
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# Implementation of a TOR Bridge

## A PROJECT REPORT

*Submitted by*

18BCI0198 – Smit Shukla

18BCI0122 – Vaidit Patel

18BCE0000 – Sejal Banka

Course Code: CSE4003

Course Title: CYBER SECURITY

Under the guidance of

**Dr. NAVAMANI T M**

**Associate Professor (Grade 1),**

**SCOPE, VIT, Vellore.**

# TABLE OF CONTENTS:

ABSTRACT.....3

1. INTRODUCTION.....4

2. LITRATURE SURVEY…….5

3. PROPOSED SYSTEM.....7

4. APPLICATION OF OUR PROJECT…….....….11

5. IMPLEMENTATION .....11

6. VISUALISATION OF THE INBOUND CONNECTIONS TO PUT TOR BRIDGE ON A WORLD.....15

7. CONCLUSION .....16

8. FUTURE WORK……..16

9. CODE SAMPLE……...…17

10. REFERENCES .....23

# ABSTRACT

The Tor network is a group of volunteer-operated servers that allows people to improve their privacy and security on the Internet. Tor's users employ this network by connecting through a series of virtual tunnels rather than making a direct connection, thus allowing both organizations and individuals to share information over public networks without compromising their privacy.

The design of the Tor network means that the IP address of Tor relays is public. However, one of the ways Tor can be blocked by governments or ISPs is by blacklisting the IP addresses of these public Tor nodes. Tor bridges are nodes in the network that are not listed in the public Tor directory, which make it harder for ISPs and governments to block them. We intend to host our own low-bandwidth Tor bridge on an Ubuntu instance, to monitor and analyze traffic in the Tor network.

# 1. INTRODUCTION

## a) Background of our Proposed Work

These are some important points that we found out during our work on this project and we used these points as reference for implementing our project:

- Bridge relays (or "bridges" for short) are Tor relays that aren't listed in the main Tor directory. Since there is no complete public list of them, even if your ISP is filtering connections to all the known Tor relays, they probably won't be able to block all the bridges.

- Most Pluggable Transports, such as obfs3 and obfs4, rely on the use of “bridge” relays. Like ordinary Tor relays, bridges are run by volunteers; unlike ordinary relays, however, they are not listed publicly, so an adversary cannot identify them easily. Using bridges in combination with pluggable transports helps to disguise the fact that you are using Tor.

- Other pluggable transports, like meek, use different anti-censorship techniques that do not rely on bridges. You do not need to obtain bridge addresses in order to use these transports.

## b) MOTIVATION

- The growing use of Internet among public masses has increased exponentially over the years and although there is a lot of research on maintaining security and anonymity, security is breached one way or another.
- Fingerprinting attacks and traffic analysis have become a common thing and hence, TOR helps us avoid all that and stay safe and anonymous online.
- But there are new methods of analyzing and fingerprinting even TOR users.
- Hence, we aim to put an end to that by making a bridge not listed on the public consensus, hence ensuring anonymity against the new type of attacks.

## c) ISSUES IN EXSISTING SYSTEM

- Your real IP address is leaked to the first node while using TOR.
- ISPs mark your use of TOR by tracking your usage.
- A malicious tech at your VPN provider could try an attempt at sending malicious code down that bridge to you.
- Fingerprinting attacks using technologies like Deep Learning continue to tighten grip on censorship circumvention alternatives.

# 2. LITRATURE SURVEY

| Title | Journal | Year | Abstracts | Drawbacks |
|---|---|---|---|---|
| Analysis of Fingerprinting Techniques for Tor Hidden Services | WPES | 2017 | The paper proposes a novel two phase approach for fingerprinting hidden services that does not rely on malicious Tor nodes. In our attack, the adversary merely needs to be on the link between the client and the first anonymization node. | Tor bridge hasn't been taken into account |
| Fingerprinting Attack on Tor Anonymity using Deep Learning | APAN | 2016 | This paper proposes a new method for launching a fingerprinting attack to analyse Tor traffic in order to detect users who access illegal websites. Our new method is based on Stacked Denoising Auto encoder, a deep-learning technology | Alternative entry points cannot be detected using this. Also, neural nets can be optimised using RNN instead of CNN |
| Mind the Gap: Towards a Backpressure-Based Transport Protocol for the Tor Network | HUB | 2016 | The current Tor design is not able to adjust the load appropriately, and we argue that finding good solutions to this problem is hard for anonymity overlays. It's due to the long end-to-end delay in such networks, combined with limitations allowable feedback due to anonymity requirements. | The destination route can be found easily thought the exit point in the network |

| Title | Journal | Year | Abstracts | Drawbacks |
|---|---|---|---|---|
| Design, implementation and test of a flexible tor-oriented web mining toolkit | ACM | 2017 | This paper presents a flexible and accessible toolkit for structure and content mining, able to crawl, download, extract and index resources from the Web using Tor network. | Normal Tor node is being used as an entry point which can be used to get the IP address of the user |
| Characterization of Tor Traffic using Time based Features | ICISS | 2017 | In this paper, a time analysis is presented on Tor traffic flows, captured between the client and the entry node. We define two scenarios, one to detect Tor traffic flows and the other to detect the application. | Monitoring the entry node of the Tor network can jeopardize the security of the IP address of the user |
| A Forensic Audit of the Tor Browser Bundle | Cornell | 2019 | The Tor browser, though, can leave behind digital artefacts which can be used by an investigator. This paper outlines an experimental methodology and provides results for evidence trails which can be used within real-life investigations | tor bridge has not been taken into account |
| Forensic Analysis of Tor Browser: A Case Study for Privacy and Anonymity on the Web | FSI | 2019 | This paper examines the extend of user privacy when he is operating on TOR network and how much exposed he is while he is on network. | N.A<br><br>. |
| Ephemeral Exit Bridges for Tor | IEEE | 2019 | This paper examines an existential threat to Tor---the increasing frequency at which websites apply discriminatory behaviour to users who arrive via the anonymity network. The main contribution is the introduction of Tor exit bridges. | N.A. |
| The onion router: Understanding a privacy enhancing technology community | IEEE | 2020 | Tor network is often monitored by law enforcement, which makes this PET is different from any other open-source initiatives. To explore this volunteer community's motivation for providing their services despite the risks, we | PET's mention weren't able to provide the level of privacy that we expect from a tor bridge relay. |

| Title | Journal | Year | Abstracts | Drawbacks |
|---|---|---|---|---|
| | | | conducted an online survey. Study results reveal that one of the main motivations for these volunteers is to advocate and provide privacy for online users. | |

**SUMMARY:**

The final takeaway from all the research paper we read is the TOR is a brilliant implementation of Privacy Enhancing Technologies (PETs) but a serious flaw that we figured out from above papers is the mention of the TOR relays in the public directory where the intruder in the network or the ISP of the user can easily track their entry relay from where they enter the TOR network. Once they figure out the entry node, neural networks can be constructed and optimized for the purpose of tracking the different relays in the network and figuring out the final destination of the user on the network i.e. their exit relay.

By implementing tor, the very cause of the problem can be avoided as the relays of the tor bridges are not mentioned in the public directory, moreover using then to enter the network of tor relays will make sure that the ISP of the user or the cracker cannot track the entry relay of the user.

# 3. PROPOSED SYSTEM

## a. OVERVIEW

The Tor network is a group of volunteer-operated servers that allows people to improve their privacy and security on the Internet. Tor's users employ this network by connecting through a series of virtual tunnels rather than making a direct connection, thus allowing both organizations and individuals to share information over public networks without compromising their privacy.

The design of the Tor network means that the IP address of Tor relays is public. However, one of the ways Tor can be blocked by governments or ISPs is by blacklisting the IP addresses of these public Tor nodes. Tor bridges are nodes in the network that are not listed in the public Tor directory, which make it harder for ISPs and governments to block them. We will also be setting up a system to analyze the TOR Traffic on our bridge.

## b. SYSTEM ARCHITECTURE

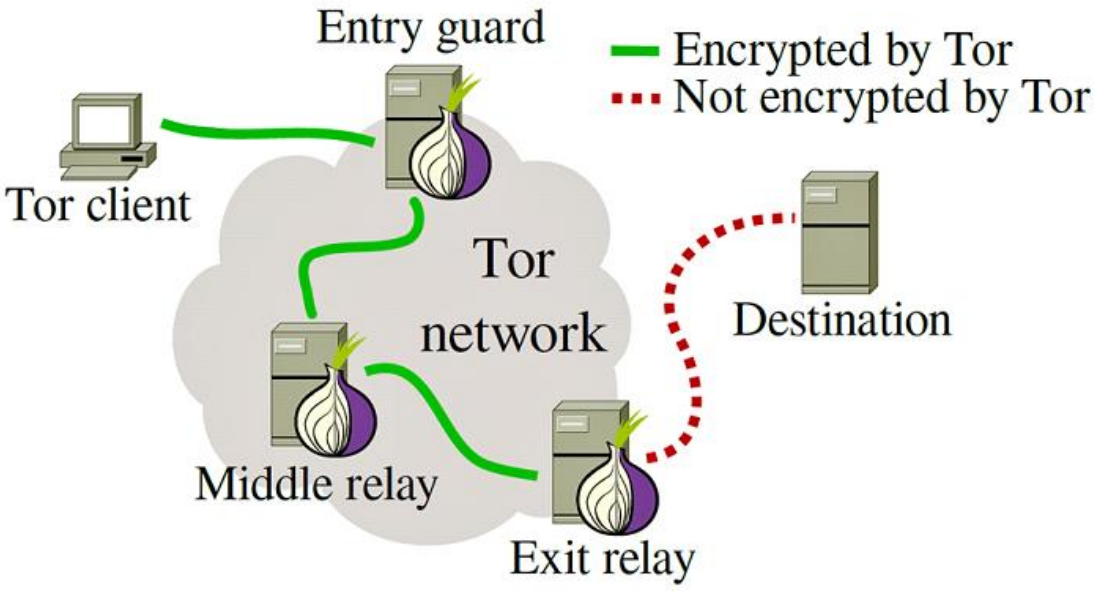


Figure 1: System Architecture of TOR network

The figure 1 above gives us a brief idea on how devices on the TOR network function and their architecture.

TOR network is an implementation of onion routing, which encrypts and then randomly bounces communications through a network of relays run by volunteers around the globe. These onion routers employ encryption in a multi-layered manner to ensure perfect forward secrecy between relays, thereby providing users with anonymity in a network location. That anonymity extends to the hosting of censorship-resistant content by Tor's anonymous onion service feature. Furthermore, by keeping some of the entry relays or bridge relays secret, users can evade Internet censorship that relies upon blocking public Tor relays.

Because the IP address of the sender and the recipient are not both in clear text at any hop along the way, anyone eavesdropping at any point along the communication channel cannot directly identify both ends. Furthermore, to the recipient it appears that the last Tor node (exit node), rather than the sender, is the originator of the communication.

## c. FUNCTIONAL ARCHITECTURE

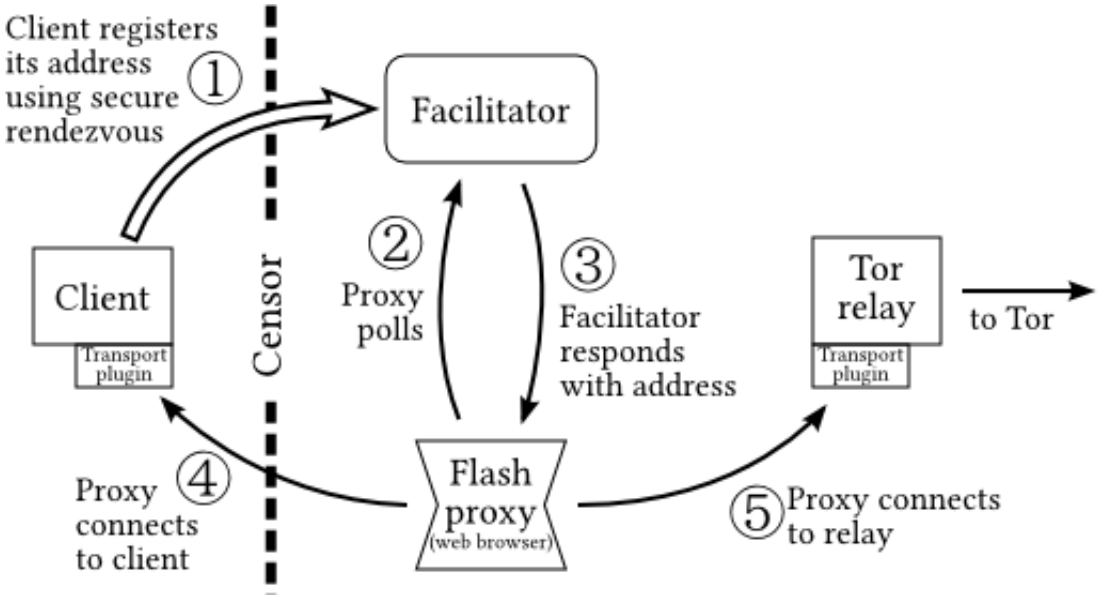


Figure 2: Functional Architecture of TOR network

Figure 2 explains the functioning of the TOR Network in presence of the facilitator which will be the TOR Bridge.

Bridges are useful for Tor users under oppressive regimes or for people who want an extra layer of security because they're worried somebody will recognize that they are contacting a public Tor relay IP address.

Bridges are relatively easy, low-risk and low bandwidth Tor nodes to operate, but they have a big impact on users. A bridge isn't likely to receive any abuse complaints, and since bridges are not listed in the public consensus, they are unlikely to be blocked by popular services. Bridges are a great option if you can only run a Tor node from your home network, have only one static IP, and don't have a huge amount of bandwidth to donate.

Client first tries to connect with the entry relay of the network which in turn send the user a proxy token signifying that a connection has been established and the user has successfully entered the TOR bridge network. The user signal is then bounced between several other relays until it reached it final destination, and the beauty of this network is that a particular node does not know the entire route taken by the signal making it very difficult to tap and eavesdrop on the connection made.

## d. DESIGN DESCRIPTION

Our project aims at hosting a Tor Bridge on a Ubuntu Virtual Private Server. This bridge serves as a door to the Tor Network. The visualization of the network and its analysis the same has been done by us. The figure 3 below gives he overall idea of the design of the TOR network in presence of our bride.

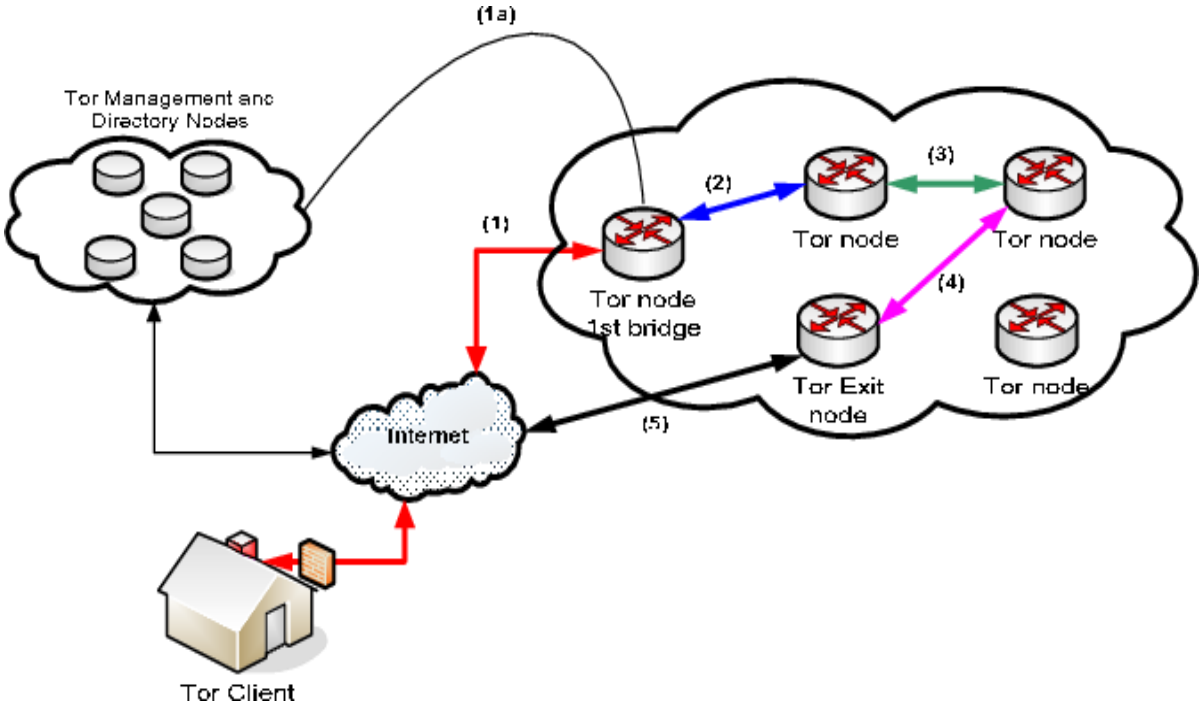


Figure 3: Model Design

## e. INNOVATION IN OUR PROJECT

- Unrestricted Access to the TOR network through the Bridge.
- No current method of tracking user in this method.
- Real-time Analysis of parameters like bandwidth usage, logs, connections, of devices on the TOR network.
- Blacklisting the user's IP address is extremely difficult.

# 4. APPLICATION OF OUR PROJECT

- Ensure your security and anonymity from all kinds of trackers like ISPs, VPNs and the government.
- A great solution for censorship circumvention.
- Access sites that are blocked in a particular country or are

# 5. IMPLEMENTATION

## a. SOFTWARE DETAILS

We will be using a cloud server hosted on Digital Ocean Droplet. It will be an Ubuntu Instance which will host our own TOR Bridge. The TOR Bridge will run as long as the server runs. After some users start using our TOR Bridge, we will analyze the traffic through our Bridge using a tool called NYX and will also be visualizing the data using a custom python code that uses different libraries to represent the IP Addresses of our users on a World Map

# b. IMPLEMENTATION SNAPSHOTS

## i. BANDWIDTH ANALYSIS WITH NYX

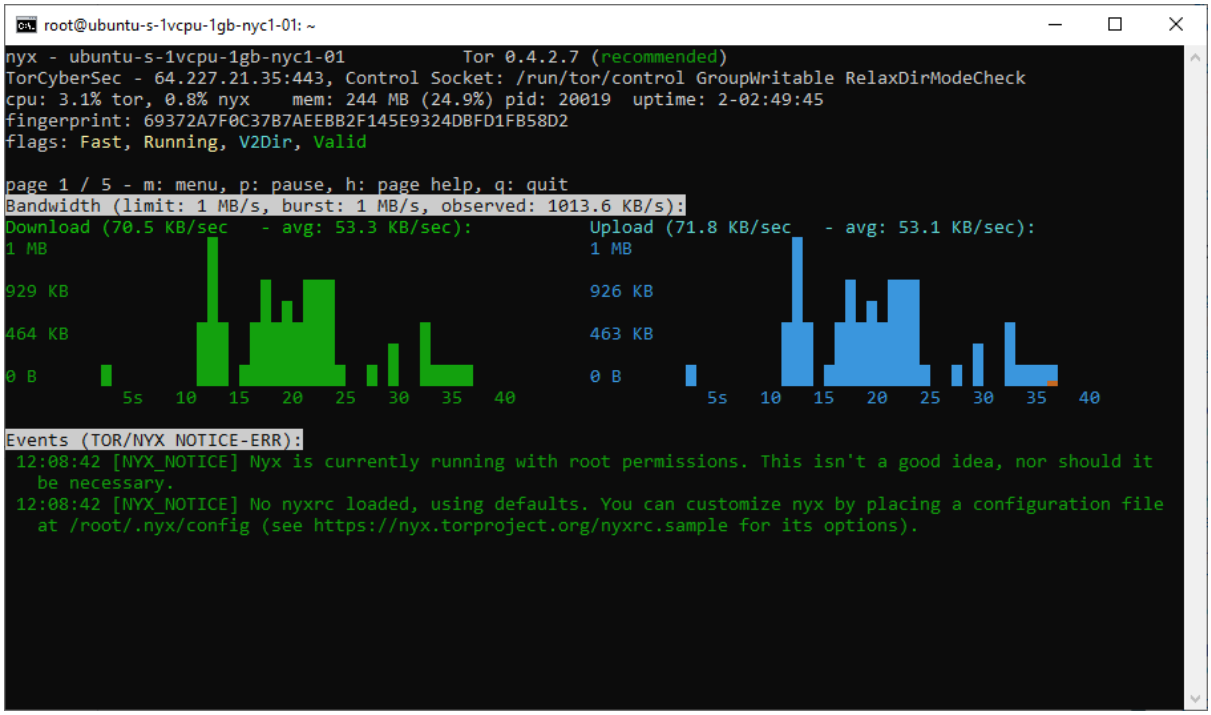


Figure 4: Bandwidth Analysis with NYX

The above Figure 4 shows the amount of bandwidth being used by TOR in terms of graph and helps us analyse it in case we need to limit the bandwidth. It shows the total data downloaded and uploaded and we can also select the time interval of the graph according to our preference.

## ii. RESOURCE ANALYSIS WITH NYX

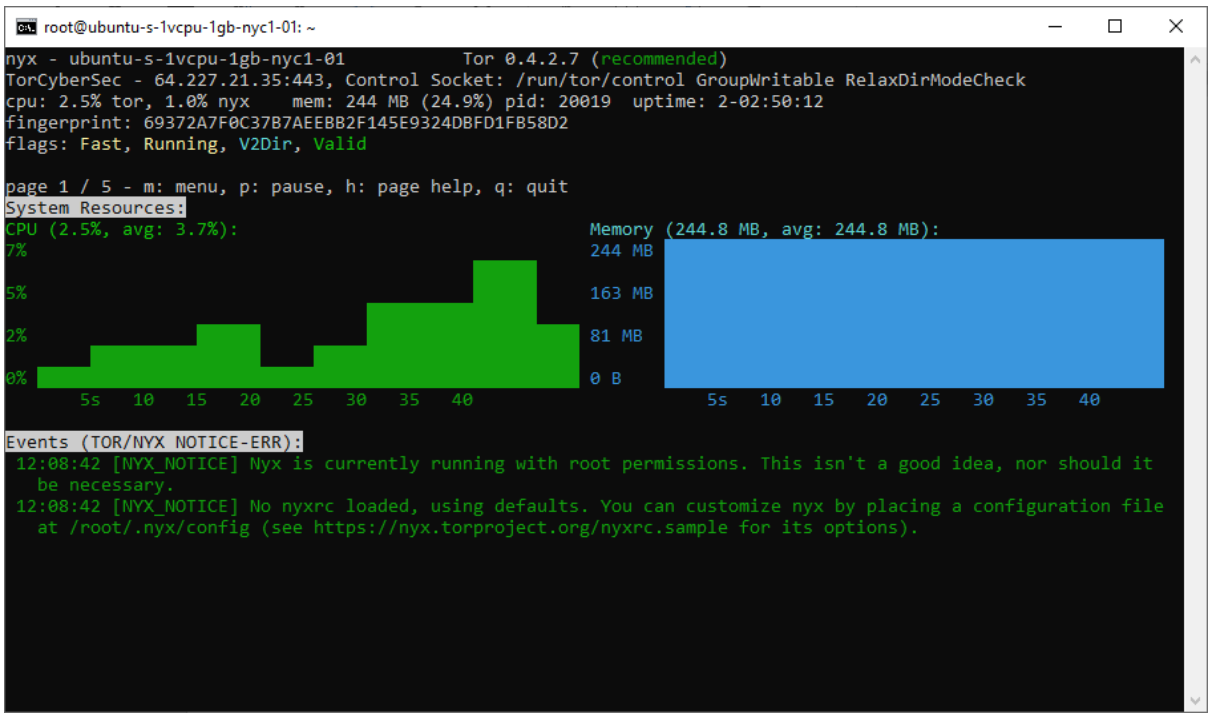


Figure 5: Resource Analysis with NYX

The above figure 5 is also a graph generated by NYX to show the system resources being utilized by TOR. Again, like the bandwidth graph, we have the option to customize the time interval here and analyze the use of our resources.

## iii. CONNECTION ANALYSIS WITH NYX

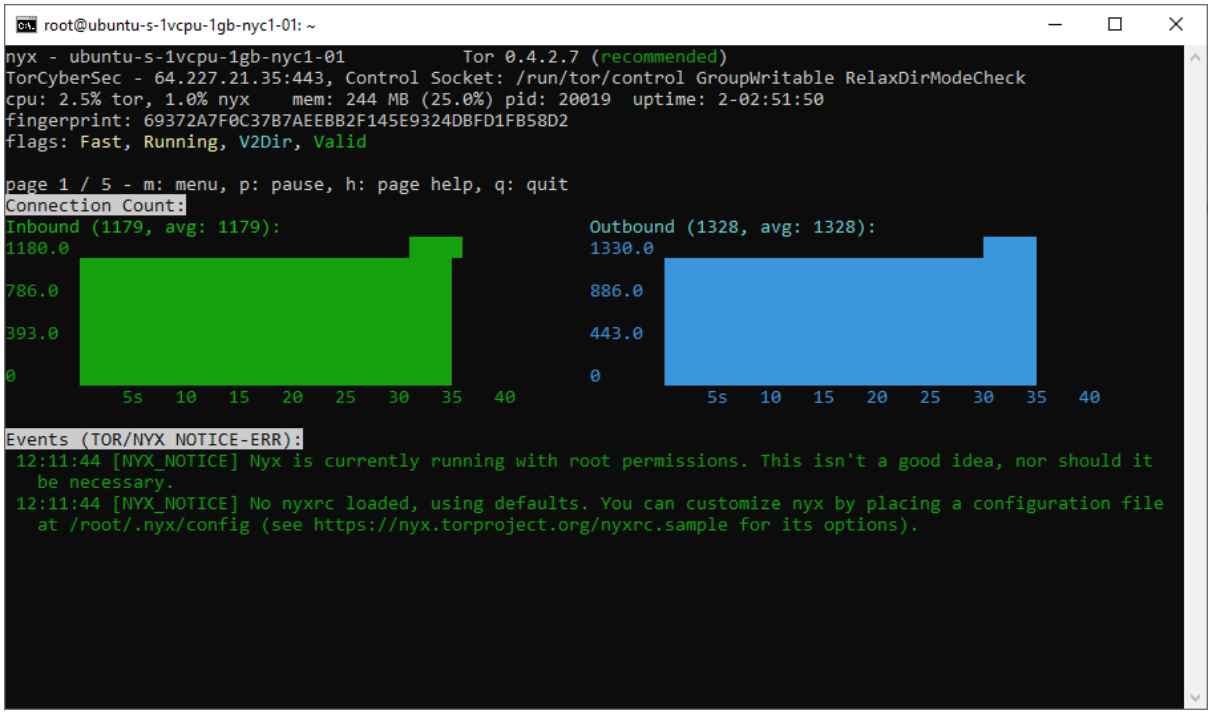


Figure 6: Connection Analysis with NYX

Figure 6 shows a graph generated for the inbound and outbound connections to our TOR Bridge with respect to time. As, seen above in the image, it shows that we have 1179 inbound connections and 1328 outbound connections. It helps us analyze how our number of connections change with time

# 6. VISUALISATION OF THE INBOUND CONNECTIONS TO PUT TOR BRIDGE ON A WORLD

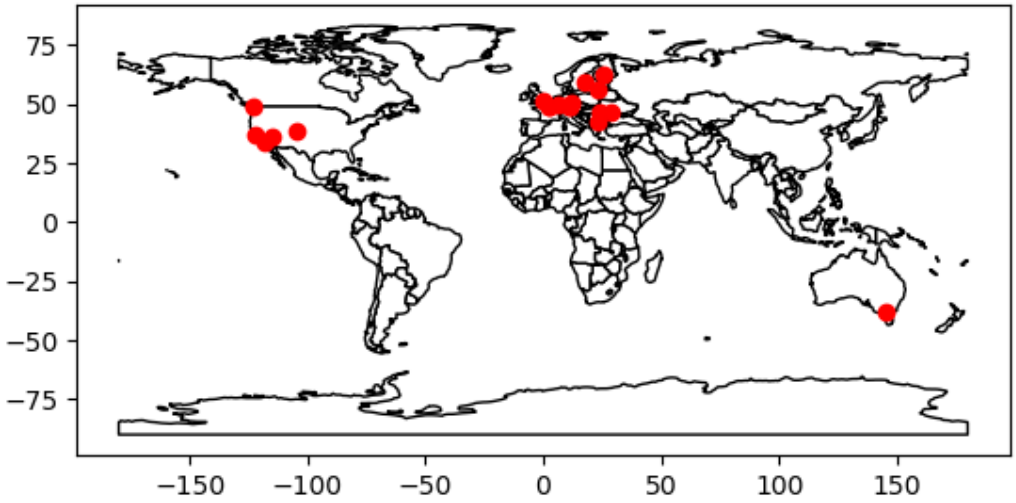


Figure 7: Visualizing the users connected to the TOR Bridge

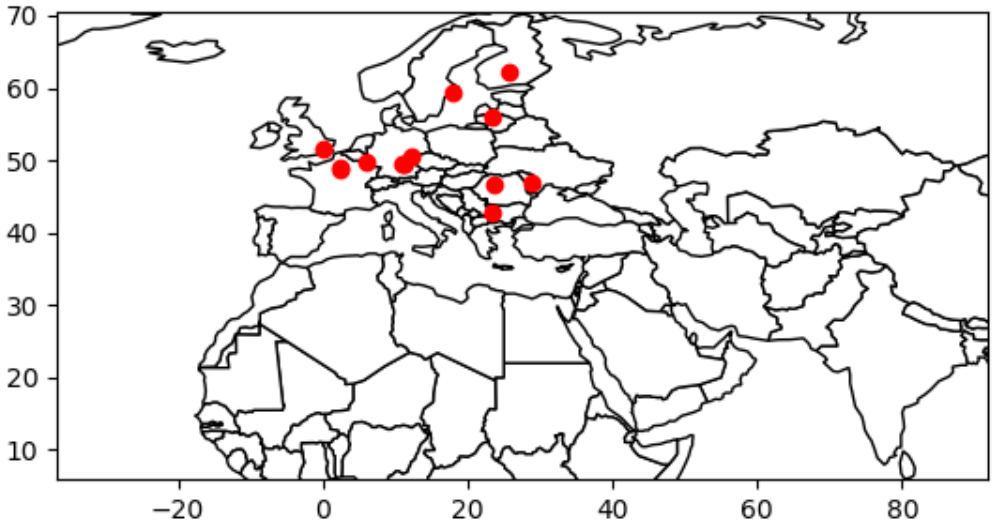


Figure 8: A zoomed in view of the Visual Map of IP Addresses TOR Users

The figures 7 and 8 above are the visual representation of the IP Addresses of the people using our TOR Bridge. We wrote a custom python code that visualizes the users that are using our TOR Bridge through their IP Addresses. The IP Address gives a very good idea about their location which can then be represented on a map.

## 7. CONCLUSION

This project helped us in learning extensively about the Tor Network, about the functioning of Tor relays, bridges and exits which are the backbone of the Tor Network. Also, analysis of the connections, resources and bandwidth helped us to draft a basic theory about the average usage of the former parameters in a typical Tor Bridge.

The addition of bridges to Tor is a step forward in the blocking resistance race and hence gave us an opportunity to contribute towards internet anonymity and access for all.

## 8. FUTURE WORK

More nodes can be generated which will create a TOR bridge network which in turn can be optimized to reduce the latency, noise or any other kind of delays in the network. By optimizing this network, it may make the current method obsolete and pave a path for the users to use a safe internet where neither their activity cannot be traced nor their location of access.

## 9. CODE SAMPLE

**Section 1: TOR CONFIGURATION:**

```
## Tor opens a SOCKS proxy on port 9050 by default -- even if you don't
## configure one below. Set "SOCKSPort 0" if you plan to run Tor only
## as a relay, and not make any local application connections yourself.
SOCKSPort 0
ControlPort 9051
```

CookieAuthentication 1

# Default: Bind to localhost:9050 for local connections.

#SOCKSPort 192.168.0.1:9100

# Bind to this address:port too. ## Entry policies to allow/deny SOCKS requests based on IP address. ## First entry that matches wins. If no SOCKSPolicy is set,

we accept

## all (and only) requests that reach a SOCKSPort. Untrusted users who

## can access your SOCKSPort may be able to learn about the connections

## you make.

#SOCKSPolicy accept 192.168.0.0/16 #SOCKSPolicy accept6 FC00::/7 #SOCKSPolicy reject *

## Logs go to stdout at level "notice" unless redirected by something

## else, like one of the below lines. You can have as many Log lines as ##

you want. ##

## We advise using "notice" in most cases, since anything

more verbose

## may provide sensitive information to an attacker who obtains the logs.

##

## Send all messages of level 'notice' or higher to

/var/log/tor/notices.log

Log notice file /var/log/tor/notices.log

## Send every possible message to /var/log/tor/debug.log Log debug file /var/log/tor/debug.log

## Use the system log instead of Tor's logfiles

#Log notice syslog

## To send all messages to stderr:

#Log debug stderr

## Uncomment this to start the process in the background... or use

## --runasdaemon 1 on the command line. This is ignored on

Windows;

## see the FAQ entry if you want Tor to run as an NT service.

#RunAsDaemon 1

## The directory for keeping all the keys/etc. By default, we store

## things in $HOME/.tor on Unix, and in Application Data\tor on Windows.

#DataDirectory /var/lib/tor

## The port on which Tor will listen for local connections from Tor

## controller applications, as documented in controlspec.txt. #ControlPort 9051

## If you enable the controlport, be sure to enable one of these

## authentication methods, to prevent attackers from accessing it.

#HashedControlPassword
16:872860B76453A77D60CA2BB8C1A7042072093276A3D701AD684053EC4 C

#CookieAuthentication 1

################ This section is just for location-hidden services ###

## Once you have configured a hidden service, you can look at the

## contents of the file ".../hidden_service/hostname" for the address ## to tell people.

##

## HiddenServicePort x y:z says to redirect requests on port x to the

## address y:z.

#HiddenServiceDir /var/lib/tor/hidden_service/ #HiddenServicePort 80 127.0.0.1:80

#HiddenServiceDir /var/lib/tor/other_hidden_service/ #HiddenServicePort 80 127.0.0.1:80

#HiddenServicePort 22 127.0.0.1:22

################ This section is just for relays #####################

#

## See https://www.torproject.org/docs/tor-doc-relay for details.

## Required: what port to advertise for incoming Tor connections.

ORPort 443

## If you want to listen on a port other than the one advertised in

## ORPort (e.g. to advertise 443 but bind to 9090), you can do it as

## follows. You'll need to do ipchains or other port forwarding

## yourself to make this work. #ORPort 443 NoListen

#ORPort 127.0.0.1:9090 NoAdvertise

## The IP address or full DNS name for incoming connections to your

## relay. Leave commented out and Tor will guess. #Address noname.example.com

## If you have multiple network interfaces, you can specify one for

## outgoing traffic to use.

## OutboundBindAddressExit will be used for all exit traffic, while

## OutboundBindAddressOR will be used for all OR and Dir connections

## (DNS connections ignore OutboundBindAddress). ## If you do not wish to differentiate, use

OutboundBindAddress to

## specify the same address for both in a single line. #OutboundBindAddressExit 10.0.0.4

#OutboundBindAddressOR 10.0.0.5

## A handle for your relay, so people don't have to refer to it by key.

## Nicknames must be between 1 and 19 characters inclusive, and must

## contain only the characters [a-zA-Z0-9].

Nickname myfirsttorbridg

## Define these to limit how much relayed traffic you will allow. Your

## own traffic is still unthrottled. Note that RelayBandwidthRate must

## be at least 75 kilobytes per second.

## Note that units for these config options are bytes (per second), not

## bits (per second), and that prefixes are binary prefixes, i.e. 2^10, ## 2^20, etc.

#RelayBandwidthRate 100 KBytes  # Throttle traffic to

100KB/s (800Kbps)

#RelayBandwidthBurst 200 KBytes # But allow bursts up to 200KB (1600Kb)

**SECTION 2: VISUALISATION TOOL WITH PYTHON**

import grequests import argparse import os from platform import system as os_name
import pandas as pd import requests import sqlite3

lookup_data = []

def gen_list_urls(relay_data, points):

urls = ['http://ip-api.com/json/'+row[1] for row in relay_data[:points]]

return urls

def lookup(urls): """

:param: ipaddress to look up

:returns: A dict containing all information post lookup """

res = (grequests.get(u) for u in urls) consol_res = grequests.map(res)  return consol_res

def index_data_from_cache_file(path): """

For reading from cache.sqlite file

:param: path to the cache file

:returns: A list containing tuples where each tuple represents a row [(), (), ()]

"""

conn = sqlite3.connect(path) c = conn.cursor()

relay_data = c.execute('SELECT * FROM relays')

relay_data = relay_data.fetchall()

return relay_data def gen_data_frame(path, points):

raw_data = index_data_from_cache_file(path)

urls = gen_list_urls(raw_data, points)

all_responses = lookup(urls)

cities = []

countries = []

lats = []

longs = []

for res in all_responses:

res_data = res.json() cities.append(res_data["city"])

lats.append(res_data["lat"])

longs.append(res_data["lon"])

countries.append(res_data["country"])

df = pd.DataFrame({'City': cities,

'Country': countries, 'Latitude': lats, 'Longitude': longs}

}

return df

# 10. REFERENCES


- Official Documentation of Tor - https://2019.www.torproject.org/docs/bridges.html.en
- Nyx Documentation - https://nyx.torproject.org/
- Forninet Blog - https://www.fortinet.com/blog/threat-research/dissecting-tor-bridges-pluggable-transpo
- Panchenko, Andriy, et al. "Analysis of fingerprinting techniques for tor hidden services." *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*. 2017.
- Abe, Kota, and Shigeki Goto. "Fingerprinting attack on Tor anonymity using deep learning." *Proceedings of the Asia-Pacific Advanced Network* 42 (2016): 15-20.
- Tschorsch, Florian, and Björn Scheuermann. "Mind the gap: Towards a backpressure-based transport protocol for the Tor network." *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*. 2016.
- Celestini, Alessandro, and Stefano Guarino. "Design, implementation and test of a flexible tor-oriented web mining toolkit." *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*. 2017.
- Lashkari, Arash Habibi, et al. "Characterization of tor traffic using time based features." *ICISSp*. 2017
- Jadoon, A. K., Iqbal, W., Amjad, M. F., Afzal, H., & Bangash, Y. A. (2019). Forensic Analysis of Tor Browser: A Case Study for Privacy and Anonymity on the Web. Forensic Science International.
- Matt Muir, Petra Leimich, William J. Buchanan " A Forensic Audit of the Tor Browser Bundle
- Zhang, Zhao, et al. "Ephemeral Exit Bridges for Tor." 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2020.
- Saputra, Ferry Astika, Isbat Uzzin Nadhori, and Balighani Fathul Barry. "Detecting and blocking onion router traffic using deep packet inspection." 2016 International Electronics Symposium (IES). IEEE, 2016