ELEC-H417 – TOR Project

Authors:

Eva DUBAR

Kassem KHALIL

Fadi NABBOUT

Dimitrios ROSSI

Module leader:

Prof. Jean-Michel DRICOT

Teaching Assistants:

Denis VERSTRAETEN

Wilson DAUBRY

Abstract:

This report aims to present the project for ELEC-H417 and discuss the implementation of the assignment requirements as well as its results and key findings. Firstly, an introduction to the project assignment is provided with a brief description of its main goals. Then the TOR network architecture and its main features as well as the challenge-response authentication process are briefly presented to aid with understanding the implementation process. The topology of the network and a description of its implementation in python is then given. The main issues encountered are also discussed.

1 – Introduction

This project aims to provide practical understanding of the subjects seen in the module and the onion router (TOR) network via its implementation in python.

The main goals of the project are the implementation of a TOR network in python as well as the implementation of a centralised server that accommodates a *challenge-response* authentication for a client. The TOR network must ensure the anonymity of a client borrowing the network on its way to a destination server (the latter being any destination of choice).

In Chapter 2, the working principles of the TOR network and the challenge-response authentication are presented. In Chapter 3, the topology of the ideal network and the python implementation is presented. The actual resulting implementation on multiple devices in a local network is also discussed as a way to aid with the understanding of the files found in the repository . Finally, a short conclusion on the project is given.

2 – TOR and Challenge-Response authentication

TOR is a type of network that works in a Peer-to-Peer (P2P) manner allowing the anonymity of users borrowing it. This anonymity is achieved via the "bouncing" of packets entering the network off of a certain amount of nodes inside it. The packets are crypted throughout the network. Additionally, the IP of the original client borrowing the network remains hidden when the messages exit the network.

According to [1] here are various types of TOR nodes: guard, middle, exit and bridge. Middle nodes are the most common, they simply act as routers for the network. Any one can choose to get involved/support TOR by becoming a middle node at no safety costs (IP address is not shown as source of traffic when viewing network from the outside). Guard nodes are almost identical to middle nodes, the main difference being the fact that a client "subscribing" to the network choses a middle node to use as its entry point to the network. The guard node is updated every few months. The exit relay is a node that allows the exit of packets from the TOR network and to the destination. These ones are also very similar to middle nodes, however, "people who run exit relays should be prepared to deal with complaints, copyright takedown notices, and the possibility that their servers may attract the attention of law enforcement agencies" as their IP address is interpreted as the source of the traffic. Bridge nodes are not explained as they fall outside the scope of this project. Figure 1 shows the general topology of the TOR network well as where encryption takes place.

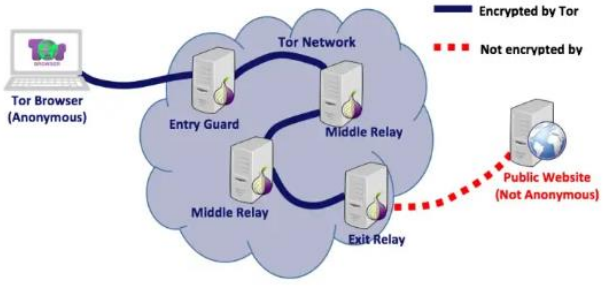

Figure 1 TOR network topology [2]

(Due to time constraints, challenge-response is not discussed)

3 – Implementation

3.1 – Ideal implementation

The implementation of TOR using python in the scope of this project is done by the deployment of several python files to each act as a part of the network.

The ideal implantation of the network consists of a few key components (python files): “client.py”, “server.py”, “central.py”, “node.py”, “exit.py”.

“client.py” represents the user wishing to reach a destination sever via TOR. It’s role is to identify the central TOR server, authenticate itself using the challenge-response method and connecting to its assigned guard node.

“server.py” is simply a destination of choice (e.g. a website). It’s role is to reply to the incoming packets as it would normally, it is mainly used in this project to test whether anonymity of the client is ensured via TOR.

“central.py” is the central server “overseeing” the network. It is responsible for keeping track of the nodes in the network, setting circuits (path taken withing the network) and authenticating any client that wants to use the network with challenge-response.

“node.py” is the P2P middle relay. As a first step, it tries to join the network by advertising its IP and stating its intention to join. After that, it listens for responses coming from the other nodes constituents of the network to create its list of nodes and servers. However, its main use is the one of listening to incoming packets, further encrypting them and relaying them along the circuit defined.

“exit.py” is almost identical to a normal node, with the main difference being it stating its intent to be the exit node. It listens to incoming packets and makes them exit the network and reach their final destination. Traffic exiting this node is not encrypted as previously shown in Figure 1.

3.2 – Actual implementation

Unfortunately, due to time constraints, the final implementation varies from the ideal one as described above. Multiple attempts have been made and are described below in Sections 3.2.1-3.2.3.

Implementations described in Sections 3.2.1 and 3.2.2 are tested on multiple devices in a local network as a test-bench. To recreate the experiment, one must update the IP addresses of the devices used. In general, it is also possible to implement these nodes on one single machine by setting the IP as 127.0.0.1 (“localhost”) or via the implementation of multiple virtual machines (both methods not discussed further as they were not used/implemented)

3.2.1 – Initial Client-Relay node-Server implementation

This first implementation simply acts as a proof-of-concept of a P2P connection using three component: the client, the server and the node connecting them (acting as both a guard for the client and the exit of

the network). In the repository, it is found under the folder “Implementation 1”. Running the python files shows that the client sends a message and receives a response from the server through the node.

3.2.2 – Addition of Guard and Exit with encryption.

This second implementation separates the guard and exit nodes. It specifically includes “client.py”, “server.py”, “guard.py” and “exit.py”. It also attempts to encrypt the message between them. A test script is provided to implement run the python files on multiple devices. Despite the encryption not working, the working principle when it comes to P2P is the same as described in Section 3.2.1. Meaning that modifying the files to work without the encryption step yields satisfying results in terms of the communication and the anonymity of the client when seen from the server.

3.2.3 – Attempt at using stem

Stem is a library that provides a TOR network architecture. It is meant to be used simply to implement TOR on a machine that would be used to join the network. In this case, the library is used to create a “tor_network.py” file that establishes the connection between the provided “client.py” and “server.py” files. Despite specific instructions to avoid the use of pre-made TOR libraries, this attempt is included here simply as a display of the possibilities that one can find out there.

4 – Conclusion

The TOR network is an effective way to ensure anonymity when surfing the web. It is highly flexible and secure, and also provides a way for any users interested to contribute to the project by allowing their machines to be used as nodes in the network.

In this project, it can be seen how such a network can be implemented from scratch using a few basic blocks.

References

[1] EFF, “What is a Tor Relay,” Electronic Frontier Foundation, [Online]. Available: https://www.eff.org/pages/what-tor-relay#:~:text=Tor%20relays%20are%20also%20referred,%2C%20exit%20relays%2C%20and%20bridges.

[2] A. Samhuri, “Hosting Anonymous Website on Tor Network,” [Online]. Available: https://medium.com/axon-technologies/hosting-anonymous-website-on-tor-network-3a82394d7a01.