Phishing Blocker Project - Analytics

Release v1

Jun 28, 2020

Basic

Welcome :)

These are the documents for Phishing Blocker Project - Analytics.

You can get the source code of this website via GitHub.

CHAPTER 1

Library Reference

This is an auto-generate reference of Analytics.

You can make sense of Analytics how to work through these documents.

1.1 Analytics

class libs.**Analytics**(*config: str*)

_deep_analyze(*url: str*)
Analyze URL with PageView

Parameters **url** – URL that latest get via *requests*

Returns float of the-trust-score between 0 to 1

analyze(*data: dict*)
Do analysis from URL sent by message with databases

Parameters **data** – dict from message decoded

Returns dict to response

check_from_database(*url: str*, *host: str = None*)
Check URL whether existed in database

Parameters

- **url** – URL from request
- **url_hash** – URL hashed
- **host** – host from URL decoded

Returns trust_score or NoneType

gen_sample()
Generate PageView samples with trustlist

Returns

start(*port: int = 2020*)
Start web service

Parameters **port** – integer of port to listen online

Returns

stop()
Shutdown web service

Returns

update_blacklist_from_phishtank()
Update database for blacklist from PhishTank

Returns

1.2 Callback

class libs.callback.**WebServer**(*pbp_handle*)
Web service of API protocol

_server_response(*data: dict*)
Handle responses from web service

Parameters **data** – dict from message decoded

Returns dict to response

static listen(*port: int*)
Start listen on web services

Returns

server_response(*message: str*)
Check responses from web service

Parameters **message** – string of JSON format

Returns dict to response

1.3 Data

class libs.**Data**(*pbp_handle*)
To control MySQL for PBP

check_blacklist(*url: str*)
To check URL whether exists in blacklist

Parameters **url** – URL

Returns dict of URL and Mark-Date or NoneType

check_trust_domain(*domain: str*)
To check URL whether exists in trust_domain list

Parameters **domain** – domain

Returns string of UUID or NoneType

check_trustlist(*url: str*)
To check URL whether exists in trustlist

Parameters **url** – URL

Returns string of UUID or NoneType

check_warnlist(*url: str*)
To check URL whether exists in warnlist

Parameters **url** – URL

Returns dict of URL, similar URL and Mark-Date or NoneType

clean_result_cache()
Clean result caches

Returns True

find_page_by_view_signature(*signature: str*)
Search URL by view_signature in trustlist

Parameters **signature** – string hashed

Returns URL or NoneType

find_result_cache_by_url_hash(*url_hash: str*)
Search cache by url_hash in result_cache

Parameters **url_hash** – URL hashed

Returns float of the-trust-score or NoneType

get_urls_from_trustlist()
Fetch all URL in trustlist

Returns list of URL

get_view_narray_from_trustlist()
Fetch all target_view_narray in trustlist

Returns dict of URL and NumPy Array

mark_as_blacklist(*url: str*)
Mark URL to blacklist by Database

Parameters **url** – URL to mark

Returns True

mark_as_blacklist_mass(*urls: list*)
Mark URLs to blacklist by Database

Parameters **url** – URLs to mark

Returns True

mark_as_warnlist(*url: str*, *origin_url: str*)
Mark URL to warnlist by PageView

Parameters

- **url** – URL to mark

- **origin_url** – the URL similar to

Returns True

upload_result_cache(*url_hash: str*, *score: float*)
Upload the-trust-score to cache

Parameters

- **url_hash** – URL hashed
- **score** – float of the-trust-score

Returns

upload_view_sample(*url: str*, *view_signature: str*, *view_data: str*)
Upload ViewSample for PageView

Parameters

- **url** – URL of Sample
- **view_signature** – string hashed with view_data
- **view_data** – string of num array base64 encoded

Returns True

1.4 Initialize

class libs.initialize.**Initialize**(*pbp_handle*)

_Initialize__config_checker(*env: bool*)
Load and check settings from shell environment or config file

Returns

_Initialize__mysql_checker()
Check tables existed with the database

Returns

1.5 Tools

class libs.**Tools**

static check_ready()
Check status that service is ready or not

Returns bool of status

static error_report()
Report errors as message

Returns string

static get_time(*time_format: str = '%b %d %Y %H:%M:%S %Z'*)
Get datetime with format

Parameters **time_format** – string of format codes

Returns

static lists_separate(*lists: list*, *numbers: int*)
Split lists to average

Parameters

- **lists** – list you want to separate
- **numbers** – numbers in part you want

Returns

static logger(*error_msg*, *silent: bool = True*)
Journal or print error message

Returns

static set_ready(*status: bool*)
Set status whether service is ready or not

Parameters **status** – bool of status

Returns

1.6 Google Safe Browsing Client

class libs.survey.**GoogleSafeBrowsing**(*google_api_key: str*)
Google Safe Browsing Client https://safebrowsing.google.com/

get_database()
Get database from Google Safe Browsing

Returns dict

lookup(*urls: list*)
To check URLs from Google Safe Browsing

Parameters **urls** – list of URLs

Returns dict

1.7 OpenDNS PhishTank Client

class libs.survey.**PhishTank**(*username: str*, *api_key: str*)
OpenDNS PhishTank Client https://www.phishtank.com/

get_database()
Get database from PhishTank

Returns dict

lookup(*url: str*)
To check URLs from PhishTank

Parameters **url** – URL

Returns dict

1.8 View

class libs.survey.**View**(*pbp_handle*)

analyze(*target_url: str*)
Analyze URL

Parameters **target_url** – URL

Returns URLs similar to in trustlist

generate()
Generate samples

Returns

1.9 Browser

class libs.survey.page_view.browser.**BrowserRender**(*capture_browser: str*)
The main solution.

To render web page from QTWebEngine with blink2png, but we plan using Gecko/Servo to replace someday.

class libs.survey.page_view.browser.**BrowserAgent**(*capture_browser: str*)
As a backup solution.

To capture web page via Selenium with webdriver. The class will allow you to use your browser as the agent to take a screenshot from it.

1.10 Image

class libs.survey.page_view.image.**Image**(*pbp_handle*)
Handle images for PageView

capture(*url: str*)
Capture Web Page by URL

Parameters **url** – URL to capture

Returns string hashed and NumPy Array

rank(*target_num_array: str*)
To rank URL not registered if it same/similar to someone in trustlist.

Parameters **target_num_array** – NumPy Array

Returns URLs that similar to the target

signature(*hex_digest: str*)
Match PageView signature from database

Parameters **hex_digest** – string hashed

Returns URL or NoneType

class libs.survey.page_view.image.**WebCapture**(*config: dict*)
To take screenshot for PBP.

static _WebCapture__set_browser_simulation(*type_id: str*)
Set Browser Simulation by ID

Parameters **type_id** – Type ID

Returns class object

delete_page_image(*output_image: str = 'out.png'*)
To delete the image of the URL you provided

Parameters **output_image** – Output path (optional)

Returns bool

get_page_image(*target_url: str*, *output_image: str = 'out.png'*)
To get the image of the URL you provided

Parameters

- **target_url** – The target URL
- **output_image** – Output path (optional)

Returns bool

static image_compare(*img1: removed*, *img2: removed*)
To compare image using structural similarity index

Parameters

- **img1** – Image object
- **img2** – Image object

Returns float of the similar lever

static image_object(*path: str*)
Create NumPy Array

Parameters **path** – The Image Path

Returns NumPy Array

static image_object_from_b64(*b64_string: bytes*)
Import NumPy Array by base64

Parameters **b64_string** – base64 NumPy Array dumped

Returns NumPy Array

CHAPTER 2

Guide

The manual will lead you to install Analytics,

show how to connect Analytics ,and tell you the usage.

2.1 Installation

2.1.1 Database required

Analytics using MySQL or MariaDB as its data driver.

Install one of them, and create a database with any name you like, then import initialize.sql to the database.

Filling the information for connect to the database into *config.ini* as config.sample.ini did.

2.1.2 Selections

Production

In order to security reason, ought not to using without docker for decreasing danger on the host server.

Build and Install with Docker

- Clone from the source repository

  git clone https://github.com/star-inc/pbp-analytics.git
- Configure *config.ini*.
- Follow these commands:

```
sudo docker build -t pbpa .
sudo docker run –network=host –detach pbpa
```

Easy Install

Please register the API key of the public databases Analytics using.

The command will help you create and run Analytics.

```
sudo docker run \
    -e PBP_CFG=1 \
    -e PBP_MySQL_host=<Database Host> \
    -e PBP_MySQL_database=<Database Name > \
    -e PBP_MySQL_user=<Database Username> \
    -e PBP_MySQL_passwd=<Database Password> \
    -e PBP_SafeBrowsing_google_api_key=<Google API Token> \
    -e PBP_PhishTank_username=<PhishTank Username> \
    -e PBP_PhishTank_api_key=<PhishTank API Token> \
    -e PBP_WebCapture_capture_type=1 \
                                    —name=pbpa –network=host –detach starinc/pbp-analytics
```

Development

For improving and researching on the platform.

Requirement

```
Ubuntu >= 18.04
python == 3.7
pip >= 19.2
```

Installation

- Clone from the source repository

  ```
  git clone https://github.com/star-inc/pbp-analytics.git
  ```

- Configure *config.ini*.
- Follow these commands:

  ```
  python3.7 -m pip install requirements.txt
  python3.7 main.py
  ```

Enjoy for using and developing.

2.2 Callback Status Code

- *200* Success With *url* And *trust_score* Tag
- *201* Success With *msg* Tag
- *202* Success Without Any Response
- *400* No *version* Tag Found From Request
- *401* Request Decode Error
- *403 requests* Got Error
- *404* URL Requested Not Found
- *405* URL Requested Was Not HTML
- *500* Empty Response

Correct Request:

{

“version”:1,

“url”: “https://example.org/”

}

CHAPTER 3

Indices and tables

- genindex
- search

Index

Symbols

A

B

C

D

E

F

G

I

L