Implementation of a Tor Bridge

Ishitva Verma[#1], Saloni Uday Parekh[#2], Damayanti Chattopadhyay[#3]

[#]*Department of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India*

[1]ishitva.verma2018@student.ac.in
(18BCI0155)

[2]saloniuday.parekh@vitstudent.ac.in
(18BCI0212)

[3]damayanti.c2018@vitstudent.ac.in
(18BCI0222)

1. 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 analyse traffic in the Tor network.

2. BACKGROUND OF THE WORK – THE PROBLEM

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

3. OVERVIEW OF THE WORK

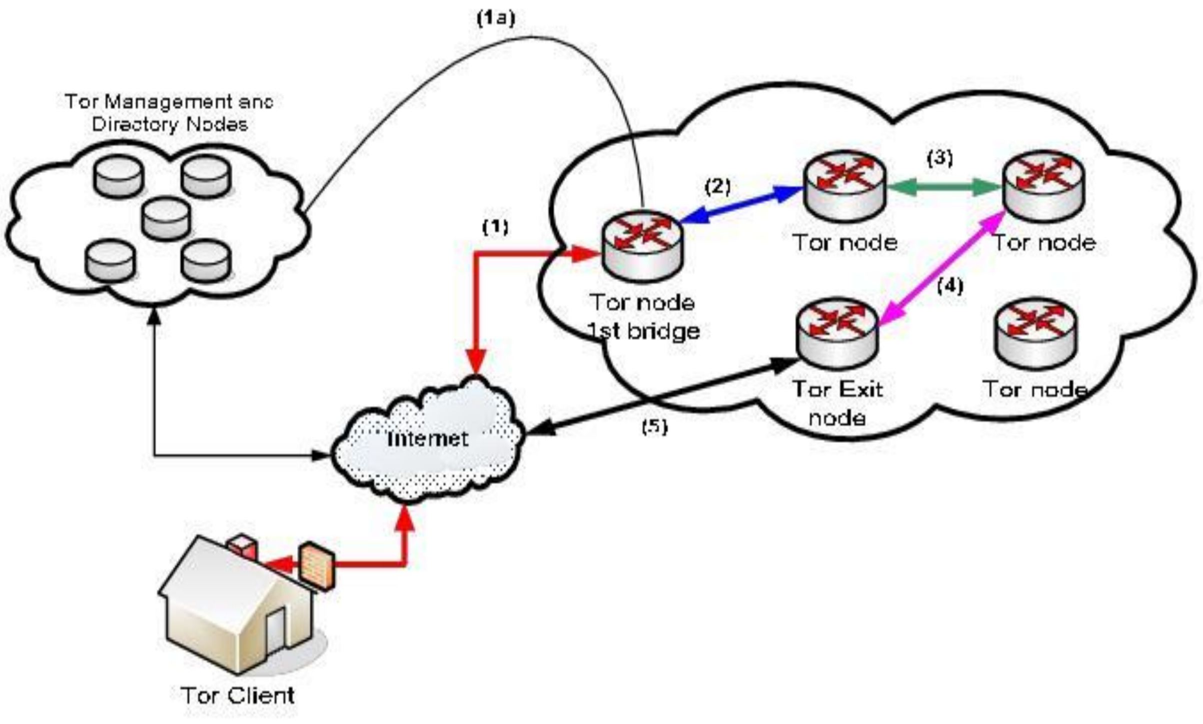
3.1 PROBLEM DESCRIPTION

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.

3.2 DESIGN DESCRIPTION

Our project aims at hosting a Tor Bridge on a Linux Virtual Private Server. This bridge serves as a door to the Tor Network.



4. IMPLEMENTATION

4.1 DESCRIPTION ABOUT THE SECURITY MODULES

4.1.1 SECURITY THREATS DEALT

Creating a tor bridge between you and your VPN provider. A malicious tech at your VPN provider could try an attempt at sending malicious code down that bridge to you. Your ISP wouldn't know you where using Tor, but your VPN provider would, and they could be logging more information than your ISP.

4.1.2 SECURITY MECHANISMS USED

One should connect to the VPN first, then Tor. The other way around there is a potential to leak your real IP address and it exposes your use of Tor to your ISP instead of your use of VPN. It is generally better to only expose your use of VPN to your ISP than a Tor connection.

4.2 SOURCE CODE

We have divided the code into two sections as for the part of hosting our own TOR Network and the other section is Python for visualizing the inbound connections to our Tor Bridge.

Section 1:

4.2.1 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


## 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 debug stderr

## Uncomment this to start the process in the background...
```

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

## 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.

## 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 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 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:

4.2.2 VISUALISATION TOOL WITH PYTHON

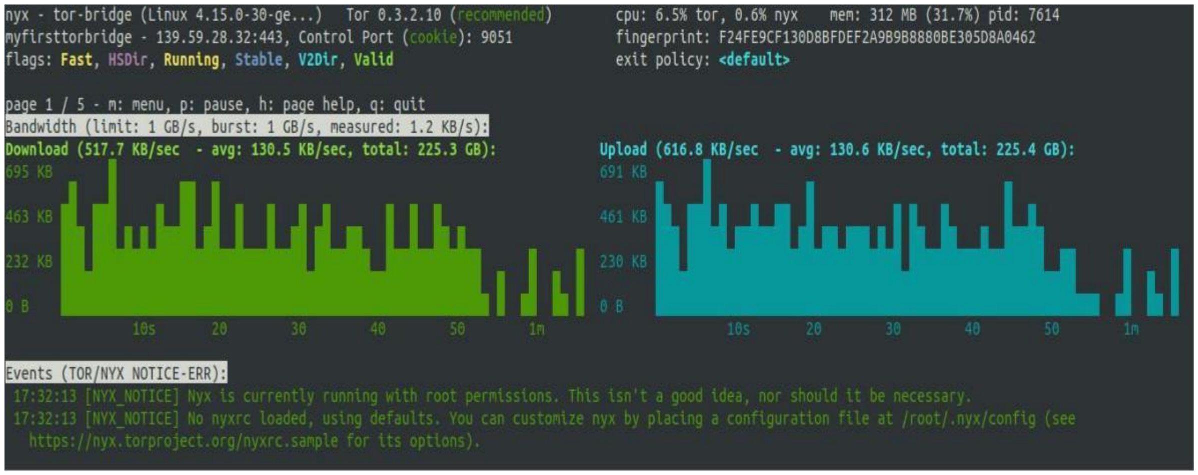
```
import grequests import argparse
import os from platform import
system as os_name import pandas as
pd import requests import sqlite3

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
```

4.3 TEST CASES WITH IMPLEMENTATION SNAPSHOTS

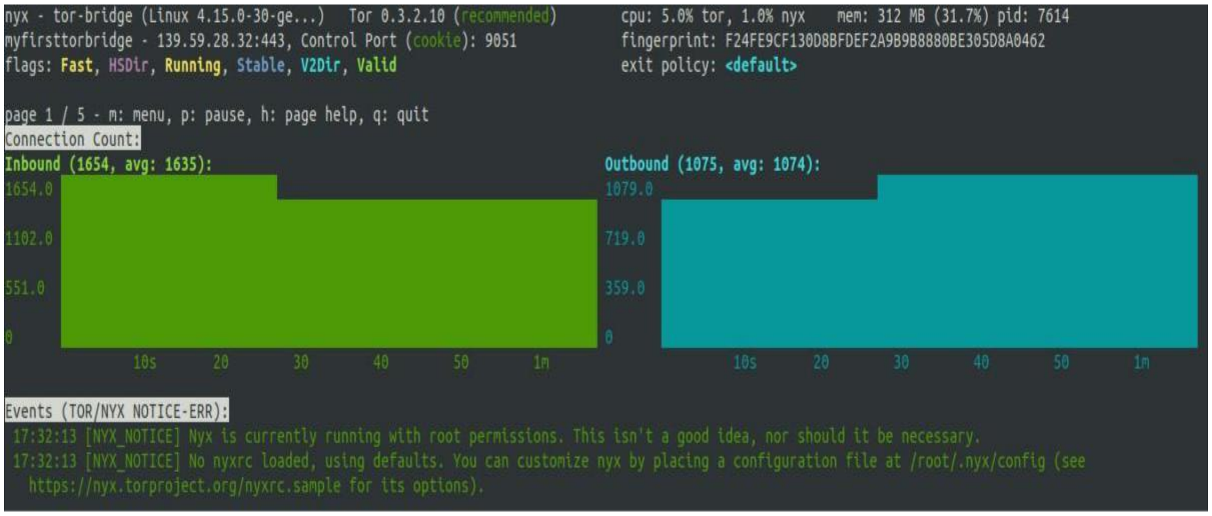
4.3.1 BANDWIDTH ANALYSIS WITH NYX



```
nyx - tor-bridge (Linux 4.15.0-30-ge...)   Tor 0.3.2.10 (recommended)          cpu: 6.5% tor, 0.6% nyx    mem: 312 MB (31.7%) pid: 7614
myfirsttorbridge - 139.59.28.32:443, Control Port (cookie): 9051               fingerprint: F24FE9CF130D8BFDEF2A9B9B8880BE305D8A0462
flags: Fast, HSDir, Running, Stable, V2Dir, Valid                              exit policy: <default>

page 1 / 5 - m: menu, p: pause, h: page help, q: quit
Bandwidth (limit: 1 GB/s, burst: 1 GB/s, measured: 1.2 KB/s):
Download (517.7 KB/sec  - avg: 130.5 KB/sec, total: 225.3 GB):              Upload (616.8 KB/sec  - avg: 130.6 KB/sec, total: 225.4 GB):
695 KB                                                                      691 KB

463 KB                                                                      461 KB

232 KB                                                                      230 KB

0 B                                                                         0 B
            10s     20      30      40      50      1m                                  10s     20      30      40      50      1m

Events (TOR/NYX NOTICE-ERR):
 17:32:13 [NYX_NOTICE] Nyx is currently running with root permissions. This isn't a good idea, nor should it be necessary.
 17:32:13 [NYX_NOTICE] No nyxrc loaded, using defaults. You can customize nyx by placing a configuration file at /root/.nyx/config (see
   https://nyx.torproject.org/nyxrc.sample for its options).
```

Figure 2: Bandwidth Analysis with NYX

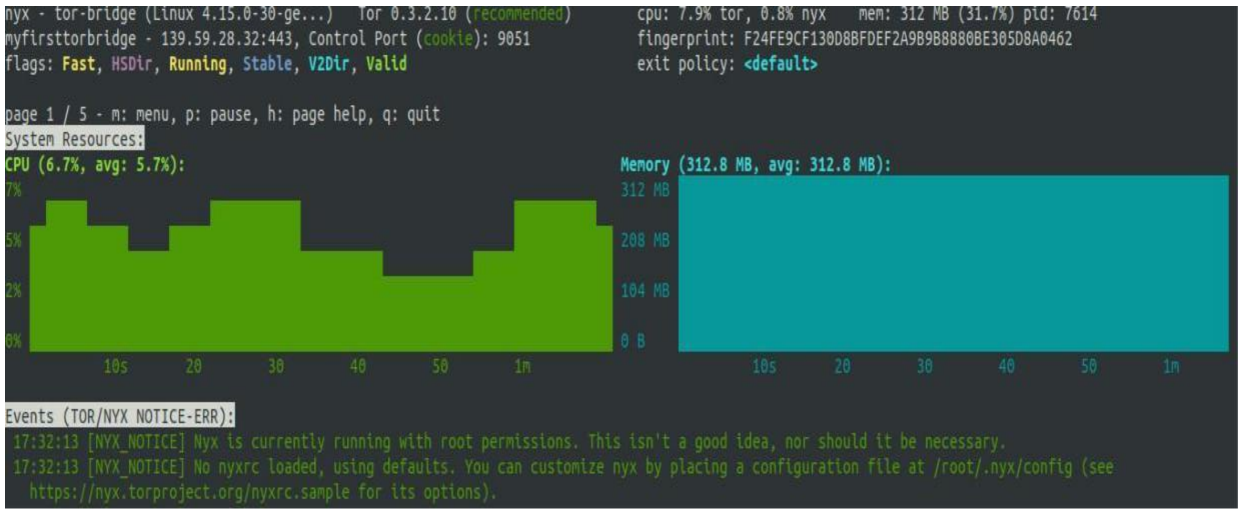
4.3.2 CONNECTION ANALYSIS WITH NYX



```
nyx - tor-bridge (Linux 4.15.0-30-ge...)   Tor 0.3.2.10 (recommended)          cpu: 5.0% tor, 1.0% nyx    mem: 312 MB (31.7%) pid: 7614
myfirsttorbridge - 139.59.28.32:443, Control Port (cookie): 9051               fingerprint: F24FE9CF130D8BFDEF2A9B9B8880BE305D8A0462
flags: Fast, HSDir, Running, Stable, V2Dir, Valid                              exit policy: <default>

page 1 / 5 - m: menu, p: pause, h: page help, q: quit
Connection Count:
Inbound (1654, avg: 1635):                                                  Outbound (1075, avg: 1074):
1654.0                                                                      1079.0

1102.0                                                                      719.0

551.0                                                                       359.0

0                                                                           0
            10s     20      30      40      50      1m                                  10s     20      30      40      50      1m

Events (TOR/NYX NOTICE-ERR):
 17:32:13 [NYX_NOTICE] Nyx is currently running with root permissions. This isn't a good idea, nor should it be necessary.
 17:32:13 [NYX_NOTICE] No nyxrc loaded, using defaults. You can customize nyx by placing a configuration file at /root/.nyx/config (see
   https://nyx.torproject.org/nyxrc.sample for its options).
```

Figure 3: Connection Analysis with NYX

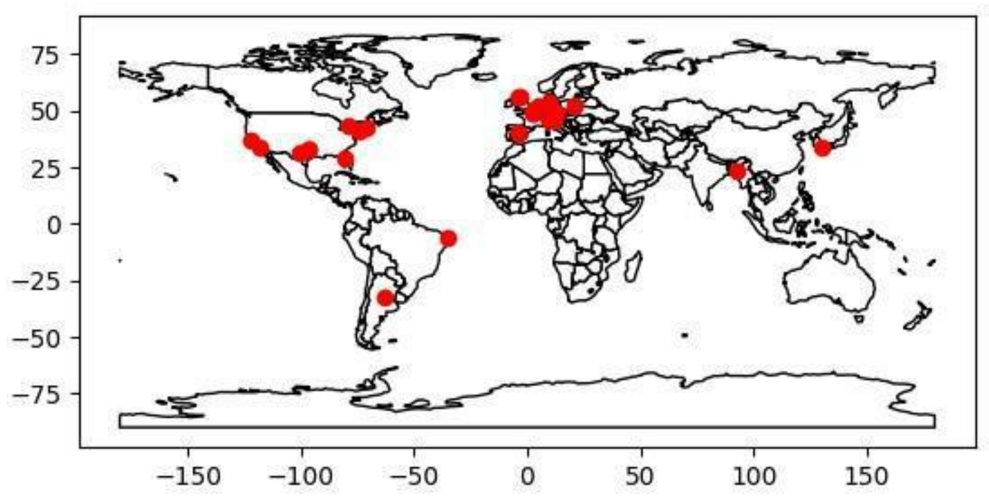
4.3.3 RESOURCE ANALYSIS WITH NYX



```
nyx - tor-bridge (Linux 4.15.0-30-ge...)   Tor 0.3.2.10 (recommended)        cpu: 7.9% tor, 0.8% nyx    mem: 312 MB (31.7%) pid: 7614
myfirsttorbridge - 139.59.28.32:443, Control Port (cookie): 9051            fingerprint: F24FE9CF130D8BFDEF2A9B9B8880BE305D8A0462
flags: Fast, HSDir, Running, Stable, V2Dir, Valid                           exit policy: <default>

page 1 / 5 - m: menu, p: pause, h: page help, q: quit
System Resources:
CPU (6.7%, avg: 5.7%):                                                    Memory (312.8 MB, avg: 312.8 MB):
7%                                                                        312 MB
5%                                                                        208 MB
2%                                                                        104 MB
0%                                                                        0 B
        10s    20    30    40    50    1m                                          10s    20    30    40    50    1m

Events (TOR/NYX NOTICE-ERR):
 17:32:13 [NYX_NOTICE] Nyx is currently running with root permissions. This isn't a good idea, nor should it be necessary.
 17:32:13 [NYX_NOTICE] No nyxrc loaded, using defaults. You can customize nyx by placing a configuration file at /root/.nyx/config (see
   https://nyx.torproject.org/nyxrc.sample for its options).
```

Figure 4: Resource Analysis with NYX

4.3.4 VISUALISATION OF THE INBOUND CONNECTIONS TO PUT TOR BRIDGE ON A WORLD MAP



5. 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.

6. REFERENCES

1. Saleh, Saad & Qadir, Junaid & Ilyas, Muhammad. (2018). Shedding Light on the Dark Corners of the Internet: A Survey of Tor Research. Journal of Network and Computer Applications. 114. 10.1016/j.jnca.2018.04.002.
2. Panait, Ioana-Cristina & Pop, Cristian & Sirbu, Alexandru & Vidovici, Adelina & Simion, Emil. (2016). TOR - Didactic Pluggable Transport. 10006. 225-239. 10.1007/978-3-319-47238-6_16.
3. R. Dingledine and N. Mathewson. Tor protocol specification. https: //gitweb.torproject.org/torspec.git/tree/tor-spec.txt.
4. A. Houmansadr, C. Brubaker, and V. Shmatikv. The Parrot Is Dead: Observing Unobservable Network Communications. In IEEE Symposium on Security and Privacy, 2013.
5. K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Protocol Misidentification Made Easy with Format-Transforming Encryption. In ACM Conference on Computer and Communications Security, 2013