Tortazo Documentation

Release 1.1

Adastra

September 02, 2014

Contents

Tortazo is a Audit and Attack framework for TOR's deep web. Supports the integration with other frameworks and tools like Nessus, W3AF, Nikto, Metasploit and others, but focusing in TOR relays and hidden services as targets.
Contents:

CHAPTER 1

About Tortazo - Gentle Introduction.

1.1 What?

Tortazo is a tool written in Python to perform pentesting activities through the TOR deep web. Allows the integration with other well known frameworks available in the market and any python developer could write plugins to execute attacks against hidden services and relays in TOR.

1.2 How?

Tortazo is written in Python language using a lot of libraries to perform pentesting activities. This project is almost entirely "I+D" because there's few tools publicly available to audit the hidden services or relays in TOR. The researching and innovative ideas are much appreciated because, there's a lot of work and things to implement in Tortazo.

1.3 Why?

The anonymous networks are the favorite "tool" of criminals and this is a shame because networks like TOR, I2P or Freenet weren't designed to protect killers, narcos, pedofiles and that kind of people. The initial idea of this project, is develop a tool to compromise the identity of that kind people. ¿How? A lot of them, usually are not aware of the vulnerabilities included in their boxes. A lot of them, just exposes hidden services with the "defaults" because they're not security professionals and usually they are just end-users with basic knowledge about computing. A lot of them just starts TOR and exposes their machines as relays in the TOR network or creates websites as TOR hidden services without any security consideration. This is a good "starting point" to try to expose them and the purpose of Tortazo is to include a lot of features to find that kind of flaws and bring a bridge between the TOR network and the "good" hackers.

1.4 When?

This project was initiated in early 2014 and actually is being developed just by me (@jdaanial aka. Adastra). Initially was a simple prototipe to test the features included in Stem library for TOR. (https://stem.torproject.org/) Stem is a impresive python library which uses the TOR controller protocol to manage a TOR instance. However, also includes utilities to querying the TOR authoritative directories and download the descriptors with the information about the relays running in TOR. On other hand, there's a lot of libraries and tools to perform pentesting activities which will be perfect against some vulnerable web applications in the deep web. Tortazo allows the integration from some of the most known of this tools and frameworks

1.5 Who?

I'm a software developer and security enthusiast. Just a guy who spent his time playing with libraries, programming languages, tools, security techniques, network protocols and anything related with computers :-) What I like:

- I like to read almost about everything.
- I like the free speech.
- I like free software.
- I like the hacker philosophy.
- I like the hacktivism.
- I like to write code.
- I like to find and fix bugs in code.
- I like to improve code.
- I like the reverse engineering.
- I like to talk with people about things that matter. (obviously, this exclude football, politics, tv shows and other bullshit).
- I like the freedom. Everyone should be free to do anything, but without affecting the freedom of others.

What I dislike:

- I dislike the bugs.
- I dislike the awful code.
- I dislike the people lazy.
- I dislike the mediocrity.
- I dislike the authoritarianism.
- I dislike some rock stars "selling smoke" in conferences and other events. We need more mentors and less security rock stars. Sadly, this is the worst thing that I've found in the infosec environment in my country and other places.

1.6 Contact?

Sure, writes an email to: debiadastra [at] gmail.com I'll reply as soon as possible. Also, you can follow me in Twitter. @jdaanial

1.7 Legal Warning!!

I've developed this tool to improve my knowledge about TOR and Python. I'm a security enthusiast and I hope that you use this tool with responsibility, but if that is not the case, I'm not responsible for the use (or misuse) of this tool. If you found vulnerabilities or any kind of flaw in any non-malicious exit node of TOR, please, send the report to the admin of the relay, in this way you can contribute to build solid and secure TOR circuits for all of us.

OK, are you ready? Go and read about Tortazo and start to use it

CHAPTER 2

Getting Started

Stem is a powerful library written in Python to perform various operations against TOR Clients and Directory Authorities. The information gathered using Stem could be very useful to an attacker to gather information about the relays available in the TOR network. Tortazo is an open source project to gather information about ExitNodes in the TOR Network, perform bruteforce attacks against services like FTP or SSH and create a Botnet with the compromised ExitNodes over SSH. Tortazo includes a lot of features in the plugins form to perform pentesting activities against TOR relays and hidden services in the deep web. In this documentation, you'll see in detail all the features included in Tortazo. The main objetive of this project is establish a bridge between the TOR deep web and the Python hackers. Let's execute python scripts against TOR!

2.1 Instalation and Dependencies

To use Tortazo, you can use the latest stable release located in "bin" directory. However, if you want to use the development version located in the GIT repository you'll need the following dependencies:

- Python 2.6 or higher: http://python.org
- Twisted: https://twistedmatrix.com/trac/
- Paramiko: https://github.com/paramiko/paramiko
- Python-Nmap http://xael.org/norman/python/python-nmap/
- Python-shodan: https://github.com/achillean/shodan-python
- Stem: https://stem.torproject.org/
- TxTorCon: https://txtorcon.readthedocs.org
- Plumbum: https://pypi.python.org/pypi/plumbum
- Fabric: http://docs.fabfile.org/en/1.8/
- Requests: https://pypi.python.org/pypi/requests
- IPython: http://ipython.org/
- PyNessus-rest: https://github.com/Adastra-thw/pynessus-rest
- PySNMP: http://pysnmp.sourceforge.net/
- IRLib: https://github.com/gr33ndata/irlib
- Jinja2: http://jinja.pocoo.org/docs/intro/
- BeautifulSoup: http://www.crummy.com/software/BeautifulSoup/

- Scrapy: http://scrapy.org/
- PyFiglet: https://github.com/pwaller/pyfiglet

2.2 Other Dependencies

Among the Python libraries needed to use Tortazo, you'll need some tools to use plugins or some execution modes in Tortazo.

- Nmap: Mandatory dependency in Gather Information mode. http://nmap.org/
- Nessus: Mandatory dependency to use Nessus from Tortazo. (NessusPlugin). http://www.tenable.com/products/nessus/
- Metasploit Framework: Mandatory dependency to use Metasploit Framework from Tortazo (MetasploitPlugin). http://www.metasploit.com/
- Nikto: Mandatory dependency to use Nikto from Tortazo (NiktoPlugin). http://www.cirt.net/Nikto2/
- Nexpose: Mandatory dependency to use Nexpose from Tortazo (NexposePlugin). https://www.rapid7.com/products/nexpose/

2.3 Usage of Tortazo: Execution Modes

There are four execution modes in the current version of Tortazo, each of this allows gathering information from the TOR network and performing attacks against relays and hidden services. The execution modes are the following.

2.3.1 Gathering Information

This is the most basic execution mode, which will download the descriptors from the latest consensus generated by the TOR directory authorities and then allows applying some filters on the information downloaded. Finally, launches an scan with Nmap against the data filtered, identifying open ports and a lot of details about the target, you can use every option included in Nmap scanner, including the Scripting Engine (NSE). The results will be stored in the local database of Tortazo and if you runs Tortazo multiple times in this mode, more information get stored in database. The larger samples of data have more chances of getting positive results against any of the relays registered, so it's recommended to run Tortazo in this mode multiple times. Read more about gather information in Tortazo *Mode Gather Information.*

2.3.2 Botnet Mode

This mode is used to execute commands over a set of SSH servers compromised using the "bruter" plugin. As you can see in *plugins-management-label* bruter plugin is used to execute dictionary attacks against multiple services in relays or hidden services. If the dictionary attack against a SSH server is successful, the plugin writes the details of the compromised server in the file "<TORTAZO_DIR>/tortazo_botnet.bot". In this mode, Tortazo will read that file to create the bots in the context of the botnet. You can run parallel commands against the entirely botnet or exclude bots to run the commands just over some machines. Read more about Botnet mode in Tortazo *Botnet mode in Tortazo.*

2.3.3 Database Mode

If you have enough information in your database, you can use it to perform direct attacks using some of the available plugins in Tortazo. In this mode, there are no connections to the TOR directory authorities to gather information about

the relays that conforms the network, instead, Tortazo will use the information stored in database. Read more about Database mode in Tortazo *Database mode in Tortazo*

2.4 Plugins management

The plugins in Tortazo are the best way to integrate external routines written in Python directly in the framework, allowing to any Python developer write his own tools to perform audits against hidden services and TOR relays. There are various plugins already developed which integrates tools like Nessus, W3AF, Metasploit Framework, among others and custom plugins to perform pentesting activities. Read more about the development and usage of plugins in Tortazo *plugins-management-label*

2.5 Repository Mode

In this mode, Tortazo will try to generate ONION addresses and then tests if the generated addresses point to a hidden service in the deep web. Every onion address is composed by 16 characters and the valid chars are the full alphabet in lowercase and the digits between 2 and 7, as you can imagine, the amount of ONION addresses that could be generated is *VERY, VERY HUGE* and this is why the repository execution mode will not finish in few hours, could take days or even weeks generate and test every "possible" onion address. Actually, the execution of this mode, depends *A LOT* of the information that you have about of an address and the processing capacity of your machine. If you represents a government, maybe this kind of limitations related to processing capacity are less severe compared with the restrictions of any natural person. However, please read more about the development and usage of plugins in Tortazo *repository-mode-label*

CHAPTER 3

Mode Gather Information.

3.1 Gather information about exit relays in TOR

This is the simplest mode of execution in Tortazo. In this mode, Tortazo will perform an Nmap scan and the results for every exit node in the Directory Authorities or in the local TOR instance will be stored in the local database used by Tortazo. Below you'll see various switches to explain how to use the script Tortazo.py. On other hand, is much, much faster use the local descriptors instead of connect with the directory authorities (directly or using mirrors, both connection types are very slow) also depending on the number of users connected, sometimes the servers are busy and the connections with the directory authorities will fail or will be unstable. Anyways, you can use any method to gather information, but the fact is that the directory authorities have a lot of information about new relays available and the local descriptors have a short set of exit nodes. So, you should choose wisely your weapon!

Note: KEEP IN MIND:

- In the new versions of the TOR Client, by default the microdescriptors will be downloaded to compose the circuits. This is good because the circuit construction is much faster, but the information about the relays is very limited. So, if you plan to gather information with Tortazo using the TOR Client, you need to set the "UseMicrodescriptors" switch to "0" in your torrc file.
- If you want to use the TOR Client to gather information, you'll need to open the Control Port to use the Stem Controller (and set a password for security issues) to connect with the local instance and get the server descriptors. Please, check your torrc file.
- The connections with the TOR Authorities could be very slow and sometimes unstable.
- You can use all the features included in Nmap, NSE Scripts even, just by using the switch "-a/–scan-arguments"
- If you want to use Shodan, you'll need a valid developer key. That value should be written into a file in a single line and use the switches "–use-shodan" and "–shodan-key"
- In botnet mode, Tortazo will not perform connections against TOR (neither local nor remote)

3.1.1 Gather information examples.

Ok, now lets see some examples about the use of Tortazo in this execution mode.

Show the available Options

Shows the help banner of Tortazo:

```
python Tortazo.py -h
```

Connecting to the Mirror servers of TOR

- Connect with the TOR Authorities using the mirrors servers (-d / –use-mirrors).
- Enable the "verbose" mode (-v / –verbose).
- Scan the exit nodes which operating system is Windows (-m / –mode windows):

```
python Tortazo.py -d -v -m windows
python Tortazo.py --use-mirrors --verbose --mode windows
```

Connecting to the TOR servers (authorities)

- Connect to the TOR Authorities directly.
- Enable the "verbose" mode (-v / –verbose)
- Scan the exit nodes which operative system is Linux (-m / –mode linux):

```
python Tortazo.py -v -m linux
python Tortazo.py --verbose --mode linux
```

Specify the number of relays to fetch from the descriptors downloaded

- Connect with the TOR Authorities directly.
- Enable the "verbose" mode (-v / –verbose).
- Scan the exit nodes which operative system is Linux (-m / –mode linux)
- Fetch the first 30 nodes from the list of exit nodes found (this value by default is very short: 10):

```
python Tortazo.py -n 30 -v -m linux
python Tortazo.py --servers-to-attack 30 --verbose --mode linux
```

Custom Nmap scan

- Connect with the TOR Authorities directly.
- Enable the "verbose" mode (-v / –verbose)
- Scan the exit nodes which operative system is Linux (-m / –mode linux)
- Fetch the first 30 nodes from the list of exit nodes found
- Performs the Nmap scan with the specified options "-sSV -A -n":

```
python Tortazo.py -n 30 -v -m linux -a "-sSV -A -n"
python Tortazo.py --servers-to-attack 30 --verbose --mode linux --scan-arguments "-sSV -A -n"
```

Connect with an Local instance of TOR.

- Connect to the Local instance of TOR and use the exit nodes stored in the local descriptors (-c / –use-circuit-nodes)
- Enable the "verbose" mode (-v / –verbose)
- Scan the exit nodes which operative system is Linux (-m / –mode linux):

```
python Tortazo.py -v -m linux -c
python Tortazo.py --verbose --mode linux --use-circuit-nodes
```

Specify an fingerprint to filter

- Connect with the TOR Authorities directly.
- Enable the "verbose" mode (-v / –verbose)
- Scan the exit nodes which operative system is Linux (-m / –mode linux)

- Fetch the first 30 nodes from the list of exit nodes found
- Perform the Nmap scan with the specified options "-sSV -A -n"
- Filter by FingerPrint (-e / –exit-node-fingerprint):

```
python Tortazo.py -n 30 -v -m linux -a "-sSV -A -n" -e FFAC0F4C85052F696EBB9517DD6E2E8B830835DD
python Tortazo.py --servers-to-attack 30 --verbose --mode linux --scan-arguments "-sSV -A -n" --
```

Using Shodan to Gather information about the relays found

- Connect with the TOR Authorities directly.
- Enable the "verbose" mode (-v / –verbose)
- Scan the exit nodes which operative system is Linux (-m / –mode linux)
- Fetch the first 30 nodes from the list of exit nodes found
- Performs an Nmap scan with the specified options "-sSV -A -n"
- Use Shodan (-s / –use-shodan) with the specified developer key (-k / –shodan-key). The key must be stored in a text file in a single line:

```
python Tortazo.py -n 30 -v -m linux -a "-sSV -A -n" -s -k /home/developer/shodanKeyFile
python Tortazo.py --servers-to-attack 30 --verbose --mode linux --scan-arguments "-sSV -A -n" --
```

CHAPTER 4

Database mode in Tortazo

Tortazo uses an internal SQLite database to store the TOR relays scanned with Nmap and hidden services discovered by the onion repository mode. The switch "-D / –use-database" allows to use the database records instead to perform a scan. This switch is very useful to use with some plugins which requires targets to attack or enumerate.

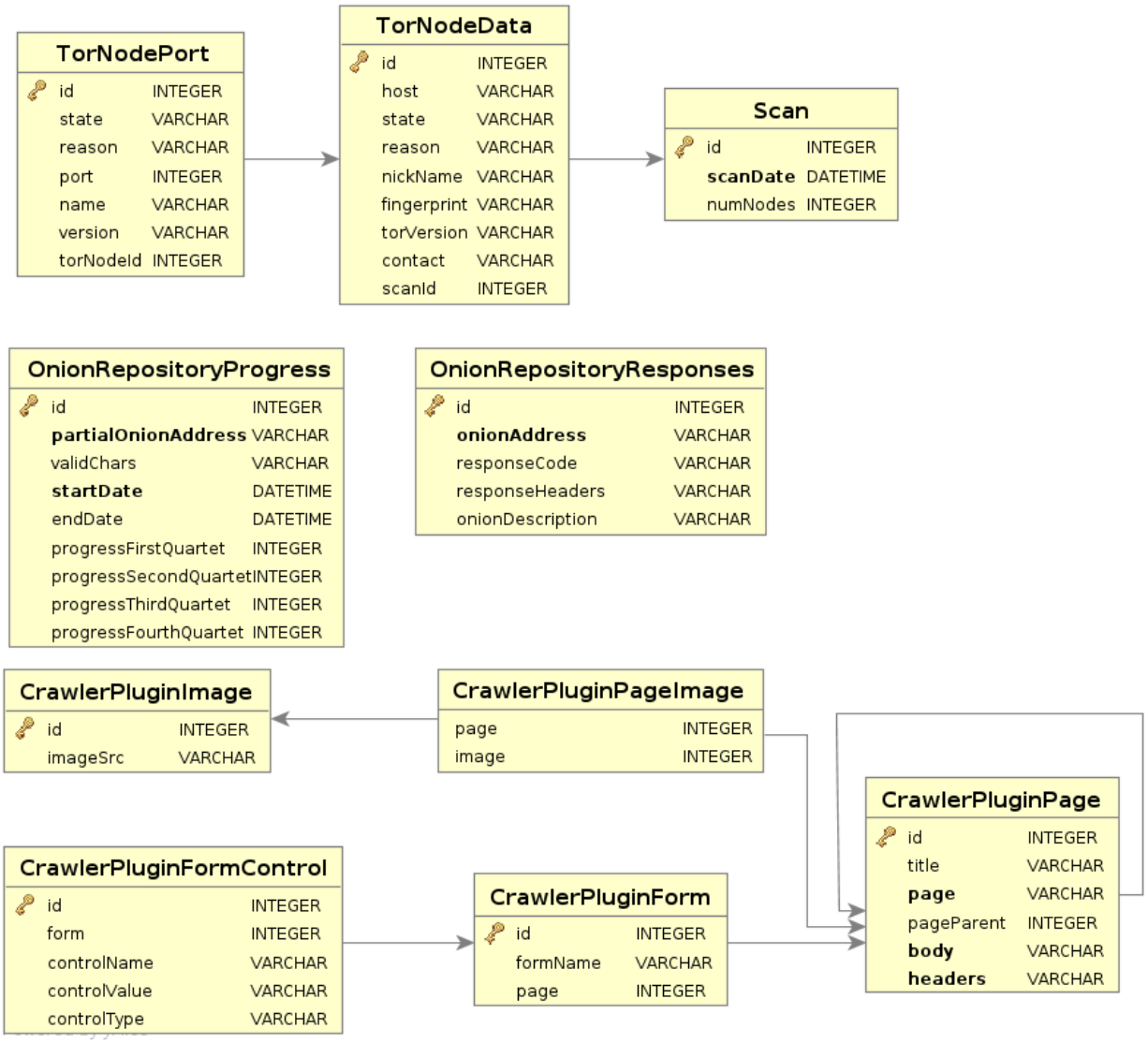
The core tables used by Tortazo in the version 1.1 are:

- Scan: Stores the date and number of relays found in every scan performed by Nmap.
- TorNodeData: Stores the information of relays found and the state of every one.
- TorNodePort: Stores the information of port states (closed, filtered or opened).
- OnionRepositoryResponses: Stores the responses from the HTTP connections performed against the hidden services in Onion Repository.
- OnionRepositoryProgress: Stores the progress for the Incremental search in the Onion Repository mode.

On other hand, the Crawler plugin stores the responses directly in the database, using the following structure

- CrawlerPluginPlage: Stores the pages and HTTP responses.
- CrawlerPluginImage: Stores the images found in the crawling process.
- CrawlerPluginPageImage: Stores the relation between the images found and the pages.
- CrawlerPluginForm: Stores the basic information about the forms found in the pages crawled by the plugin.
- CrawlerPluginFormControl: Stores the controls which are included in every form found.

The database schema in this mode is the following.


TorNodePort
id INTEGER
state VARCHAR
reason VARCHAR
port INTEGER
name VARCHAR
version VARCHAR
torNodeId INTEGER
TorNodeData
id INTEGER
host VARCHAR
state VARCHAR
reason VARCHAR
nickName VARCHAR
fingerprint VARCHAR
torVersion VARCHAR
contact VARCHAR
scanId INTEGER
Scan
id INTEGER
scanDate DATETIME
numNodes INTEGER
OnionRepositoryProgress
id INTEGER
partialOnionAddress VARCHAR
validChars VARCHAR
startDate DATETIME
endDate DATETIME
progressFirstQuartet INTEGER
progressSecondQuartet INTEGER
progressThirdQuartet INTEGER
progressFourthQuartet INTEGER
OnionRepositoryResponses
id INTEGER
onionAddress VARCHAR
responseCode VARCHAR
responseHeaders VARCHAR
onionDescription VARCHAR
CrawlerPluginImage
id INTEGER
imageSrc VARCHAR
CrawlerPluginPageImage
page INTEGER
image INTEGER
CrawlerPluginPage
id INTEGER
title VARCHAR
page VARCHAR
pageParent INTEGER
body VARCHAR
headers VARCHAR
CrawlerPluginFormControl
id INTEGER
form INTEGER
controlName VARCHAR
controlValue VARCHAR
controlType VARCHAR
CrawlerPluginForm
id INTEGER
formName VARCHAR
page INTEGER

CHAPTER 5

Botnet mode in Tortazo.

The botnet mode is specified with the option -z/–zombie and depends on the tortazo_botnet.bot file. In this mode, Tortazo will read that file and then, tries to perform SSH Connections using the hosts and credentials defined in that file. Every line in "tortazo_botnet.bot" have the next format:

```
host:user:passwd:port:nickname
```

The library "Fabric" is used to connect with the SSH servers and execute commands across a set of SSH Servers. When you specify the switch -z/–zombie option, Tortazo will not connect with the TOR authoritative directories, just will read the tortazo_botnet.bot file and then will try to open a shell on the specified bot (-o/–open-shell switch) or runs a command (-r/–run-command switch). In this mode, you must specify the nicknames that will be excluded from the botnet (comma separated) or the keyword "all" to include all bots from tortazo_botnet.bot file.

5.1 Botnet mode examples.

Using this mode is very simple, but the file located in <TORTAZO_DIR>/tortazo_botnet.bot must have a list of bots with the connection details of every bot for each line.

Execute commands across the entirely botnet Runs the commands: "id; uname -a; uptime":

```
python Tortazo.py -v -z all -r "id; uname -a; uptime; w"
python Tortazo.py --verbose --threads 10 --zombie-mode all --run-command "id; uname -a; uptime; w"
```

Open a shell in the specified bot Using the shell identifier to open a new console in the specified host.:

```
python Tortazo.py -v -z all -o
python Tortazo.py --verbose --zombie-mode all --open-shell
```

Note: KEEP IN MIND:

Obviously, the credentials in the tortazo_botnet.bot file should be valid for every host registered. If the credentials are not valid, "Fabric" will resolve the authentication method (password or public key) and will require that you enter the password or passphase.

CHAPTER 6

Supported Switches in Tortazo.

The following list is a summary of the core switches supported by Tortazo v1.1 and their usage.

6.1 Simple Switches.

The following is a list of single switches which doesn't receive any value. These switches allows to activate features in Tortazo.

6.1.1 Common Switches for all modes

- **-v / –verbose**: Activates the debug mode. Shows debug, info and error messages. It's very useful to see a full trace of actions performed by Tortazo and is recommended to use. However, in some cases this option shows many traces, for example, the plugin "BruterPlugin" uses Paramiko library to execute Brute Force attacks against relays and hidden services with an SSH Server up and running and Paramiko shows a very detailed information about every connection when the debug mode is active.
- **-U / –use-localinstance**: Tortazo can start a new instance of TOR automatically using the switch "-T / –tor-localinstance". Use the switch "-U / –use-localinstance" if you want to use the socks proxy and other settings defined in the instance started by Tortazo.

6.1.2 Switches for Gathering Information

- **-b / –brute**: Deprecated in v1.1. Replaced by the plugin "BruterPlugin". Actually this switch is not supported
- **-d / –use-mirrors**: By default, Tortazo uses the authoritative directories of TOR and with this option, Tortazo will perform a connection with the mirrors of the authoritative directories to get the last consensus available.
- **-s / –use-shodan**: Allows to use ShodanHQ service to gather information about every relay found (up and running) in the descriptors downloaded from the TOR authorities or their mirrors up and running or stored in database. The switch "-k / –shodan-key" must be specified.
- **-c / –use-circuit-nodes**: Instructs to Tortazo to connect to a local instance of TOR through the control port of that instance instead of connect with authoritative directories or their mirrors. Any TOR client (i.e. TorBrowser) will connect with the authoritative directories to download the last consensus and build new virtual circuits with the TOR relays included in the descriptors. Tortazo use the information downloaded by that TOR instance (client) and will perform the actions specified by the other switches used. Note that the TOR instance should use the property "UseMicrodescriptors" with the value "0" in the "torrc" file used to start the instance. This is important to Tortazo, because in this way, the TOR instance will download the "Server Descriptors" from the authoritative directories instead of the "Micro Descriptors". In recent versions of TOR, by default the client

will download the "Micro Descriptors" with much less information about the relays in the network, this default behaviour should be overwritten and allows Tortazo to get as much information as he can from the descriptors downloaded.

6.1.3 Switches for Database Mode

- **-D / –use-database**: Tortazo always stores in a SQLite database every scan performed against the relays found in the descriptors downloaded. This switch uses the records stored in database and avoids performing connections to the TOR authoritative directories. The option "-s / –scan-identifier" allows to specify the number of scan and recover the records associated with that scan identifier. The database is located in "<TORTAZO_DIR>/db/tortazo.db".
- **-C / –clean-database**: Deletes every record stored in the database.

6.1.4 Switches for Botnet Mode

- **-o / –open-shell**: This option is used in "Botnet Mode", which is activated with the switch "-z / –zombie-mode" and allows to create a new interactive shell with the bot entered by the user.

6.1.5 Switches for Plugins management

- **-L / –list-plugins**: List of plugins loaded in Tortazo. Shows author, description, version, etc.

6.2 Valued Switches.

The following is a list of valued switches which receive arguments.

6.2.1 Common Switches for all modes

- **-T <path_to_torrc> / –tor-localinstance <path_to_torrc>**: Start a new local TOR instance with the "torrc" file specified. Usually, the user will specify the switch "-U / –use-localinstance" too.

6.2.2 Switches for Gathering Information

- **-n <number_of_relays> / –servers-to-attack <number_of_relays>**: The number of relays returned by the TOR authoritative directories usually is very large. The user uses this switch to specify the limit, a maximum number of relays which will be used by Tortazo. The default value is "10" if it's not specified. Note that 10 relays is a very low value, the user should use this switch and specify a higher value.
- **-t <number_of_threads>/ –theads<number_of_threads>**: Deprecated in v1.1. Replaced by the plugin "BruterPlugin". Actually this switch is not supported.
- **-m <os> / –mode <os>**: Filter the platform (operative system) of the relay to attack. The accepted values are: "windows", "linux", "darwin", "freebsd", "openbsd", "bitrig","netbsd". Not case-sensitive.
- **-f <password_file> / –passwords-file <password_file>**: Deprecated in v1.1. Replaced by the plugin "BruterPlugin". Actually this switch is not supported.
- **-k <shodan_key_file> / –shodan-key <shodan_key_file>**: Used with the "-s / –use-shodan" to perform queries with Shodan using the IP address of the relays found. This switch receives a text file, which contain a unique line with the developer key used by the Shodan API to perform queries. More info: https://developer.shodan.io/

- **-l <list_of_ports> / –list-ports <list_of_ports>**: Comma-separated list of ports to scan with Nmap. The scan internally will use the Nmap switch "-p" to specify this list of ports.
- **-a <nmap_arguments> / –scan-arguments <nmap_arguments>**: Specify the arguments used by Nmap to perform every scan on the relays founded.
- **-e <relay_fingerprint> / –exit-node-fingerprint**: Specify an fingerprint to filter the exit nodes received in the dataset (Data from descriptors or Data in the local database.) If the fingerprint is not equals to any relay, Tortazo will finish without any result. This option should be used to perform direct attacks against a known exit node.
- **-i <controller_port> / –controller-port <controller_port>**: If the user want to perform connections against a TOR local instance to get and parse descriptors, should use the switch "-c / –use-circuit-nodes" as you've seen above. However, if the local instance uses a non-default controller port, this switch allows specifying it.

6.2.3 Switches for Database Mode

- **-S <scan_identifier> / –scan-identifier <scan_identifier>**: Specify the scan identifier in the Scan table. Tortazo will use the relays related with the scan identifier specified with this switch. This switch should be used with the switch "-D / –use-database".

6.2.4 Switches for Botnet Mode

- **-z <excluded_bots> / –zombie-mode <excluded_bots>**: Tortazo supports the Botnet mode over SSH. In this mode, Tortazo will read the file "tortazo_botnet.bot" located in "<TORTAZO_DIR>/tortazo_botnet.bot" where every line of the file is a SSH server compromised using the "BruterPlugin" against relays with SSH servers with usernames and passwords easy to guess. This switch enables the Botnet Mode and allows selecting the nicknames that will be excluded. (Nicknames included in the tortazo_botnet.bot). For instance, "-z Nickname1,Nickname2" excludes the bots with nicknames "Nickname1" and "Nickname2" and "-z all" allows to include all nicknames in the Botnet Mode. In this mode, Tortazo will not perform any kind of query against the TOR authoritative directories, instead will try to execute parallel commands against the bots loaded. The user usually would like to specify the command to execute against the bots using the switch "-r <command> / –run-command <command>" or open an interactive shell with the switch "-o / –open-shell".
- **-r <command> / –run-command <command>**: Execute a command across the hosts of the botnet. Requieres the -z/–zombie-mode. example: –run-command "uname -a; uptime"

6.2.5 Switches for Plugins management

- **-P <plugin_name> / –use-plugin <plugin_name>**: Loads the interpreter for the specified plugin. The name of the plugin must be registered in Tortazo and the interpreter loaded will contain the functions and elements available in the plugin. This elements allows the interaction with the plugin and are easily accessible by IPython interpreter.
- **-A <plugin_args> / –plugin-arguments <plugin_args>**: Arguments to execute the specified plugin with the switch -P / –use-plugin. List of key/value pairs separated by colon. Used to overwrite the values of the config file for the project located in config/config.py. Example= nessusHost=127.0.0.1,nessusPort=8834,nessusUser=adastra,nessusPassword=adastra

6.2.6 Switches for Repository Mode

- **-R <serviceType> / –onion-repository <serviceType>**: Start Tortazo in Onion Repository Mode. The valid values are: HTTP, SSH, FTP and ONIONUP. The value "ONIONUP" tries to use the online service https://onionup.com/ to check if the onion addresses generated have an hidden service up and running.

- **-W <Number of workers> / –workers-repository <Number of workers>**: Number of processes used to process the ONION addresses generated.
- **-V <chars> / –validchars-repository <chars>**: Valid characters to use in the generation process of onion addresses. Default: All characters between a-z and digits between 2-7
- **-O <partialOnionAddress> / –onionpartial-address <partialOnionAddress>**: Partial address of a hidden service. Used in Onion repository mode.

CHAPTER 7

Available Plugins in Tortazo

There's some plugins integrated in Tortazo and you can use them immediately just by loading the plugin in the interpreter using the switch "-P / –use-plugin".

7.1 Plugins to Gather Information and enumeration of hidden services and TOR relays

7.1.1 infoGathering. TODO IN 1.2!

Plugin Name: infoGathering

Definition: plugins.infogathering.infoGatheringPlugin.infoGatheringPlugin

Description:

Plugin with some functions to gather information about the relays located in the plugin's context. The source of the information could be from the last scan performed by Tortazo or from Database records stored in previous scans.

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()

infoGathering Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P infoGathering -U -T config/config-example/torrc-example
```

7.1.2 stemming

Plugin Name: stemming Definition: plugins.enumeration.deepWebStemmingPlugin.deepWebStemmingPlugin Description:

Basic tasks of stemming module against hidden services in the TOR network. Uses IRL library to find terms in hidden services in the TOR network.

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()
simpleStemmingAllRelays	Stemming with the specified terms along the relays loaded in the plugin. Searches in websites against common ports, like 80,8080,443 or in an specific port.	self.simpleStemmingAllRelays("drugs kill killer hitman")
stemmingHiddenService	Stemming with the specified terms in the onion address defined.	self.stemmingWebSite("http://torlinkbgs6aabns.oni "drugs kill killer")

stemming Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P stemming -U -T config/config-example/torrc-example
```

7.1.3 crawler

Plugin Name: crawler

Definition: plugins.enumeration. deepWebCrawlerPlugin.deepWebCrawlerPlugin

Description:

This plugin uses Scrapy Framework to crawl a hidden service in TOR network. By default, the rules used follow every link in the specified website and downloads the contents found, however the user could overwrite this behavior specifying custom XPATH rules. The first action performed by the plugin is create a new Socat tunnel in the local machine in the port 8765 by default. The endpoint will be the hidden service specified by the user, but the crawler will performs the requests directly against the local machine through the Socat tunnel created. This is very useful to route the requests from the local machine to the TOR network transparently. Also, the user could specify arguments to overwrite the XPATH rules for content extraction and the pages that the crawler should visit before to start the process.

The website structure will be stored in database and the contents will be downloaded in local file system in the path "<TORTAZO_DIR>/onionSites/<hiddenServiceName>/"

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()
setExtractorRulesAllow	Sets the XPATH rules to specify the allowed pages to visit and analyse. This value will be passed to the "allow" attribute of the class: "scrapy.contrib.linkextractors.LinkExtractor"	self.setExtractorRulesAllow("index.php|index.jsp")
setExtractorRulesDeny	Sets the XPATH rules to specify the disallowed pages to visit and analyze. This value will be passed to the "deny" attribute of the class: "scrapy.contrib.linkextractors.LinkExtractor"	self.setExtractorRulesDeny("index.php|index.jsp")
setCrawlRulesLinks	Sets the XPath rules to extract links from every webpage analyzed. Default value should be enough to almost every case, however you can use this function to overwrite this value. Default: '//a/@href'	self.setCrawlRulesLinks('//a[contains(@href, "confidential")]/@href')
setCrawlRulesImages	Sets the XPath rules to extract images from every webpage analyzed. Default value should be enough to almost every case, however you can use this function to overwrite this value. Default: ' //img/@src'	self.setCrawlRulesImages('//a[contains(@href, "image")]/@href')
compareWebSiteWithHiddenWebSite	Compares the contents of a website in clear web with the contents of a web site in TOR's deep web. The return value will be a percent of similitude between both sites.	self.compareWebSiteWithHiddenWebSite("http://exit relay-found.com/", "http://gai12dase4sw3f5a.onion/")
compareRelaysWithHiddenWebSite	This function will perform an HTTP connection against every relay found. If the response is a HTTP 200 status code, performs an HTTP connection against the hidden service specified and compares the contents of both responses. The return value will be a percent of similitude between both sites.	self.compareRelaysWithHiddenWebSite("http://gai12
crawlOnionWebSite	This function executes a crawler against the specified hidden service. The following parameters allow to control the behaviour of the crawler: hiddenWebSite: The hidden site to crawl. This is a mandatory parameter. hiddenWebSitePort: Port for the hidden site to crawl. Default value: 80 socatTcpListenPort: Port for the Socat proxy. Default value: 8765 crawlImages: Search and download the images from every page. Default value: True. crawlLinks: Search and visit the links found in every page. Default value: True. crawlContents: Download and save in local file system the contents of every page found. crawlFormData: Search the forms in every page and store information in database. deepLinks: Number of Links that the crawler will visit in deep. useRandomUserAgents: Use a	• self.crawlOnionWebSite("http://gai12dase4sw3 • self.crawlOnionWebSite("http://gai12dase4sw3 hiddenWebSitePort=8080, crawlImages=False) • self.crawlOnionWebSite("http://gai12dase4sw3 crawlFormData=False)

crawler Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P crawler -U -T config/config-example/torrc-example
```

7.1.4 shodan

Plugin Name: shodan

Definition: plugins.infogathering.shodanPlugin.shodanPlugin

Description:

Plugin used to perform tests against Shodan service using the information gathered by Tortazo. This plugin is much more flexible that the switch "-s / –use-shodan".

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()
setApiKey	Sets the API Key string.	self.setApiKey("XXXXXXXXXXXX")
setApiKeyFile	Sets the API Key file. Reads the first line of the file and then sets the API Key string.	self.setApiKeyFile("/home/apiKeyFile")
basicSearchQuery	Performs a basic search with Shodan. By default prints the 10 first results	self.basicSearchQuery("OpenSSL 1.0.1", 15)
basicSearchAllRelays	Performs a basic search with Shodan against all TOR relays. Uses the "net" filter.	self.basicSearchAllRelays("OpenSSL 1.0.1")
basicSearchByRelay	Performs a basic search with Shodan against the specified TOR relay.	self.basicSearchByRelay("OpenSSL 1.0.1", "80.80.80.80")
basicSearchByNickname	Performs a basic search with Shodan against the specified TOR NickName.	self.basicSearchByNickname("OpenSSL 1.0.1", "TORNickName")

shodan Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P shodan -U -T config/config-example/torrc-example
```

7.2 Plugins to Pentesting and attack hidden services and TOR relays

7.2.1 bruter

Plugin Name: bruter

Definition: plugins.bruteforce.bruterPlugin.bruterPlugin

Description:

This plugin is used to perform dictionary attacks against TOR relays and hidden services. Supports brute forcing against services like SSH, FTP, SNMP and SMB.

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()
setDictSeparator	Sets an separator for the dictionary file. Every line en the file must contain <user><separator><passwd>.	self.setDictSeparator(":")
sshBruterOnRelay	Bruteforce attack against an SSH Server in the relay entered. Uses FuzzDB if the dictFile is not specified.	self.sshBruterOnRelay('37.213.43.122', dictFile='/home/user/dict')
sshBruterOnAllRelays	Bruteforce attack against an SSH Server in the relays founded. Uses FuzzDB if the dictFile is not specified.	self.sshBruterOnAllRelays(dictFile='/home/user/dict')
sshBruterOnHiddenService	Bruteforce attack against an SSH Server in the onion address entered. Uses FuzzDB if the dictFile is not specified.	self.sshBruterOnHiddenService("5bsk3oj5jufsuii6.onion", dictFile="/home/user/dict")
ftpBruterOnRelay	Bruteforce attack against an FTP Server in the relay entered. Uses FuzzDB if the dictFile is not specified.	self.ftpBruterOnRelay("37.213.43.122", dictFile="/home/user/dict")
ftpBruterOnAllRelays	Bruteforce attack against an FTP Server in the relays founded. Uses FuzzDB if the dictFile is not specified.	self.ftpBruterOnAllRelays(dictFile="/home/user/dict")
ftpBruterOnHiddenService	Bruteforce attack against an FTP Server in the onion address entered. Uses FuzzDB if the dictFile is not specified.	self.ftpBruterOnHiddenService("5bsk3oj5jufsuii6.onion", dictFile="/home/user/dict")
smbBruterOnRelay	Bruteforce attack against an SMB Server in the relay entered. Uses FuzzDB if the dictFile is not specified.	self.smbBruterOnRelay("37.213.43.122", dictFile="/home/user/dict")
smbBruterOnAllRelays	Bruteforce attack against an SMB Server in the relays founded. Uses FuzzDB if the dictFile is not specified.	self.smbBruterOnAllRelays(dictFile="/home/user/dict")
smbBruterOnHiddenService	Bruteforce attack against an SMB Server in the onion address entered. This function uses socat to create a local Socks proxy and route the requests from the local machine to the hidden service.	self.smbBruterOnHiddenService("5bsk3oj5jufsuii6.onion", servicePort=139, localPort=139, dictFile="/home/user/dict")
snmpBruterOnRelay	Bruteforce attack against an SNMP Server in the relay entered. Uses FuzzDB if the dictFile is not specified.	self.snmpBruterOnRelay("37.213.43.122", dictFile="/home/user/dict")
snmpBruterOnAllRelays	Bruteforce attack against an SNMP Server in the relays founded. Uses FuzzDB if the dictFile is not specified.	self.snmpBruterOnAllRelays(dictFile="/home/user/dict")
httpBruterOnSite	Bruteforce attack against a web site. Uses FuzzDB if the dictFile is not specified.	self.httpBruterOnSite("http://eviltorrelay.com/auth/", dictFile="/home/user/dict")
httpBruterOnHiddenService	Bruteforce attack against an onion site (hidden site in TOR's deep web). Uses FuzzDB if the dictFile is not specified.	self.httpBruterOnHiddenService("http://5bsk3oj5jufsuii6.oni dictFile="/home/user/dict")

bruter Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P bruter -U -T config/config-example/torrc-example
```

7.2.2 heartBleed

Plugin Name: heartBleed

Definition: plugins.attack.heartBleedPlugin.heartBleedPlugin

*Description: *

Perform HearthBleed Extension vulnerability tests. This plugin allows to discovery TOR relays with this vulnerability.

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()
setTarget	Set the relay for the HeartBleed attack. Check the targets using the function "printRelaysFound". Default port: 443.	self.setTarget("1.2.3.4")
setTargetWithPort	Set the relay and port for the HeartBleed attack. Check the targets using the function "printRelaysFound".	self.setTarget("1.2.3.4", "8443")
startAttack	Starts the HeartBleed attack against the specified target.	self.startAttack()
startAttackAllRelays	Starts the HeartBleed attack against all relays loaded in the plugin. Default port: 443	self.startAttackAllRelays()

heartBleed Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P heartBleed -U -T config/config-example/torrc-example
```

7.3 Plugins for integration with Third-Party tools

7.3.1 w3af

Plugin Name: w3af

Definition: plugins.thirdparty.w3afPlugin.w3afPlugin

Description:

W3AF is a powerful scanner focused on discovering vulnerabilities and attack in

web applications. As is written in Python and has a GNU/GPL license, you can use the classes and utilities from any script in Python. In this case, the plugin does not only covers the features included in w3af, but also allows the execution of audits in web applications that are hosted in the deep web. In the official release of W3AF, you can't use any site on the deep web whose target address is an ONION TLD. Using this plugin, allows you to do that.

Function Name	Description	Usage Example
help	Shows the banner help.	self.help()

Table 7.1 – continued from previous page

Function Name	Description	Usage Example
showPluginsByType	List of available plugins filtered by type.	self.showPluginsByType("audit")
showPluginTypes	List of available plugin types.	self.showPluginTypes()
getEnabledPluginsByType	Enabled plugins by types.	self.getEnabledPluginsByType("a
getPluginTypeDescription	Description for the plugin type specified.	self.getPluginTypeDescription("a
getAllEnabledPlugins	List of enabled plugins.	self.getAllEnabledPlugins()
enablePlugin	Enable a plugin.	self.enablePlugin("blind_sqli","a
disablePlugin	Disable a plugin.	self.disablePlugin("blind_sqli","a
enableAllPlugins	Enable all plugins.	self.enableAllPlugins("audit")
disableAllPlugins	Disable all plugins.	self.disableAllPlugins("audit")
getPluginOptions	Get Options for the plugin specified.	self.getPluginOptions("audit","bl
setPluginOptions	Set Options for the plugin specified.	self.setPluginOptions("audit","ev
getPluginStatus	Check if the specified plugin is enabled.	self.getPluginStatus("audit","eval
setTarget	Sets the target for the attack (clear web).	self.setTarget("http://www.target.
setTargetDeepWeb	Sets the target in the Deep eb of TOR.	self.setTarget("http://torlongonio
startAttack	Starts the attack.	self.startAttack()
listMiscConfigs	List of Misc Settings.	self.listMiscConfigs()
setMiscConfig	Sets a Misc Setting.	self.setMiscConfig("msf_location
listProfiles	List of Profiles.	self.listProfiles()
useProfile	Use a Profile.	self.useProfile("profileName")
createProfileWithCurrentConfig	Creates a new Profile with the current settings.	self.createProfileWithCurrentCon
modifyProfileWithCurrentConfig	Modifies an existing profile with the current settings.	self.modifyProfileWithCurrentCo
removeProfile	Removes an existing profile.	self.removeProfile("profileName"
listShells	List of Shells.	self.listShells()
executeCommand	Executes a command in the specified shell.	self.executeCommand(1,"lsp")
listAttackPlugins	List of attack plugins.	self.listAttackPlugins()
listInfos	List of Infos in the Knowledge Base of W3AF.	self.listInfos()
listVulnerabilities	List of Vulns in the Knowledge Base of W3AF.	self.listVulnerabilities()
exploitAllVulns	Exploits all vulns in the Knowledge Base of W3AF.	self.exploitVulns("sqli")
exploitVuln	Exploits the specified Vuln in the Knowledge Base of W3AF.	self.exploitVulns("sqli",18)

w3af Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P w3af -U -T config/config-example/torrc-example -A
```

7.3.2 nessus

Plugin Name: nessus

Definition: plugins.thirdparty.nessusPlugin.nessusPlugin

Description:

This plugin is responsible for executing the authentication process against a Nessus instance and allows you to use the full features of the Nessus engine against the repeaters

analyzed by Tortazo. It has the functions necessary to list the available plugins, manage policies, users, create specific scans, scheduled scans and query reports generated by Nessus. To carry out the interaction between Tortazo and Nessus, the pynessus-rest library is used; which has been developed primarily to meet the needs of this plugin and directly uses the functions available in the latest version of Nessus REST API. In this way, you can run the same tasks that are available from the web interface enabled on Nessus. Connection and authentication must be declared in

the properties file located in <TORTAZO_DIR>/config.py, which should specify the details for the connection to the server; these details include the address and port of the Nessus server and the credentials required to access. On other hand, if you want overwrite the configuration values without change the properties file, you can use the switch "-A / –plugin-arguments" with the special keywords "nessusHost", "nessusPort", "nessusUser", "nessusPassword".

Function Name	Description
help	Shows the banner help.
serverLoad	Shows details about the load of the server. Number of opened sessions and memory usage, etc.
feed	Return the Nessus Feed.
serverSecureSettingsList	List of Server Secure Settings.
serverRegister	Registers the Nessus server with Tenable Network Security.
serverLoad	Server Load and Platform Type.
serverUuid	Server UUID.
userAdd	Create a new user. The third parameter defines the user as administrator (1) or regular user (0).
userEdit	Edit the user specified. The third parameter defines the user as administrator (1) or regular user (0).
userDelete	Delete the user specified. The third parameter defines the user as administrator (1) or regular user (0
userChpasswd	Change the password for the user specified. The third parameter defines the user as administrator (1
usersList	List of users.
pluginsList	List of plugins.
pluginAttributesList	List of plugins attributes for plugin filtering.
pluginDescription	Returns the entire description of a given plugin.
pluginsAttributesFamilySearch	Filters against the family of plugins.
pluginsAttributesPluginSearch	Returns the plugins in a family that match a given filter criteria. Check the Nessus documentation to
pluginsMd5	List of plugin file names and corresponding MD5 hashes.
policyList	List of available policies, policy settings and default values.
policyDelete	Delete the policy specified.
policyCopy	Copies an existing policy to a new policy.
policyDownload	Download the policy from the server to the local system.
scanAllRelays	Create a new scan with all relays loaded.
scanByRelay	Create a new scan with the specified relay.
scanStop	Stops the specified started scan.
scanResume	Resumes the specified paused scan.
scanPause	Pauses the specified actived scan.
scanList	List of scans.
scanTemplateAllRelays	Create a new scan template (scheduled) with all relays loaded.
scanTemplateByRelay	Create a new scan template (scheduled) with the specified relay.
scanTemplateEditAllRelays	Edit the scan template specified with all relays loaded.
scanTemplateEditByRelay	Edit the scan template specified with the specified relay.
scanTemplateDelete	Delete the scan template specified.
scanTemplateLaunch	Launch the scan template specified.
reportList	List of available scan reports.
reportDelete	Delete the specified report.
reportHosts	List of hosts contained in a specified report.
reportPorts	List of ports and the number of findings on each port.
reportDetails	Details of a scan for a given host.
reportTags	Tags of a scan for a given host.
reportAttributesList	List of filter attributes associated with a given report.

nessus Plugin example

Interaction Example:

```
sudo python Tortazo.py -v -D -P nessus -U -T config/config-example/torrc-example
sudo python Tortazo.py -v -D -P nessus -U -T config/config-example/torrc-example -A nessusHost=192.16
```

CHAPTER 8

Plugin Development in Tortazo

Develop a plugin in Tortazo is a very simple task, which is composed by the following steps:

1. Create a Python file in <TORTAZO_DIR>/plugins/<CATEGORY> where category is the root module which better describes the actions of your plugin (attack, bruteforce, enumeration, infogathering, etc.) In <TORTAZO_DIR>/plugins you'll see the module "examples" where you can create your Python file to follow this guide.

2. Open the Python file created in the previous step and creates a new class which will extend from the class "core.tortazo.pluginManagement.BasePlugin.BasePlugin". BasePlugin class defines the elements to integrate Python routines with Tortazo, so every plugin developed in Totazo must be an sub-class of BasePlugin class. Also, you should define a constructor (__init__) with the parameter "torNodes" which will be used by Tortazo to inject the dataset of relays loaded in the execution context; as probably already you know, data are the relays found in the current scan performed by Tortazo or database records from previous scan, depends on the switches used. The following script could be a valid example

```
from core.tortazo.pluginManagement.BasePlugin import BasePlugin

class TestPlugin(BasePlugin):
    def __init__(self, torNodes=[]):
        BasePlugin.__init__(self, torNodes, 'examplePlugin')
        self.setPluginDetails('testPlugin', 'Example of a plugin in Tortazo.', '1.0', 'Adastra: @jdaa
        self.info("[*] examplePlugin Initialized!")
    def __del__(self):
        self.info("[*]testPlugin Destroyed!")
```

Note that you should use the function "setPluginDetails" to define the settings for the plugin (name, description, version and author).

3. Your plugin is almost done. Now you need to register it in Tortazo. Edit the file <TORTAZO_DIR>/pluginsDeployed.py and add your new plugin in the dict structure defined in the script. You need to add the name for your plugin and the class that you've defined in the previous step. For example

```
plugins = {
#OTHER PLUGINS LOADED IN TORTAZO
#Now, the definition of your plugin.
"testingPlugin": plugins.examples.testing.TestPlugin"
}
```

Assuming that you've created the module "testing" inside the module "plugins.examples", the class "TestPlugin" will be loaded in Tortazo when you use the switch "-P / –use-plugin". Execute the following command to check if your plugin can be successfully loaded.:

```
python Tortazo.py -v -D -P testingPlugin
```

If you can see the IPython interpreter loaded, you're done. Your simple plugin is now integrated in Tortazo. Please, note that if your python script has compilation errors, the load process will fail, so you should verify that your program don't have any errors before trying to load it in Tortazo.

8.1 Utilities in Tortazo for Plugin Development.

When you create a plugin, the functions declared in that plugin will require access to hidden services and TOR relays. You can't access to any hidden service without a connection with TOR and a Socks proxy up and running. The utilities included in Tortazo manages this issues for you and you can create and start your own TOR instance (with a Socks Proxy to browse in the TOR deep web) or indicate to Tortazo that executes an local instance and uses the Socks Proxy settings to connect with any hidden service. The main utility to perform connections to SMB, SSH, HTTP, FTP, and other services in the TOR deep web is the attribute "serviceConnector" defined in the class "BasePlugin". The class "ServiceConnector" includes some functions to performs connections to services in TOR and uses utilities to manage connections like "socat" and "connect-socks". The functions declared in "ServiceConnector" are the following:

Function Name	Description
startLocalSocatTunnel(self, tcpListen,hiddenServiceAddress,hiddenServicePort, socksPort='9150')	Starts a local socat tunnel using the "TCP4-LISTEN" switch. The command executed will have the following format: <TORTAZO_INSTALL>/plugins/utils/socat/socat TCP4-LISTEN:<tcp_port>,reuseaddr,fork SOCKS4A:127.0.0.1:<hidden_service_onion_address>:<hidden_service_onion_port &
anonymousFTPAccess(self,host, port)	Tries to perform an FTP anonymous connection in the host and port specified.
performFTPConnection(self, host, port, user, passwd)	Tries to perform an FTP connection using the user and password specified.
performSSHConnection(self, host, port, user, passwd, brute=False)	Tries to perform an SSH connection using the user and password specified. If the parameter "brute" is True, Tortazo will append the connection settings to the tortazo_botnet.bot file.
performSSHConnectionHiddenService(self, onionService, port, user, passwd)	Tries to perform an SSH connection using the user and password specified against a hidden service.
performSNMPConnection(self, host, port=161, community='public')	Tries to perform an SNMP connection using the community name specified.
performSMBConnection(self, host='127.0.0.1', port=139, user="", passwd="")	Tries to perform an SMB connection using the user and password specified. If the connection is successful, lists the shared resources in the server.
performHTTPAuthConnection(self, url, user, passwd)	Tries to perform an HTTP connection using the user and password specified against a protected resource in the server. This function checks if the resource has Basic or Digest authentication.
performHTTPConnectionHiddenService(self, onionUrl, headers={}, method="GET", urlParameters=None, auth=None)	Tries to perform an HTTP connection against a hidden service. The caller of the function could specify headers, url parameters and authentication details as needed.
performHTTPConnection(self, siteUrl, headers, method="GET", urlParameters=None, auth=None)	Tries to perform a HTTP connection against the web site specified. The caller of the function could specify headers, url parameters and authentication details as needed.
setSocksProxySettings(self, socksHost, socksPort)	Sets the socks proxy settings. Host and Port where the TOR socks proxy is running.
setSocksProxy(self)	Enable the socks proxy defined by the function "setSocksProxySettings" and allows to route every connection through the TOR socks proxy.
unsetSocksProxy(self)	Disable the socks proxy defined by the function "setSocksProxySettings" and allows to perform every connection directly with the service, without using the TOR socks proxy.

CHAPTER 9

GENERAL FAQs

9.1 ¿What is this?

Please, read the gentle introduction of Tortazo. *About Tortazo - Gentle Introduction.*

9.2 I have problems when I run "Tortazo.py" script.

Please, check the dependencies and verify that your environment satisfies them.

9.3 ¿This is free?

Yes, free as the air you breathe. This project is licensed under GNU/GPLv2

9.4 ¿How can I help you?

Fine! if you want to help, you can:

- read the documentation and improve the documents.
- Find errors
- Test the framework
- Report bugs and if you can, write code.

You can write me an email to debiadastra [at] gmail.com

CHAPTER 10

SPECIFIC FAQs

10.1 I get "Import Errors" when I run the Tortazo.py script.

Please, check the dependencies of Tortazo and then proceed to install the Python libraries *Instalation and Dependencies*.

10.2 I have problems running the onion repository mode and some plugins which perform connections against hidden services in TOR, ¿what am I doing wrong?

Every connection against the Deep web, must to use a SOCKS proxy to route the traffic through a TOR circuit. To do that, you must start a local instance of TOR and setting the "SocksPort" property in the configuration file "torrc" used to start TOR. On other hand, the default value for that port in Tortazo is "9150" and you can change that value manually editing property "socksPort" in the configuration file "config/config.py" (If you're using the development version, you can't change that value using the executable distributed). Also, if you want to start automatically a new instance of TOR when you're executing Tortazo, just specify the switches -T / –tor-localinstance and -U / –use-localinstance. Check the switches available in Tortazo *Supported Switches in Tortazo.*

10.3 I get an error when try to loading a Plugin

Check two things: 1. The user that runs the command must read and write over the structure directories where Tortazo runs. Check the permissions of your user. 2. The argument passed to the switch -P / –use-plugin must include a valid plugin registered in the application. To see the modules enabled in Tortazo, check the configuration file "pluginsDeployed.py"

10.4 Oh man, the onion repository mode has been running for the last "n" hours and I don't have any result ¿Am I doing something wrong?

Well, this could be something normal. Please, check the onion repository documentation *repository-mode-label* Also, you must have a TOR local instance up and running with the SocksPort property enabled.

10.5 When I run some functions of the plugins “crawler” or “hiddenService” twice, I get the error “ReactorNotRestartable”.

The crawler plugin uses Scrapy Framework (http://scrapy.org/) which uses Twisted for every connection and network process. Twisted have an element called “reactor” which is designed to not be “restartable”, so if you run the function “crawlOnionWebSite” from the crawler plugin twice, you’ll get that error. You should exit from the plugin interpreter and run the plugin again.

10.6 I’m trying to use shodan to gather information about the relays found, but I get errors

To use Shodan, you’ll need a valid Shodan Key, which you can get if created a new Shodan account. http://www.shodanhq.com/ Also, the shodan key must be included in a plain-text file (in just one line) and use the switch -k / –shodan-key If you use the Shodan plugin available, you’ll have another extended options to perform searches against shodan. *Supported Switches in Tortazo.*

10.7 I get an strange error... ¿What can I do?

Well, your question is very ambiguous, don’t you think? If after read the documentation *Getting Started* and read this FAQ. If you can not solve it, please contact me.

CHAPTER 11

Indices and tables

- *genindex*
- *modindex*
- *search*