**Ben-David Benyamin** 13/10/2021

**URLhausAPI**

Table of Contents

[Options and usage 2](#_Toc85252557)

[Querying options 3](#_Toc85252558)

[Query recent URLs 3](#_Toc85252559)

[Query recent payloads 4](#_Toc85252560)

[Query URL information 5](#_Toc85252561)

[Query host information 6](#_Toc85252562)

[Query payload information 6](#_Toc85252563)

[Query tag information 6](#_Toc85252564)

[Query signature information 7](#_Toc85252565)

[Download malware 7](#_Toc85252566)

[Data manipulation and filtering 7](#_Toc85252567)

[Fetch 8](#_Toc85252568)

[Output to file 8](#_Toc85252569)

[Regex filtering 9](#_Toc85252570)

# Options and usage

usage: main.py [-h] [-G] [-P] [-V] [-L] [--url] [--sha256] [--urlId]

[--host] [--payloadmd5] [--payloadsha256] [--tag]

[--signature]

Hello, this program is meant to aid in basic and advanced analytics,

collecting and tracking and sharing of malware URLs, helping network

administrators and security analysts to protect their network and customers

from cyber threats.

options:

-h, --help show this help message and exit

-G, --getRequest create a get request to the API

-P, --postRequest create a post request to the API

-V, --version show program's version number and exit

-L, --limit Used to limit the number of entries received by the API

call, example "-L 3" OR "-limit 3"

--url the url that the user wants to query

--sha256 the hash of the payload that you want to download

--urlId the ID of the url that is queried

--host the name of the queried host

--payloadmd5 the md5 hash of the payload that is queried

--payloadsha256 the sha256 hash of the payload that is queried

--tag the tag of the database entry that was searched

--signature the signature that will be used to query the API

# Querying options

This tool gives the user the full access to the API of URLhaus, we will go through all the possibilities that are available to the user.

## Query recent URLs

Querying for the most recent additions that were made in the past 3 days, this call is restricted to 1000 entries (image 1).

There is the ability to cap the number of entries that will be returned by this call (image 2).

A picture containing text

Description automatically generated

Image 1: the use of the U and urls flags, the two variations can be seen.

The user can use the -G flag to send a GET request or he can user the –getRequest flag as well, they serve the same purpose.

As mentioned above the U value that is passed to -G or –getRequest has two variants, the user can use ether U or urls to query for the recently added URLs.

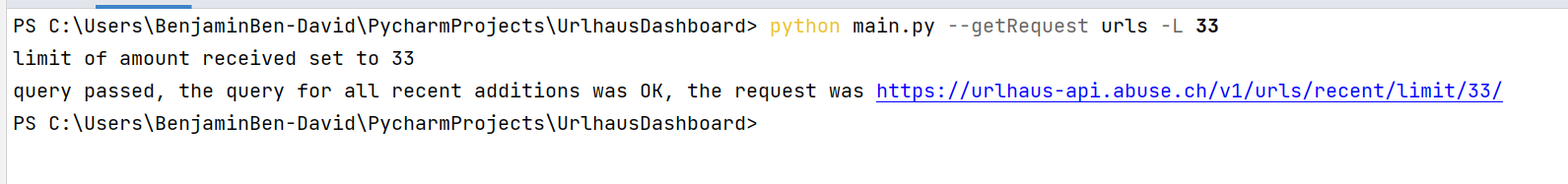


Image 2: the use of limits on the number of entries received from the AIP call.

There is the option to limit the amount of database entries that are retrieved from the query, this is possible by using the -L or –limit flag, the value of the flag must be an integer and its value must be between 1 and 1000 (max entries that can be fetched).

## Query recent payloads

To retrieve a list of recent payloads (recent payloads seen by URLhaus), the API will return a list of recent payloads from the past 3 days, this call is restricted to 1000 entries (image 3).

There is the ability to cap the number of entries that will be returned by this call (image 4).

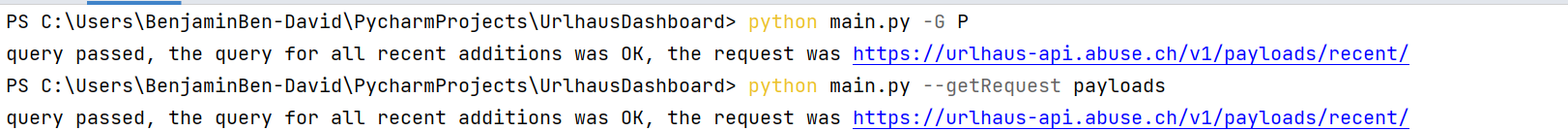


Image 3: the use of the P and payloads flags, the two variations can be seen.

The user can use the -G flag to send a GET request or he can user the –getRequest flag as well, they serve the same purpose.

As mentioned above the P value that is passed to -G or –getRequest has two variants, the user can use ether P or payloads to query for the recently added URLs.

A picture containing diagram

Description automatically generated

Image 4: the use of limits on the number of entries received from the AIP call.

There is the option to limit the amount of database entries that are retrieved from the query, this is possible by using the -L or –limit flag, the value of the flag must be an integer and its value must be between 1 and 1000 (max entries that can be fetched).

## Query URL information

To retrieve information about an URL, the URL must be supplied in the commend line (image 5), there is the possibility to use the URL ID if it is known instead of the URL (image 6).

In the case that a url will not be provided the user will be prompted with a message saying that a url must be provided for the search, or if the –url was used but not supplied (image 7).

Graphical user interface, text, application

Description automatically generated

Image 5: the use of the U and url flags, the two variations can be seen.

This kind of query uses the –url flag, it must include a valid url so the search will be successful, by valid we mean a url that exists in the records of URLhaus and not an empty string, empty strings will be handled, and a message will be shown to the user.

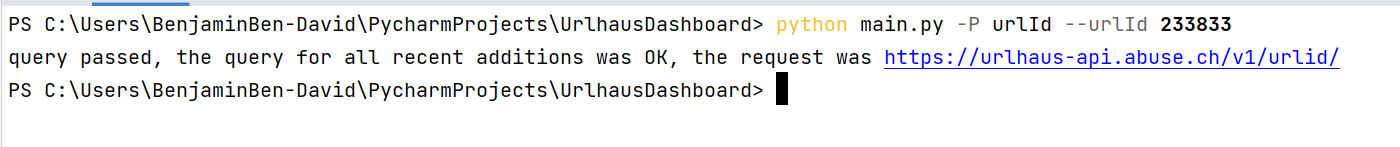


Image 6: the use of the urlId flag being passed with the matching URLs ID.

This kind of query uses the variable urlId as a parameter that will be passed to the -P flag, this will make a post request to the API searching for the URL ID that will be passed by the user in the –urlId flag that will follow.

A picture containing Word

Description automatically generated

Image 7: the user being notified that ether the –url flag was empty or not used at all.

## Query host information

To retrieve information about a host, the host’s name must be supplied in the command line (image 8).

Text

Description automatically generated with low confidence

Image 8: the use of the H and host flags, the two variations can be seen.

This kind of query will use the –host flag to search the URLhaus database, it will retrieve every entry that has the value passed in the –host flag as its value, the use of H or host as the value passed to the -P flag server the same purpose.

## Query payload information

To retrieve information about a payload (malware sample) that URLhaus has retrieved, there are two types of data types that can be passed to ID the payload that the user wants to search, the 1st is a MD5 hash, the 2nd is a SHA256, one of these data types must be givens to send a request to the API (image 9), if the user will not supply ether one of the options given to him a message saying that a hash is needed will be shown (image 10).

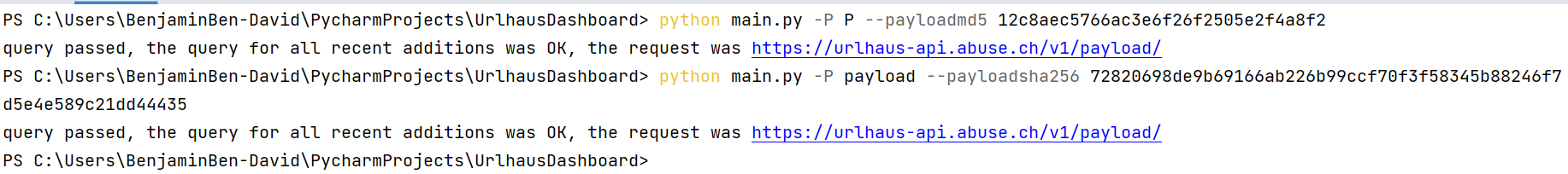


Image 9: the use of the P and payload flags, the two variations can be seen, as well the use of both MD5 hash and SHA256 for the search vector.

A picture containing text

Description automatically generated

Image 10: a message being shown to the user asking for a hash as a input for the search.

## Query tag information

To retrieve information about a tag, there are tags that are associated with each entry on the database, the user needs to enter a value to the –tag flag, this will filter only the entries that have the tag that was given by the user (image 11).

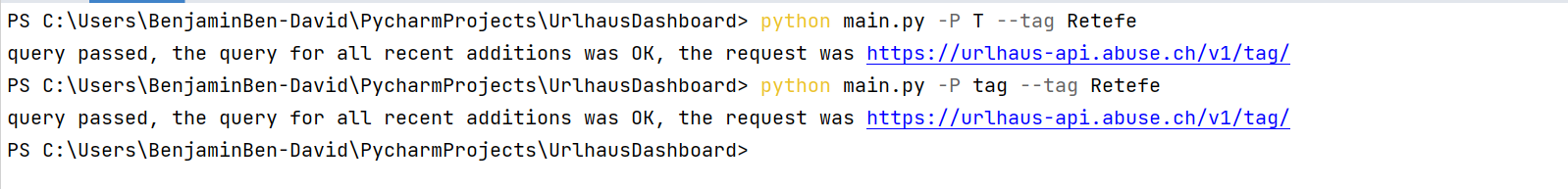


Image 11: the use of the P and payload flags, the two variations can be seen.

## Query signature information

URLhaus tries to identify the malware family of a payload (malware sample) served by malware URLs.

Unlink tags, the signature is something that the reporter of the malware URL cannot influence. To retrieve information using this method the user will need to pass the signature as value to the – signature flag (image 12).

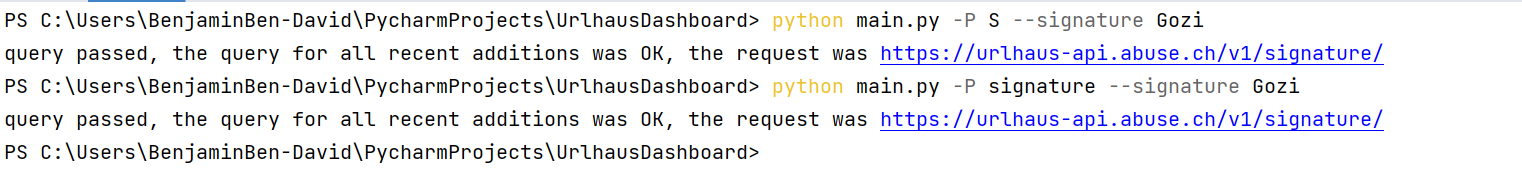


Image 12: the use of the P and payload flags, the two variations can be seen.

## Download malware

There is the ability to download a zip file that contains the wanted malware, the user must supply the SHA256 hash in the –sha256 flag for the get request to be successful, the zip containing the malware will be called payload.zip and will not be protected by any kind of password, please act with caution (image 13).

Graphical user interface, text, application

Description automatically generated

Image 13: the zip file downloaded and containing the wanted malware.

# Data manipulation and filtering

There are several options that we are offering the user to focus its search and filter specifically what the user needs, ether using the key to the value or using regular expressions in order to filter them out.

## Fetch

We gave the user the ability to mark the kyes that he wants to collect there value and the program will filter through all the response and will detect when it sees key that the user stated that he is interested in, then it will append it to a list of all the other matching values, at the end of the matching it will be sent back to the main program as data (image 14).

Graphical user interface, application

Description automatically generated

Image 14: the program in debug, showing that only the URLs were taken from the full API response.

## Output to file

We gave the user the ability to export the results that he received to a file, the types that are supported at the moment are, json, xlsx, CVS, html, xml, to choose the file type just enter the name of the file with the wanted extension attached to it for example, test.xml, check.json, etc. (image 15)

Graphical user interface, text, application

Description automatically generated

Image 15: the result being outputted as a json file the current active directory

## Regex filtering

We can filter all the values that are received by the API with regular expression as well, this comes as a combination of two flags the --fetch flag and the –regex (OR -R) flag, both are needed to create a regular expression filtering (image 16).

Graphical user interface, application

Description automatically generated

Image 16: the result of the use of regex and fetch to get a specific pattern on the URLs.