[ELEC-H417] - Project Communication Networks

Implementing TOR

Arno De Haseleer, Rogier De Nys, Sam Mousavi, Warre De Winne

23/12/2022

Contents

1 Introduction

In the scope of the Communication Networks course, engineering students were tasked to design and implement a TOR network. TOR is a network type that allows anonymous connection to a centralized server. The objectives of this project was to gain an insight and practical understanding of networking and cryptographic principles studied in class.

The TOR network to be implemented has several key characteristics:

- **Peer-to-Peer architecture:** all nodes connected to each other in a peer-to-peer way. The protocol should support an arbitrary number of nodes.
- **Anonymous connection from client:** client should be able to send requests and receive replies from a destination address. The source address will be anonymous to any third party intercepting the message.

2 Overview

Below is explained how the implemented TOR protocol works, but first a list of assumptions is given:

- Client knows address of TOR server by logging in on a webserver or from somewhere else. It does not need to look for it.
- Relays know address of TOR server.
- Everything is run locally on a local IP-address, but this should work for public IP-addresses too.
- Keys for encryption are normally obtained by an external service. Here, the public keys of relays are sent with their addresses to the client. Also, the relays send their public key to the TOR server.
- Everything uses UDP. This means no TCP connection can be made between the client and another client/server.
- No timeouts are set up. If one does not recieve a response, it will wait for eternity.

These assumptions make sure that the focus could be put on getting a working TOR protocol that can be run locally instead of a complex not working TOR protocol.

The project consists of 4 parts: an external server, a TOR server, relays and a Client. Full explanations of each component are given below. Here a short overview is given. First of all, the TOR server is fired up. This communicates with the relays and Client. Then relays can tell the TOR server that they are available to the TOR server. Once enough relays are added to the TOR server, the Client can ask to use the TOR services. It receives a list of random selected available relays from the TOR server, encrypts the package it wants to send and sends it to the first relay. The package then hops between relays until it reaches the external server. The response from the server is then sent back through the relays and finally arrives at the Client. The Client then decrypts the package and has the response from the external server.

3 Scripts

3.1 Client

The Client script does 2 things: it requests a list of a given amount of relays from the TOR server and sends an encrypted package to the first relay on the list.

The Client sends a message to the TOR server if it wants to use the TOR service. It also specifies how many relays it wants to use. Then the server responds by sending the Client a random list of relays with their addresses and their public keys.

Then the Client uses the addresses and public keys to encrypt the package it wants to send. The encryption layers are added in reverse order of the path of the package, such that each relay can use their own private key on the outside layer of the package it receives. It first takes the original package and adds the address of the server it wants to reach with that package. Then it takes the public key of the last relay in the list and encrypts the package. After the encryption, the address of the second to last relay is added to the package. Then the public key of the third to last relay is used to encrypt the package and its address is added to the package. This is done for each relay except the first one. After all the encryption layers for the relays are added (except the first one), the whole package is encrypted one last time using the public key of the first relay. After this the package is sent to the address of the first relay.

Finally the Client waits for a packet from the first relay. Using the public keys of the relays, the packet is decrypted. First it uses the public key of the first relay, then the second relay and so on. After the Client went through all the keys, the decrypted response is achieved.

Note that if the package is lost between relays, no answer will arrive at the Client, and since no timeout is implemented, the Client will wait an infinite amount of time. Also, if the TOR server cannot be reached, an error message is printed.

3.2 Relays

All relays are run in a single script. All relays can send and receive packets at the same time using a loop that loops over all the relays.

When the relays script is started, one can add relays to the TOR network. Each relay that is activated sends a message to the TOR server signifying that it is available. With this message, the public key of this relay is also sent. The TOR server can respond with multiple answers. If the relay is already in the database of the TOR server, a message saying that the relay is already is present in the database is send. If the relay is not yet in the TOR server database, and the key is included in the package, then the TOR server responds saying that the relay is added to the database. If the server cannot be reached, an error message is printed.

If all the wanted relays are added to the database, then the relays can be set on active an start receiving and sending packages. When a packet is received, the relay will encrypt, using its private key, or decrypt, using its public key, the package depending on the source and destination address. The relay has a list of source and destination addresses of packages that went through the relay. This means that when a Client sends a package through the relay, each relay remembers where the package comes from and what its destination is. So when the external server responds to the last relay in the list, the relay that is in direct connection with the server,

it checks if the source address of the response matches with the destination address of the packet sent to the server. If they match, the relay will then encrypt the package using its private key and send it to the source of the original package. This way, the response from the server can find its way back to the client while being encrypted. Note that when the original package from the Client goes through a relay, that relay uses its private key to decrypt the package. Then it can extracted at the destination for the next hop of the package.

3.3 TOR Server

The TOR server has one main purpose. It keeps a database of all the available relays. If a relay sends to the server that it is available, the server checks whether the relay is already in the database using the address of the relay. If the relay is not yet added and if the relay has also sent its public key, then the relay is added to the database.

If a Client sends a request for a specific amout of relays, then the TOR server checks whether enough relays are available. Here a minimum amount of five relays is implemented. This minimum ensures that the package hops through enough relays to make sure that the communication is anonymous. If enough relays are available, the TOR server sends then the relay addresses and public keys to the Client.

If the server receives any message that it cannot process, a bogus answer is sent.

3.4 External Server

The external server is just a destination that the Client can send a package to. It responds by sending a simple reponse including the original package from the Client.

4 Challenges

Here is an overview of the multiple challenges that had to be overcome to get a working TOR protocol.

The first challenge was implementing a `try` and `except` for every time someone sends a package. This prevents the scripts from crashing if an address is not available.

Next is the problem of running all the relays at the same time using only one script. There were several attempts at solving this problem. The main attempt consists of trying to run each relay in parallel using the parallel processing library in python. This kept failing even after many attempts. So a deep dive on google was performed. Here a code that can run multiple sockets was found. Using this, all packets from and to relays can be processed.

Another challenge was getting the address of the next relay from the packet sent by the Client or another relay. The destination address of the next hop has to be extracted from a string since each layer is encoded. From this string one has to select the right parts in order to send the package to the next destination.

5 Improvements To Be Made

There are quite a few improvements one could make, but due to the time window for the given assignment, they could not be implemented. Here is an overview of the possible improvements:

First of all, one could add a timeout if a someone is waiting for a response. Second, messages between the relays and TOR server or between the Client and TOR server could be encrypted as well. Also, a TCP connection cannot be made since everything uses UDP. One could set a a TCP connection to the relay closest to the external server and that relay can use UDP to send everything back to the client. Finally, the TOR server could check periodically if the relays are still up and running.

The relays can have trouble processing multiple packages. When multiple packages are send over the same relays, and if the relay is waiting for multiple responses, then the relay has a list of all the addresses of the packages where the relay expects a response. But if the destination address of the original packages are the same, then the relay could send the wrong response to the wrong destination address(or source address of the original package) since the source address of the response is the same. One could reserve each relay for a single Client, or make sure that a relay never sends multiple packages to the same address, while it is still waiting for a response from that address.

6 Conclusion

Implementing a TOR protocol is far from easy. One has to take a whole lot into account in order to even provide a sufficient protocol. Just getting everything up and running without adding the encryption proved to be a big task. Due to the project deadline, not a lot of attention to detail is spent. But the main principles of a TOR protocol are successfully implemented. If more time was given, a more complete protocol could be delivered.