's installation script, go to DARK_COLLECTOR/CONTROLLER and type **sudo python3 DC.py**.
- For the Dark Brain, go to DARK_BRAIN/CONTROLLER and type **sudo python3 DB.py**.
- For the server, go to DARK_BRAIN/Serveur and type **sudo python3 installation.py.**

For each of the scripts, instructions should be followed.

2.2 Main page

The main page contains graphics representing three KPIs. The first one shows the percentage of content that is pertinent for keyword's algorithm search, the second one, the number of URLs collected and the third the number of webpages contained in the database.

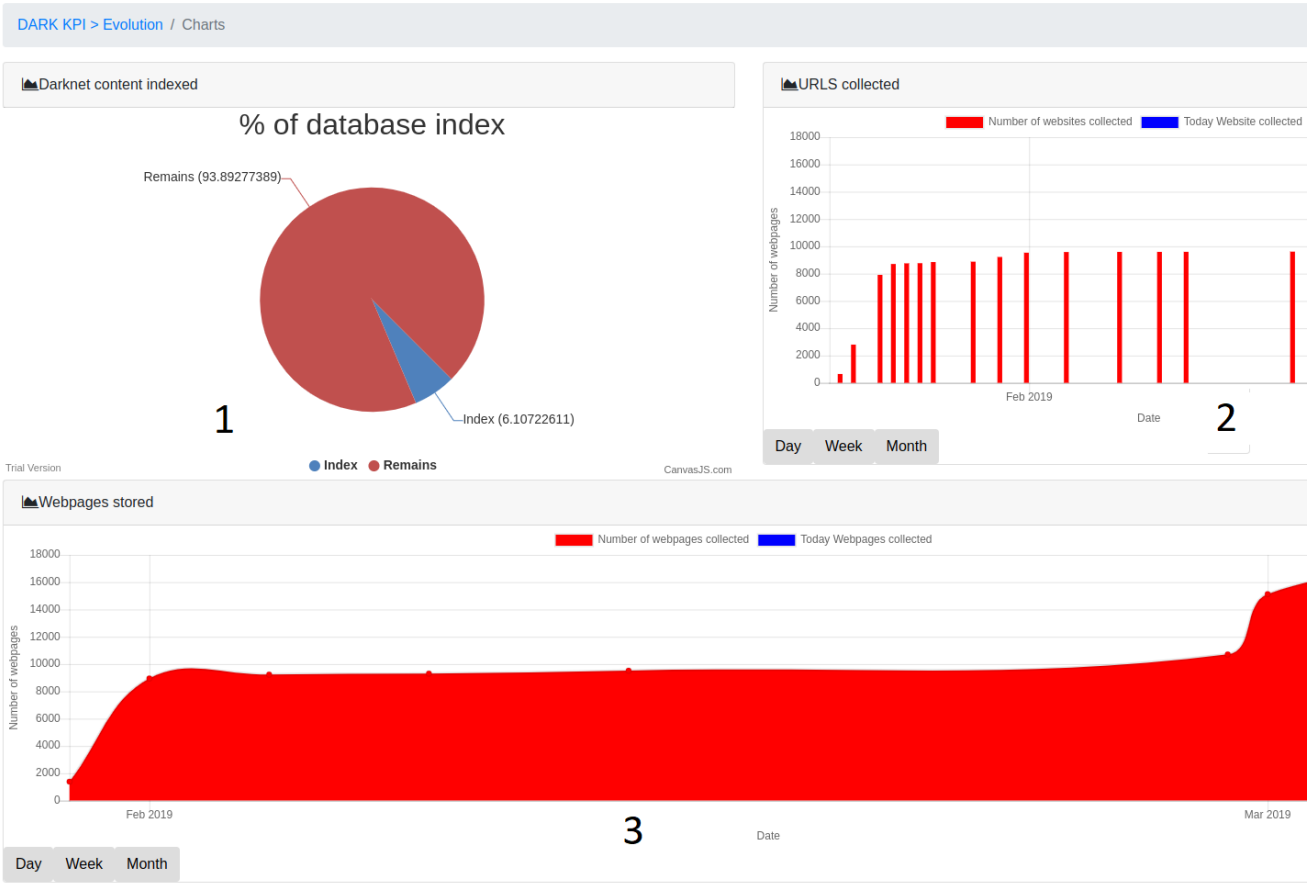


Figure 1: The DNAS main page

2.2.1 Database's indexing KPI

After the download of a webpage from the darknet, it is sent to the main server where it is stored with minor modifications, but the Dark Brain cannot use it for analysis yet.

To resolve that problem a mapping between words and the webpage containing them is provided through an indexer. The 'Vulnerability Search Algorithm' will use this mapping to perform keyword's algorithm searching.

2.2.2 Darknet's URL finding KPI

One of the most common reasons for the DNAS to become inactive is the lack of darknet websites to scan. Meaning that when the DNAS crawls a website it also collects external darknet URLs for further scanning. If all the websites contained in the DNAS database have been crawled, the DNAS will be put in sleep mode. The graph represents the number of darknet URLs found over a period of time.

2.2.3 Third KPI

The last KPI shows the number of webpages that have been stored in the database over a period of time.

2.2.4 Interpretation

A low percentage of indexing (see 2.2.1) could be a sign that the Indexer Dark Brain module has a problem. If no evolution is shown through the second graph, that might be a sign that the DC_Frontier module has crashed. The last graph is linked to the Dark Crawler module and could be interpreted in the same way.

2.3 DARK Collector

As the DNAS is composed of several modules, a central python script known as the 'controller' is installed on every machine containing the DNAS, and which interacts with the interface see Figure 2.

Every line in the table represents a controller with:

- IP of the controller;
- a unique ID that identifies the controller;
- number of Tor instances;
- action to open a Tor instance (press the '+' button);
- action to open a new module (press 'Start new').

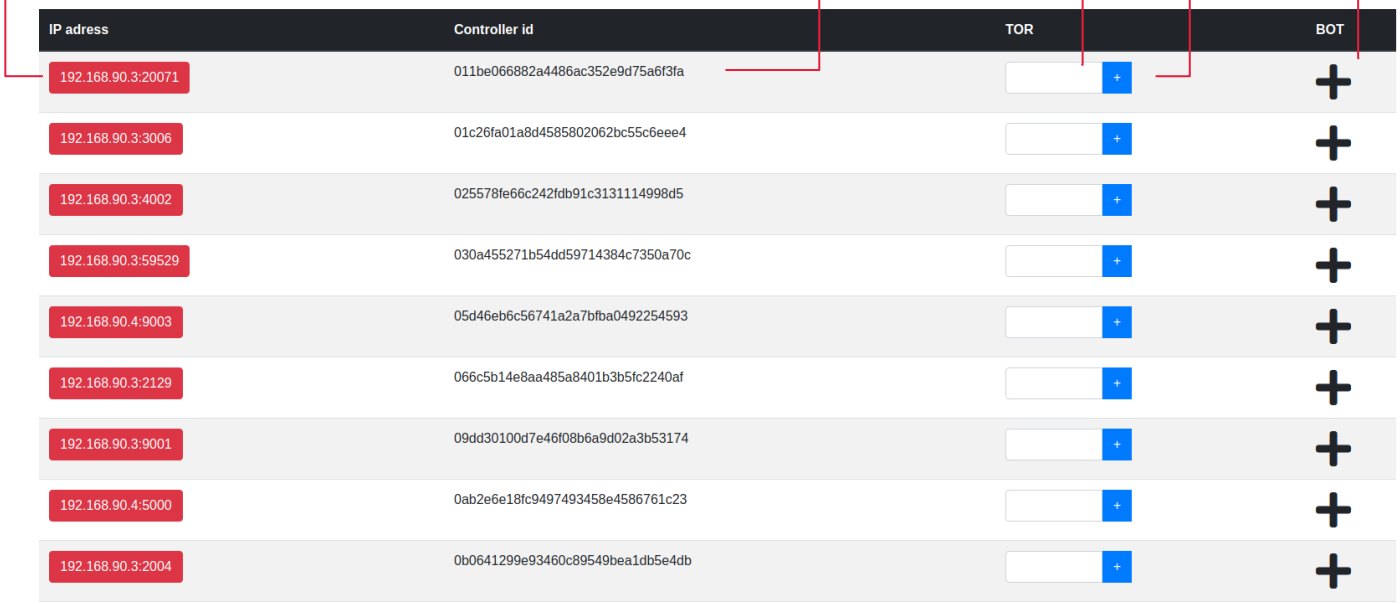


Figure 2: DC overview

The user can start new modules by pressing the 'Start new' button. Then a window appears showing two possibilities. The first one will deploy a crawler to collect darknet URL (see 2.3.1) and the second will deploy a crawler that will collect the webpages from darknet website (see 2.3.2).

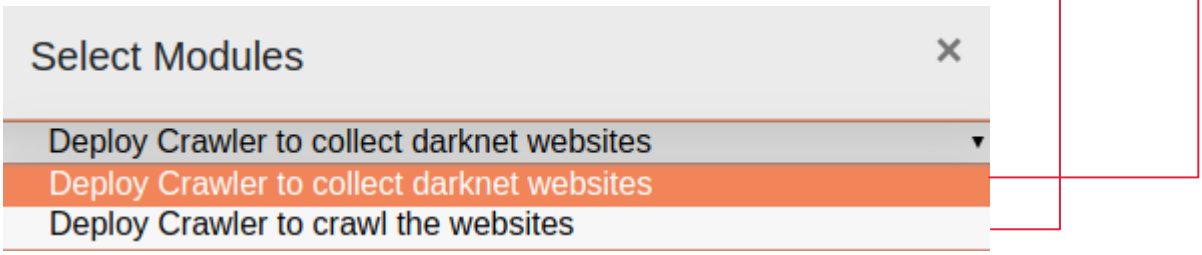


Figure 3: Dark Collector crawlers

2.3.1 'Deploy crawler to collect darknet websites':

The aim of this module is to crawl and focus only on the URLs that are found on a website. After pressing the 'Deploy' button a list of parameters will appear.

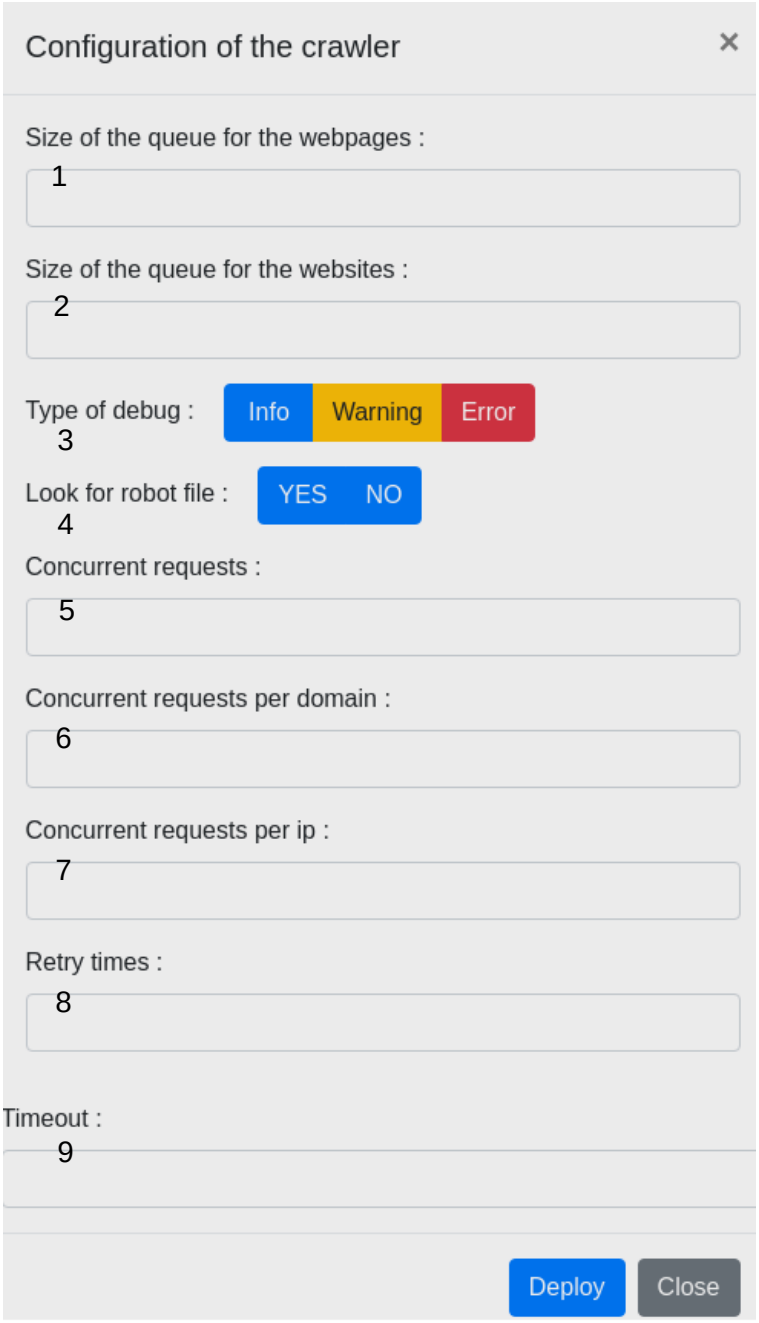


Figure 4: First crawler configuration

1. Size of the queue for the webpages:
   - Must be put at 0.
2. Size of the queue for the websites:
   - The modules need to stock the URL which they find on the website before sending it to the main server. This means that instead of sending URLs one-by-one to the server, it locally stores several darknet URLs and then sends them. In this field, the user needs to specify the number of URLs that can be locally stocked before being transmitted.
3. Debug:
   - The user chooses the type of debug. Info would be the more precise, it his often used by the developer to resolve issues with the module. As a user the appropriate method would be Warning or Error, Warning would send back warning plus other main KPIs and Error would send back critical error and some KPIs.
4. Robot file:
   - Some websites have a file containing information known as ‘robot file’ which aims to help bots crawling their website. If the file is present, the user can order the bot to follow it.
5. Concurrent requests:
   - The user must enter the maximum number of requests that the module could send in parallel.
   - Warning: A too high number could lead to an error in the module.
6. Concurrent requests per domain:
   - Same as before but for a single domain, if this number is too high that might cause the website to block any additional requests from DNAS.
7. Concurrent requests per IP:
   - Only useful if the URL is an IP.
8. Retry time:
   - This field specifies the number of times the module needs to reiterate the request if that one has failed.
9. Timeout:
   - This field specifies after how many seconds a timeout event is triggered.

2.3.2 'Deploy crawler to collect darknet websites':

The aim of this module is to go to darknet website and get darknet URLs from it. All the parameters are the same as describe above plus the following:

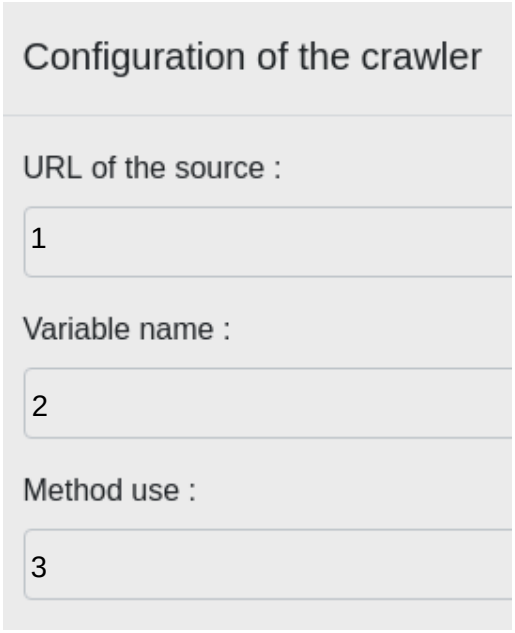


Figure 5: Second crawler configuration

1. URL of the source:
   - The user needs to enter the URL of the website he would like to crawl but without the parameter that could be in the URL. For example, if he has a website 'http://test.onion?q=test' he must enter everything before the question mark, so 'http://test.onion'.
2. Variable name:
   - If the website uses a get method to perform his search, the user will need to enter the variable name that is between the question mark and the equal, so in our example it is q.
   - If the website uses a post method, it will be trickier: the user will have to inspect the packet that is sent to the darknet website in order to find the variable name.
3. Method:
   - After the user has filled in any form on the internet, it is sent back to a server. The data that the user types in are sent to the website using two kinds of methods:
     - Get: The information is contained in the URL.
     - Post: The information is masked to the user.

To have more details on the module running on the machine, the user needs to click on the button containing the IP address.

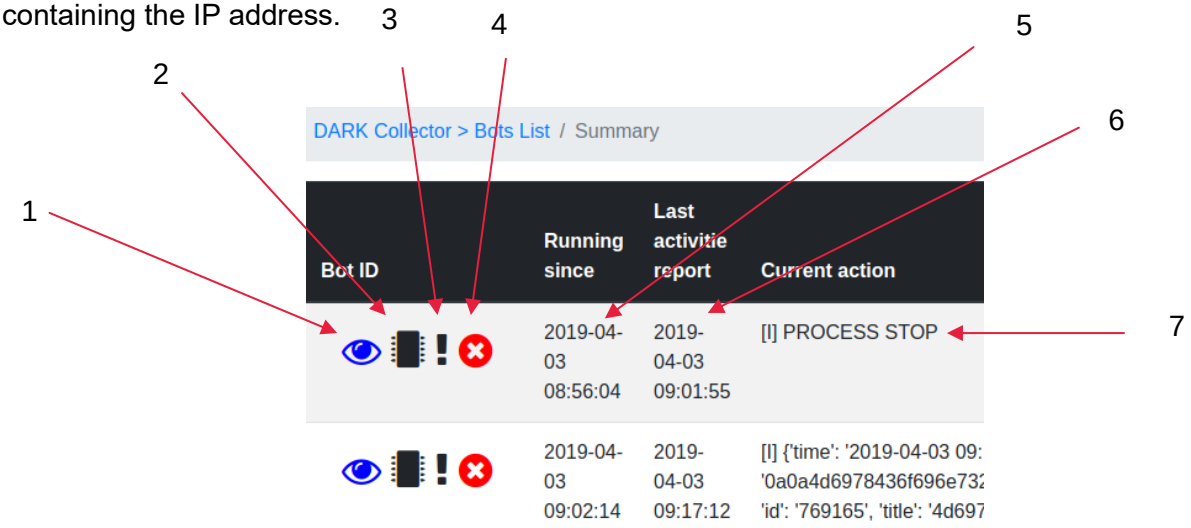


Figure 6: Information on modules

1. Crawling's information:
   - A popup will show up containing two graphs. The first shows the number of webpages downloaded per hour and the second the number of websites discovered per hour.

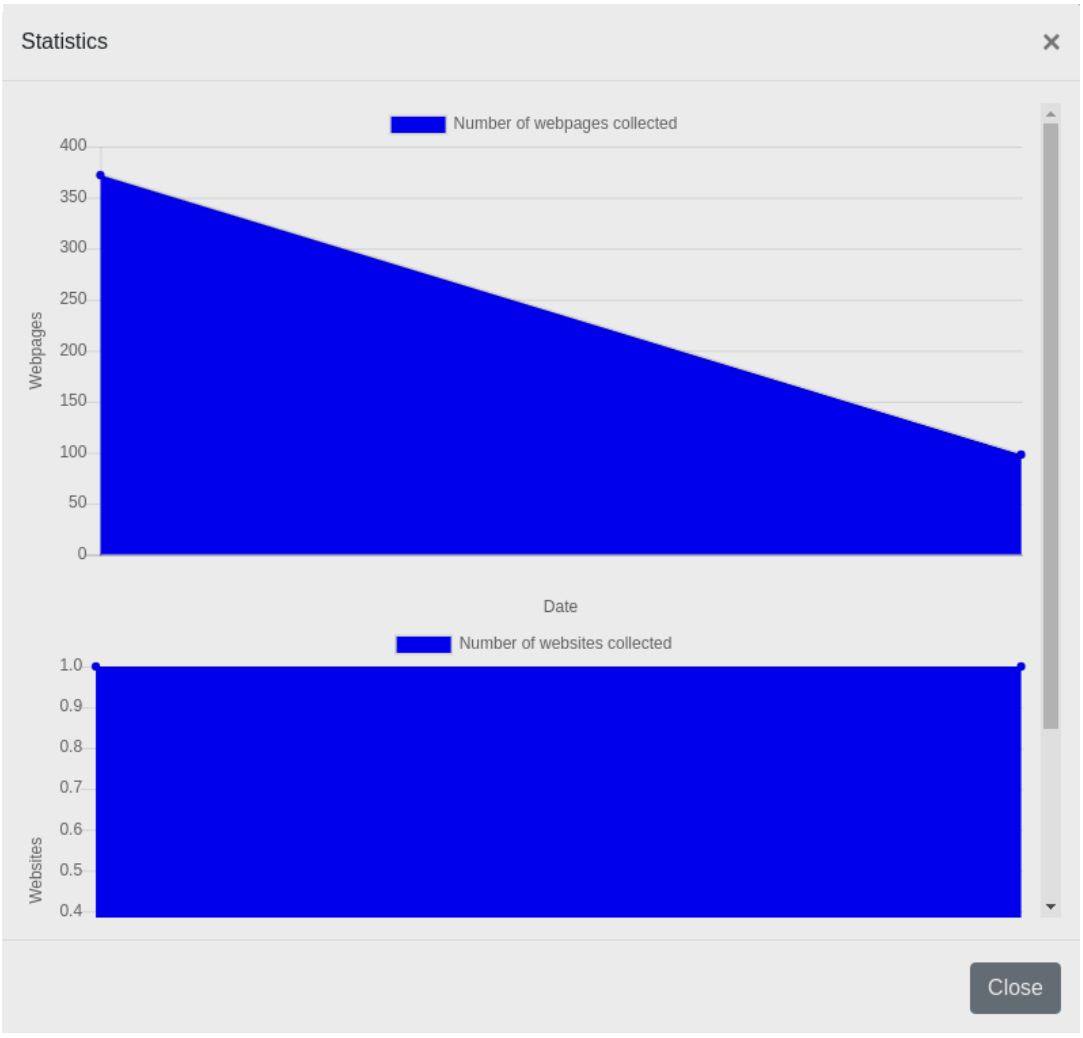


Figure 7: Crawling statistics

2. Performance:
   - Two graphs will appear, the first showing usage of CPU and the second usage of memory.

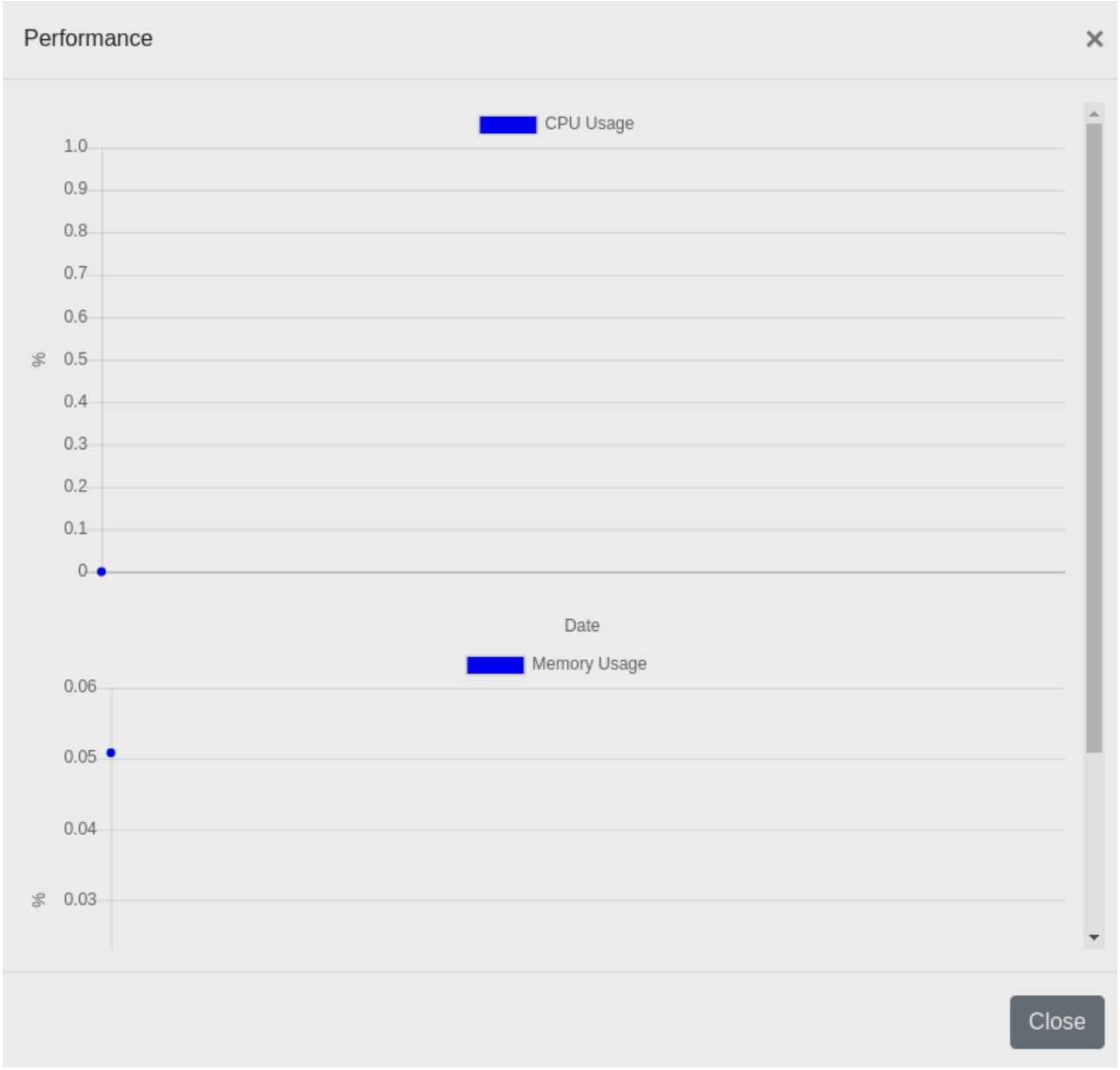


Figure 8: CPU and memory usage

3. Errors:
   - A table containing an error message and the time at which it appears will show up.

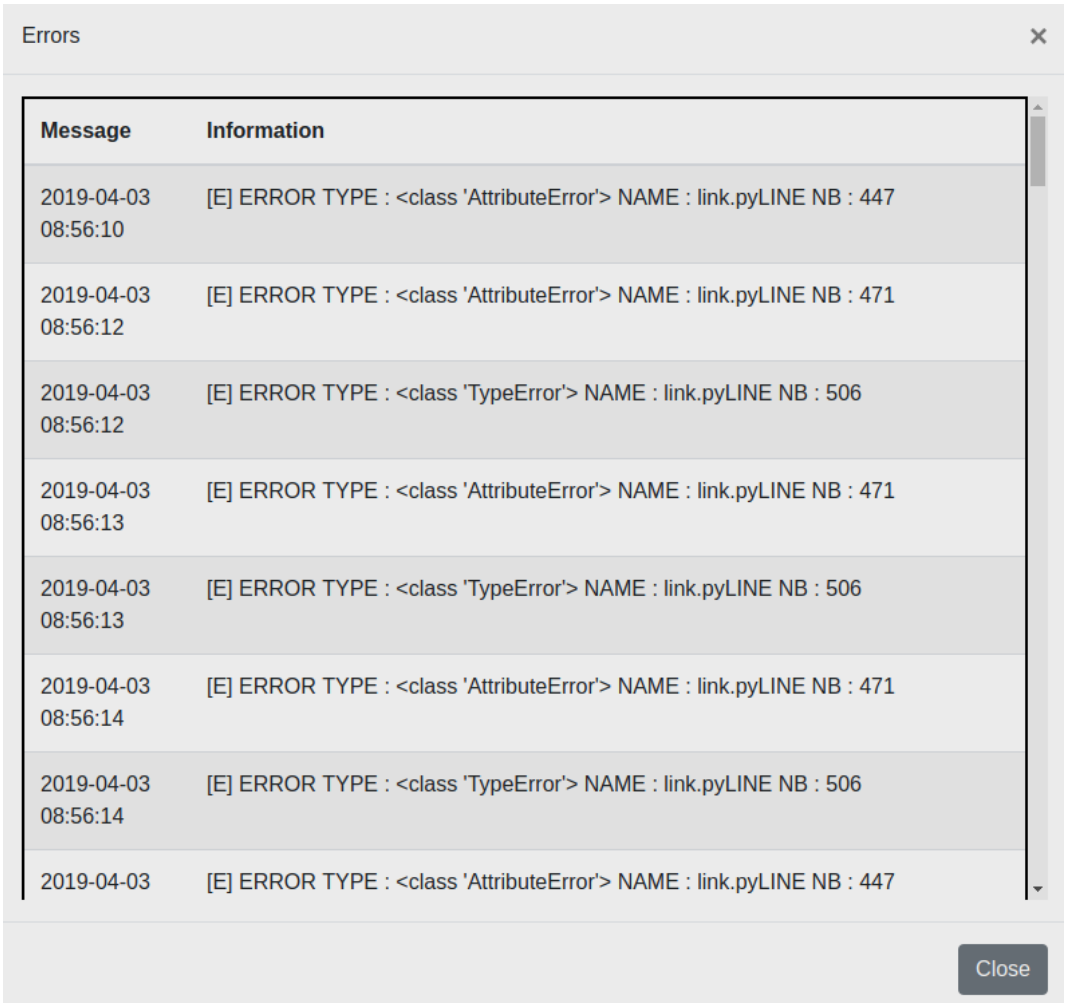


Figure 9: Module errors

4. Stop:
   - If the user presses this button, the module will stop its actions.
5. Start time:
   - Tells the user when the module was launched.
6. Last message:
   - Tells the user when the last message was received.
7. Current action:
   - Gives information on the activity.

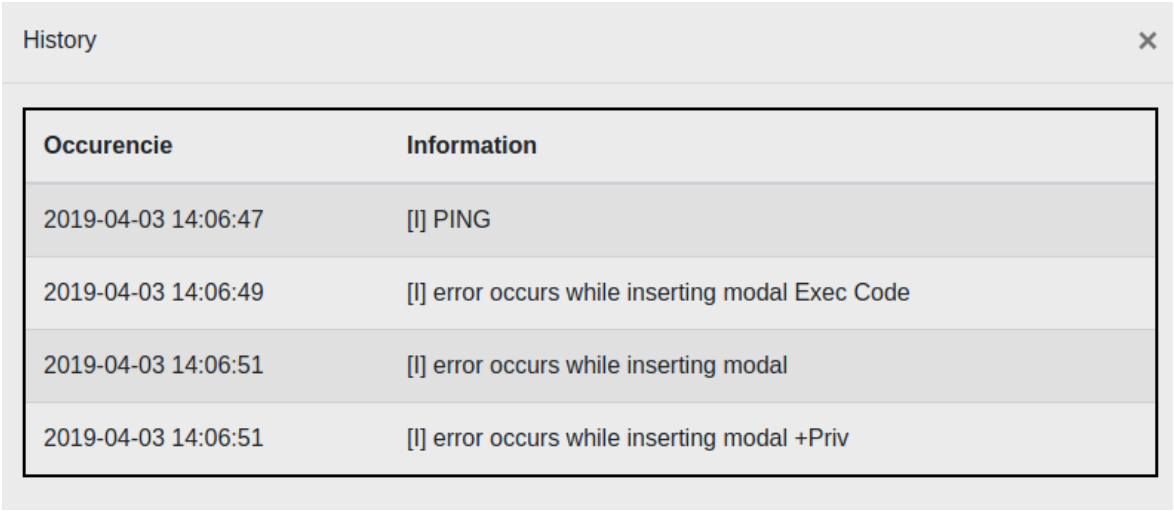
2.4 DARK Brain

The DARK Brain focuses on the analytic part, so we don't have any TOR instances to deploy. Instead the user could deploy:

- Vulnerability Search Algorithm and IA Modal Builder:
  - This module will gather all information on CVEs already existing in the NVD database.
  - If the user presses on the '*eye button*', a graph showing the number of CVEs per hour will appear.

Figure 10: CVE graph

- IA Modal Builder:
  - The 'IA Modal Builder' will construct CVE Modal based on the information collected by the CVE Crawler.
  - Pressing the '*eye button*' will show the history of the module.



Occurencie	Information
2019-04-03 14:06:47	[I] PING
2019-04-03 14:06:49	[I] error occurs while inserting modal Exec Code
2019-04-03 14:06:51	[I] error occurs while inserting modal
2019-04-03 14:06:51	[I] error occurs while inserting modal +Priv

Figure 11: Dark Brain module history

- Indexer:
  - The aim of this module is to take the content of the darknet and to apply transformation so that keyword research can be performed.
  - By pressing on the 'eye button', a graph showing the number of webpages per hour will appear.

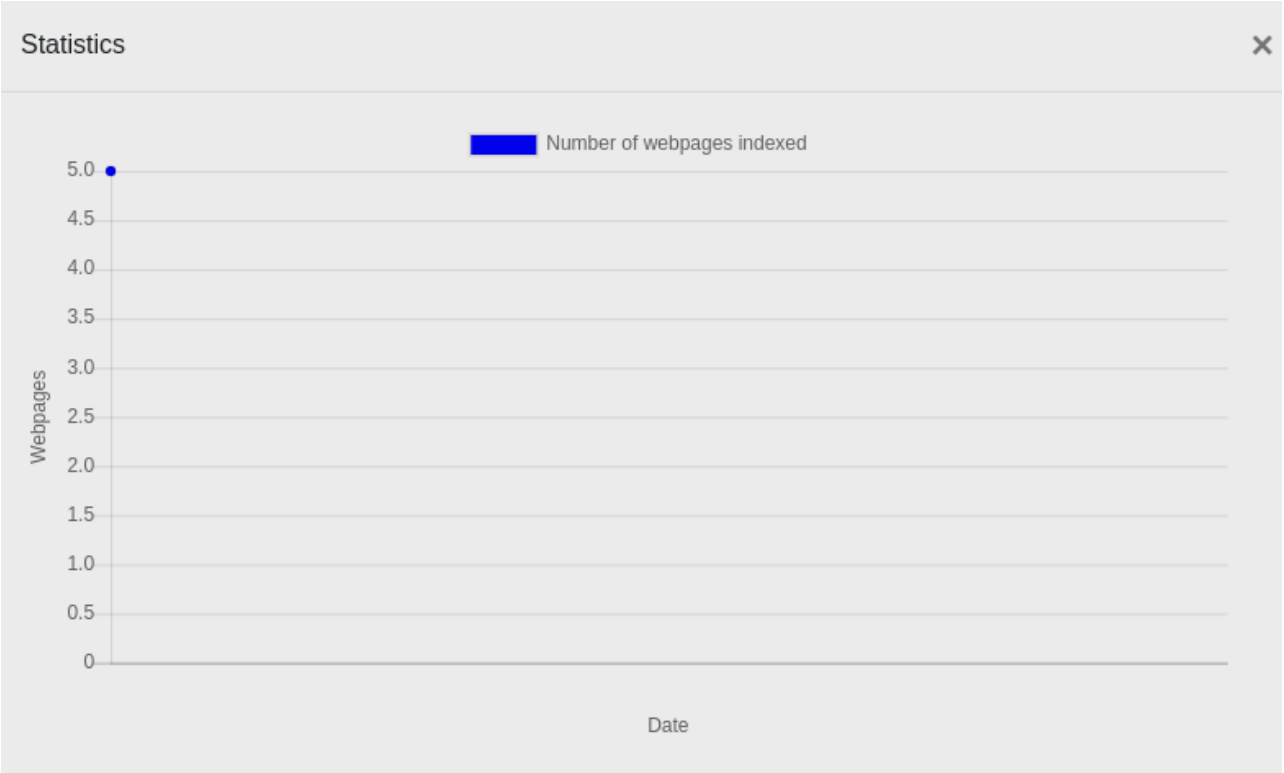


Figure 12: Indexer performance

The usage of the dashboard for the Dark Brain part is the same as for the Dark Collector, except that the user cannot start any TOR Instance.

2.5 Dark Search

The user can do searches based on keywords by clicking the tab ‘Search’. He can enter one or more keywords in the search box and then click on the button ‘Search’.

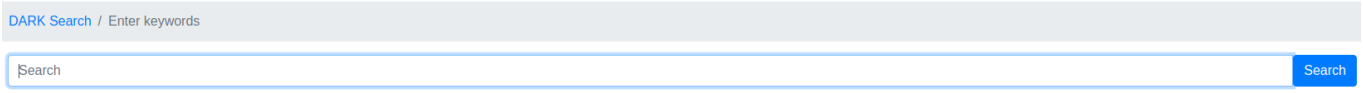


Figure 13: DNAS Search engine

After a few seconds, the results will appear. Every result contains the title of the page and if the user clicks on it, a box containing the full webpage appears.

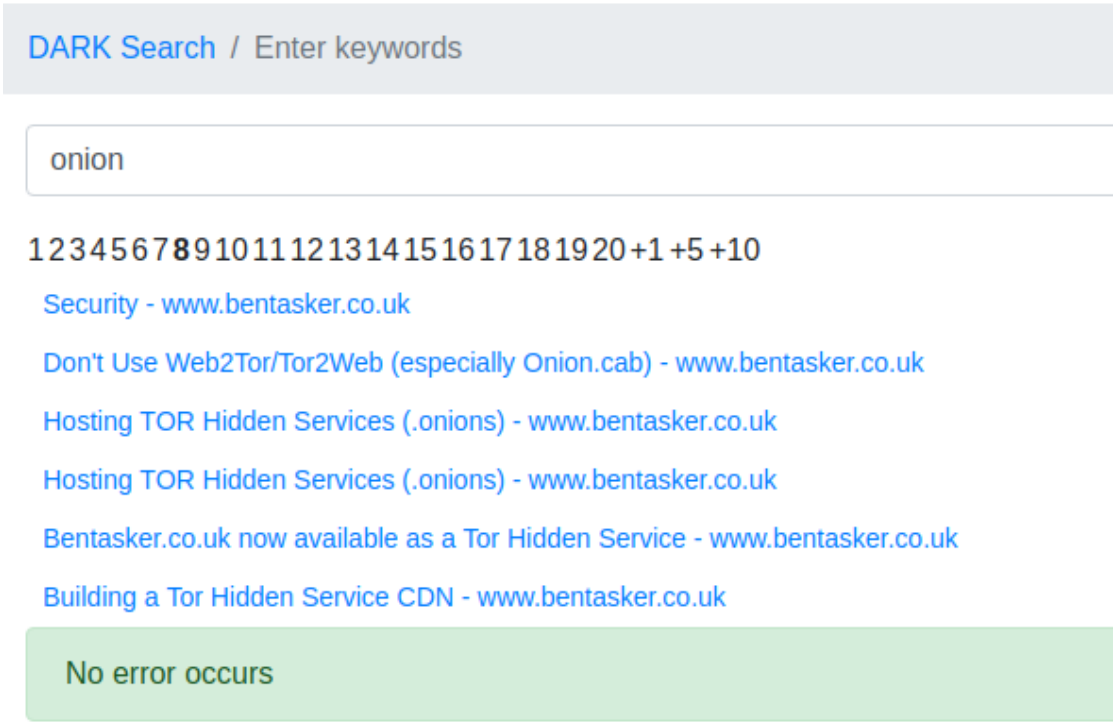


Figure 14: Search result

2.6 Information for user

On every page of the dashboard in the upper corner, three categories of message will be present.

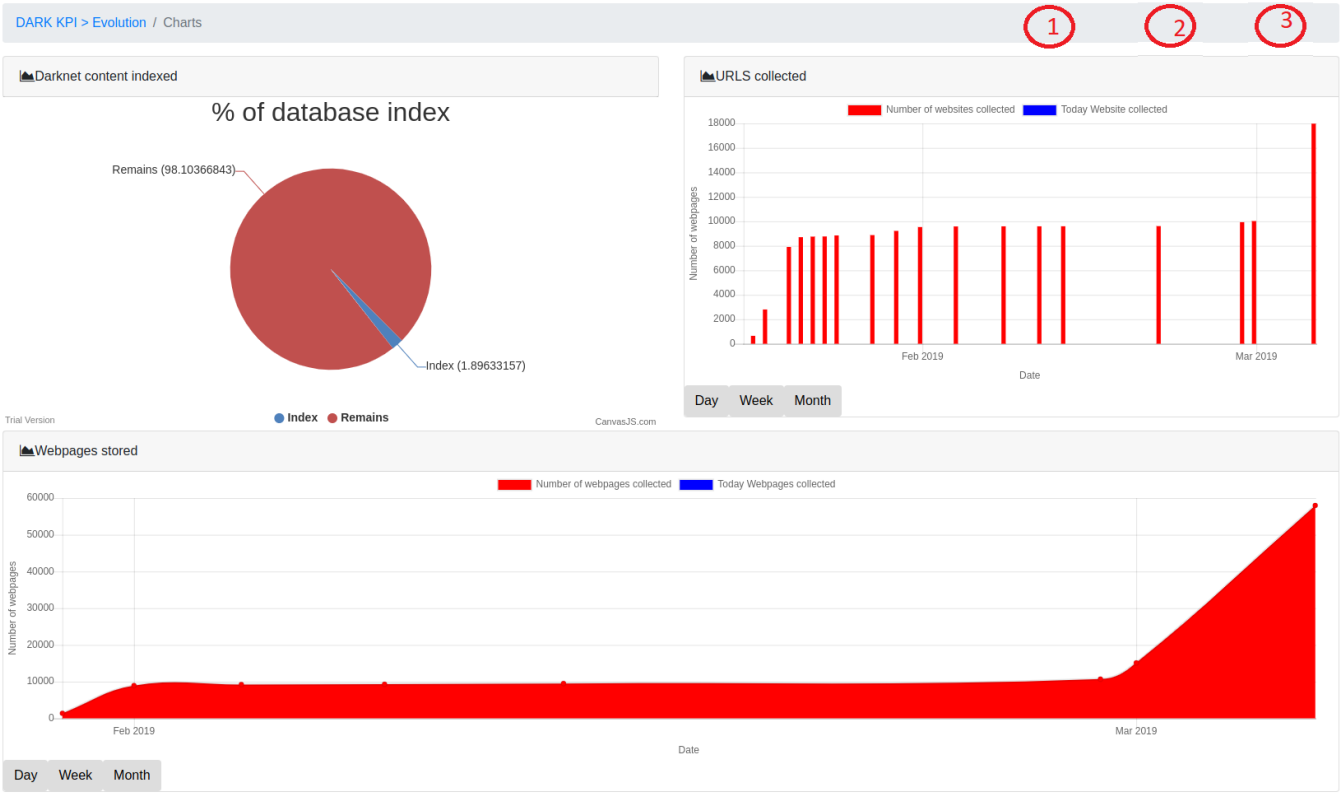


Figure 15: Logs

- Message: When the DNAS finds a vulnerability, he will send a message to the user.
- Warning: When a temporary problem occurs, the user could have more information by clicking on the warning button (e.g. a website that doesn't respond).
- Error: When a problem occurs that will affect the whole system permanently, a message will be put in this category (e.g. crash of a module).

3 For developers

Most of the DNAS is coded in Python. To have a good view of the different scripts I advise to open the project with Visual Studio Code.

There are two subfolders that match the two modules of DNAS, one containing scripts for DC and the other one for DB.

Figure 16: DNAS Main folder

3.1 Common component

As the DC and the DB can be run on different machines than the server, both of them have an API that offers communication to the main server, the class containing the script is stored in the same folder than the module with a name beginning with API.

In order to prevent any crashes every module will have his CPU and memory usage monitored through a class named 'ressourcesMonitor' and stored in the file monitor.py. The class will have in input the process ID of the process and every 10 minutes a KPI will be sent to the server.

3.2 Dark Collector

The main script of DC is named 'controller.py'. It is a webserver running with Flask that interacts with the main server to manage the two types of crawlers. Every crawler uses the Scrapy library which respects the architecture shown below (Figure 17).

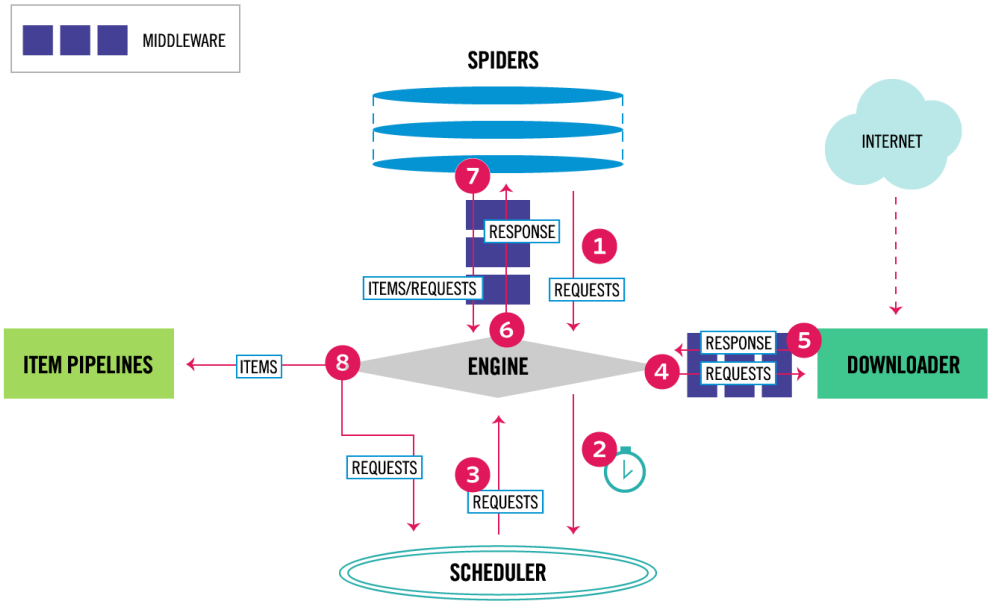


Figure 17: Scrapy architecture

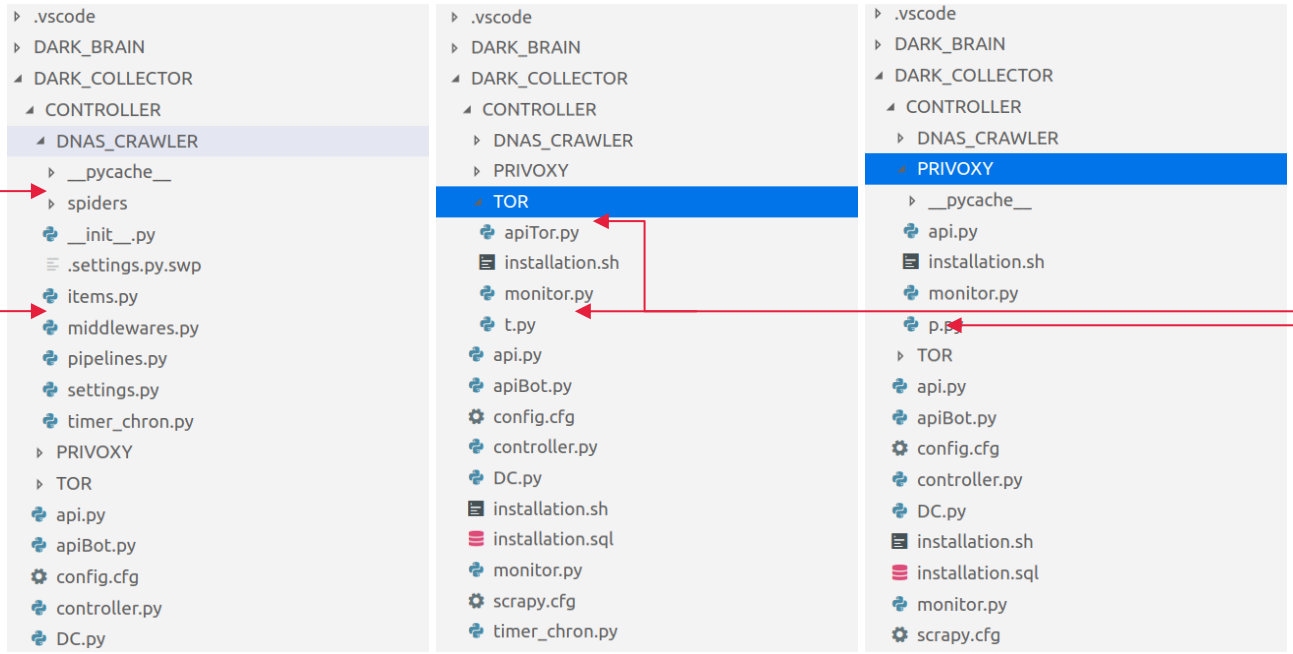


Figure 18: DC folder structure

- Spiders: It is the part of the crawler that gets responses from the website and extracts data to send it to the main server where it will be stored. The two spiders' scripts are in the file DARK_COLLECTOR\CONTROLLER\DNAS_CRAWLER\spiders\link.py.
- Middleware: It is an intermediary framework between the two spiders of darknet. It selects a TOR proxy so the packet can access the Darknet.
- TOR: The script that manages the TOR process is contained in the file DARK_COLLECTOR\CONTROLLER\TOR\t.py. It selects a port on which the TOR proxy will listen and store that information on a local database so PRIVOXY can retrieve it. The API which manages communication between the TOR process and the main server is in the file apiTor.py.
- PRIVOXY: The issue with the TOR proxy is that it only supports communication through SOCK socket, so PRIVOXY will convert the SOCK proxy into an http proxy, it will store the port on which it listens in the database and the middleware will get this information to send packet to TOR network. The file containing the PRIVOXY's management script is in the file DARK_COLLECTOR\CONTROLLER\PRIVOXY\p.py.

3.3 Dark Brain

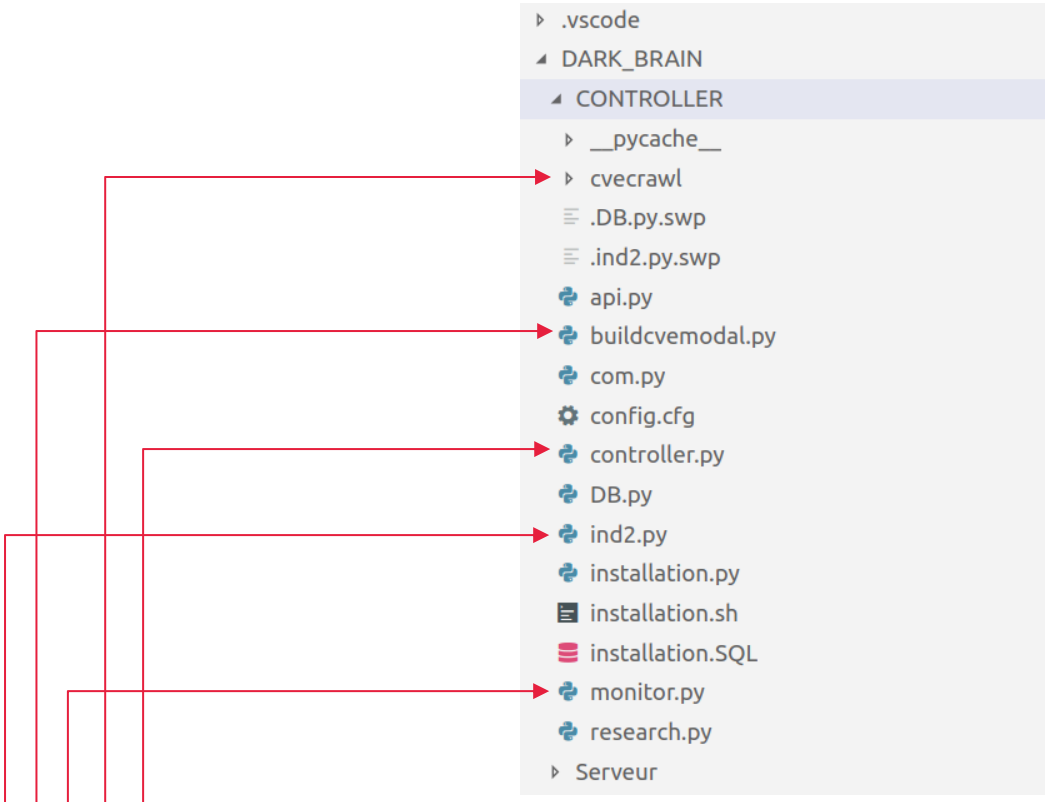


Figure 19: DB Folder structure

- Like the DC, the Dark Brain controller script is saved in the file DARK_BRAIN\CONTROLLER\controller.py.
- The Dark Brain has one crawler that is in the folder DARK_BRAIN\CONTROLLER\cvecrawl which targets the website https://www.cvedetails.com/ and triggers information on every CVE over the past years. The information will then be used to build an artificial intelligence modal.
- The class making the construction of the IA modal is contained in DARK_BRAIN\buildcvemodal.py, every 30 minutes it gets back the information that the CVE crawler has stored and it builds the modal, in consequence the modal will be constantly updated.
- Vulnerability Search Algorithm which does the search in the database based on an association of two types of keywords, the first one is related to the asset that could be targeted by attack and the second to the way an attack is performed (e.g.: DDos and PLC). When it finds a vulnerability, it sends this information to the VMS.
- The indexer script creates a database associating every word with the webpage where it is contained.