Bachelor's thesis

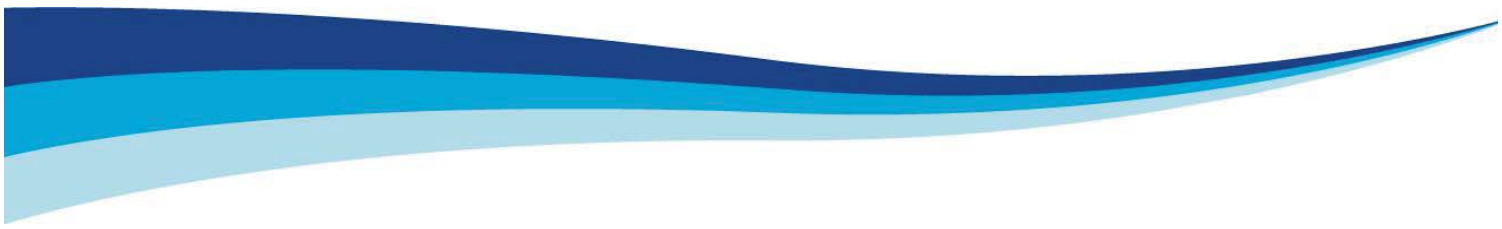
Degree Programme in Information Technology

Internet Technology

2016

Alexandr Vitosinschi

PROTECTING PRIVACY USING TOR



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme in Information Technology | Internet Technology

2016| 50 Pages

Instructor: Ossi Väänänen

Alexnadr Vitosinschi

PROTECTING PRIVACY USING TOR

Most books about network security and secure communication discuss three main topics: availability, confidentiality, and integrity. The focus of this thesis is confidentiality which can be achieved by applying cryptographic algorithms to the message. Nevertheless, it is still possible to track down the source and destination of the message although who is talking to whom can also be sensitive information. There are many technologies that help to preserve privacy online.

One of the most common technologies is The Onion Router or Tor. This thesis focuses on examining the concept of onion routing.

Before discussing the concept of onion routing, first and foremost, this thesis discusses the importance of privacy and the underlying concept of communication between the web-server and client. The final part of this thesis concentrates on configuring a Tor node using Raspberry-Pi and the Google Cloud platform as well as making traffic captures and analysis.

KEYWORDS:

Privacy, Onion Routing, Tor, Internet, HTTP

CONTENTS

APPENDICES

FIGURES

TABLES

DIAGRAMS

LIST OF ABBREVIATIONS (OR) SYMBOLS

DNS	Domain Name System.
HTML	HyperText Markup Language.
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
IRC	Internet Relay Chat
OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
IP	Internet Protocol
UDP	User Datagram Protocol
RFC	Request For Comments
VoIP	Voice over IP
Tor	The Onion Router
TLS	Transport Layer Security
OR	Onion Router
OP	Onion Proxy
VPN	Virtual Private Network
ISP	Internet Service Provider
POP3	Post Office Protocol version 3
IMAP	Internet Message Access Protocol
R-Pi	Raspberry-Pi
IT	Information Technology

1 INTRODUCTION

In the last twenty years, technological advancements made the production of electronic devices cheaper. This had an influence on the development of the Internet. Nowadays many can afford buying a device which has access to the Internet. The phenomenon of the Internet changed the way we live. It has changed the way we communicate, shop, travel, research, look for news and weather forecasts, the way we apply for jobs. Before that, one had to be physically present at the shop to buy a product, or visit a travel agency in order to book a trip. Now it is possible to do all of these just by using a device which has access to the Internet, a web browser, and an Internet connection. It is hard to believe but everything one is doing on the Internet is tracked. The web-browser stores the web-browsing history, while servers contain a log with an IP address and timestamp. In addition, an ISP can eavesdrop on the traffic and see what one is browsing. The Internet is no more anonymous nowadays. Large IT companies, such as Google, Facebook, and Microsoft offer a variety of services for free. However, are those services actually free? The answer is no. Users pay with their privacy. The services offered by the above mentioned companies gather tons of personal data. These include phone numbers, location data, messages, pictures and search history. The problem is the way this data is used. If user personal data is leaked or a third party gains access to the data, it may cause problems to people in their real life. Another problem lies within the government's desire to control the Internet. A good example is China which has placed strong internet censorship and its citizens do not have access to particular resources. Most countries prosecute people for illegal actions on the internet. Tracking down a person works the following way: law-enforcing agencies make an inquiry to the owner of the resource where an illegal action has happened in order to acquire the IP address of the person who committed the illegal action. The IP address reveals the identity of the host who commited the illegal action. Another inquiry is made to ISP in order to reveal the legal

name of the user to whom this IP address is assigned. Technically, this means that all tracking is based on the IP address. In some countries, an illegal action can be a post or a comment about a hot political topic. That is why it is important to maintain privacy and anonymity online in order to preserve the freedom of speech. In order to become anonymous on the internet, it is possible to use VPN, Proxy servers, I2P, Tor, Freenet and many other software packages or technologies. The drawback of Proxy servers and VPNs is that these are centralized systems, meaning that in case the Proxy server or VPN gateway are compromised, it is possible to to trace down the user. Tor is a distributed system and was designed to protect the user's identity as well it is extremely popular. These were main reasons for choosing Tor for this thesis.

The second chapter of this thesis discusses the importance of privacy and how private information makes a user vulnerable. The third chapter is focused on explaining the way communication happens between a client and a server. The third chapter also compares the OSI and the TCP/IP models, and explains the difference between TCP and UDP and the way DNS works and its purpose. The fourth chapter begins with a brief overview of Tor and concentrates on in-deep technical details on how Tor works. Attack vectors on Tor network are described in the end of the fourth chapter. The fifth chapter presents the way to configure a Tor node using a Raspberry-Pi and a tor node using a virtual machine. The end of the chapter describes the process of capturing Tor traffic and analyzing it using Wireshark.

2 PRIVACY

The web-privacy issue is quite young. The phenomenon of the World Wide Web or the Internet has significantly changed the way we live. In the past decade, the technological advancements made our life much easier but, on the other hand, we have to pay a huge price for that by sharing our private information.

The term “Privacy” is defined as “the quality or state of being apart from company or observation or freedom from unauthorized intrusion”[1].
By applying the meaning of term Privacy to the Internet, it means that nobody is spying on oneself when browsing the Internet. In other words, nobody is checking one's browser history or reading the private messages or taking other actions that intrude private life of a person.

One of the main principles of democracy is that “You have the right to have your own beliefs, and to say and write what you think” [2] and “No one can tell you what you must think, believe, and say or not say” [3]. It can be hard for some people to express their beliefs or what they think while they know that they are constantly watched.

2.1 The importance of privacy

Nowadays, many people use smartphones. A good example of tracking is a smartphone. The smartphone has several features, such as GPS, which technically tracks one whenever he or she goes somewhere. The Figure 1 shows where the author has been during Saturday 28th of May 2016. It is a screenshot from the timeline feature in Google Maps.

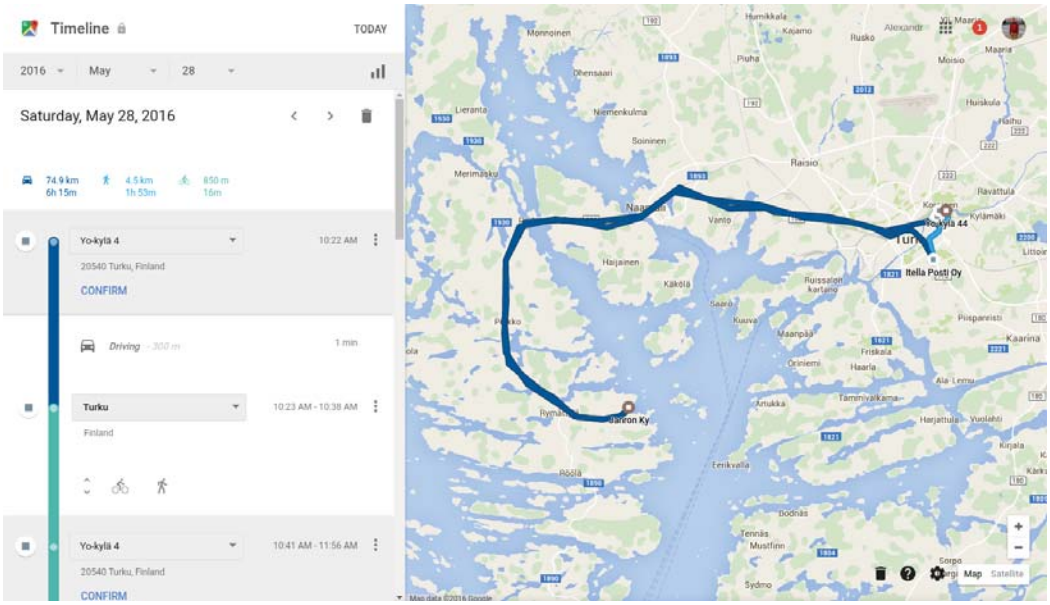


Figure 1. Google Maps timeline

How does that information make one vulnerable? Probably this particular piece of information does not, but there are actionsthat one is doing consistently, such as going to school, work or shopping. Those actions are tracked by the phones on everyday basis. By gathering a huge amount of data, it is possible to find the patterns. In this case, it will be possible to predict where a particular person will be and at what time with quite great accuracy. The issue with private information can be stated in the following question: what happens if this data falls into wrong hands?

The privacy issue also applies to web-browsing. It is possible to extract valuable information based on one's search history or browser history and look for patterns there. The following screenshot (Fig. 2) represents the author's Google search history, for his personal Google account. From the following screenshot, it is possible to find out that author is most probably studying or working in IT industry, who speaks Russian (ru.wikipedia.com) and who is interested in programming (stackowerflow.com) and network engineering (cisco.com) as well as Linux (ask.ubuntu.com).

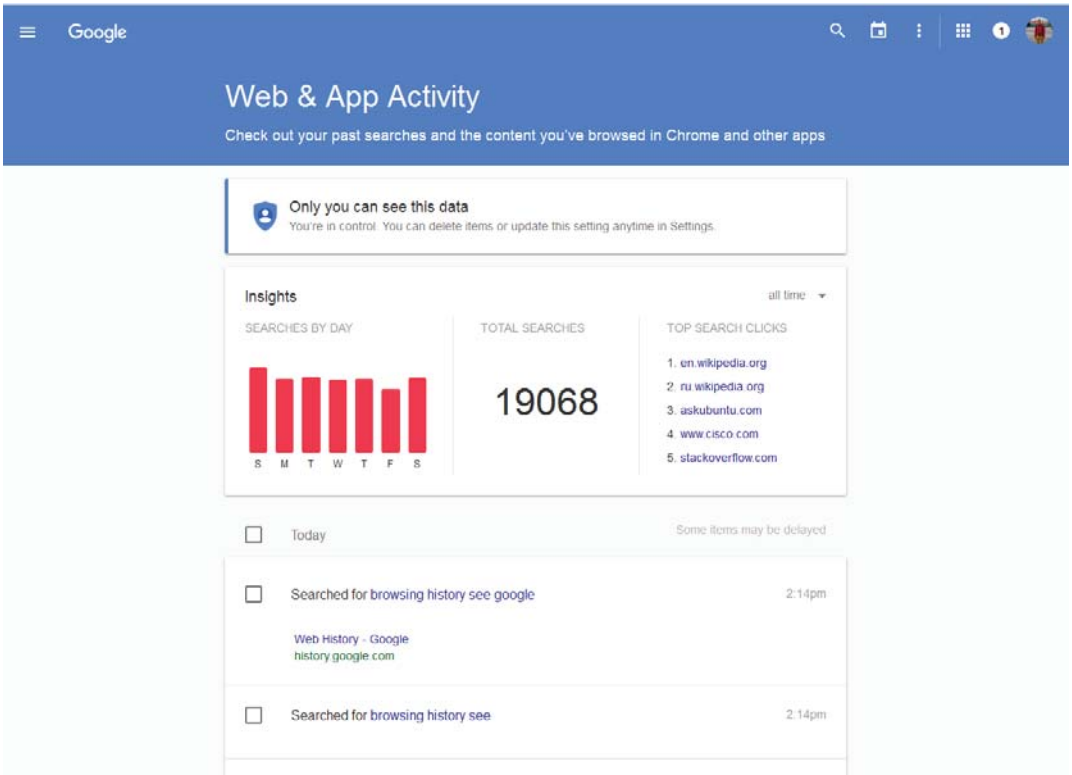


Figure 2. Google search history

There are plenty of open source tools that are designed to protect one's anonymity such as: Freenet, I2P and Tor. All those software packages have their own advantages and disadvantages. The author has choosed Tor because it is the most popular and the fastest, in terms of bandwidth, anonymity software package.

3 CLIENT-SERVER MODEL

Tor software supports various communication application layer protocols, such as: HTTP, SMTP, SSH, IRC and so on [4]. This thesis is focusing mainly on HTTP, thus, it is important to explain the communication model between a web-server and client.

3.1 Client Server communication basics

When discussing communication between two parties, there usually exist a source and destination. Each of them has its own address. When Host A sends a message addressed to Host B, Host A indicates the destination address of Host B and the source address of itself (Fig. 3). This is done in order for Host B to know where to sent the reply message.

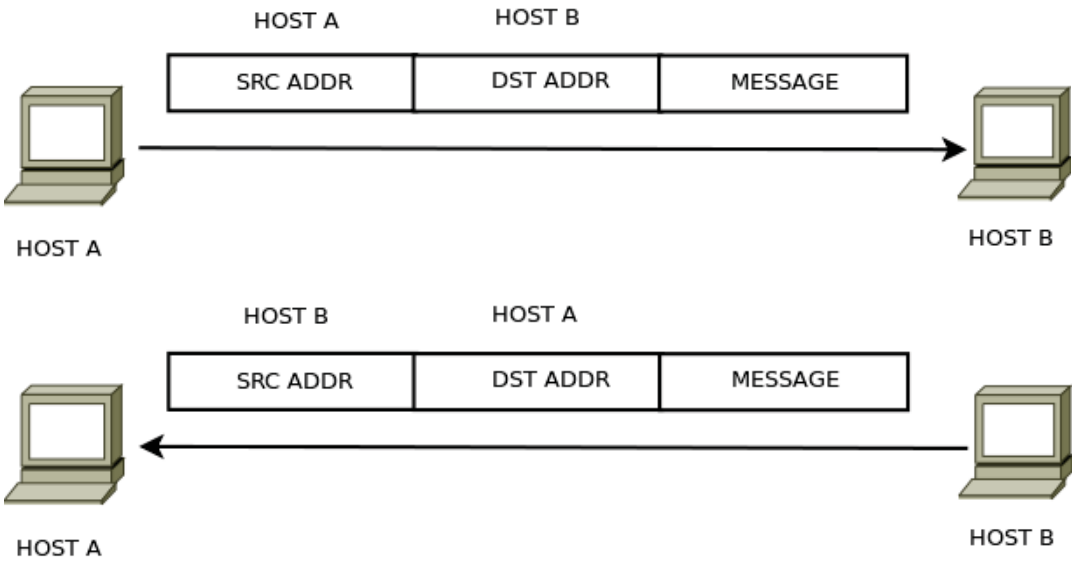


Figure 3. Communication between two hosts

Host B receives the message, processes it, and replies to Host A. The source and destination addresses have changed in the reply message (Fig. 3).

The logic works for client-server communication in the Internet when a user is accessing a web-page.

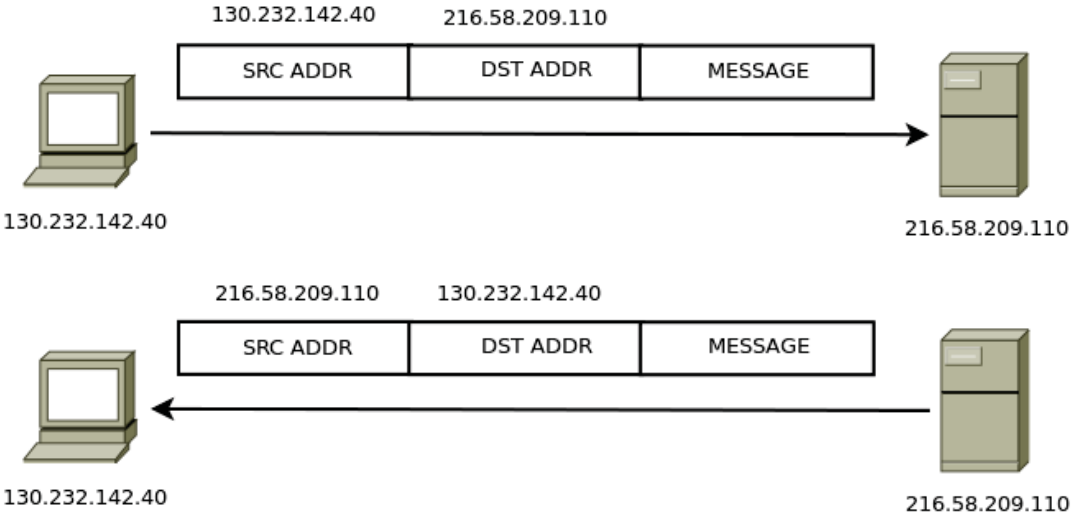


Figure 4. Communication between a client and the server

A user is sending a request and indicates its address as source. The server receives the request, processes it and sends a reply to the user. The IP address unveils the identity of the user, the server also logs the request, thus the requester is no more anonymous in the case described in Figure 4.

3.2 OSI and TCP/IP moldels

When talking about communication models, it is necessary to mention the OSI (Open System Interconnection) and the TCP/IP (Transmission Control Protocol/ Internet Protocol) models.

Communication models are used to explain how computers communicate with each other, while protocols are set of rules used for communication.

The following figure (Fig.5) compares the two above mentioned models and protocols used.

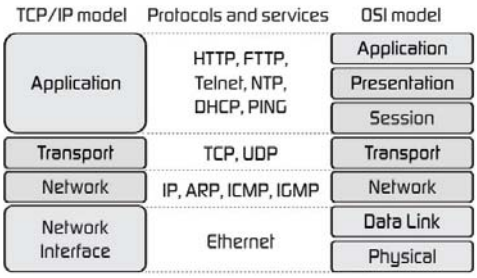


Figure 5. TCP/IP vs OSI model

Although the two models are quite similar, the TCP/IP is used in production while OSI is used as reference. A more detailed explanation of OSI and TCP/IP models is out of the scope of this thesis and can be found in the ITU-T Recommendation X.200 (11/93) [ISO/IEC 7498-1:1994] [5] and RFC1180[6] respectively.

3.3 Encapsulation

In most cases when two hosts are communicating over the network, there is an application which generates data. An example of such an application is a web-browser. It generates a web request. Then this data is passed to the Transport layer. The concept of encapsulation is used in this case, meaning that it adds a header and sometimes a trailer to the data.

The following figure (Fig. 6) demonstrates the UDP encapsulation concept:

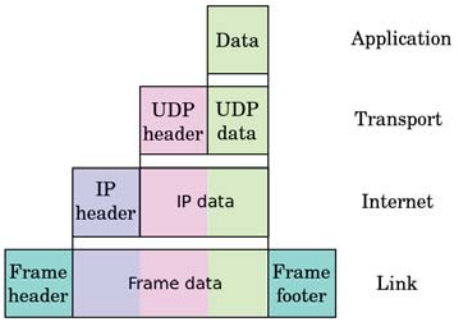


Figure 6. Encapsulation concept

3.4 Transport Control Protocol (TCP) vs User Datagram Protocol (UDP)

Two most common transport layer protocols are TCP and UDP [7]. There is significant difference between these two. The UDP is lightweight and connectionless, while TCP is connection-oriented, meaning that TCP must establish a connection between the two parts before the data can be sent [8]. In other words, when an application uses TCP as Transport, it needs to receive a

confirmation that the destination has successfully received the message. UDP does not support this feature and is thus considered unreliable.

TCP is used when reliability is needed. Applications such as VoIP (Voice Over IP) are loss tolerant and use UDP as the Transport layer protocol. TCP establishes the connection to the server, using the three way handshake (Fig. 7):

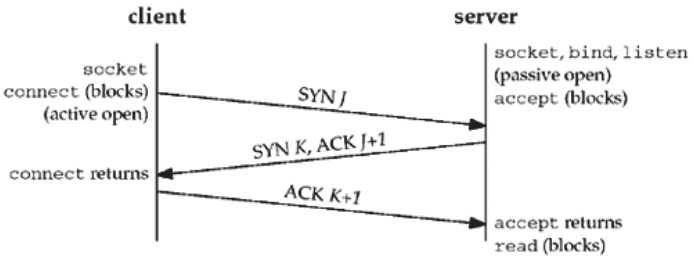


Figure 7. Three-way handshake

The difference between TCP and UDP becomes more obvious when comparing the headers of both protocols in Figures 8 and 9:

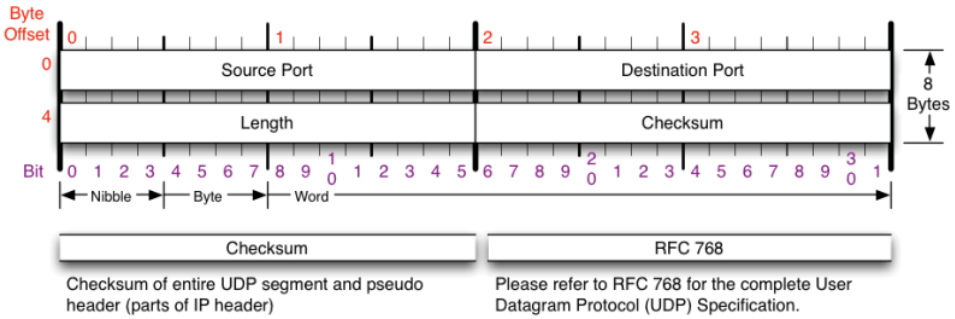


Figure 8. UDP header

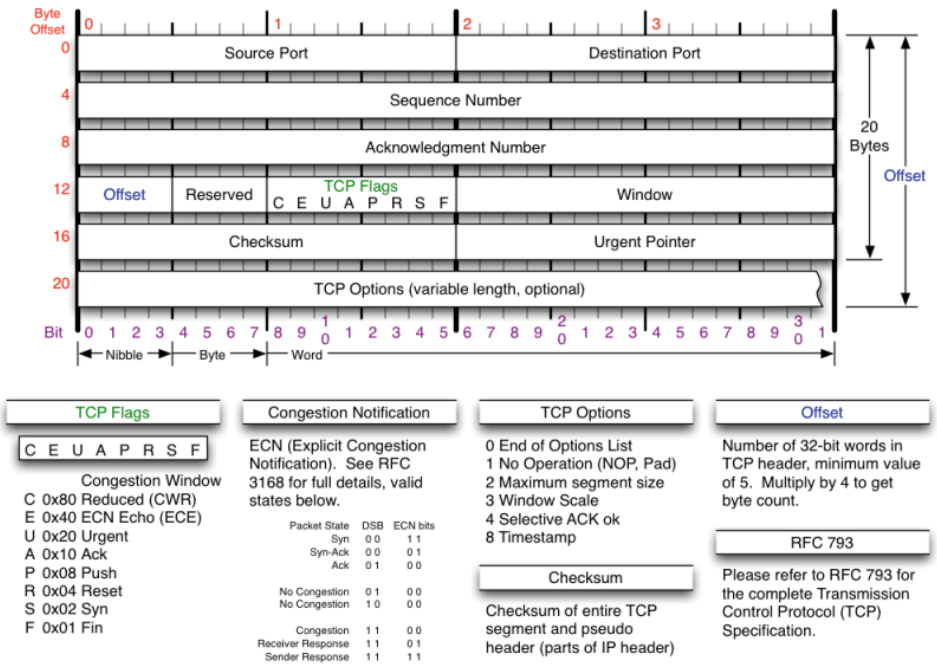


Figure 9. TCP header

It is possible to notice that TCP has a longer overhead and more options than UDP. The notable fields in both protocol headers are the Source and Destination ports. The port defines a service which runs on the server. There is a list of well-known ports described in RFC 1700 [9]. The HTTP server is using port 80 as default. The combination of port and IP address is called a socket.

After data has been processed at the Transport layer it is passed to Internet or the Network layer. The Network layer is concerned about adding the source and destination IP addresses as the header. Afterwards, the packet is passed to Data Link layer and transmitted but the Data Link layer is out of scope of this thesis.

3.5 Domain Name System

The web-browsing starts when a user inputs the address of a web-site in the address bar and hits Enter. In order to successfully communicate with the server, the client needs to know servers IP address. This function is

implemented by the Domain Name System (DNS) [10]. It is used for looking up the IP address based on domain name, such as www.lib.ru

The Figure 10 is a Wireshark capture of a DNS request:

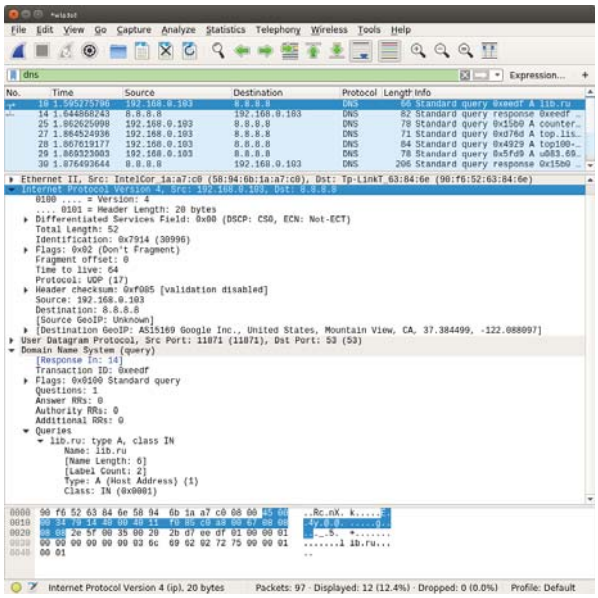


Figure 10. DNS request

In response, the DNS replies with the IP address of the requested domain:

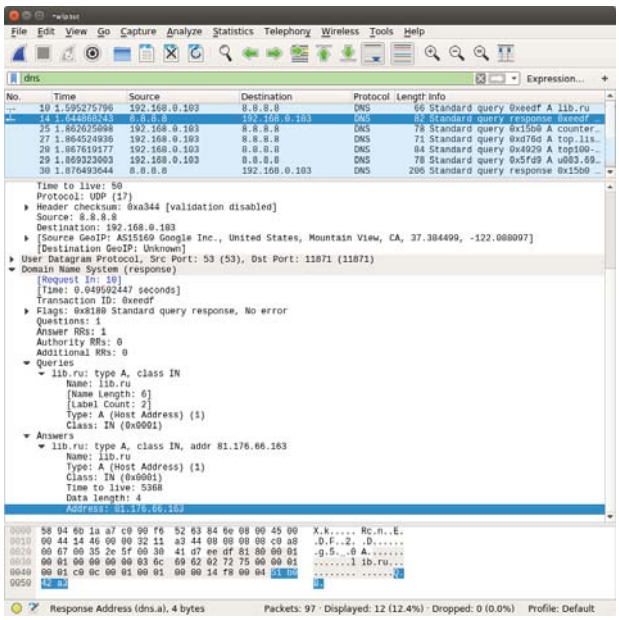


Figure 11. DNS reply

3.6 Hyper Text Transfer Protocol

The client machine obtains the IP address of the www.lib.ru server which is 81.176.66.163

Now the HTTP GET request can be formed.

The following figure (Fig. 12) represents an HTTP GET request:

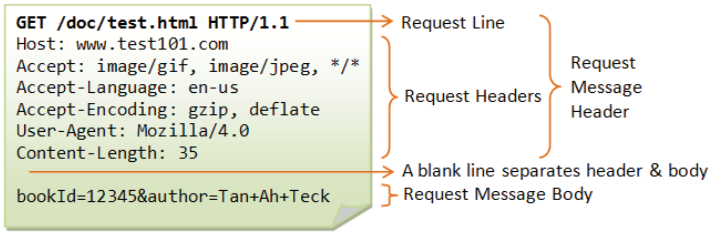


Figure 12. HTTP GET request

The client gets the following reply (Fig. 13) if the request was successful:

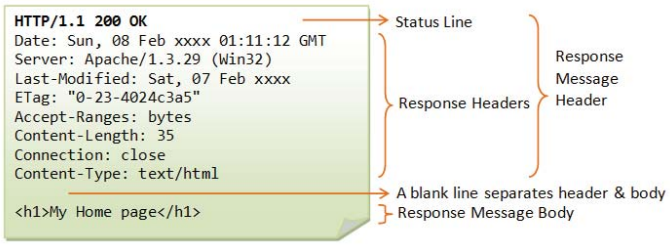


Figure 13. HTTP Response

When the HTTP request if formed, it is passed to the Transport layer. The Transport layer establishes the connection to the server using TCP in this case. It uses the source and destination IP address to do that. The source is the IP address of the host and the destination is the IP address of the server. Thus, the server is able to track down who was accessing the the particular service at particular time because it is able to identify the IP address of the originator.

4 ONION ROUTING

After briefly describing the underlying technologies behind the client server communication model used by HTTP, it is possible to proceed to the concept of Onion Routing and Tor.

4.1 Onion Routing history

The history of Tor network starts with the development of the concept of Onion Routing. The concept of Onion Routing was designed by the U.S. Naval Research Laboratory in the mid-1990s [11]. Tor was designed in 2002 and it is known as “Second generation onion routing” . The purpose behind the Tor is to protect the anonymity of its users. Tor, presented as low-latency communication service, considering the time it takes to apply all cryptographic operations and to push the packet through the network, it is considerably slower than typical connection which does not use Tor network, but still the system provides a reasonable trade-off between reliability, efficiency and anonymity. Using Tor makes it more difficult to observe the internet activity of a user which includes web-site visits, e-mails and other forms of communication [12].

4.2 Onion Routing overview

Onion Router connections are protocol independent and consist of three phases: *connection setup*, *data movement* and *connection teardown*. [13]

- *Connection setup*: it is the first stage where the client machine or initiator creates a socalled onion which defines the route of the packet, in other words, the initiator chooses the Tor nodes that the packet will travel through.
- *Data movement*: it is the second stage when the client pushes the data through the Tor network.

- *Connection teardown*: is the third phase when the connection is closing after the client chooses to not move the data through the Tor network.

Onion by itself is a layered data structure which defines the properties of the connection at each node along the path such as: cryptographic algorithms and keys that shall be used during data movement phase.

Every onion router along the path uses its public key to decrypt the entire onion that it receives and its duty is to pass data from one connection to another after applying some specific cryptographic operation.

The Tor network consists of nodes. There are three types of nodes: *guard node*, *relay node,* and *exit node* (Fig. 14).

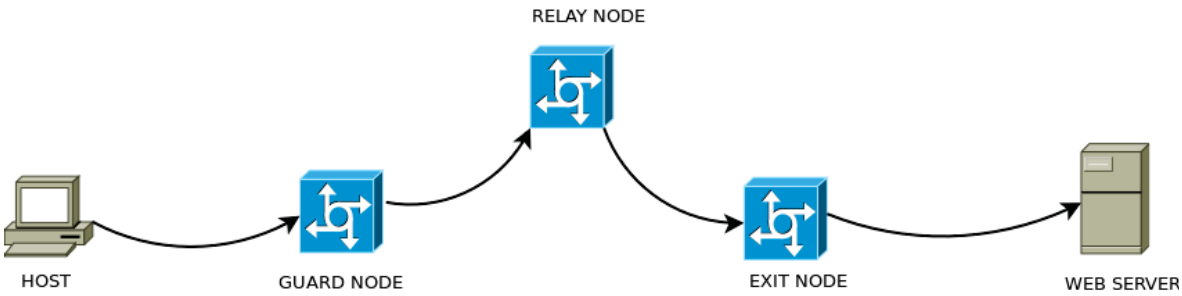


Figure 14. Tor topology

- The guard node is the only node that knows the host's identity. It acts as an entry point to the Tor network.
- The relay node is used to relay the packets to the exit node, which makes the system more secure to identity revealing attacks.
- The exit node is used to relay the traffic to the requested resource.

Taking out the technical details, the Tor network works as follows. Figure 15 is a graphical representation of Tor topology:

1. Host gets a list of available nodes from the Tor directory (request is encrypted)
2. Host selects the nodes and creates the circuit
3. Host relays the packets through the Tor network
4. The packets received by the requested resource have a source IP address of the exit node.

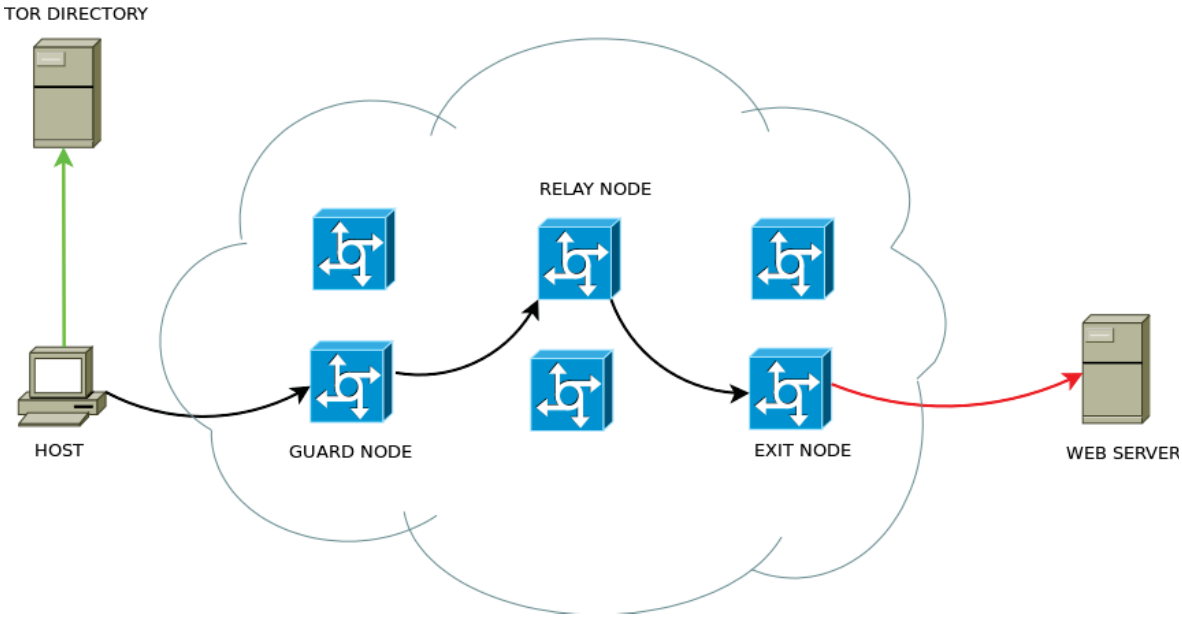


Figure 15. Tor topology including Tor directory

In addition,Tor is used to hide the identity of the resource. This is called Hidden service. It has a special domain name. The logic behind the hidden service is that it uses a node called Rendezvous point which is used to secure the identity of the initiator and the resource as well. The initiator uses an identifier, which is a 16-character name derived from server's public key [14].

4.3 Encryption in Tor

Tor is a distributed relay network designed to make TCP connections anonymous. The initiator chooses the path through the Tor network and builds a so called circuit in which each Onion Router knows only a next-hop node and previous-hop node but has no information about other nodes involved in the circuit. Tor is a layered network. Each node in the circuit does not require any special privileges to run the Tor process. Every node maintains a TLS connection to every other node. Onion Proxy (OP) is a process run by the user. The scope of OP is to establish circuits across the onion network and manage connections from the application layer. [15]

There are two keys that are managed by the onion router. The first one is a long-term identity key which is used to sign the TLS certificates and the onion router's description such as: bandwidth, address, exit policy etc. The second one is a short-term onion key which is used to set up a circuit, decrypt user requests and negotiate ephemeral keys.
The TLS protocol is also using the short-term keys to establish a short-term link key when communicating between the onion routers. The keys are rotated independently and periodically in order to mitigate the risk in case the keys are compromised.

The unit of communication in the Tor network is a fixed-size cell.

ORs use TLS connections to communicate with each other and with user's OPs. The choice of TLS here is dictated by the following criteria:

- TLS encrypts the data
- TLS prevents data to be modified

4.4 Tor cell structure

The traffic in ToR network is passing in fixed-size (512 bits) cells. The Figure 16 represents the structure of a Tor cell:

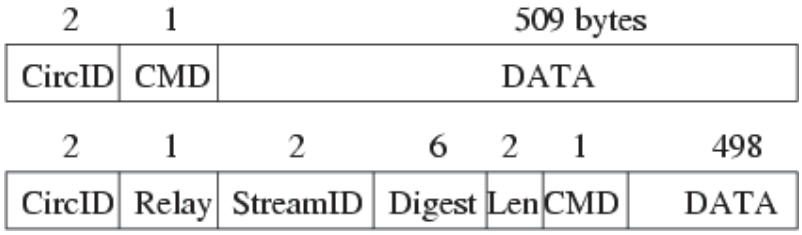


Figure 16. Tor control and relay cell structure

The cell is divided into two parts: *header* and *payload*. The header consists of circuit ID which helps identifying which circuit does this particular cell belong to, because a single TLS connection can be carrying cells from multiple circuits and command which is used to tell the OR what particular action should it apply to the cell

Based on the command, the cell can be either a *control cell* or a *relay cell*. The control cell commands are: *create*, *destroy* or *padding*. *Create* stands for creating a circuit, *destroy* for destroying it and *padding* is used for the keepalive messages. Relay cells have an additional streamID header, a Digest header (checksum) and a Len header (size of payload). The contents of the cell are encrypted or decrypted using the 128-bit AES cipher as the cell travels between the nodes. The CMD header contains a relay command which can be one of the following:

- Relay data
- Relay begin
- Relay end
- Relay teardown
- Relay connected
- Relay extend
- Relay truncate
- Relay sendme
- Relay drop

4.5 Tor circuit construction process

The initial design of onion routing implies that each TCP stream will have its own OR circuit. Building a circuit for each TCP stream is costly in terms of computing resources due to applying cryptographic operations. The Tor design implies that one circuit can be shared by multiple TCP streams. To minimise linkability between their streams, the user's OP builds a new circuit periodically, usually once in a minute[16].

The Figure 17 represents the way a user's OP constructs a circuit:

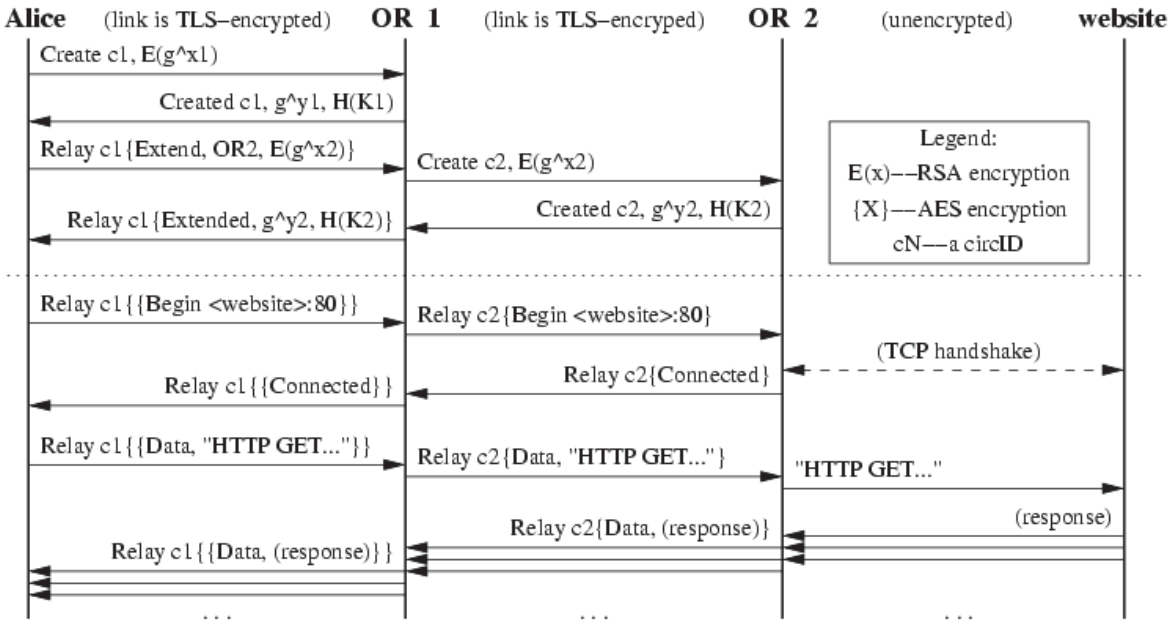


Figure 17. Tor key-exchange process

The circuit construction process is incremental. Each node has to negotiate a symmetric key with the other one. Firstly, Alice's OP creates a circuit with the OR1. Then OR1 creates a circuit with OR2 and forwards the information about newly created circuit to Alice. The only node which knows all the keys that are used to communicate between nodes is Alice. This is done to prevent man-in-the-middle attacks, in case a node is compromised. Each OR in the circuit knows only its key [17].

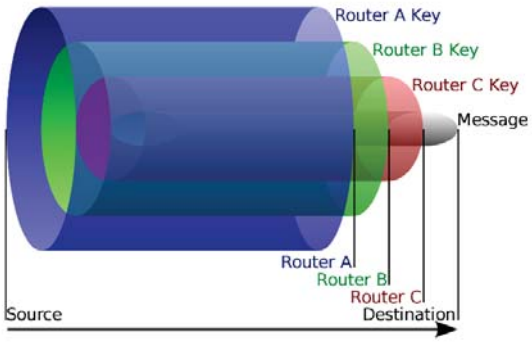


Figure 18. Layered encryption concept

When Alice has established the circuit, she can start sending relay cells. When an OR receives a relay cell, it looks up the appropriate circuit and decrypts the cell header and payload in order to extract the session key for this particular circuit. Then the OR checks if the digest is valid. In case it is, the cell is processed further. OPs treat incoming relay cells similarly: they unwrap the relay header and payload with the session keys shared with every other OR in the circuit. If the digest is valid, the cell must have originated at the OR whose encryption has just been removed.
When the OR replies to Alice, it uses the key shared with Alice to encrypt the relay header and payload and sends the cell towards Alice along the circuit. The next OR to receive the cell adds a further layer of encryption.
In order to tear down the circuit, Alice needs to send a destroy cell. When this cell is received by the OR, it closes the circuit. [18]

4.6 Comparing Tor network with VPN

It is possible to compare the Tor network with VPN. Both of those services provide anonymit, but Tor is distributed. Thus it provides more security in case a node is compromised [19].In the case of VPN, all the traffic generated by the client machine goes through the VPN gateway. Most likely the connection between the client and the VPN gateway is encrypted. At the VPN gateway, data is decrypted and sent to its destination.

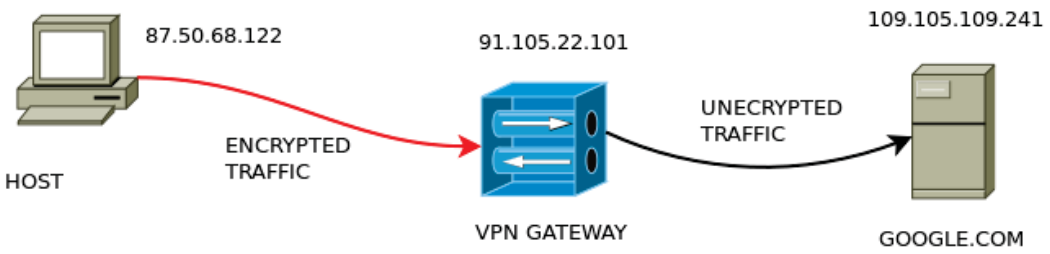


Figure 19. VPN topology

The VPN gateway in the figure above (Fig. 19) acts as a trusted third party, but what happens if the VPN gateway is compromised? The answer is simple, the connections can be tracked down, and it is possible to uncover the identity of the host.

The Tor network is distributed. Each OR uses its own encryption key which is known only to the specific OR and to the initiator. In this case, if a Tor node is compromised, it is still not possible to see raw data because there are used at least three layers of encryption.
The Figure 20 is a representation of the Tor network:

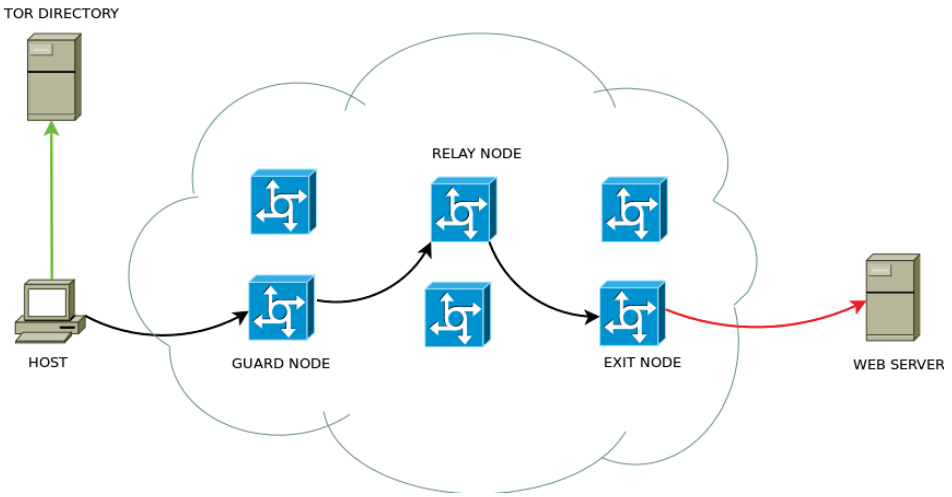


Figure 20. Tor topology

All the communication between the nodes is encrypted, as well as the link between the host and the guard node. The sole unencrypted link is between the exit node and the web-server. However, by eavesdropping traffic at exit node, it is still not likely to discover the identity of the host.

4.7 Attack vectors on Tor

The attacks on the Tor network can be divided into two categories: passive and active attacks.
The passive attacks are:

- *Observing user traffic patterns*

  In case there is a compromised Tor node, it is possible to eavesdrop the packets that are relayed by it but because of layered encryption, this will not reveal the source or destination of data, neither will it reveal the information contained in the message. On the other hand, this method reveals the traffic patterns with further processing. This gives an attacker the opportunity to reveal a user's identity although this requires to have more than one compromised Tor node and a great deal of processing.

- *End-to-end timing correlation*

  End-to-end timing correlations are barely hidden by Tor. This requires to process the traffic patterns and it can end up in revealing the initiator and the responder’s identity.

- *End-to-end size correlation*

  Size correlation attacks are based on simple packet counting.

Active attacks are:

- *Compromise keys*

  An attacker who knows the TLS session key is able to see the control cells and the encrypted relay cells and unwrap one layer of encryption, but a periodic key rotation limits the risk of this attack.

- *Run a recipient*

  An attacker who runs a web server can learn the timing patterns of the users which connect to it and can introduce random patterns to the responses.

- *Run a hostile OR*

  A compromised node can create circuits through itself or alter traffic patterns to affect traffic at other nodes.

- *Tagging attacks*

  A compromised node could tag the traffic, but integrity checks prevent this attack.

- *Replacing contents of unauthenticated protocols*

  A compromised exit node can impersonate the target resource, such as the web-server, thus causing the unauthenticated protocol such as HTTP to believe that it is the real requested resource. The initiator should be using the protocols with end-to-end encryption such as HTTPS.

- *Replay attacks*

  There are protocols that are vulnerable to replay attacks, but Tor is resistant to this kind of attacks because it will result in different negotiated session keys.

5 CONFIGURING A TOR NODE

The practical part of this thesis will focus on configuring a Tor node using Raspberry-Pi and capturing traffic. It is possible to run a Tor node as bridge, relay, or exit.

5.1 Types of Tor nodes

There are three types of Tor nodes in the network: *a bridge node*, *a relay node* and *an exit node*. This practical part concentrates on configuring a relay and an exit node, as the only difference between the two is that the relay does not have an exit policy.

A bridge is a node that is not listed in theTor database. It lets other users to access normal relays. There is no publicly available list of all bridges, so it is impossible to say how many there are out there on the Internet. The main purpose of the bridge is to help people in such countries like China where Tor is being blocked by the ISPs.
Running a bridge will use a small amount of traffic. The following configuration makes a Tor node run as a bridge.

```
vim /etc/tor/torrc

      SocksPort 0
      ORPort 9001
      BridgeRelay 1
      ExitPolicy reject *:*
```

A relay node is used to relay the packets in the Tor network from initiator to the exit. There is a special relay called guard node, which the initiator uses to enter the Tor network.

The following configurations should be applied to the /etc/tor/torrc file in order to make it run as a Relay node.

```
ORPort 9001
Nickname tortestnode
RelayBandwidthRate 512 KB
RelayBandwidthBurst 1024 KB
ContactInfo alexandr.vitosinschi@edu.turkuamk.fi
ExitPolicy reject *:*
```

An exit node is a special node where traffic leaves the Tor network and it is forwarded to the internet. The exit nodes are an extremely important part of the Tor network because they are the linking point between the Tor network and the Internet. The tricky part in running a Tor exit node is that Tor can be used for illegal actions and the traffic will appear to be generated from a random user's location. This can cause problems with law-enforcement agencies.
In order to configure a Tor exit node, the following configuration should be applied to /etc/tor/torrc:

```
ORPort 9001
Nickname tortestnode
RelayBandwidthRate 512 KB
RelayBandwidthBurst 1024 KB
ContactInfo alexandr.vitosinschi@edu.turkuamk.fi
ExitPolicy accept *:20-23     # FTP, SSH, telnet
ExitPolicy accept *:53        # DNS
ExitPolicy accept *:79-81     # HTTP
ExitPolicy accept *:110       # POP3
ExitPolicy accept *:143       # IMAP
ExitPolicy accept *:194       # IRC
ExitPolicy accept *:220       # IMAP3
ExitPolicy accept *:443       # HTTPS
ExitPolicy reject *:*
```

Where `ORPort` is port which Tor service is listening for incoming connections, `Nickname` is the name of Tor node, `RelayBandwidthRate` defines the bandwidth which can be used by Tor to relay traffic, `RelayBandwidthBurst` is the peak bandwidth which can be used by Tor to relay traffic, `ExitPolicy` is used to relay traffic from Tor network to the Internet. It is possible to tweak the

exit policy in order to allow specific traffic to specific resources. For example, to allow web traffic to www.lib.ru, it is required to use the following exit policy:

```
ExitPolicy accept 81.176.66.163:80
```

Where 81.176.66.163 is the IP address of lib.ru server and 80 is the HTTP port. To allow the web traffic to any destination, it is required to configure an exit policy in the following way:

```
ExitPolicy accept *:80
```

The asterisk (*) stands for `any` in configuration file. It is also possible to configure the hidden service using Tor which is a server with a hidden IP address. The hidden service cannot be accessed through the internet but only through the Tor network.

5.2 Configuring theTor node using Raspberry-Pi

As mentioned before, the author used the Raspberry-Pi 3 to configure the Tor node. The first step is installing OS. It is possible to download a Debian Linux (Raspbian) distribution for R-Pi from the official website[20] and write it to an microSD card which is afterwards plugged into R-Pi. If everything is configured properly, the user will be able to access R-Pi through SSH.

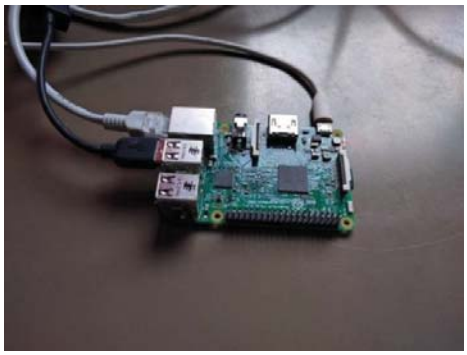


Figure 21. Raspberry-Pi 3

The author connected R-Pi (Fig. 21) to local router, accessed it by SSH and configured so it could support the following topology (Fig. 22):

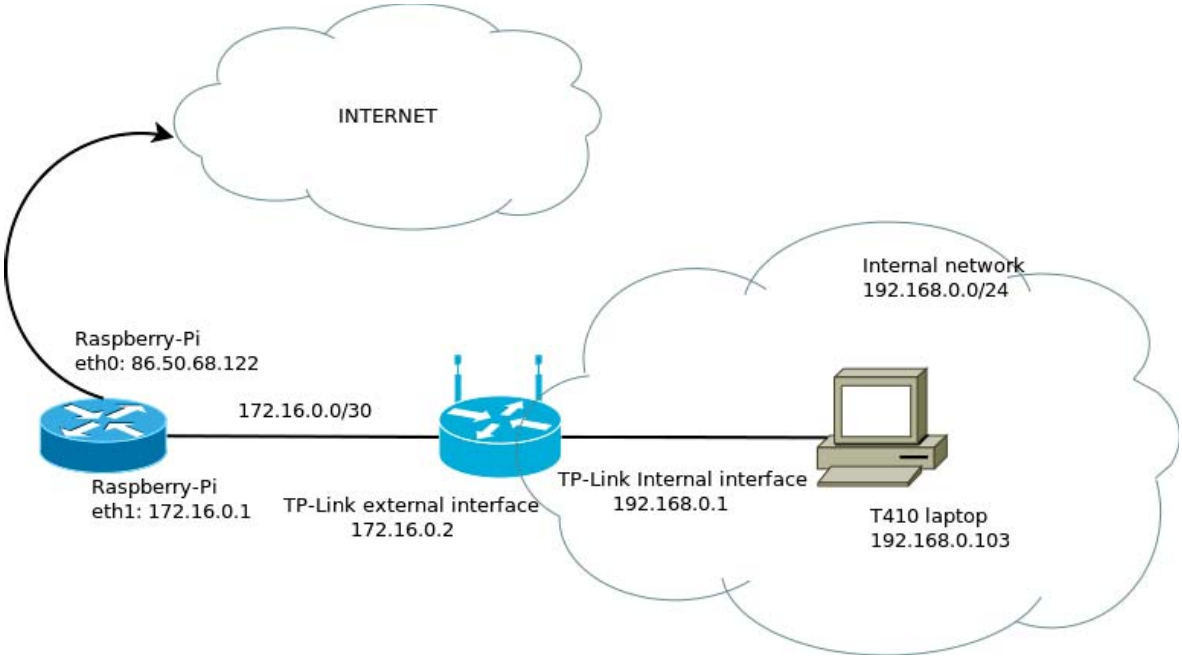


Figure 22. Home network topology

Configuring the network interfaces on Raspberry-Pi was carried out as follows:

```
pi@raspberrypi:~ $ sudo ifconfig eth1 172.16.0.1

pi@raspberrypi:~ $ sudo ifconfig eth1 netmask 255.255.255.252

pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:73:bf:3e
          inet
addr:86.50.68.122  Bcast:86.50.69.255  Mask:255.255.254.0
          inet6 addr: fe80::b0b0:9267:5b21:694a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23342 errors:0 dropped:32 overruns:0 frame:0
          TX packets:448445 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5116457 (4.8 MiB)  TX bytes:28085876 (26.7 MiB)

eth1      Link encap:Ethernet  HWaddr 00:b5:6d:00:98:db
          inet addr:172.16.0.1  Bcast:172.16.0.3  Mask:255.255.255.252
          inet6 addr: fe80::1a60:949c:5a93:9d72/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:373 errors:0 dropped:0 overruns:0 frame:0
          TX packets:293 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:51985 (50.7 KiB)  TX bytes:58301 (56.9 KiB)
```

It is required to edit the `/proc/sys/net/ipv4/ip_forward` file and change the `0` to `1` in order to enable forwarding feature.

Second step is to configure the iptables to do the NAT[21]:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o eth1 -m state --state
RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

Third step in the configuration process is installing the Tor:

```
pi@raspberrypi:~ $ sudo apt-get update
pi@raspberrypi:~ $ sudo apt-get install tor
```

After the installation process is completed it is possible to start editing the configuration file which is based in the `/etc/tor` directory

```
pi@raspberrypi:~ $ vim /etc/tor/torrc
```

In our case, the author configured the Tor node with the following settings[24]:

```
ORPort 9001
Nickname testnodetest
RelayBandwidthRate 150 KB
RelayBandwidthBurst 250 KB
ContactInfo alexandr.vitosinschi@edu.turkuamk.fi
ExitPolicy accept *:20-23     # FTP, SSH, telnet
ExitPolicy accept *:53        # DNS
ExitPolicy accept *:79-81     # HTTP
ExitPolicy accept *:110       # POP3
ExitPolicy accept *:143       # IMAP
ExitPolicy accept *:194       # IRC
ExitPolicy accept *:220       # IMAP3
ExitPolicy accept *:443       # HTTPS
ExitPolicy reject *:*
```

After applying changes it is required to restart the Tor service. It is done using the following command:

```
pi@raspberrypi:~ $ sudo service tor restart
```

It is a good practice to check the if Tor service is running. It is done by the following command:

```
pi@raspberrypi:~ $ service tor status
● tor.service - LSB: Starts The Onion Router daemon processes
   Loaded: loaded (/etc/init.d/tor)
   Active: active (running) since Fri 2016-06-03 20:49:36 UTC;
14h ago
  Process: 9473 ExecReload=/etc/init.d/tor reload (code=exited,
status=0/SUCCESS)
  Process: 939 ExecStart=/etc/init.d/tor start (code=exited,
status=0/SUCCESS)
```

```
  CGroup: /system.slice/tor.service
          └─969 /usr/bin/tor --defaults-torrc
/usr/share/tor/tor-service-def...
```

The author has used tcpdump in order to capture the traffic which goes through the Tor node.

```
tcpdump -i eth0 -s 65535 -w /home/pi/capture.cap
```

At this point, it is a good idea to check the Tor log file. Although it may seem that the process is running, there is a possibility that the node can not be accessed by other nodes. It is necessary to check the log to verify if the service is running.

```
Jun 04 12:54:20.000 [notice] Tor 0.2.5.12 (git-3731dd5c3071dcba) opening log
file.
Jun 04 12:54:20.000 [notice] Parsing GEOIP IPv4 file /usr/share/tor/geoip.
Jun 04 12:54:21.000 [notice] Parsing GEOIP IPv6 file /usr/share/tor/geoip6.
Jun 04 12:54:21.000 [notice] Configured to measure statistics. Look for the *-
stats files that will first be written to the data directory in 24 hours from
now.
Jun 04 12:54:21.000 [notice] Caching new entry debian-tor for debian-tor
Jun 04 12:54:21.000 [notice] Caching new entry debian-tor for debian-tor
Jun 04 12:54:23.000 [notice] Your Tor server's identity key fingerprint is
'testnodetest 16EA88ED4AC04AF14FB810CECC327C0CD2A97C99'
Jun 04 12:54:23.000 [notice] Bootstrapped 0%: Starting
Jun 04 12:54:28.000 [notice] We now have enough directory information to build
circuits.
Jun 04 12:54:28.000 [notice] Bootstrapped 80%: Connecting to the Tor network
Jun 04 12:54:29.000 [notice] Guessed our IP address as 86.50.69.28 (source:
194.109.206.212).
Jun 04 12:54:30.000 [notice] Bootstrapped 85%: Finishing handshake with first hop
Jun 04 12:54:30.000 [notice] Bootstrapped 90%: Establishing a Tor circuit
Jun 04 12:54:30.000 [notice] Tor has successfully opened a circuit. Looks like
client functionality is working.
Jun 04 12:54:30.000 [notice] Bootstrapped 100%: Done
Jun 04 12:54:30.000 [notice] Now checking whether ORPort 86.50.69.28:9001 is
reachable... (this may take up to 20 minutes -- look for log messages indicating
success)
Jun 04 13:09:48.000 [notice] Your network connection speed appears to have
changed. Resetting timeout to 60s after 18 timeouts and 133 buildtimes.
Jun 04 13:14:30.000 [warn] Your server (86.50.69.28:9001) has not managed to
confirm that its ORPort is reachable. Please check your firewalls, ports,
address, /etc/hosts file, etc.
```

The last line of the log reveals the problem. The configured Tor node is not reachable by the other nodes.

It is also possible to check if the node is running through a web-site[22].

```
% TOR Node Checker Tool
% Copyright(c) 2016, Daniel Austin MBCS.
%
% Hello, 86.50.68.122, pleased to meet you.
%
% Checking IP: 86.50.68.122
%
Status: NAK
% The IP address is *not* a TOR Node.
```

This means that the Tor node is not visible to other Tor nodes. The author assumes that the problem is caused by the firewall which resides on the campus network and author has neither access nor permissions to change its configuration.

5.3 Configuring the Tor node using Google Cloud platform

In order to successfully capture Tor traffic, the author decided to use the Google Cloud platform by creating a virtual server with Ubuntu 16.04 LTS installed.

It is quite easy to create a virtual machine using the platform, it is possible to find the instructions on the Google Cloud platform web-site.

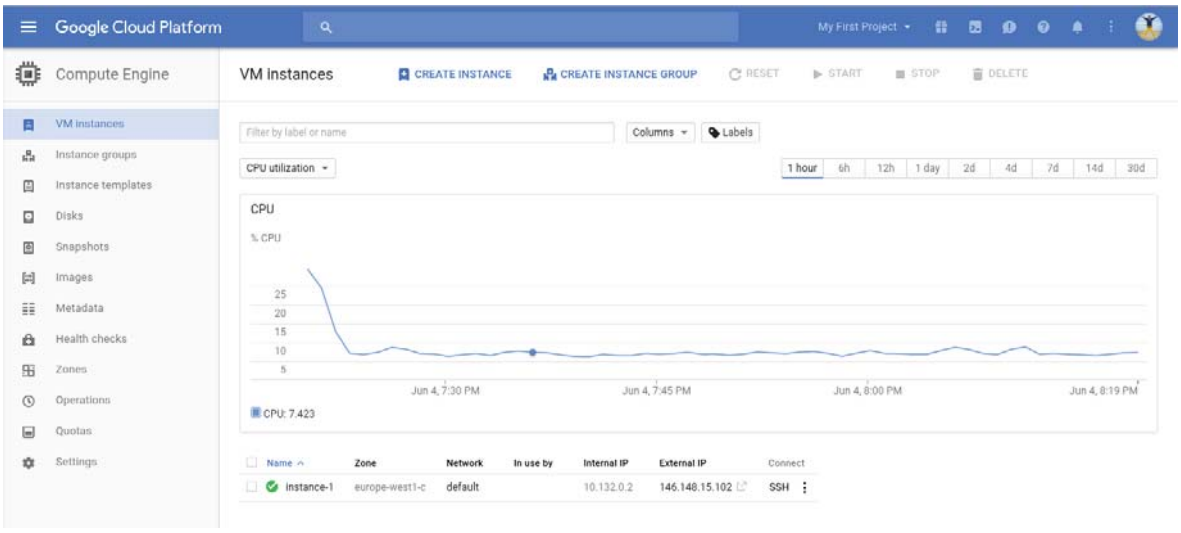


Figure 23. Google cloud platform web-interface

When the virtual machine is created, it is possible to ssh into it and install Tor. Afterwards, apply changes to the /etc/tor/torrc configuration file[24]:

```
vitosinschi@instance-1$ sudo apt-get update
vitosinschi@instance-1$ sudo apt-get upgrade
vitosinschi@instance-1$ sudo apt-get install tor
vitosinschi@instance-1$ sudo vim /etc/tor/torrc
```

```
ORPort 9001
Nickname gcloudnodetest
RelayBandwidthRate 1024 KB
RelayBandwidthBurst 1024 KB
ContactInfo alexandr.vitosinschi@edu.turkuamk.fi
ExitPolicy accept *:20-23  # FTP, SSH, telnet
ExitPolicy accept *:53     # DNS
ExitPolicy accept *:79-81  # HTTP
ExitPolicy accept *:110    # POP3
ExitPolicy accept *:143    # IMAP
ExitPolicy accept *:194    # IRC
ExitPolicy accept *:220    # IMAP3
ExitPolicy accept *:443    # HTTPS
ExitPolicy reject *:*
```

After applying changes to the configuration file it is required to restart Tor service and check if Tor service is up and running. It is done in the way shown below:

```
vitosinschi@instance-1:~$ sudo service tor restart
vitosinschi@instance-1:~$ sudo service tor status
● tor.service - Anonymizing overlay network for TCP (multi-instance-master)
  Loaded: loaded (/lib/systemd/system/tor.service; enabled; vendor preset:
enabled)
  Active: active (exited) since Sat 2016-06-04 15:25:29 UTC; 2h 1min ago
 Process: 21201 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
Main PID: 21201 (code=exited, status=0/SUCCESS)
   Tasks: 0 (limit: 512)
  Memory: 0B
     CPU: 0
  CGroup: /system.slice/tor.service
```

Afterwards, it is required to check Tor log file:

```
vitosinschi@instance-1:~$ tail /var/log/tor/log
Jun 04 15:25:29.000 [notice] Tor 0.2.7.6 (git-605ae665009853bd) opening log file.
Jun 04 15:25:29.326 [warn] OpenSSL version from headers does not match the version we're
running with. If you get weird crashes, that might be why. (Compiled with 1000207f:
OpenSSL 1.0.2g  1 Mar 2016; running with 1000207f: OpenSSL 1.0.2g-fips  1 Mar 2016).
Jun 04 15:25:29.347 [notice] Tor v0.2.7.6 (git-605ae665009853bd) running on Linux with
Libevent 2.0.21-stable, OpenSSL 1.0.2g-fips and Zlib 1.2.8.
Jun 04 15:25:29.347 [notice] Tor can't help you if you use it wrong! Learn how to be
safe at https://www.torproject.org/download/download#warning
```

```
Jun 04 15:25:29.348 [notice] Read configuration file "/usr/share/tor/tor-service-
defaults-torrc".
Jun 04 15:25:29.348 [notice] Read configuration file "/etc/tor/torrc".
Jun 04 15:25:29.351 [notice] Based on detected system memory, MaxMemInQueues is set to
1272 MB. You can override this by setting MaxMemInQueues by hand.
Jun 04 15:25:29.351 [warn] Tor is running as an exit relay. If you did not want this
behavior, please set the ExitRelay option to 0. If you do want to run an exit Relay,
please set the ExitRelay option to 1 to disable this warning, and for forward
compatibility.
Jun 04 15:25:29.352 [notice] Opening Socks listener on 127.0.0.1:9050
Jun 04 15:25:29.353 [notice] Opening Control listener on /var/run/tor/control
Jun 04 15:25:29.353 [notice] Opening OR listener on 0.0.0.0:9001
Jun 04 15:25:29.000 [notice] Parsing GEOIP IPv4 file /usr/share/tor/geoip.
Jun 04 15:25:29.000 [notice] Parsing GEOIP IPv6 file /usr/share/tor/geoip6.
Jun 04 15:25:29.000 [notice] Configured to measure statistics. Look for the *-stats
files that will first be written to the data directory in 24 hours from now.
Jun 04 15:25:29.000 [notice] Your Tor server's identity key fingerprint is
'gcloudnodetest E6CDFAD7397842D4604878CC6C2E0EE60A03E353'
Jun 04 15:25:29.000 [notice] Bootstrapped 0%: Starting
Jun 04 15:25:32.000 [notice] Bootstrapped 80%: Connecting to the Tor network
Jun 04 15:25:32.000 [notice] Signaled readiness to systemd
Jun 04 15:25:32.000 [notice] Guessed our IP address as 146.148.15.102 (source:
194.109.206.212).
Jun 04 15:25:33.000 [notice] Bootstrapped 85%: Finishing handshake with first hop
Jun 04 15:25:33.000 [notice] Bootstrapped 90%: Establishing a Tor circuit
Jun 04 15:25:33.000 [notice] Tor has successfully opened a circuit. Looks like client
functionality is working.
Jun 04 15:25:33.000 [notice] Bootstrapped 100%: Done
Jun 04 15:25:33.000 [notice] Now checking whether ORPort 146.148.15.102:9001 is
reachable... (this may take up to 20 minutes -- look for log messages indicating
success)
Jun 04 15:25:33.000 [notice] Self-testing indicates your ORPort is reachable from the
outside. Excellent. Publishing server descriptor.
Jun 04 15:25:34.000 [notice] Performing bandwidth self-test...done.
Jun 04 15:26:34.000 [warn] http status 400 ("Authdir is rejecting routers in this
range.") response from dirserver '194.109.206.212:80'. Please correct.
Jun 04 17:16:22.000 [warn] http status 400 ("Authdir is rejecting routers in this
range.") response from dirserver '194.109.206.212:80'. Please correct.
```

And final step is using on-line tool [22] to check if the Tor node is running and reachable:

```
% TOR Node Checker Tool
% Copyright(c) 2016, Daniel Austin MBCS.
%
% Hello, 130.232.142.40, pleased to meet you.
%
% Checking IP: 146.148.15.102
%
Status: ACK
Exit-Node: NAK
% TOR-Name: gcloudnodetest
% TOR-Onion-Port: 9001
% TOR-Flags: Running Valid
% TOR-Exit-Node: NAK
% TOR-Version: Tor 0.2.7.6
% TOR-Full-Version: Tor 0.2.7.6 on Linux
% TOR-Uptime: 0
% TOR-Bandwidth-Average-Bytes: 1048576
% TOR-Bandwidth-Burst-Bytes: 1048576
% TOR-Contact: alexandr.vitosinschi@edu.turkuamk.fi
%
```

5.4 Capturing traffic

At this point, it is possible to start the packet capture:

```
sudo tcpdump -i ens4 -s 65535 -w 04-06-cap4.cap &
```

The author used the `tcpdump` tool to capture the traffic. Running `man tcpdump` in command line of linux machine will give more in-deep information about the tool. In the case above, `-i ens4` is the capture interface, `-s 65535` indicates that the `tcpdump` is capturing full size packets and `-w 04-06-cap4.cap` is the file where captured packets are stored.

After a while, the capture was downloaded on the author's laptop and the analys was carried out using Wireshark.

It is possible to acquire the list of all IP addresses in the capture file by going to *Statistics* > *Endpoints* and exporting it is as CSV

This is an example of output. The part of the list is attached as Appendix 1.

Table 1. IP address list of captured packets

Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	AS Number	Country	City	Latitude	Longitude
54.230.15.17	20	57254	15	55659	5	1595	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
54.230.15.55	10	52936	9	52617	1	319	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
54.230.15.76	14	69798	12	69160	2	638	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
54.230.15.137	4	4620	2	3982	2	638	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995

It is possible to see that the table contains the IP address, number of packets, AS number, country of origin, and geographical coordinates of the IP address owner which in many cases is the ISP.

5.5 Analyzing captured traffic

The captured traffic contains several non-Tor related packets, thus, it is necessary to filter them out. As the Tor OR port is 9001, the traffic was filtered based on the port. This capture contains only the relayed packets between the Tor nodes. Thus now it is possible to see all the adjacent Tor connections.

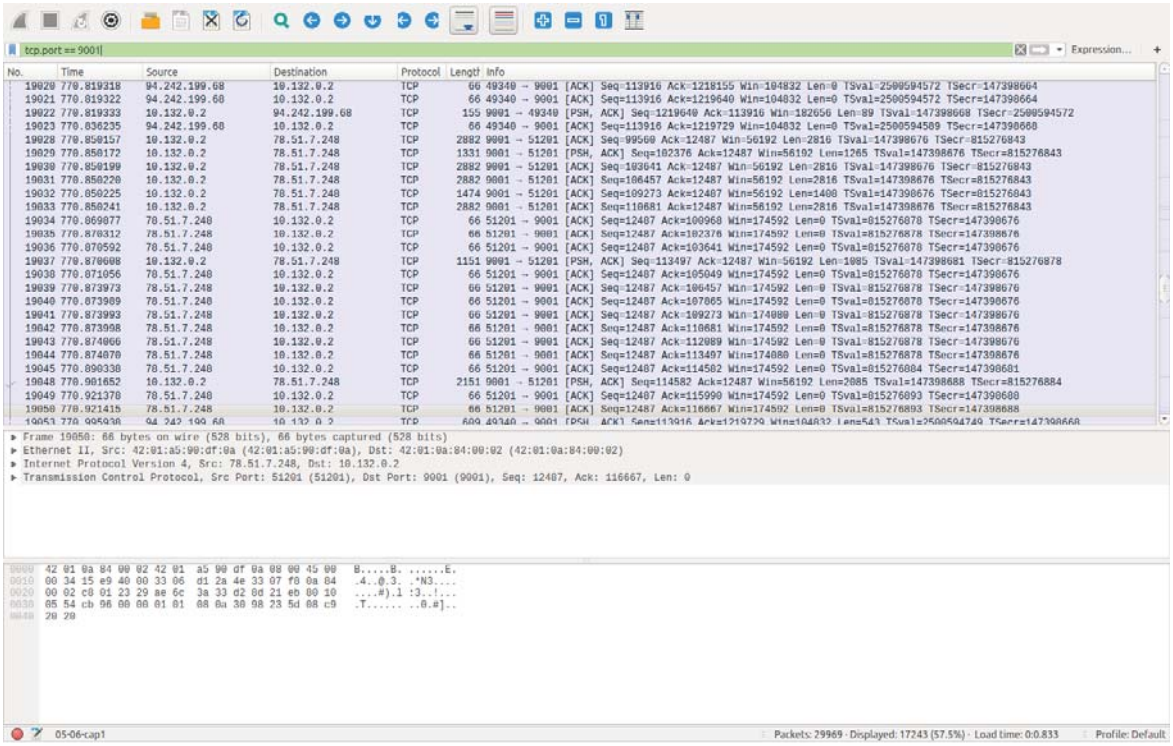


Figure 24. Filtering traffic based on the TCP 9001 port

It is also possible to find out for which of the adjacent ORs there are Tor nodes or hosts. The most common Tor relay configuration is using port 9001 or 443 as OR port, thus based on that, it is possible to say that those nodes which do not have port 9001 or 443 open are most probably hosts or initiators. It is also important to keep in mind that port 443 is used for HTTPS, which means that although the port is open, it can still be a host which runs a web-server.
This is a preliminary filtering. Assuming that both 443 and 9001 ports are closed, it means that it is a host. If one of the ports is open, it is necessary to use the online tool to check if this is actually a Tor node.

In order to check if port 9001 is open, the author used nmap, the address list of all connections on port 9001, and a simple bash script.

```
cat ./Documents/address_list | while read in; do nmap -p
9001,443 $in; done > ./Documents/node_check_9001
```

The script provides the following output:

```
Starting Nmap 7.01 ( https://nmap.org ) at 2016-06-08 21:58
EEST
Nmap scan report for 95.85.8.226
Host is up (0.032s latency).
PORT     STATE  SERVICE
443/tcp  open   https
9001/tcp closed tor-orport


Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

Starting Nmap 7.01 ( https://nmap.org ) at 2016-06-08 21:58
EEST
Nmap scan report for hosted-by.yourserver.se
(95.215.45.128)
Host is up (0.039s latency).
PORT     STATE  SERVICE
443/tcp  closed https
9001/tcp closed tor-orport
```

The nmap file contains entries for each of 522 IP addresses that were using port 9001 as destination, thus for simplicity's sake, the author has focused on few entries.

Let us dig deeper and find out if 95.85.8.226 is a Tor node. In order to do that, it is possible to use the online tool www.dan.me.uk/torcheck?ip=95.85.8.226 [22].

The following output is given:

```
% TOR Node Checker Tool
% Copyright(c) 2016, Daniel Austin MBCS.
%
% Hello, 193.166.134.13, pleased to meet you.
```

```
%
% Checking IP: 95.85.8.226
%
Status: ACK
Exit-Node: NAK
% TOR-Name: ccrelaycc
% TOR-Onion-Port: 443
% TOR-Directory-Port: 80
% TOR-Flags: Fast Guard HSDir Running Stable V2Dir Valid
% TOR-Exit-Node: NAK
% TOR-Version: Tor 0.2.4.27
% TOR-Full-Version: Tor 0.2.4.27 on Linux
% TOR-Uptime: 9555465
% TOR-Bandwidth-Average-Bytes: 52428800
% TOR-Bandwidth-Burst-Bytes: 104857600
% TOR-Bandwidth-Estimated-Bytes: 22697827
% TOR-Contact: 0xa7e9fe6cf073fda2 CristianCantoro
```

This is a running Tor node which uses 443 as its OR port.

The second interesting case is 95.215.45.128 where both ports are closed. The online tool [22] gives the following output:

```
% TOR Node Checker Tool
% Copyright(c) 2016, Daniel Austin MBCS.
%
% Hello, 193.166.134.13, pleased to meet you.
%
% Checking IP: 95.215.45.128
%
Status: NAK
% The IP address is *not* a TOR Node.
```

This means that 95.215.45.128 is not a Relay node or Exit node. Most probably it used the author's node as a Guard node, in other words, an entry point to the Tor network. By filtering the packets based on this IP address, the author obtained the following:

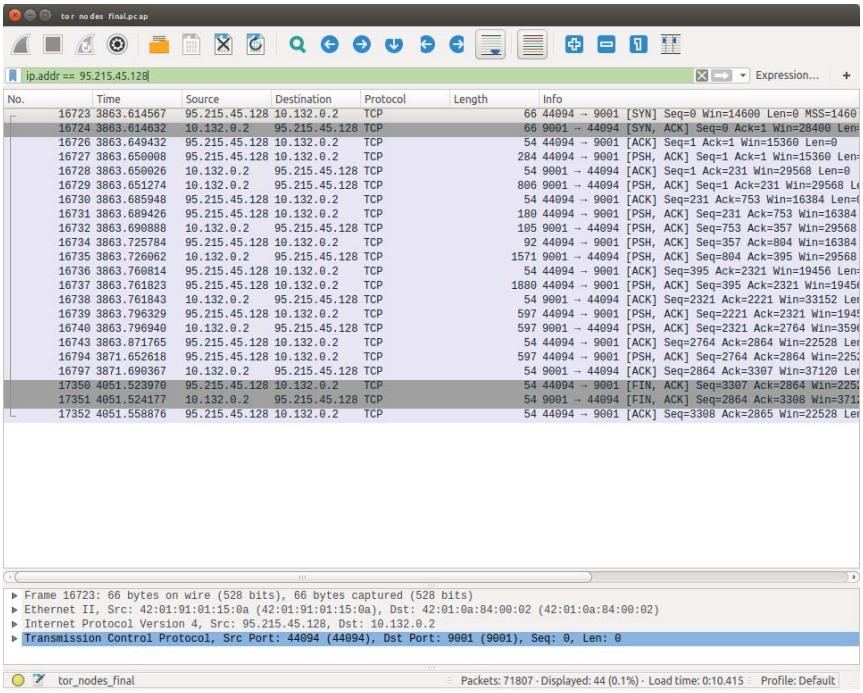


Figure 25. Filtering based on IP address

This is a TCP connection. It is possible to see how 95.215.45.128 establishes the connection, then forwards data, and closes the connection.
Wireshark has a feature to follow the TCP connection. Using this feature revealed the following:

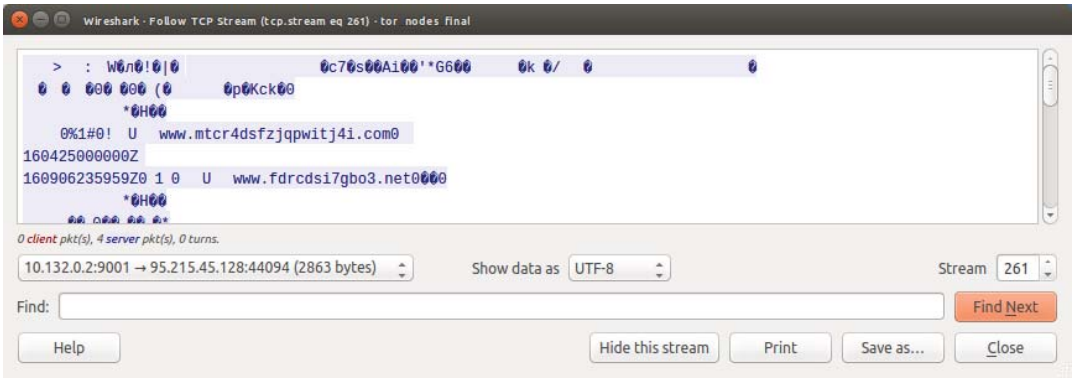


Figure 26. TCP flow

There are two domain names in this capture:

- www.mtcr4dsfzjqpwitj4i.com
- www.fdrcdsi7gbo3.net

By using `nslookup`, it is possible to find out if these domain names are valid. Unfortunately, it gave no information about the domain names above. At this point, it is possible to make an assumption that this connection was used to access a Hidden service.

Assuming that 95.215.45.128 is an initiator of Tor circuit, his or her true identity still can be hidden by NAT.

Based on the IP address of all 522 connections, it is possible to classify them by the country of origin.

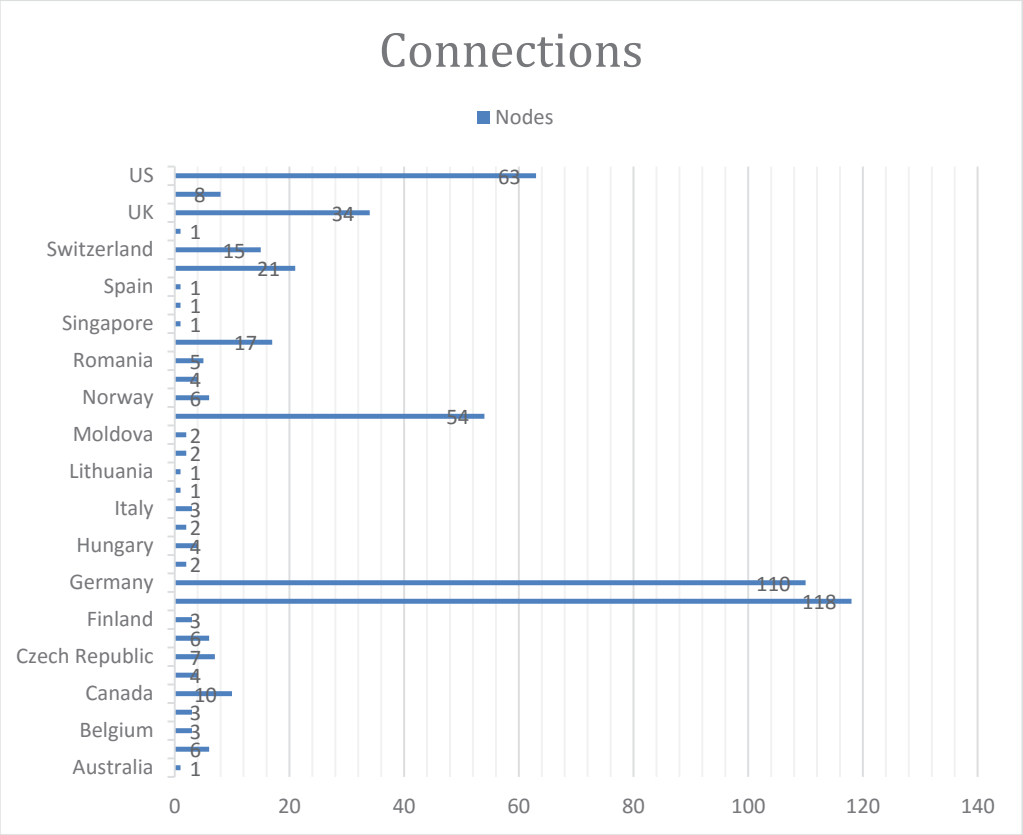


Diagram 1. Tor connections and their country of origin

One of the main objectives of this thesis was to inspect the captured packets and find out what web-sites Tor users are visiting. In order to do that, the author

has configured the exit policy in such way that it would be possible to capture unencrypted HTTP traffic. After filtering it out, the author found that there are several Google search requests.

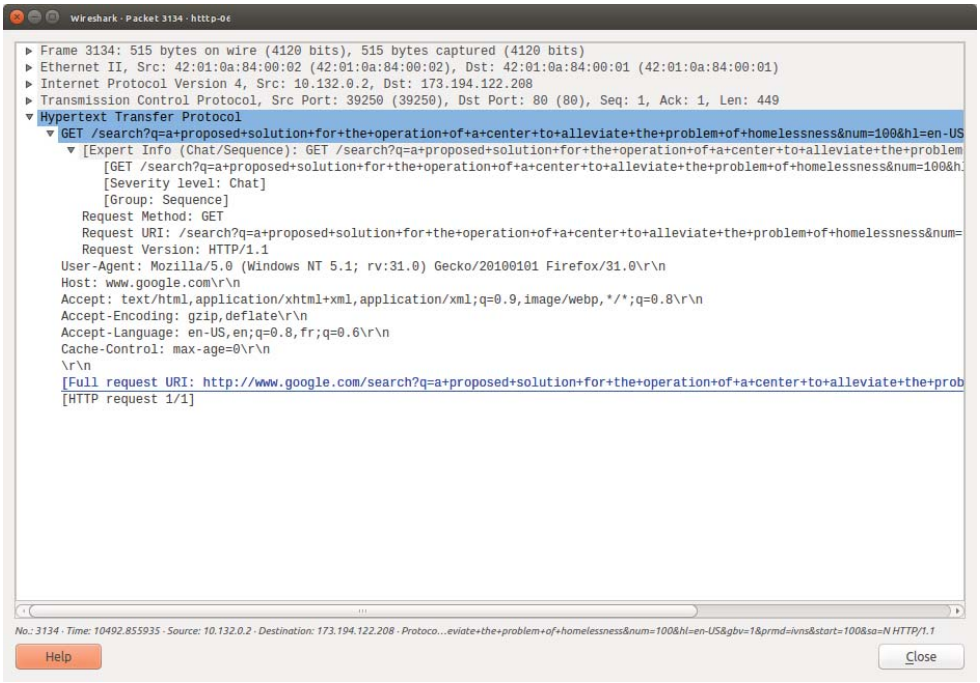


Figure 27. Inspecting captured packet with Wireshark

The search queries that were captured included:

- a proposed solution for the operation of a center to alleviate the problem of homelessness
- the ban on wearing burqas in france in public spaces
- absorption sfr numericable numericable strategy history
- the new european union members and the euro advantages and disadvantadges of joining the euro an assessment
- strategic marketing caroll fashion house china
- rushdie a bend in indian history
- "D'autres, s'orientent vers la piscine ou jouissent de la vue sur la mer, omniprésente sur ce territoire insulaire. Le culte de l'eau, on le retrouve encore au temple du bien-être : le Spa"
- the evolution of the british parliamentary system under the influence of the majority phenomenon

But let us take a closer look to the web-session with 54.230.15.76. In the beginning, it was obvious that someone was accessing Amazon website

through Tor, because this IP address belongs to Amazon. After taking a closer look in the packet revealed that it was a wrong assumption.

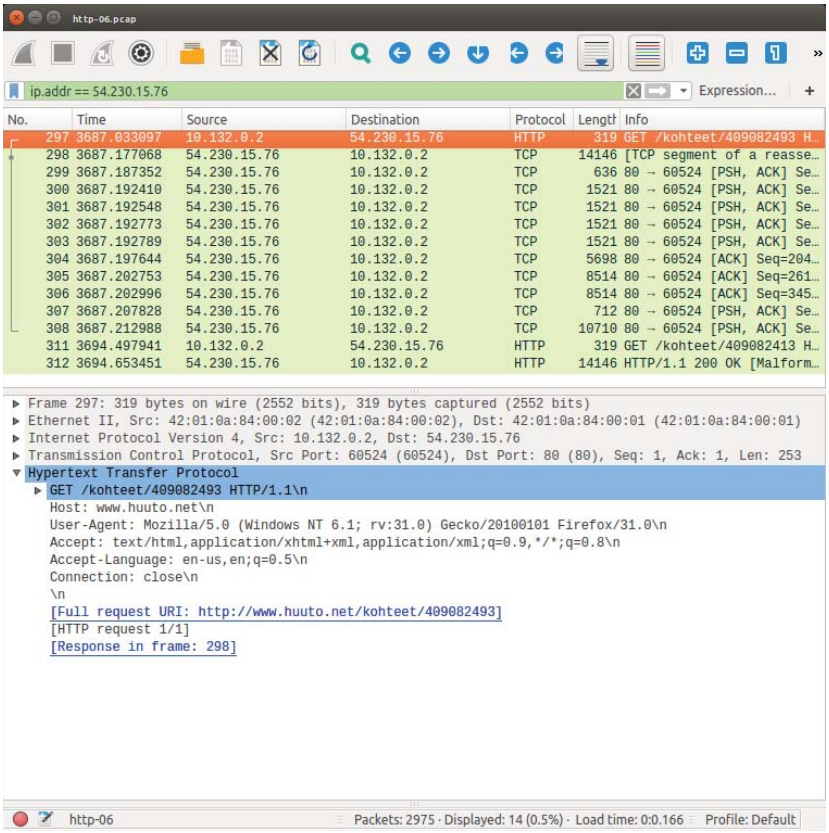


Figure 28. Filtering based on IP address

It is possible to see that the user had accessed www.huuto.net which is a Finnish online auction web site. It is also possible to reveal that user was using a Windows machine. Because the IP address belongs to Amazon, it is possible to assume that the web-site is running on Amazon Web Services.

6 TOR FROM A USER PERSPECTIVE

As a normal user who cares about anonymity online and does possess knowledge of Linux OS, it is still possible to use Tor by downloading theTor browser from the torproject.org website.

*The **Tor Browser** lets you use Tor on Windows, Mac OS X, or Linux without needing to install any software. It can run off a USB flash drive, comes with a pre-configured web browser to protect your anonymity, and is self-contained (portable).* [23]

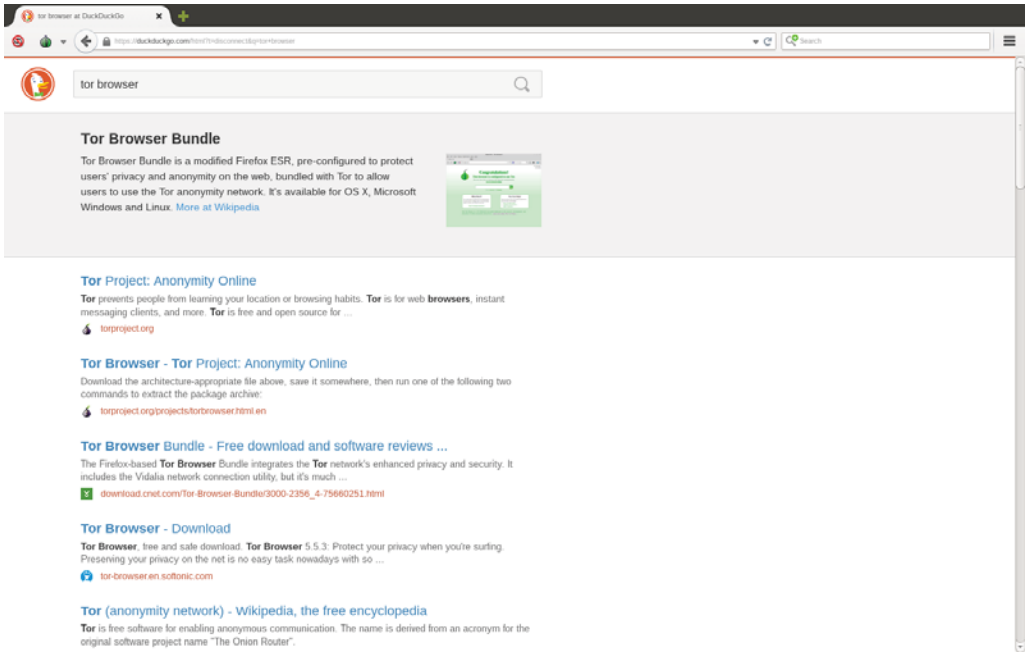


Figure 29. Tor browser

7 CONCLUSIONS & RECOMMENDATIONS

Privacy is a great issue nowadays. There are myriads ofonline services that collect users' private data and use it for marketing purposes. The problem arises when this data is used against the user, for example in court. There are numerous anonymizing technologies available which minimize the impacted private data collection. The most known are Tor, I2P and Freenet. It is also possible to use VPN gateway or a Proxy-server but both are centralized systems, thus in case these are compromised, it is possible to reveal the identity of the user.

This thesis has described the communication process between a web-server and a client. Additionally, it has also described technical specifications of onion routing and the difference between normal web-browsing and web browsing using Tor. The practical part of the thesis consisted of installing the Tor software on Raspberry-Pi and making it a Tor exit node. Unfortunately, it was not a success because of the firewall, which author did not have access to. Instead of using Raspberry-Pi, there was a Tor node created using Google Cloud Platform. The traffic dumps were made on a virtual server and analyzed on a local computer. From the traffic dump,s it was possible to detect adjacent Tor connections. Some of the connections were coming from nodes, while others were from hosts. It was also interesting to capture the exit traffic and see what resources users use to access through Tor.

Tor users should be aware that there are many compromised nodes on the Tor network. A malicious node is a node where the owner tries to implement a passive or active attack in order to deanonymize the identity of the user. Detecting a malicious node is not simple because it is hardly possible to detect a passive attack. It is possible to assume that these exit nodes that have an exit policy configured only for the protocols that do not use end-to-end encryption, such as HTTP, are malicious. There is a possibility that the owner of that node

is capturing the unencrypted traffic at the exit and uses it to detect identity of the user. Tor users should not be using any service that does not support end-to-end encryption, because the unencrypted message can contain the information which provides identity of the user, such as nickname, surname, e-mail address etc. Although revealing the identity of the user is possible, it requires an arsenal of resources, such as running multiple Tor nodes. Furthermore it is possible to reveal the identity of a random user but a targeted attack on anonymity is much more difficult.

Running a Tor exit node can cause legal problems. All the traffic exiting the node and forwarded to the internet is seen to be originating from the person who runs the node. The traffic can carry some malicious code or can be a part of DoS attack, for example. The authorities may contact the node owner and interrogate him or her regarding this matter.

For a non-technical user who does not run the Tor node but wants to use Tor to access the Internet, it is required to install the Tor web-browser in order to use the Tor service.

REFERENCES

1. Merriam-Webster: Dictionary and Thesaurus. Available from: <http://www.merriam-webster.com/dictionary/privacy> [On-line; accessed: 15-May-2016]

2. Lecture at Hilla University for Humanistic Studies - January 21, 2004 ”What is Democracy?” Available from: <https://web.stanford.edu/~ldiamond/iraq/WhaIsDemocracy012004.htm> [On-line; accessed: 15-May-2016]

3. Lecture at Hilla University for Humanistic Studies - January 21, 2004 ”What is Democracy?” Available from: <https://web.stanford.edu/~ldiamond/iraq/WhaIsDemocracy012004.htm> [On-line; accessed: 15-May-2016]

4. P. Syverson, G. Tsudik, M. Reed, C. Landwerhr – 2000, ”Towards an Analysis of Onion Routing Security”. Available from: <http://www.dtic.mil/dtic/tr/fulltext/u2/a465255.pdf> [On-line; accessed: 15-May-2016]

5. X.200 : Information technology - Open Systems Interconnection - Basic Reference Model: The basic model – 1994. Available from: <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items> [On-line; accessed: 15-May-2016]

6. T. Socolofsky, C. Kale – January 1991. RFC 1180, ”A TCP/IP Tutorial”. Available from: <https://tools.ietf.org/html/rfc1180> [On-line; accessed: 15-May-2016]

7. A. Rodriguez, J. Gatrell, J. Karas, R. Peschke – August 2001 “TCP/IP Tutorial and Technical Overview -2001” Chapter 5. Available from: <http://www.cs.virginia.edu/~cs458/material/Redbook-ibm-tcpip-Chp5.pdf [On-line; accessed: 15-May-2016]

8. Information Sciences Institute, University of Southern California – September 1981. “RFC 793, TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION”. Available from: <https://www.ietf.org/rfc/rfc793.txt> [On-line; accessed: 15-May-2016]

9. J. Reynolds J. Postel – October 1994. “RFC 1700, Assigned numbers” p. 15. Available from: <https://www.ietf.org/rfc/rfc1700.txt> [On-line; accessed: 15-May-2016]

10. P. Mockapetris – November 1987. “RFC 1035, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION”. Available from: <https://www.ietf.org/rfc/rfc1035.txt> [On-line; accessed: 15-May-2016]

11.M. Reed, P. Syverson, D. Goldschlag – May 1998 ” Anonymous Connections and Onion Routing”. IEEE JOURNAL ON SELECTED AREAS

IN COMMUNICATIONS, VOL. 16, NO. 4. Available from: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=668972>[On-line; accessed: 27-May-2016]

12. R. Dingledine, N Mathewson, P. Syverson - 2004 - DTIC Document ”Tor: The Second-Generation Onion Router”. Available from: <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf> [On-line; accessed: 27-May-2016]

13. P. Syverson, G. Tsudik, M. Reed, C. Landwehr – 2001, ”Towards an Analysis of Onion Routing Security”. Available from: <http://www.dtic.mil/dtic/tr/fulltext/u2/a465255.pdf> [On-line; accessed: 27-May-2016]

14. R. Dingledine, N. Mathewson, P. Syverson - 2004 - DTIC Document ”Tor: The Second-Generation Onion Router”. Available from: <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf> [On-line; accessed: 27-May-2016]

15. R. Dingledine, N. Mathewson, P. Syverson - 2004 - DTIC Document ”Tor: The Second-Generation Onion Router”. Available from: <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf> [27 May 2016]

16. R. Dingledine, N. Mathewson, P. Syverson - 2004 - DTIC Document ”Tor: The Second-Generation Onion Router”. Available from: <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf> [On-line; accessed: 27-May-2016]

17. R. Dingledine, N. Mathewson, P. Syverson - 2004 - DTIC Document ”Tor: The Second-Generation Onion Router”. Available from: <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf> [On-line; accessed: 27-May-2016]

18. R. Dingledine, N Mathewson, P Syverson - 2004 - DTIC Document ”Tor: The Second-Generation Onion Router”. Available from: <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf> [On-line; accessed: 27-May-2016]

19. J. Appelbaum, M. Ray, K. Koscher, I. Finder ”vpwns: Virtual Pwned Networks”. Available from: <https://www.usenix.org/system/files/conference/foci12/foci12-final8.pdf> [On-line; accessed: 27-May-2016]

20. Raspberry-Pi official web-site. <https://www.raspberrypi.org/downloads/raspbian/> [On-line; accessed: 3-June-2016]

21. Quick-Tip: Linux NAT in Four Steps using iptables. <http://www.revsys.com/writings/quicktips/nat.html> [On-line; accessed: 3-June-2016]

22. Daniel Austin MBCS web-site <https://www.dan.me.uk/torcheck>. [On-line; accessed: 3-June-2016]

23. Official Tor project website. Available from: <https://www.torproject.org/projects/torbrowser.html.en> [On-line; accessed: 15-May-2016]

24. Tor configuration example on Github. Available from: <https://gist.github.com/pdp7/7e44c1772e9e7dbba211> [On-line; accessed: 3-June-2016]

APPENDIX

Appendix 1. IP addresses of connections which used port 9001

Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	AS Number	Country	City	Latitude	Longitude
95.85.8.226	79	33237	41	14660	38	18577	AS200130 Digital Ocean, Inc.	Netherlands	Amsterdam, 07	52.349998	4.9167
95.85.34.137	87	42100	46	17555	41	24545	AS200130 Digital Ocean, Inc.	Netherlands	Amsterdam, 07	52.349998	4.9167
95.85.19.46	85	41960	47	17596	38	24364	AS200130 Digital Ocean, Inc.	Netherlands	Amsterdam, 07	52.349998	4.9167
95.78.76.147	33	11602	17	6046	16	5556	AS42116 JSC ER-Telecom Holding	Russian Federation	Chelny, 73	55.003799	50.054901
95.71.126.230	307	103729	168	52185	139	51544	AS29456 PJSC Rostelecom	Russian Federation	Belgorod, 09	50.610001	36.580002
95.215.45.128	22	7381	13	4020	9	3361	AS52173 Makonix SIA	Sweden	-	59.329399	18.0686
95.213.236.154	1100	2345631	415	102681	685	2242950	AS49505 OOO Network of data-centers Selectel	Russian Federation	-	55.75	37.6166
95.211.225.167	86	37628	45	16820	41	20808	AS60781 LeaseWeb Netherlands B.V.	Netherlands	Amsterdam, 07	52.349998	4.9167
95.211.216.9	96	41645	50	17318	46	24327	AS60781 LeaseWeb Netherlands B.V.	Netherlands	Amsterdam, 07	52.349998	4.9167
95.183.48.12	35	12275	19	6181	16	6094	AS197988 Solar Communications GMBH	Switzerland	-	47	8
95.141.83.146	13	3573	6	2025	7	1548	AS49409 DataOppdrag AS	Norway	-	59.950001	10.75
94.75.93.34	33	11568	18	6078	15	5490	AS203889 SZELAG SLAWOMIR FIRMA USLUGOWO HANDLOWA	Poland	-	52.233299	21.016701
94.242.58.51	34	11707	18	6134	16	5573	AS43317 OOO Fishnet Communications	Russian Federation	-	55.75	37.6166
94.242.58.2	37	12954	20	6789	17	6165	AS43317 OOO Fishnet Communications	Russian Federation	-	55.75	37.6166
94.242.58.122	34	11677	18	6121	16	5556	AS43317 OOO Fishnet Communications	Russian Federation	-	55.75	37.6166
94.242.255.112	83	36622	43	16267	40	20355	AS5577 root SA	Luxembourg	-	49.75	6.1667
94.242.228.174	15	4248	7	2091	8	2157	AS5577 root SA	Luxembourg	-	49.75	6.1667
94.23.219.192	11	2898	6	1482	5	1416	AS16276 OVH SAS	France	-	48.860001	2.35
94.23.204.175	33	11219	18	5888	15	5331	AS16276 OVH SAS	France	-	48.860001	2.35
94.100.18.162	232	188186	119	21473	113	166713	AS35017 Swiftway Sp. z o.o.	Netherlands	-	52.366699	4.9
93.238.143.111	35	11770	20	6263	15	5507	AS3320 Deutsche Telekom AG	Germany	Kleve, 07	51.786499	6.1375
93.219.66.168	90	37641	48	17154	42	20487	AS3320 Deutsche Telekom AG	Germany	Berlin, 16	52.516701	13.4
93.186.200.68	106	41779	58	19111	48	22668	AS24961 myLoc managed IT AG	Germany	-	51	9
93.186.14.70	48	16426	26	8302	22	8124	AS9063 VSE NET GmbH	Germany	-	51	9
93.180.157.154	89	38465	44	17411	45	21054	AS34011 domainfactory GmbH	Germany	-	51	9
93.170.77.174	376	366242	183	35370	193	330872	AS49981 WorldStream	Czech Republic	-	50.083302	14.4167
93.158.216.142	160	68945	81	30663	79	38282	AS50673 Serverius Holding B.V.	Netherlands	-	52.366699	4.9
93.115.97.242	198	81341	106	34716	92	46625	AS197922 Techcrea Solutions Sarl	France	-	48.860001	2.35

93.115.91.66	205	167945	106	23230	99	144715	AS3223 Voxility S.R.L.	Anonymous Proxy	-	0	0
93.104.212.253	85	36796	43	16326	42	20470	AS8767 M-net Telekommunikations GmbH	Germany	-	51	9
93.104.209.61	34	11626	18	6070	16	5556	AS8767 M-net Telekommunikations GmbH	Germany	-	51	9
92.51.245.131	45	19261	25	8728	20	10533	AS31122 Digiweb Ltd.	Ireland	Dublin, 07	53.333099	-6.2489
92.39.246.45	1289	1599794	607	119903	682	1479891	AS34274 ELB MULTIMEDIA	France	-	48.860001	2.35
92.249.143.119	104	65191	56	12861	48	52330	AS20845 DIGI Tavkozlesi es Szolgaltato Kft.	Hungary	Budapest, 05	47.5	19.0833
92.222.181.123	33	11592	18	6102	15	5490	AS16276 OVH SAS	France	Paris, A8	48.866699	2.3333
92.222.153.147	415	468018	219	34670	196	433348	AS16276 OVH SAS	France	Paris, A8	48.866699	2.3333
92.221.62.34	15	4248	8	2157	7	2091	AS29695 Altibox AS	Norway	Nærbø, 14	58.665501	5.6379
92.208.208.186	35	11759	19	6203	16	5556	AS3209 Vodafone GmbH	Germany	Berlin, 16	52.556499	13.3911
91.78.225.145	92	37294	48	16692	44	20602	AS8359 MTS PJSC	Russian Federation	Moscow, 48	55.752201	37.615601
91.55.245.213	1071	2345350	377	100679	694	2244671	AS3320 Deutsche Telekom AG	Germany	-	51	9
91.236.116.87	52	17154	27	8815	25	8339	AS42708 Portlane AB	Sweden	-	59.329399	18.0686
91.231.86.204	222	128978	108	37232	114	91746	AS197726 Ukrainian Internet Names Center LTD	Ukraine	-	50.450001	30.5233
91.219.29.142	312	230024	184	205274	128	24750	AS3254 Lucky Net Ltd	Ukraine	-	50.450001	30.5233
91.219.28.211	33	11579	17	6006	16	5573	AS196682 FLP Kochenov Aleksej Vladislavovich	Netherlands	Meppel, 01	52.700001	6.2
91.219.237.19	11	2898	6	1482	5	1416	AS56322 ServerAstra Kft.	Hungary	-	47.4925	19.051399
91.195.125.0	33	11532	17	5976	16	5556	AS49352 Domain names registrar REG.RU, Ltd	Russian Federation	-	55.75	37.6166
91.180.213.84	252	195177	143	24587	109	170590	AS5432 Proximus NV	Belgium	-	50.849998	4.35
91.134.223.223	90	39162	50	16312	40	22850	-	Bulgaria	-	42.700001	23.3333
91.122.31.175	335	103197	169	49871	166	53326	AS8997 PJSC Rostelecom	Russian Federation	Saint Petersburg, 66	59.894402	30.2642
91.121.85.130	92	38872	49	17585	43	21287	AS16276 OVH SAS	France	-	48.860001	2.35
91.121.84.137	34	11663	19	6156	15	5507	AS16276 OVH SAS	France	-	48.860001	2.35
91.121.23.100	33	11612	17	6039	16	5573	AS16276 OVH SAS	France	-	48.860001	2.35
91.109.29.120	33	11550	18	6060	15	5490	AS28753 Leaseweb Deutschland GmbH	Germany	-	51	9
90.110.233.162	93	37939	48	17271	45	20668	AS3215 Orange S.A.	France	-	48.860001	2.35
89.46.196.84	85	36802	44	16381	41	20421	AS31034 Aruba S.p.A.	Italy	Arezzo, 16	43.4995	11.9109
89.46.101.181	21	7546	11	4015	10	3531	AS9009 M247 Ltd	Romania	-	47.200001	23.049999
89.22.96.47	34	12078	19	6378	15	5700	AS31342 Alvotech GmbH	Germany	-	51	9
89.163.235.163	34	11248	18	5880	16	5368	AS24961 myLoc managed IT AG	Germany	-	51	9
89.16.176.158	93	76905	46	11652	47	65253	AS35425 Bytemark Limited	United Kingdom	Manchester, I2	53.5	-2.2167
88.198.51.101	100	41890	52	17455	48	24435	AS24940 Hetzner Online GmbH	Germany	Nürnberg, 02	49.4478	11.0683
88.198.151.8	11	2898	6	1482	5	1416	AS24940 Hetzner Online GmbH	Germany	-	51	9
88.193.237.145	159	76026	81	31654	78	44372	AS1759 TeliaSonera Finland Oyj	Finland	Tampere, 15	61.5	23.75

Appendix 2. IP addresses of connections which used port 80

Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	AS Number	Country	City	Latitude	Longitude
10.132.0.2	2975	5985393	633	282583	2342	5702810	-	-	-	-	-
54.230.15.17	20	57254	15	55659	5	1595	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
54.230.15.55	10	52936	9	52617	1	319	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
54.230.15.76	14	69798	12	69160	2	638	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
54.230.15.137	4	4620	2	3982	2	638	AS16509 Amazon.com, Inc.	United States	Seattle, WA	47.5839	-122.2995
62.140.236.139	8	4747	3	2576	5	2171	AS28917 LLC TRC FIORD	Russian Federation	Moscow, 48	55.752201	37.615601
66.228.37.24	488	120660	244	70272	244	50388	AS63949 Linode, LLC	United States	Absecon, NJ	39.489899	-74.477303
81.19.88.115	2	4592	1	4218	1	374	AS24638 Rambler Internet Holding LLC	Russian Federation	-	55.75	37.6166
84.234.75.97	5	14470	4	14131	1	339	AS58003 Planeetta Internet Oy	Finland	Helsinki, 13	60.2229	24.861
86.59.21.38	2	356	1	95	1	261	AS8437 Tele2 Telecommunication GmbH	Austria	-	48.200001	16.366699
87.108.60.73	7	10937	3	9514	4	1423	AS29154 TelecityGroup Finland Oy	Finland	-	60.170799	24.9375
87.108.60.74	13	64061	10	63023	3	1038	AS29154 TelecityGroup Finland Oy	Finland	-	60.170799	24.9375
88.212.196.104	4	2004	2	1057	2	947	AS39134 United Network LLC	Russian Federation	-	55.75	37.6166
91.207.59.161	69	263091	39	250241	30	12850	AS48061 CJSC RuTube	Russian Federation	-	55.75	37.6166
91.216.107.158	10	12077	5	9837	5	2240	AS16347 ADISTA SAS	France	-	46.141701	-0.2218
95.211.98.147	10	15410	9	15027	1	383	AS60781 LeaseWeb Netherlands B.V.	Netherlands	-	52.366699	4.9
104.239.240.75	2	1297	2	1297	0	0	AS33070 Rackspace Hosting	United States	San Antonio, TX	29.488899	-98.398697
131.188.40.189	155	740664	114	705927	41	34737	AS680 Verein zur Foerderung eines Deutschen Forschungsnetzes e.V.	Germany	Erlangen, 02	49.588799	11.0098
154.35.175.225	256	1328047	249	1322369	7	5678	AS14987 Rethem Hosting LLC	United States	-	38	-97
162.159.251.201	2	3520	1	3183	1	337	AS13335 CloudFlare, Inc.	United States	San Francisco, CA	37.769699	-122.393303
172.217.18.234	2	14227	1	13846	1	381	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
173.194.32.145	61	51060	30	35290	31	15770	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
173.194.32.161	52	97061	22	81227	30	15834	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
173.194.122.194	62	129735	30	109137	32	20598	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
173.194.122.208	63	155946	29	137708	34	18238	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
173.194.122.212	152	377084	74	336855	78	40229	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
184.105.139.85	2	306	1	139	1	167	AS6939 Hurricane Electric, Inc.	United States	Fremont, CA	37.515499	-121.896202
188.187.188.153	4	6550	2	5764	2	786	AS41786 JSC ER-Telecom Holding	Russian Federation	Yoshkar-ola, 45	56.638802	47.8908
192.124.249.13	13	50669	11	50001	2	668	AS30148 Sucuri	United States	-	38	-97
192.229.233.25	4	17705	2	16964	2	741	AS15133 EdgeCast Networks, Inc.	United States	Santa Monica, CA	34.011902	-118.468201
193.0.170.23	1	171	0	0	1	171	AS58116 Mamba CJSC	Russian Federation	-	55.75	37.6166

193.23.244.244	39	39739	19	27115	20	12624	AS50472 Chaos Computer Club e.V.	Germany	-	51	9
194.109.206.212	1186	1811385	1165	1787290	21	24095	AS3265 XS4ALL Internet BV	Netherlands	-	52.366699	4.9
194.226.130.229	1	451	0	0	1	451	AS52016 CJSC TNS Gallup AdFact	Russian Federation	-	55.75	37.6166
199.254.238.52	117	200925	107	190198	10	10727	AS16652 Riseup Networks	United States	-	38	-97
212.47.219.161	100	148240	98	147500	2	740	AS3327 Linx Telecommunications B.V.	Estonia	-	59	26
213.180.193.119	2	847	1	461	1	386	AS13238 Yandex LLC	Russian Federation	-	55.75	37.6166
216.58.208.206	4	7270	2	6017	2	1253	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
216.58.209.168	2	14248	1	13846	1	402	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
216.58.209.206	5	13748	2	12410	3	1338	AS15169 Google Inc.	United States	Mountain View, CA	37.419201	-122.057404
217.159.201.10	22	77485	20	76857	2	628	AS3249 Telia Eesti AS	Estonia	-	59	26