Project On

Implementation Of Tor Bridge

B.Tech in
Computer Science and Engineering

By

Aviral Goyal- 19BCE0883
Hemaksh Chaturvedi - 19BCE2222
Aakansha Gauatam - 19BCE2663

Under the guidance of
UmaDevi K S

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
FALL- 2021

TABLE OF CONTENTS:

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.

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.

We intend to host our own low-bandwidth Tor bridge on a Linux instance, to monitor and analyze traffic in the Tor network.

INTRODUCTION TO DOMAIN

Background of our Proposed Work

These are some key points that we discovered while working on this project, and we utilized them as a guide for putting it together:

- The use of "bridge" relays is required by most Pluggable Transports, such as obfs3 and obfs4. Bridges, like regular Tor relays, are run by volunteers; but, unlike regular relays, they are not advertised publicly, making it difficult for an enemy to find them. The use of bridges in conjunction with pluggable transports can help hide the fact that you're using Tor.

- Other pluggable transports, such as meek, employ anti-censorship strategies that aren't based on bridges. To use these transports, you do not need to obtain bridge addresses.

- Tor relays that aren't mentioned in the main Tor directory are known as bridge relays (or "bridges" for short). Even if your ISP is filtering connections to all known Tor relays, they won't be able to ban all the bridges because there isn't a complete public list of them.

Motivation

- The popularity of the Internet among the general population has exploded in recent years, and despite extensive study into maintaining security and anonymity, security is hacked in some way.

- Fingerprinting attacks and traffic analysis have grown commonplace, and TOR helps us stay safe and anonymous online by preventing them.

- 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

Issues in existing 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.

LITERATURE 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	Alternative entry points cannot be detected using this. Also, neural nets can be optimised using RNN

			method is based on Stacked Denoising Auto encoder, a deep-learning technology	instead of CNN
Mind the Gap: Towards a Backpressu re- 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, implement ation 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
Characterizat ion 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 .

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	PET's mention weren't able to provide the level of privacy that we expect from a tor bridge relay.

			services despite the risks, we	

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 take away 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.

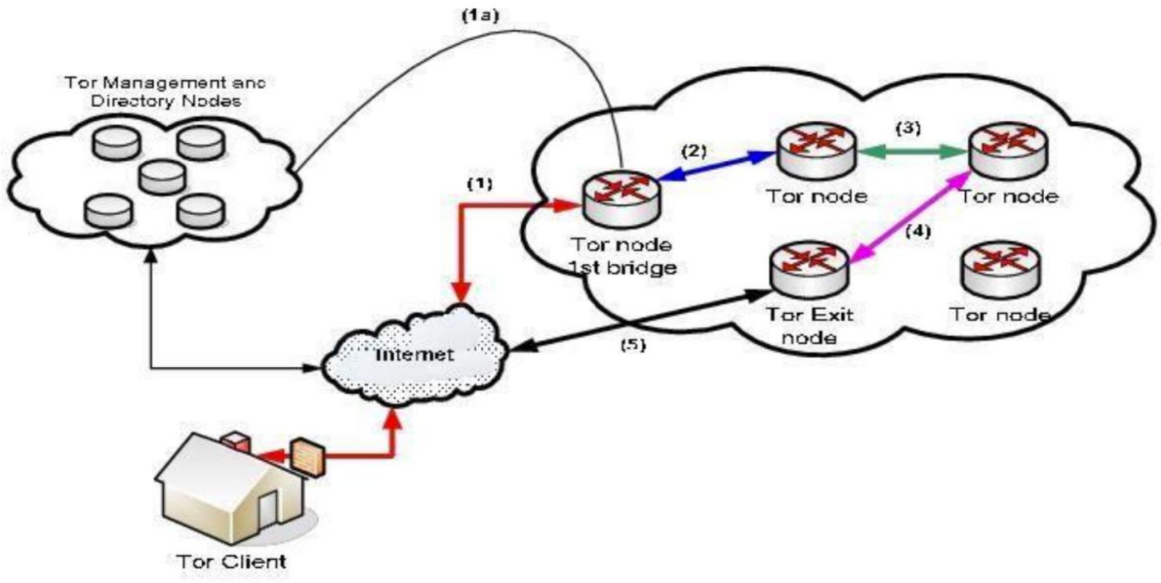
PROPOSED WORK

These are some important points that we found out and we will use 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.

- 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.



System Architecture

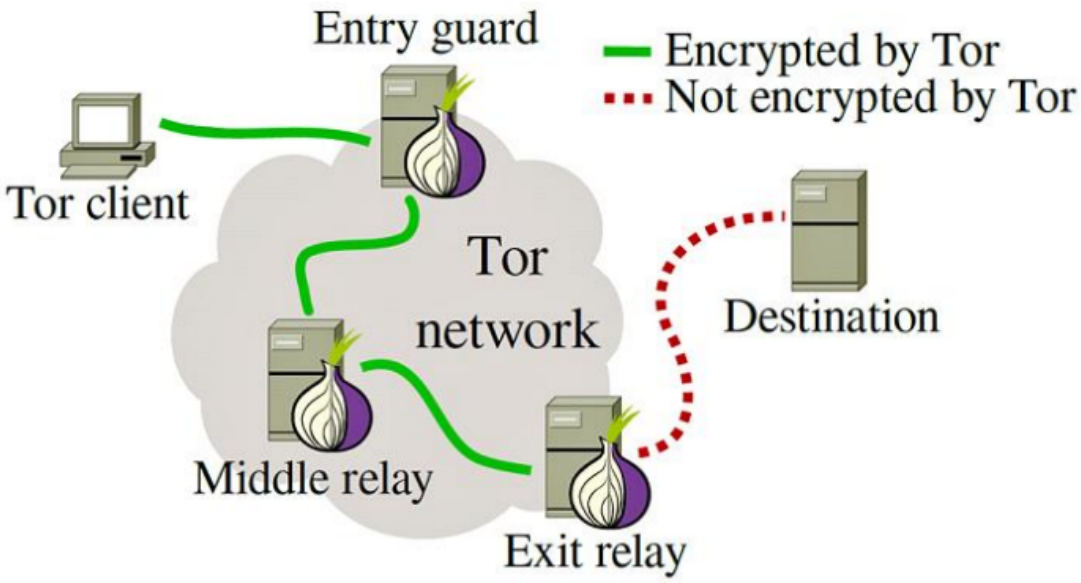


Figure 1: System Architecture of TOR network

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.

Functional Architecture

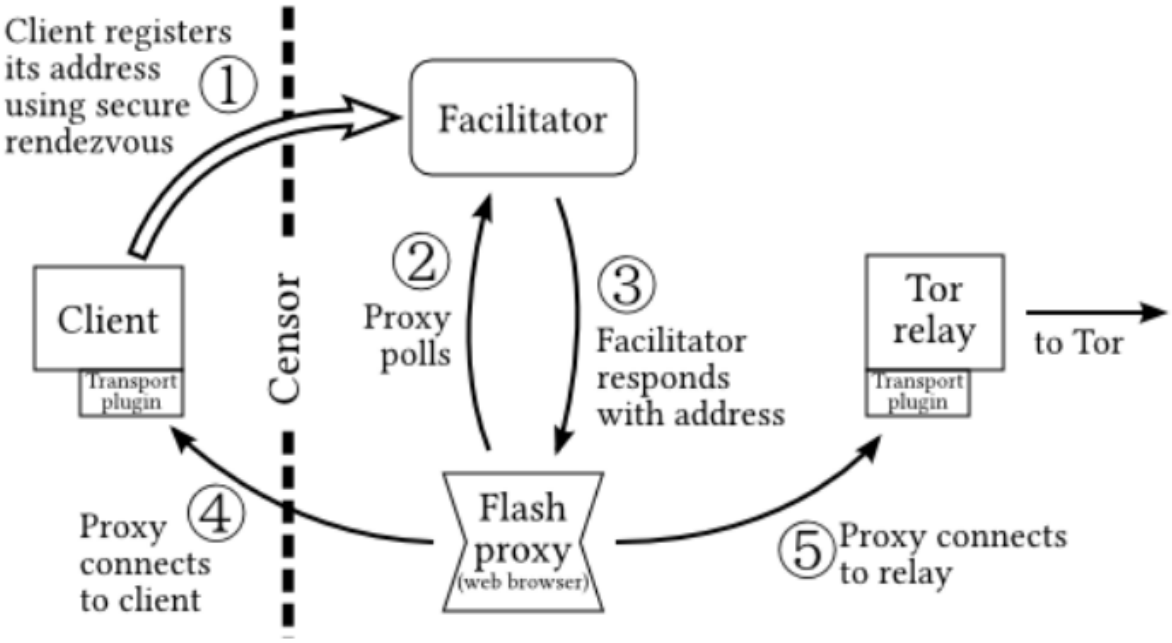


Figure 2: Functional Architecture of TOR network

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 sends 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 reaches its 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.

Design Description

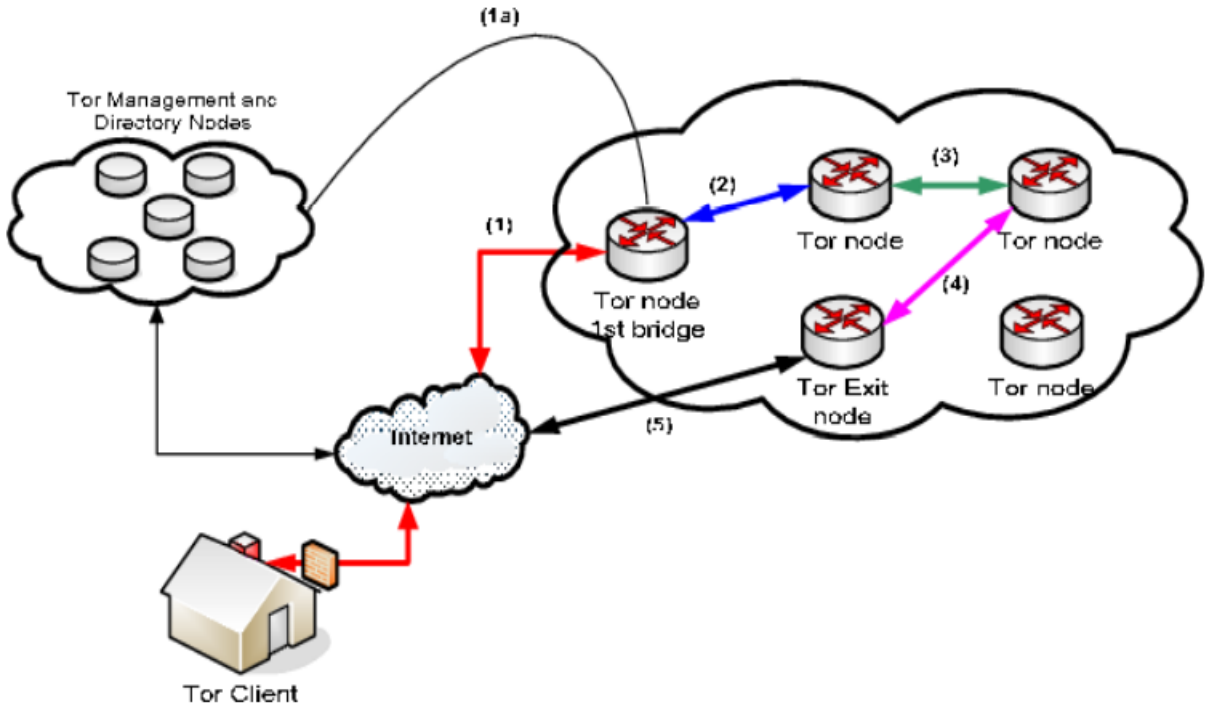


Figure 3: Model Design

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 has been done by us. The figure 3 below gives the overall idea of the design of the TOR network in presence of our bride.

Innovation in our Project

- Unrestricted Access to the TOR network through the Bridge.
- No current method of tracking users 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.

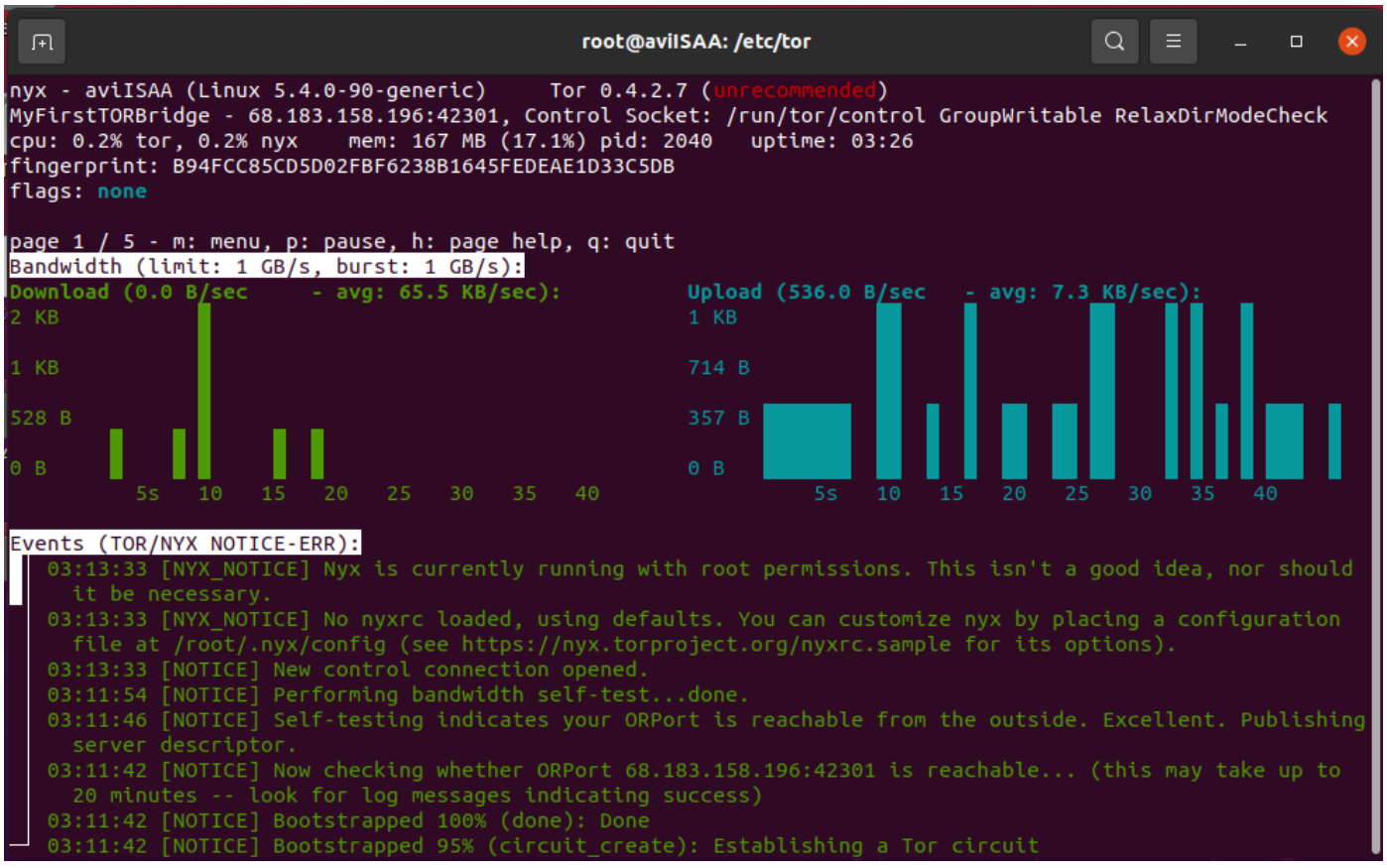
IMPLEMENTATION

Software Details

We will be using a cloud server hosted on DigitalOcean 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

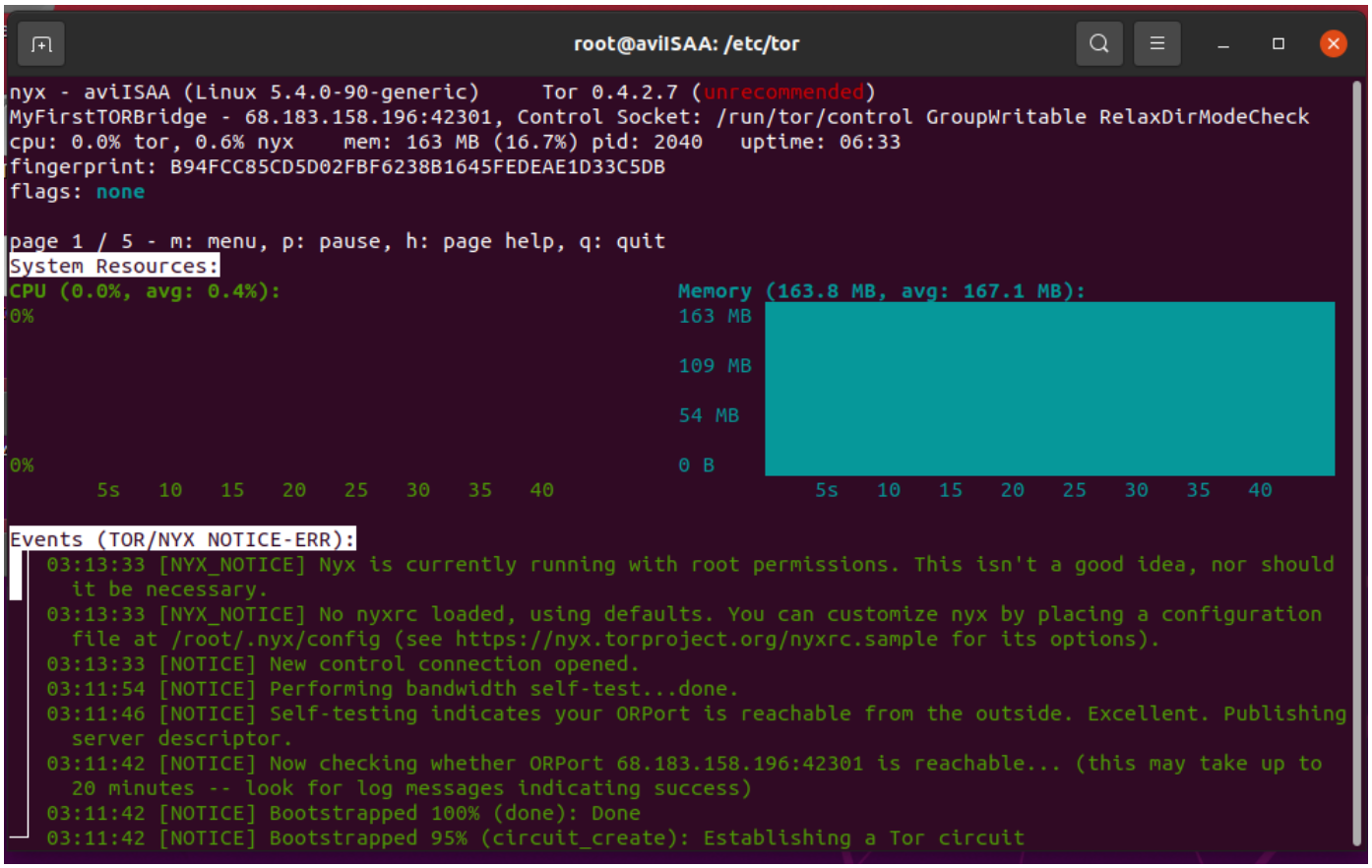
Implementation Snapshots

1. **Bandwidth Analysis with Nyx**



The above figure shows the amount of bandwidth being used by TOR in terms of graphs 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.

2. **Resource Analysis with Nyx**



The above figure 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.

3. Connection Analysis with Nyx

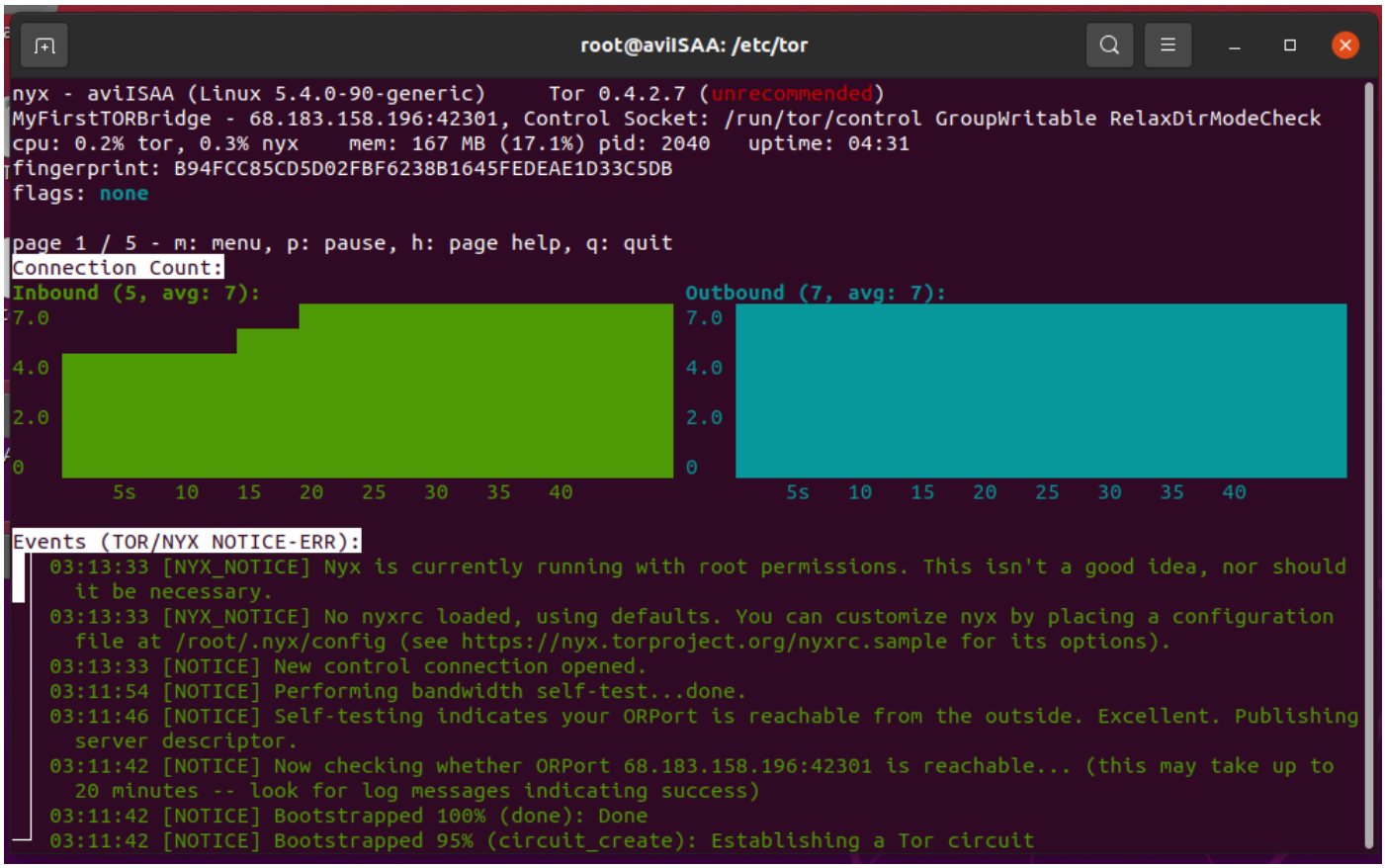
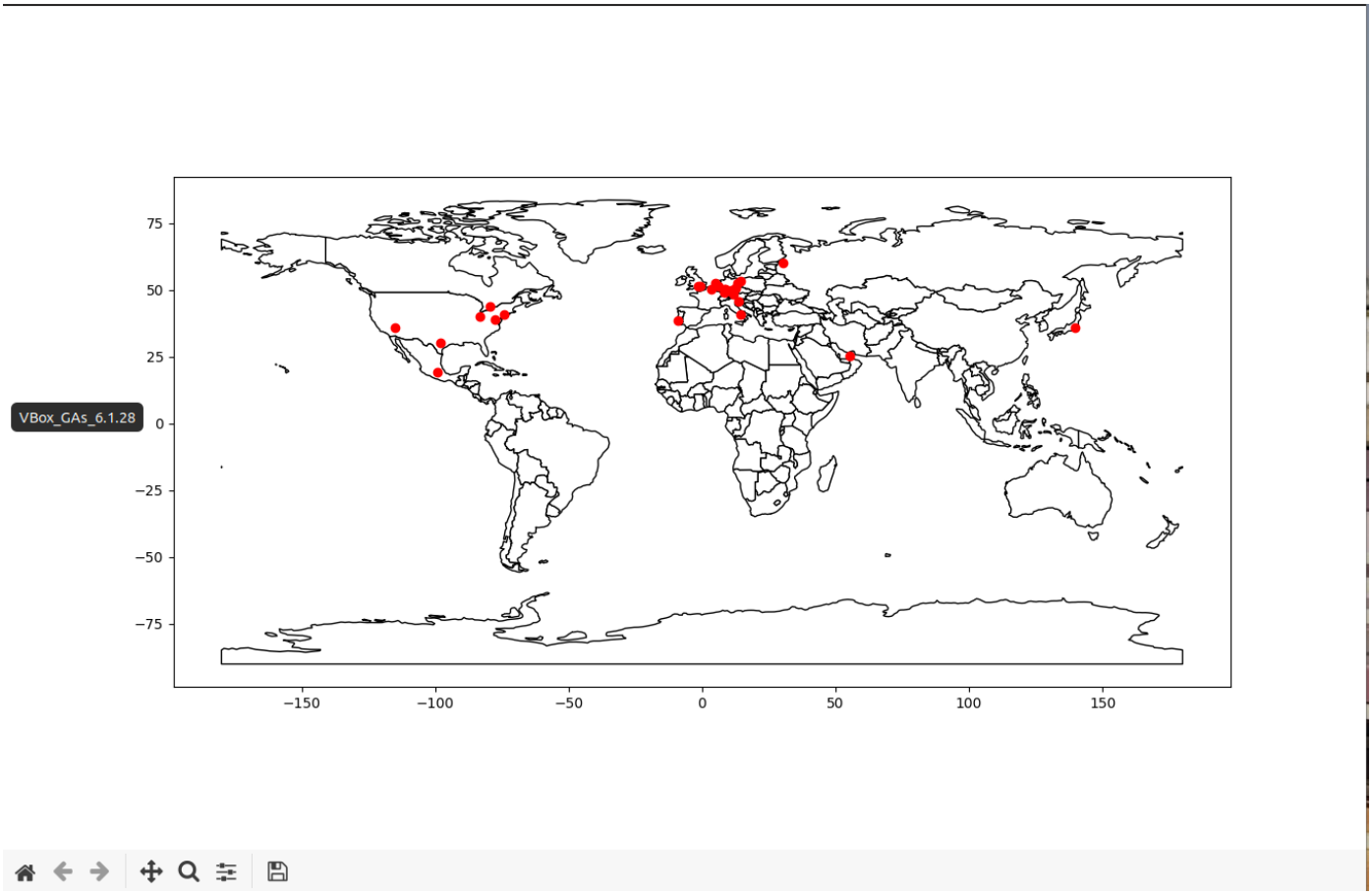


Figure shows a graph generated for the inbound and outbound connections to our TOR Bridge with respect to time. As seen above in the image, it shows that we have 1179 inbound connections and 1328 outbound connections. It helps us analyze how our number of connections change with time.

Visualization of the Inbound Connections to put TOR Bridge on a World Map



The figure above is 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.

CODE SAMPLE

TOR Configuration:

```
## Configuration file for a typical Tor user
## Last updated 9 October 2013 for Tor 0.2.5.2-alpha.
## (may or may not work for much older or much newer versions of Tor.)
##
## Lines that begin with "## " try to explain what's going on. Lines
## that begin with just "#" are disabled commands: you can enable them
## by removing the "#" symbol.
##
## See 'man tor', or https://www.torproject.org/docs/tor-manual.html,
## for more options you can use in this file.
##
## Tor will look for this file in various places based on your platform:
## https://www.torproject.org/docs/faq#torrc

## 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 9050 # 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 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 control-spec.txt.
#ControlPort 9051
## If you enable the controlport, be sure to enable one of these
## authentication methods, to prevent attackers from accessing it.
#HashedControlPassword
16:872860B76453A77D60CA2BB8C1A7042072093276A3D701AD684053EC4C
#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.
# OutboundBindAddress 10.0.0.5

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

Define these to limit how much relayed traffic you will allow. Your
own traffic is still unthrottled. Note that RelayBandwidthRate must
be at least 20 KB.
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 KB # Throttle traffic to 100KB/s (800Kbps)
#RelayBandwidthBurst 200 KB # But allow bursts up to 200KB/s (1600Kbps)

Use these to restrict the maximum traffic per day, week, or month.
Note that this threshold applies separately to sent and received bytes,
not to their sum: setting "4 GB" may allow up to 8 GB total before
hibernating.
##
Set a maximum of 4 gigabytes each way per period.
#AccountingMax 4 GB
Each period starts daily at midnight (AccountingMax is per day)
#AccountingStart day 00:00
Each period starts on the 3rd of the month at 15:00 (AccountingMax
is per month)
#AccountingStart month 3 15:00

Administrative contact information for this relay or bridge. This line
can be used to contact you if your relay or bridge is misconfigured or
something else goes wrong. Note that we archive and publish all
descriptors containing these lines and that Google indexes them, so
spammers might also collect them. You may want to obscure the fact that
it's an email address and/or generate a new address for this purpose.
#ContactInfo Random Person <nobody AT example dot com>
You might also include your PGP or GPG fingerprint if you have one:
#ContactInfo 0xFFFFFFFF Random Person <nobody AT example dot com>

Uncomment this to mirror directory information for others. Please do
if you have enough bandwidth.
#DirPort 9030 # what port to advertise for directory connections
If you want to listen on a port other than the one advertised in

DirPort (e.g. to advertise 80 but bind to 9091), you can do it as
follows. below too. You'll need to do ipchains or other port
forwarding yourself to make this work.
#DirPort 80 NoListen
#DirPort 127.0.0.1:9091 NoAdvertise
Uncomment to return an arbitrary blob of html on your DirPort. Now you
can explain what Tor is if anybody wonders why your IP address is
contacting them. See contrib/tor-exit-notice.html in Tor's source
distribution for a sample.
#DirPortFrontPage /etc/tor/tor-exit-notice.html

Uncomment this if you run more than one Tor relay, and add the identity
key fingerprint of each Tor relay you control, even if they're on
different networks. You declare it here so Tor clients can avoid
using more than one of your relays in a single circuit. See
https://www.torproject.org/docs/faq#MultipleRelays
However, you should never include a bridge's fingerprint here, as it would
break its concealability and potentionally reveal its IP/TCP address.
#MyFamily $keyid,$keyid,...

A comma-separated list of exit policies. They're considered first
to last, and the first match wins. If you want to _replace_
the default exit policy, end this with either a reject *:* or an
accept *:*. Otherwise, you're _augmenting_ (prepending to) the
default exit policy. Leave commented to just use the default, which is
described in the man page or at
https://www.torproject.org/documentation.html
##
Look at https://www.torproject.org/faq-abuse.html#TypicalAbuses
for issues you might encounter if you use the default exit policy.
##
If certain IPs and ports are blocked externally, e.g. by your firewall,
you should update your exit policy to reflect this -- otherwise Tor
users will be told that those destinations are down.
##

```
## For security, by default Tor rejects connections to private (local)
## networks, including to your public IP address. See the man page entry
## for ExitPolicyRejectPrivate if you want to allow "exit enclaving".
##
#ExitPolicy accept *:6660-6667,reject *:* # allow irc ports but no more
#ExitPolicy accept *:119 # accept nntp as well as default exit policy
#ExitPolicy reject *:* # no exits allowed

## Bridge relays (or "bridges") are Tor relays that aren't listed in the
## main directory. Since there is no complete public list of them, even an
## ISP that filters connections to all the known Tor relays probably
## won't be able to block all the bridges. Also, websites won't treat you
## differently because they won't know you're running Tor. If you can
## be a real relay, please do; but if not, be a bridge!
#BridgeRelay 1
## By default, Tor will advertise your bridge to users through various
## mechanisms like https://bridges.torproject.org/. If you want to run
## a private bridge, for example because you'll give out your bridge
## address manually to your friends, uncomment this line:
#PublishServerDescriptor 0

BridgeRelay 1
ORPort 42301
Log notice file /var/log/tor/notices.log
Log notice file /var/log/tor/debug.log
Log notice syslog
Log debug stderr
ServerTransportPlugin obfs4 exec /usr/bin/obfs4proxy
ServerTransportListenAddr obfs4 0.0.0.0:37198
ExtORPort auto
ContactInfo <aviralgoyal2010@gmail.com>
Nickname MyFirstTORBridge
```

Visualisation Tool with Python

helpers.py (code helps create the data frame for plotting)

```
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
```

plot.py (code helps to plot the points from data frame on world map)

```
import pandas as pd
import geopandas
from shapely.geometry import Point
import matplotlib.pyplot as plt

def generate_plot(df):
    df["Coordinates"] = list(zip(df.Longitude, df.Latitude))
    df['Coordinates'] = df['Coordinates'].apply(Point)
    gdf = geopandas.GeoDataFrame(df, geometry='Coordinates')
    world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))
    ax = world.plot(color='white', edgecolor='black')
    gdf.plot(ax=ax, color='red')
    plt.show()
    return None
```

RESULTS AND DISCUSSION

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.

REFERENCES

- https://www.theguardian.com/technology/2013/nov/05/tor-beginners-guide-nsa-browser
- https://www.eff.org/deeplinks/2014/07/7-things-you-should-know-about-tor
- https://support.torproject.org/censorship/censorship-7/
- https://blog.torproject.org/
- 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.
- Nyx Documentation - https://nyx.torproject.org/
- Forninet Blog - https://www.fortinet.com/blog/threat-research/dissecting-torbridges-pluggable-transpo