Thug Distributed Documentation

Release 0.0.1b

Akshit Agarwal, Angelo Dell'Aera, Sebastian Poeplau

September 24, 2013

CONTENTS

ThugD stands for **Thug Distributed Task Queuing Project**. It is developed under The Honeynet Project organization as a GSoC Project. It is a Distributed Version of the existing Thug Project.

ThugD is developed using **Celery** [1] for Distributing tasks(URLs) among the workers. While RabbitMQ [2] and Redis [3] works as the brokers in it. It also uses Team Cymru Community Services [4] and PyDNS [5] to query its services.

Please refer to Project Slot Page for more details. Stay tuned to Project Weekly Blog.

[1] Celery: Distributed Task Queue

[2] RabbitMQ is used as the Message Broker in Celery.

[3] Redis is used as the Backend Broker in Celery. It is preferred then default AMQP backend broker as it don't create individual queues while returning results to server.

[4] Team Cymru Community Service is used for fetching country codes from IP address of the users.

[5] PyDNS is used for DNS queries over Team Cymru's IP-to-ASN service.

CHAPTER

ONE

INTRODUCTION TO THUGD

ThugD as the name suggests stands for **Thug Distributed** adding the Distributed functionality to the Thug project.

ThugD's Architecture comprises of:

- **Server** : Containing bulk of URLs (fed by spamtraps).
- **Workers** : Thug Instances (running across the globe).

Till now Thug was working like a stand-alone tool and didn't provided any way to distribute URL analysis tasks to different workers. For the same reason it is neither able to analyze difference in attacks to users according to their geolocation (unless it is provided a set of differently geolocated proxies to use obviously).

But now with Thug Distributed we are be able to solve the problem. Now we have a Centralized Server which will be connected to all the Thug instances running across the globe and will distribute URLs (according to geolocation analysis requirements). After that the clients will consume the tasks distributed by centralized server, process them and return back the results to the server using HpFeeds.

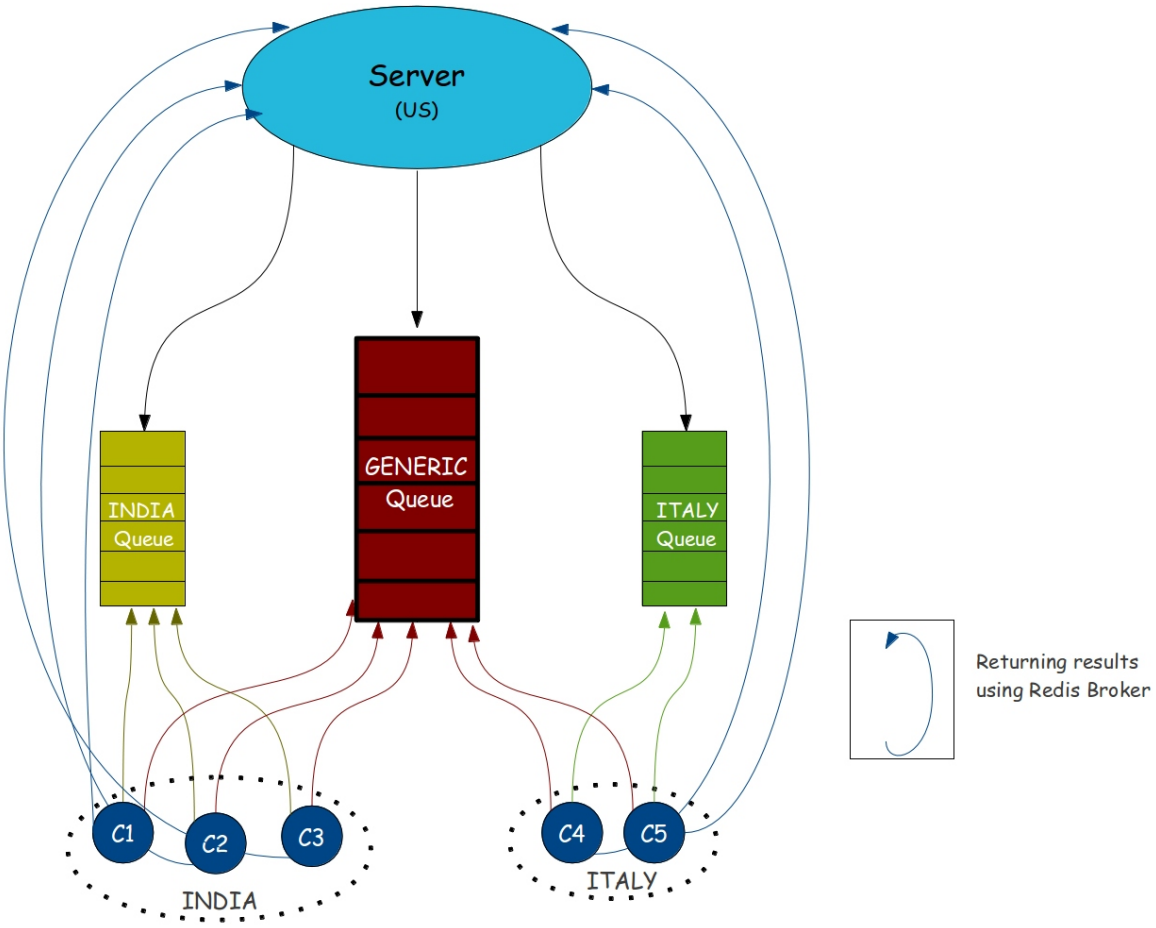
1.1 Architecture

1.2 Implementation

- ThugD maintains a Centralized Server which is fed up with bunch of URLs(collected from Spamtraps).
- Then it distributes these URLs into 2 different types of Queues(Generic & Geolocation based) according to the geolocation analysis requirements.
- While on the other side whenever a Worker(client) starts the Thug Instance anywhere across the globe it will automatically be connected to both queues(Generic and its Country Queue like: IN(India), IT(Italy)).
- Then if the URLs are present in the any of the connected queue they will be automatically fetched by the Thug Instance. It will process them and return back the results to the server.

1.3 Optimizations

- At a time **4 URLs** will be processed in PARALLEL using gevent at every Worker.
- According to Worker's System Performance some URL's will be automatically **prefetched** for later analysis, so that workers with better performance do more work and return results fastly.


Server
(US)
INDIA
Queue
GENERIC
Queue
ITALY
Queue
C1
C2
C3
INDIA
C4
C5
ITALY
Returning results
using Redis Broker

CHAPTER

TWO

LIBRARIES USED AND INSTALL

2.1 Libraries Used

- **Celery** : Asynchronous Task Queue based on Distributed message passing, which is perfect choice for our implementation.
- **RabbitMQ** : Its the Message Broker. Handles queues in Celery with capability to handle millions of tasks efficiently.
- **Flower** : Monitoring & Management tool of Celery which helps us to monitor & manage all our Thug Clients working across the globe.
- **DnsPython** : Used for DNS Querying the Team Cymru Service of IP to ASN Mapping.
- **LibRabbitMQ** : This module is installed to use optimized client written in C.

2.2 Installation

1. RabbitMQ Server

```
$ sudo apt-get install rabbitmq-server
```

2. Celery

```
$ pip install celery
```

3. Flower

```
$ pip install flower
```

4. DnsPython

```
$ pip install dnspython
```

5. LibRabbitMQ: Optimizing Worker

```
$ pip install librabbitmq
```

CHAPTER

THREE

USAGE

3.1 Server

Options provided by ThugD

```
~/thugd/src $ python run_tasks -h

Synopsis:
    ThugD: Distributed Pure Python Honeyclient Implementation

    Usage: python run_tasks.py [ thug-options ] url

    Optional Arguments:
    -h, --help                      show this help message and exit

    URL Options:
    -U  [ ...], --url  [ ...]       Enter Single/Multiple URL's to Analyze
    -uf , --url-file                File containing bunch of URL's(1 per line)

    Thug Distributed Options:
    -ia, --include-agent            Display Thug Version
    -qu  [ ...], --queue  [ ...]    Specify Queue/Queues to route URL's
                                   (*Single Queue: URL's will be routed to specified Queue,
                                    *Multiple Queues: URL's will be routed to ALL specified Queues)
    -qf , --queue-file              Specify File name containing Queue names(1 per line)

    Thug Options:
    -V, --version                   Display Thug Version
    -u , --useragent                Select a user agent(see below for values, default: winxpie60)
    -e , --events                   Enable comma-separated specified DOM events handling
    -w , --delay                    Set a maximum setTimeout/setInterval delay value (in milliseconds
    -n , --logdir                   Set the log output directory
    -o , --output                   Log to a specified file
    -r , --referer                  Specify a referer
    -p , --proxy                    Specify a proxy (see below for format and supported schemes)
    -l, --local                     Analyze a locally saved page
    -x, --local-nofetch             Analyze a locally saved page and prevent remotecontent fetching
    -v, --verbose                   Enable verbose mode
    -d, --debug                     Enable debug mode
    -q, --quiet                     Disable console logging
    -m, --no-cache                  Disable local web cache
    -a, --ast-debug                 Enable AST debug mode (requires debug mode)
    -t , --threshold                Maximum pages to fetch
```

```
    -E, --extensive                 Extensive fetch of linked pages
    -T , --timeout                  Timeout in minutes

    Plugins:
    -A , --adobepdf                 Specify the Adobe Acrobat Reader version (default: 9.1.0)
    -P, --no-adobepdf               Disable Adobe Acrobat Reader Plugin
    -S , --shockwave                Specify the Shockwave Flash version (default: 10.0.64.0)
    -R, --no-shockwave              Disable Shockwave Flash Plugin
    -J , --javaplugin               Specify the Java Plugin version (default: 1.6.0.32)
    -K, --no-javaplugin             Disable Java Plugin

    Classifiers:
    -Q , --urlclassifier            Specify a list of additional (comma separated) URL classifier rul
    -W , --jsclassifier             Specify a list of additional (comma separated) JS classifier rule

    Available User-Agents:
    winxpie60               Internet Explorer 6.0   (Windows XP)
    winxpie61               Internet Explorer 6.1   (Windows XP)
    winxpie70               Internet Explorer 7.0   (Windows XP)
    winxpie80               Internet Explorer 8.0   (Windows XP)
    winxpchrome20           Chrome 20.0.1132.47     (Windows XP)
    winxpfirefox12          Firefox 12.0            (Windows XP)
    winxpsafari5            Safari 5.1.7            (Windows XP)
    win2kie60               Internet Explorer 6.0   (Windows 2000)
    win2kie80               Internet Explorer 8.0   (Windows 2000)
    win7ie80                Internet Explorer 8.0   (Windows 7)
    win7ie90                Internet Explorer 9.0   (Windows 7)
    win7chrome20            Chrome 20.0.1132.47     (Windows 7)
    win7firefox3            Firefox 3.6.13          (Windows 7)
    win7safari5             Safari 5.1.7            (Windows 7)
    osx10safari5            Safari 5.1.1            (MacOS X 10.7.2)
    osx10chrome19           Chrome 19.0.1084.54     (MacOS X 10.7.4)
    galaxy2chrome18         Chrome 18.0.1025.166    (Samsung Galaxy S II,Android 4.0.3)
    galaxy2chrome25         Chrome 25.0.1364.123    (Samsung Galaxy S II,Android 4.0.3)
    linuxchrome26           Chrome 26.0.1410.19     (Linux)
    linuxfirefox19          Firefox 19.0            (Linux)
```

3.1.1 Different Methods of Distributing URL's among workers:

- **Distributing Single URL with Default Queue (generic)**: By it a single URL will be put up in the *generic* queue, from which worker can fetch the URL and after processing it will return back the results to server.

```
~/thugd/src$ python run_tasks.py -U http://www.google.com
```

- **Single URL with Single Specified Queue(India)**: In it a single URL will be put up in the Specified Queue i.e. *India* (geolocation based queue) and not in generic queue. Then whenever a worker from country India connects it will automatically fetch the URL from it and do further processing on it.

```
~/thugd/src$ python run_tasks.py -qu IN -U http://www.google.com
```

- **Single URL with Multiple Specified Queues(India, Italy, China, US)**: In it a single URL will be putted up in multiple specified queues. Therefore a copy of a single URL will be put up in multiple queues and whenever workers corresponding to that queues will be attached they will process the URLs and return back the results.

```
~/thugd/src$ python run_tasks.py -qu IN IT CN US -U http://www.google.com
```

- **Multiple URL's(Google, Twitter, Mozilla) with Single Specified Queue(India)**: This is a simple case where

multiple URL's are put up in a single specified queue like India in this case. So whenever workers corresponding to India will connect URLs will get processed by them.

```
~/thugd/src$ python run_tasks.py -qu IN -U http://www.google.com http://www.twitter.com http://www.mo
```

- **Multiple URL's(Google, Twitter, Mozilla) with Multiple Specified Queues(India, Italy, China, US)**: This is the advanced distribution as here multiple URL's will be distributed among all the specified queues. Therefore according to this case Google, Twitter, Mozilla URL will be put up in all India, Italy, China, US queues.

```
~/thugd/src$ python run_tasks.py -qu IN IT CN US -U http://www.google.com http://www.twitter.com http
```

- **Multiple URL's from file(urls.txt) with Multiple Specified Queues from file(queues.txt)**: This feature was added for reducing pain of specifying all URL's and queues manually. By this URL's and queues name would be fetched from the files specified and then every URL will be put up in every queue present in the file.

```
~/thugd/src$ python run_tasks.py -qf queues.txt -uf urls.txt
```

- **Running Thug with following prioritized Agents: Multiple URL's from file(urls.txt) with Multiple Specified Queues from file(queues.txt)**: In it every URL will be put up in every Queue with all the agent's specified below one at a time, so that we can check the difference in attacks to different browsers. Therefore for a single URL 18 URL's will be added to a queue because there are 18 different agents specified.

```
~/thugd/src$ python run_tasks.py -qf queues.txt -uf urls.txt -ia
```

Agents Priority

```
win7chrome20
win7firefox3
win7ie90
win7safazi5
osx10chrome19
osx10safari5
linuxchrome26
linuxfirefox19
win7ie80
winxpchrome20
winxpfirefox12
winxpie80
winxpsafari5
winxpie70
win2kie80
win2kie60
galaxy2chrome25
galaxy2chrome18
```

3.1.2 Run Flower(optional)

```
$ flower
```

Open http://localhost:5555/ to access the tool.

3.1.3 Checking Active Queues

```
$ sudo rabbitmqctl list_queues
```

3.2 Worker

Workers are runned to help the **Thug** project to analyze the attacks on Clients. Please run the workers on your system as Server is running up there in US, so that we can analyze the attacks on clients and can secure users from these attacks.

Its a contribution to the **Thug** Project, so be the part of the Thug Project by running worker on your system.

Move inside the **src** folder of thugd

Single Worker

```
~/thugd/src$ celery worker -A ThugD.main_server.thugd -l info -n w1
```

Multiple Workers

```
~/thugd/src$ celery multi start w1 w2 w3 -A ThugD.main_server.thugd -l info
```

CHAPTER

FOUR

DOWNLOAD

Working Project with **Thug Honeyclient** files integrated into it.

Download any one of them and follow the *Usage* commands for testing it.

4.1 ThugD Tar Ball

`thugd.tar.gz`

4.2 ThugD Zip

`thugd.zip`

CHAPTER

FIVE

CODE

Celery Configuration:

Main Server:

Finding Geolocation:

Running tasks

Calling ThugAPI functions

CHAPTER

SIX

INDICES AND TABLES

- *search*