



Report

Assignment 2



Author: Anas KWEFATI, Ludvig JOHANSSON, Pedro DINIZ

Group: 57

Semester: Autumn 2021

Course: Introduction to Information Security (INTROSEC)

Course code: ML470C

Contents

1	Design	1
1.1	Protocol	1
1.1.1	Tor	2
1.1.2	User-Agent Switcher	3
1.1.3	Proxy Chaining	4
2	Experiment results	5
2.1	Installation process	5
2.1.1	Tor Project	5
2.1.2	User-Agent Switcher	5
2.1.3	Proxy Chaining	5
2.2	Usage and misuse of tools	6
2.2.1	Tor Project	6
2.2.2	User-Agent Switcher	8
2.2.3	Proxy Chaining	9
2.3	Comparison of protection and vulnerabilities	14
3	Time estimation	16
4	Group reflection	16
	References	17

1 Design

1.1 Protocol

In this task we have decided to choose the following tools:

- Tor Project, with an emphasis placed on the Tor Browser and the underlying network.
- User-Agent switcher, which allows to spoof someone's default user-agent string.
- Proxy chaining technique.

These three tools were elected because, although varying in approach, implementation, and complexity they share the common goal to give some layer of anonymity to the user. By comparing tools with different complexity and degree of privacy protection techniques our hope is to obtain a better understanding in which ways a simpler tool might fail where a more advanced tool might not, as well as what are the drawbacks or problems are caused by each tool.

The test protocol is divided into three aspects deemed interesting for further discussion, exploration, and comparison.

- Installation Process
- Usage and Misuse of tools
- Protection and Vulnerabilities

During the installation process section, we will evaluate the installation, deployment, or otherwise equivalent method in an attempt to discover any potential threats or mishandling that could cause a breach of anonymity. During the usage and misuse section we will evaluate similar parameters as during the installation process, but instead with regards to any potential use, misuse, or also misconception of the tool that may occur when operated by a user. In the final section, protections and vulnerabilities, we will highlight in what way the three tools are similar and different with regards to the type of protection, as well as any potential vulnerabilities that has been documented or will be found by any of the experiment or comparisons conducted.

Whenever deemed sufficiently interesting, non-trivial, and feasible given our technical skills and limitations as well as within the time resource allocated for this report and assignment, the various sections will conduct experiments or give examples to illustrate any interesting observations made or discovered. For example, a conducted test to display some vulnerability found in one of the tools and the result of that test. In places where that is not possible due to any of the factors mentioned above, the report will still attempt to illustrate or describe how a test could have been made, and what these results could have been. Finally, any aspect such as vulnerabilities or protections brought up in the in the report should not be interpreted as being exhaustive of the tools.

1.1.1 Tor

The Tor browser uses the concept of onion routing and network to attempt to anonymize users' identity and protect data message integrity when using or accessing websites, messaging service, email service, or other web resources over the internet [1]. The Tor network strive to achieve a balance between usability versus privacy/security. Therefore the Tor browser is not completely anonymized and secure, nor does it state to be or strive to become one, but instead offer an easily available, more anonymous, and secure web browser than a "normal" browser for individuals and organizations with such requirements.

Consequently, an adversary that operates as an intermediate node in the tor network that receives a message cannot under normal secure operating circumstances decipher, modify the message, or figure out the message's initial node or the final destination. However, the Tor network does not directly prevent an intermediate node (or an exit node for that matter) from performing traffic analysis/timing attacks. Although it is complicated for an attacker due to multiple streams of data that are most likely being processed simultaneously on a single circuit [1]. That being said, complications are not equal to prevention. Thus, it is still possible for an attacker to analyse the timing of the request and responds to profile user and their communications [1]. It should however be noted that this is a problem for "normal" encrypted internet communication as well.

In addition to this vulnerabilities, an adversary operating as an exit node can utilize more types of attacks depending on the scenario. For example, if a user's underlying message is not encrypted using encryption such as SSL/TLS, a malicious exit node can intercept and view the message. The exit node adversary therefore would have the possibility to collect sensitive and private information in the communication [1]. Information that directly or indirectly could be used to identify the person sending the message. This is sometimes referred to as an "Exit node vulnerability". The misconceptions of viewing the Tor network and browser as being synonymous to end-to-end encryption, as well as the opportunity of using an exit node as a spy tool was demonstrated by Dan Egerstad in 2007 when he setup five Tor exit nodes and subsequently were able to capture private emails, messages, and credentials from high-ranking governments officials from India, Russia, Iran, and other nations [2].

Unfortunately we were unable to find a method to test this and the intermediate node vulnerability ourselves since it would have required us to run an exit or intermediate node, which would have cause both big technical challenges and potential legal liabilities and were deemed to be too difficult to implement and test given the resources and constraints provided. The test protocol for Tor are written below:

- The objective is to look closer at some user-centric problems caused by the infrastructure of the Tor network, and potentially the user itself.
- The test uses the Tor browser and a some custom code.
- Tests speed of request response time with custom code and Tor to various URLs and compare with a normal connection.
- Test bad configuration of Tor browser and describe implications for anonymity.

1.1.2 User-Agent Switcher

The browser extension User Agent Switcher allows to identify the device in order to render an optimized version of the page for the end-user and facilitates interaction with Web content [3]. For example, it can adjust the page font depending on if it is accessed by a smartphone or a normal computer, or it can decide on the language of the page depending on the browser's default language and so on. This sort of information can also pose a threat to the user as it can be used to identify a certain group of users with specific user-agent. For instance, when going to the website mybrowserinfo, we can see many information such as what browser, and, as well as which operating system I am currently using. These kind of data which can be called "fingerprint" can pose a threat to privacy, and further to the user's security in case of an attack. Regarding privacy, nowadays many websites use tracking tools to collect user's metadata, and creating fingerprint profile. These fingerprints are usually collected by Analytic companies, which can be used for different reasons. They are also used for targeted advertisements, or sold to some companies [4]. Regarding security, if I were an intruder, I would mainly focus on the platform used, which internet browser is used, with the version of each as it would give me ideas for attacks. For instance, specific browsers versions may have specific vulnerabilities. Also, knowing the operating system version could help the intruder into deciding for an attack vector. Then, would probably look at other third party app (plugins) if any is installed or used, so it would help also to think for different attack strategies. For instance, if Adobe Acrobat is installed on the browser or not.

We can see that knowing someone's fingerprint may become dangerous in case of an attack, as it could be used to decide on different attack vectors. Moreover, knowing someone's web fingerprint, can also play against its anonymity, as websites have the possibility to gather these data, and create unique user profiles. Then, these metadata information can be used to track those persons. Hence, I believe it is necessary to use a tool, that would allow to change our user-agent metadata, in order to help us against these issues, and avoid websites into creating a unique profile that can be used over time of someone [5].

Therefore the tool User-Agent Switcher was chosen, as it allows to change our browser's user-agent metadata. Using this method, we will appear as another device with a different browser configurations. The idea to become anonymous, is to make ourselves look less unique as possible. If this option is not possible, then we would need to change our user-agent metadata every certain time in order to not be tracked, as we would always have a different metadata. Thus, in our case we can use the option to randomly change our user-agent through the tool every 10 minutes. This would make it harder to specifically identify us, as we would always have a different fingerprint.

That being said, in this test protocol we will use a custom user-agent string which shows how we can spoof our default user-agent, and gives some privacy :

- The objective is to verify if our User-Agent fingerprint will change, which would give a layer of anonymity.
- The tool should spoof our current fingerprint to a custom fingerprint that we would decide.
- In the test, we will be using MacOSX as our operating system, and Firefox, as our browser.
- Install User-Agent Switcher on Firefox.

- Make sure to add the custom user-agent string.
- Browse to the website mybrowserinfo in order to verify if the spoof has been successful.

1.1.3 Proxy Chaining

There are different ways to attempt hiding over the internet. Making oneself anonymous using proxy chains is one of those.

Making a link to the first assignment, the use of proxy chaining allows you to use Nmap commands, for example, and not be traced back. Without proxy chaining, the attacker can be traced back because when exploring live systems and checking for open ports, the target receives information about the attacker from the TCP communication with the 3-way handshake.

Proxy chaining provide the user with network invisibility and act as a mean to perform network attacks. It can provide anonymity when done correctly. Moreover, it allows for network scanning - port, type and version scanning, vulnerability scanning, among others, while greatly diminishing the risk of being identified by targets for it.

In computer networking, is a server application that acts as an intermediary between a client requesting a resource and the server providing that resource [6].

The proxy chaining is composed of several proxies, creating a bulk of addresses between the client and server. It acts on the behalf of the client regarding requests made, which masks the true origin of the request to the resource server.

Considering the above, we decided to test proxy chaining. The formulation and a few considerations are found below:

- The objective is to check whether or not proxy chaining hides the IP address, providing anonymity.
- The proxy chaining should replace the original IP address with another existing one, the proxy chaining address.
- Check over Firefox, at google.com and what is my IP address, the proxy chaining IP address.
- Whether or not is possible to make use of Nmap commands hiding behind the proxy chaining.
- The test is made making use of the Kali Linux machine provided by the laboratory.
- A Snapshot of the system will be made prior to modifications.
- The proxy chaining will be set up through the "proxychains.conf" file.
- The test will be done following the default mode on the ProxyList within the file.

2 Experiment results

2.1 Installation process

2.1.1 Tor Project

A Tor browser executable install file can be downloaded from the “The Tor projects” website, and installed as any other web browser application. The Tor browser is supported on a number of platforms such as for Windows, macOS, Linux, and Android operating systems. It is therefore reasonable to assume that the average naive web and computer user can download, install, and operate the web browser without much complexity, similarly to any other web browser any user most likely previously have used. According to the official documentation [1] the belief is that greater usability allows for more users, which in turn increases privacy and security. Given that a complex system also has fewer users, which reduces anonymity, usability and ease of use is a security requirement for the Tor project.

2.1.2 User-Agent Switcher

The user-agent switcher extension is easy for user to install, as it is installed as any other web browser extension and will run automatically without further user configuration. However, it is always a good practice if it is possible to verify the tool’s integrity to make sure it has not been modified. I believe that if someone is not tech-savvy it would be difficult to custom the user-agent strings to what he wants.

Relying solely on this tool for full anonymity would contribute to poor handling, as this tool does not hide the IP address, or other means to gather data, such as Canvas Fingerprinting. Furthermore, the documentation is available in user-agent faq. It may be a problem for someone not speaking English, but in overall it is straight-forward, however there are no images to help the user, to make it easier for understanding. It also gives more information about the tool. Furthermore, in the privacy section, it says that it does not collect any data, personal information to identify someone.

2.1.3 Proxy Chaining

The set up of the proxy chain was done in the Kali Linux machine. It provides a service built-in for anonymity, as well as other systems that are based on penetration testing. On the terminal, the file proxychains.conf controls the configuration. It provides support for HTTP and SOCKS5, among a few others. There are different types of proxy chain that be selected, it is possible to deal with DNS anonymity as well, can be used in conjunction with servers and applications that are TCP based, such as Nmap. To make use of it, simply stating proxychains prior to the command will make sure it goes through the proxies.

The step-by-step guide of the set up can be found in the item 2.2.3- Proxy Chaining.

2.2 Usage and misuse of tools

2.2.1 Tor Project

Due to the operations, infrastructure, and protocols used in the onion routing, the Tor system is slower than a normal web browser. By default, the number of relays used to construct a circuit through the network is set to three [1]. Although not available via the GUI settings of the browser it is technically possible to both increase and decrease the number of hops by editing the source code. An increase in the number of hops will decrease the performance of the network and increase user latency, as well as the likelihood of being identified, given that only a small number of users might use the same custom hop count as the user in the Tor network [1]. This is not something a normal user likely will stumble upon or accidentally change. The potential for a breach of anonymity caused by such an event can be deemed to be small, although not minimal. To showcase the overhead connection time caused by the Tor network, a simple speed test has been created to illustrate the difference between the connection made to various websites using the request library with a normal connection and Tor connection (Figure 4). In total eight URLs were tested using Python 3.10 and the Request library to time the difference between sending a request to the URL and receiving a response. The result can be viewed in table 1 and shows that most of the connections made using Tor was clearly slower than using a normal connection. It should be noted that during some initial testing some of the connection made using Tor was much slower, with a request response time difference of up to 45 seconds. We were unable to determine the cause for these deviations. The source code can be viewed in figure 2.

URL	Normal connection (sec)	Tor connection (sec)
https://httpbin.org/get	0.400598	3.243096
https://google.com	0.082053	3.474035
https://www.microsoft.com/	0.097102	0.649161
https://www.expressen.se/	0.134945	3.615685
https://www.netflix.com/	0.356963	3.657683
https://mail.google.com/	0.078766	4.606242
https://web.whatsapp.com/	0.176681	3.321725
https://www.kth.se/	0.022468	3.813562

Figure 1: Speed test results

```
1 import requests
2
3 urls = ["https://httpbin.org/get", "https://google.com", "https://www.microsoft.com/", "https://www.expressen.se/",
4         "https://www.netflix.com/", "https://mail.google.com/", "https://web.whatsapp.com/", "https://www.kth.se/"]
5
6 proxies = {
7     'http': 'socks5://localhost:9150',
8     'https': 'socks5://localhost:9150'
9 }
10
11 for url in urls:
12     print(f"Url: {url} + \t\t")
13     f"Normal: {str(requests.get(url).elapsed.total_seconds())} \t" +
14     f"Tor: {str(requests.get(url, proxies=proxies).elapsed.total_seconds())}"
15 )
```

Figure 2: Speed test results

A potential problem of the slower request response time is that users will leave the network and revert back to a normal browser. Not only this worsens the individual user's anonymity on the web, but also every other user in the Tor network. As mentioned in the "Installation process" section Tor derives some of its anonymity through the number of users. A reduction in users, for example due to slow speeds, will cause a decrease in the overall anonymity provided by the network.

Additionally, although maybe not likely in a normal use case scenario, a user that continuously runs the Tor browser, never shuts down his/her computer, as well as never generates a new identity, the Tor browser will never purge the cookies sent from various servers made during a web surfing session. Cookies can also be maintained on the client in the event of a poor privacy settings configuration that would disable such mechanisms. Not removing the cookies could effectively remove the anonymization provided by the Tor network, since the cookies could/will continuously collect information about the user during usage of the browser and connection to the websites. For example, sending back the real IP address of the user to the server. To illustrate, we configure a Tor browser to never purge cookies during shutdown by turning off the setting "Always use private browsing mode". In figure 3 we can observe the Tor browser just after startup and notice that no cookies are currently stored. Then we visit a handful of web-pages and restart the browser and observed that in figure 4 a number of cookies is stored on the browser, and will remain running on the client browser unless steps are taken to have them removed.

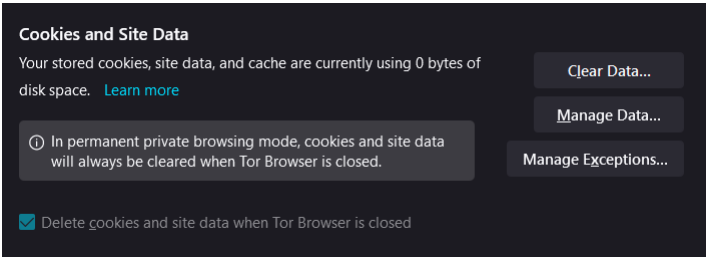


Figure 3: Cookies on startup of Tor browser with good configuration

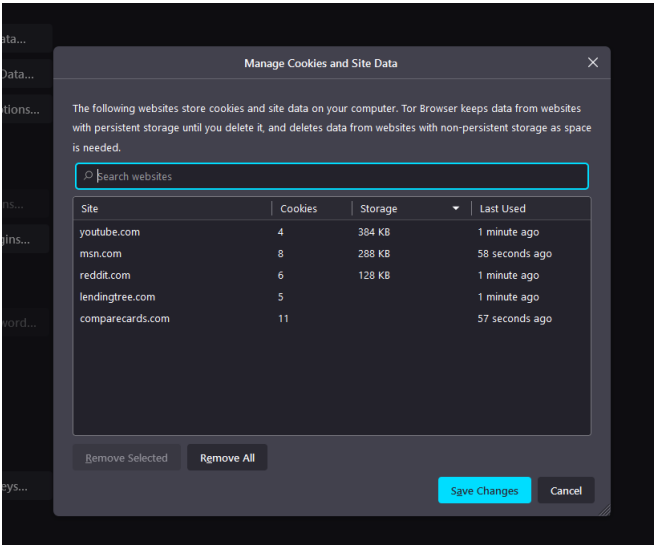


Figure 4: Cookies after shutdown and restart of Tor browser with poor configuration

2.2.2 User-Agent Switcher

The default fingerprint will be spoofed into the following custom fingerprint: **Mozilla/5.0 (Windows NT 5.1; AOL 9.0; en-US; rv:1.7.5) Gecko/20041202 Firefox/1.0**. This would make me appear using Windows XP and AOL v9.0. We can see the result in Figure 5. As can be seen, now websites see that we are using Firefox version 1, under Windows XP operating System, and we are using AOLv9. In this string, we have also the possibility to change the language of our browser, as you may have seen in Figure 5, we have put "en-US", we could have put another language. Our user-agent string has been successfully changed into the one we wanted. If we are in a certain environment where we need to have a special fingerprint, we would be able to do it. We also have the possibility to choose other options that are proposed by the tool.

Your Browser User Agent String is	
Mozilla/5.0 (Windows NT 5.1; AOL 9.0; en-US; rv:1.7.5) Gecko/20041202 Firefox/1.0	
AOL:	You are using AOL version 9
Operating System:	Microsoft
Platform:	WinXP
Internet Browser:	Firefox 1.0
Beta Version:	No
Connection Speed:	Not detectable due to an invalid reading. This typically occurs when testing over a local network connection. Speed detection is meant only for determining speeds around T3 and slower.

Figure 5: Windows XP + AOL9.0

In this tool, it protects against tracking users through user-agent fingerprints. Because we have the possibility to spoof our default one, into any other fingerprint. For instance, we can appear as a PlayStation 4. That being said, users should be aware that this tool does not bring 100% anonymity, it gives a certain layer of anonymity, but it is a good tool when combined with other tools. There's no magic tool that does everything. This tool does not cost anything, it brings more benefits than anything, as the tool should work properly, and be available as long as it is supported by the browser. This tool brings some layer of anonymity, but it should still be combined with a proxy or VPN to bring full power into having different fingerprints, and become harder to identify as your fingerprint would change every time, and the location of that fingerprint as well.

2.2.3 Proxy Chaining

First are foremost, the IP address, coming from Sweden and prior to using proxies, can be found below:

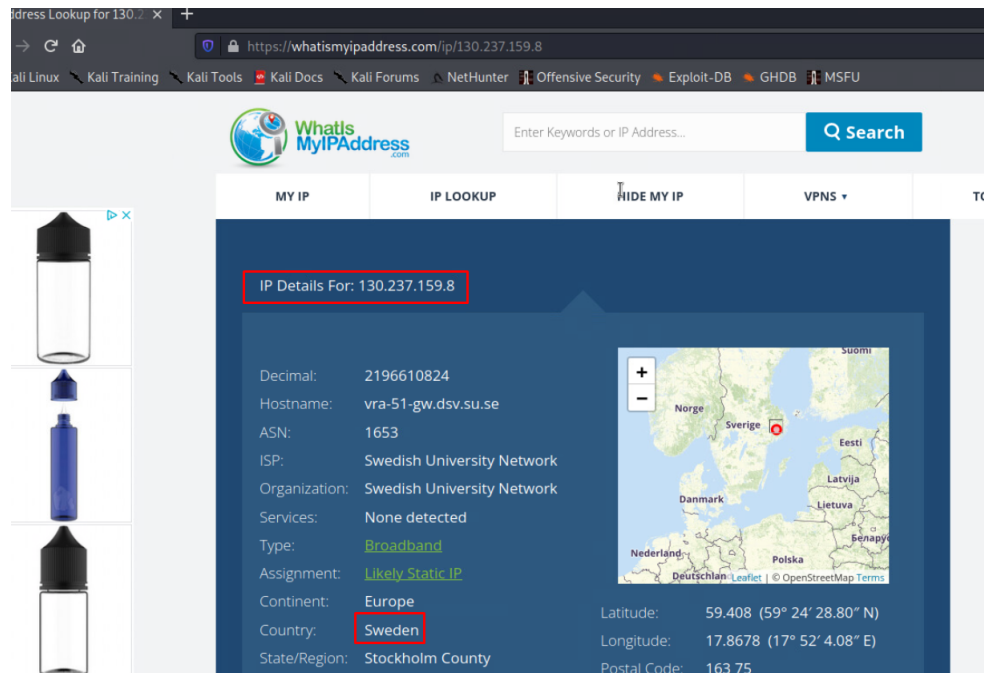
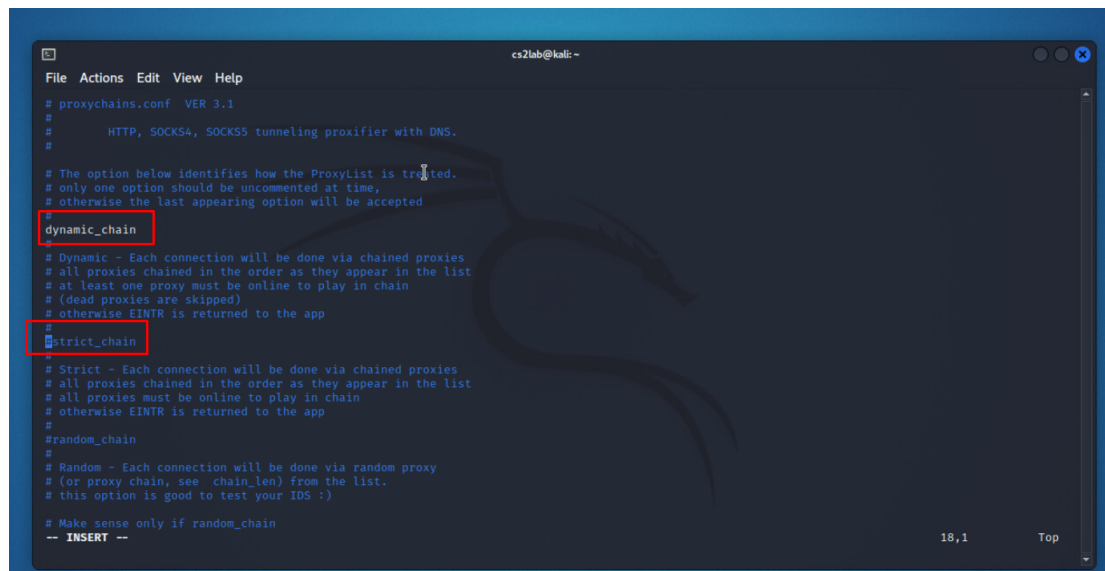


Figure 6: the IP address prior to proxy

The objective is to follow steps in order to verify a different public IP address by using proxies. There is a file within /etc/ called "proxychains.conf". It is possible to set up Proxychains through the file. We changed the default settings. Here we can see that there are three different types of chain. We selected the dynamic chain, which allows us to create a list of proxies chained proxies, while dead ones are skipped.

There are 2 other modes of chaining:

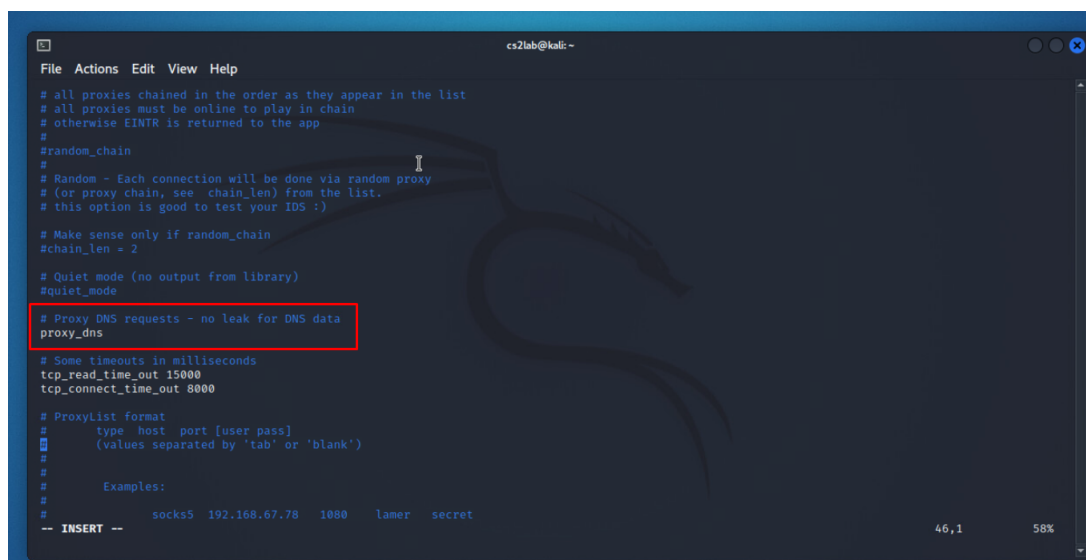
- Strict mode: which similarly to dynamic, follow the order of proxies from top to bottom on the list. However, it does not have flexibility to skip the ones that might not be live;
- Random mode: The proxies in the list are selected in random manner. Therefore the order which they are listed does not matter.



```
# proxychains.conf  VER 3.1
#
# HTTP, SOCKS4, SOCKS5 tunneling proxyifier with DNS.
#
# The option below identifies how the ProxyList is treated.
# only one option should be uncommented at time,
# otherwise the last appearing option will be accepted
dynamic_chain
#
# Dynamic - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# at least one proxy must be online to play in chain
# (dead proxies are skipped)
# otherwise EINTR is returned to the app
#
strict_chain
#
# Strict - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# all proxies must be online to play in chain
# otherwise EINTR is returned to the app
#
random_chain
#
# Random - Each connection will be done via random proxy
# (or proxy chain, see chain_len) from the list.
# this option is good to test your IDS :)
#
# Make sense only if random_chain
-- INSERT --
```

Figure 7: Dynamic Chain Proxy

The DNS goes through the chain of proxies as well. If this statement is left out, the IP address goes through proxies, which hides the client, but the DNS will not, which breaks the anonymity and reveals the identity of the client. That is a critical mistake to make, since it invalidates the purpose of attempting to hide the origin IP address behind a chain of proxies.



```
# all proxies chained in the order as they appear in the list
# all proxies must be online to play in chain
# otherwise EINTR is returned to the app
#
random_chain
#
# Random - Each connection will be done via random proxy
# (or proxy chain, see chain_len) from the list.
# this option is good to test your IDS :)
#
# Make sense only if random_chain
#chain_len = 2
#
# Quiet mode (no output from library)
#quiet_mode
#
# Proxy DNS requests - no leak for DNS data
proxy_dns
#
# Some timeouts in milliseconds
tcp_read_time_out 15000
tcp_connect_time_out 8000
#
# ProxyList format
# type host port [user pass]
# (values separated by 'tab' or 'blank')
#
# Examples:
# socks5 192.168.67.78 1080 lamer secret
-- INSERT --
```

Figure 8: the DNS goes through Proxy

The file informs the user about the format of the proxies in the list. The format is, in sequence from the left to the right:

- The type of the connection (network protocol) to be used, space;
- The host address, space;
- The port.

```

cs2lab@kali:~$ nano /etc/proxychains.conf
# Quiet mode (no output from library)
#quiet_mode

# Proxy DNS requests - no leak for DNS data
proxy_dns

# Some timeouts in milliseconds
tcp_read_time_out 15000
tcp_connect_time_out 8000

# ProxyList format
# type host port [user pass]
# (values separated by 'tab' or 'blank')
#
# Examples:
#
# socks5 192.168.67.78 1080 lamer secret
# http 192.168.89.3 8080 justu hidden
# socks4 192.168.1.49 1080
# http 192.168.39.93 8080
#
# proxy types: http, socks4, socks5
# ( auth types supported: "basic"-http "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
-- INSERT --

```

Figure 9: ProxyList

In the ProxyList, the proxies are inserted and will compose the chain. They are used from top to bottom. The more proxies added to the list, the stronger is the level of the anonymity. It should be noted that it is still possible to go through logs within proxies and try to find out about the origin IP address, but in most cases it is not a viable task, given the volume of data and possibilities to look for. Important considerations about proxy chain are made in the chapter "Comparison of Protection and Vulnerabilities". It defaults to Tor, if no other proxies are specified.

The proxy chain can be, however, specified. In this case, the default "Tor" is commented out and the chain is specified.

- The type of the connection (network protocol) to be used, space;
- The host address, space;
- The port.

```

GNU nano 5.9 /etc/proxychains.conf
# type host port [user pass]
# (values separated by 'tab' or 'blank')
#
# Examples:
#
# socks5 192.168.67.78 1080 lamer secret
# http 192.168.89.3 8080 justu hidden
# socks4 192.168.1.49 1080
# http 192.168.39.93 8080
#
# proxy types: http, socks4, socks5
# ( auth types supported: "basic"-http "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks5 127.0.0.1 9050
http 8.218.29.111 59394
http 190.85.115.78 3128
http 102.164.252.150 8080
socks5 72.221.232.155 4145
...

```

Figure 10: Proxy Chain List

When executing the proxy chain on the command, opening Firefox behind the proxies, Google recognizes the new IP, which actually the one from the last proxy in the chain. It receives a German address, which is completely independent from the initial original IP address, which was from Sweden. This is a sign that the proxy chain was successful.

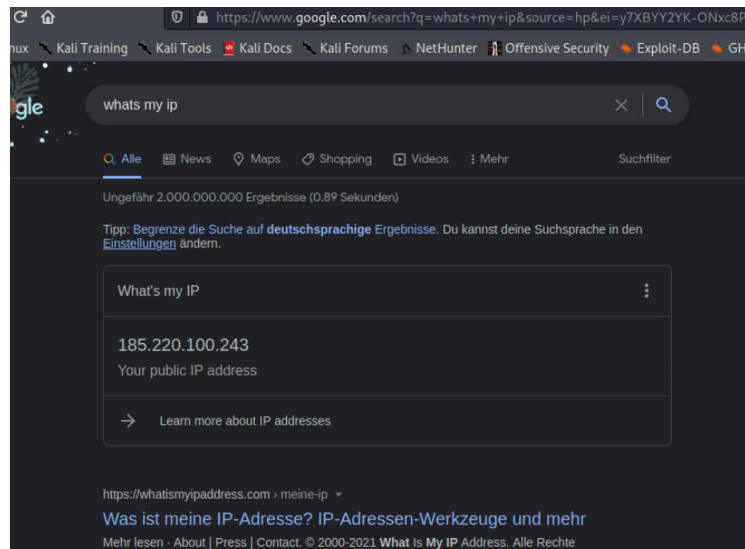


Figure 11: new IP address according to Google

The following image confirms the actual location inside Germany.

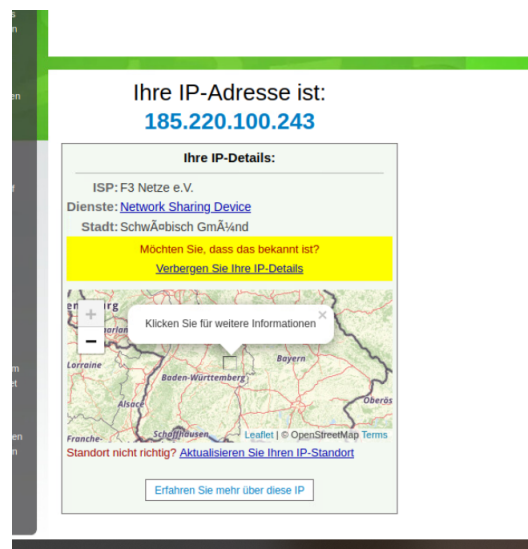


Figure 12: new public IP address from Germany

As previously stated, it is possible to use Nmap scanning commands through the proxy chain as well, providing anonymity. The following image demonstrates the use of Nmap when aiming at ports from www.Youtube.com.

```
File Actions Edit View Help
cs2lab@kali:~$ proxychains nmap -sP www.youtube.com
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-06 15:44 CET
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... www.youtube.com:80 ... OK
Nmap scan report for www.youtube.com (224.0.0.1)
Host is up (0.28s latency).
rDNS record for 224.0.0.1: all-systems.mcast.net
Nmap done: 1 IP address (1 host up) scanned in 1.22 seconds
cs2lab@kali:~$ proxychains nmap -sT -p 80,443 youtube.com
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-06 15:45 CET
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... youtube.com:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... youtube.com:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:9050 ... youtube.com:443 ... OK
Nmap scan report for youtube.com (224.0.0.1)
Host is up (0.32s latency).
rDNS record for 224.0.0.1: all-systems.mcast.net

PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
Nmap done: 1 IP address (1 host up) scanned in 1.19 seconds
cs2lab@kali:~$
```

Figure 13: Port Scanning using Nmap commands toward www.Youtube.com

2.3 Comparison of protection and vulnerabilities

Considerations about Tor and User-Agent Switcher

When comparing the protection provided from Tor with the user-agent switcher, we observe that the latter makes no attempt to obfuscate the origin of the connection thereby reducing the anonymity protection substantially. Instead, the user-agent switcher has a less complicated and thereby also more vulnerable approach which attempts to hide the identity of the user by spoofing the default user-agent metadata. As has been discussed in Section 1.1.2, when visiting any websites we leave some traces, that can be used by different websites to create user's fingerprints, which may be used for tracking. In that way, this tool, protects against tracking users through user-agent fingerprints. Because we have the possibility to spoof our default one, into any other fingerprint. For instance, we can appear as any other device. However, the users should be aware that this tool does not bring 100% anonymity, but gives a certain layer of anonymity. It is better when combined with other tools (e.g. Proxy, VPN etc.), as it would allow having different fingerprints, and thus becoming harder to identify. Indeed, users' would have fingerprints changing every certain time, with different metadata, and IP location. Doing that, would also make it harder for attacker to decide on an attack vector, as we would have different metadata, and thus hiding our real metadata.

Considerations about Proxy

There are different types of proxy. Proxies can be transparent or anonymous[7]:.

- Transparent: They are shown to be a proxy server to the destination as well as reveal the address of origin under the header of X-Forwarded-For HTTP.
- Anonymous: These proxies hide the identity of the original IP address, although still reveal that they are a proxy.

This means that even anonymous proxies aren't completely anonymous, because they do disclose the fact that they are proxies. Moreover, proxies can be targeted by hackers, since many are machines using HTTPS/SOCKS for example, and can be source for eavesdropping coming from a third party. Therefore, it be a gamble to trust proxies.

The logging of the proxy could also be enabled, registering requests as well, which are saved and could reveal information about the origin if investigated.

Proxy Chains and Tor

In comparison with Tor, proxies do not provide encryption as a general rule, while Tor does. There are different levels of encryption, and Tor covers all of them.

1. The encryption of the content of the message, through the use of SSL/ TLS. HTTPS and SOCKS5, for example, supports this. This level of encryption can be applicable to Tor as well as proxy chains, when connecting to a SSL/TLS protected web server.
2. The encryption of the headers. In a proxy chain, the HTTPS header would reveal the Forward message with the IP address of the proxies. In Tor, this is encrypted, however, in proxy chains, it is not.

Therefore there needs to be trust placed in every proxy in the chain, one after the other. This is not a concern with Tor, however.

The Tor project documentation touches upon this subject, pointing out differences [8]:

"Proxychains is a program that sends your traffic through a series of open web proxies that you supply before sending it on to your final destination. Unlike Tor, proxychains does not encrypt the connections between each proxy server. An open proxy that wanted to monitor your connection could see all the other proxy servers you wanted to use between itself and your final destination, as well as the IP address that proxy hop received traffic from. Because the Tor protocol requires encrypted relay-to-relay connections, not even a misbehaving relay can see the entire path of any Tor user".

To conclude, Proxy is a tool that is focused to change one's IP Address, whereas Tor is a full tool that is used to hide someone's identity and block any web tracking metadata by giving a default result to all websites. Where User-Agent is a tool that is used only for spoofing our default metadata, which would give a layer of anonymity, but should be combined with a VPN or Proxy to be fully powerful and obtain a better anonymity.

3 Time estimation

Time estimation for the total group work was approximately 26 hours for each group member.

4 Group reflection

In this assignment, we have learnt that obtaining anonymity is quite difficult, and we should not put all our basket into one tool in order to obtain full anonymity. Indeed, each tool gives a certain level of privacy, but they still do not bring full anonymity. The ideal way, is to combine multiple tools in order to have better level of privacy. However, when doing that the user should be aware that the chances of him being unique are increased. Additionally even a quite complex and privacy focused tool such as Tor have its limitations and vulnerabilities that can be caused both by malicious actor, but also due to misuse in configuration and usage of the tool, even though it from the outset can be perceived to be extremely secure in regards to anonymity.

References

- [1] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router.” [Online]. Available: <https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.pdf> [Kontrollerad: January 3, 2022]
- [2] P. Gray, “Embassy hacker’ dan egerstad and the tor network.” [Online]. Available: <https://www.computerweekly.com/news/2240022106/Embassy-hacker-Dan-Egerstad-and-the-Tor-network> [Kontrollerad: January 2, 2022]
- [3] Wikipedia, “User agent.” [Online]. Available: https://fr.wikipedia.org/wiki/User_agent [Kontrollerad: January 3, 2022]
- [4] Pypo, “Browser fingerprinting / de quoi parle t-on ?” [Online]. Available: <https://www.pypo.eu/solutions/navigation-securisee/browser-fingerprinting-de-quoi-parle-t-on/> [Kontrollerad: January 3, 2022]
- [5] Revoltenum, “Révolte numérique is watching you ! les traces que vous laissez...” [Online]. Available: <http://www.revoltenumerique.herbesfolles.org/2011/09/03/revolte-numerique-is-watching-you-les-traces-que-vous-laissez/> [Kontrollerad: January 3, 2022]
- [6] Wikipedia, “Proxy server.” [Online]. Available: https://en.wikipedia.org/wiki/Proxy_server [Kontrollerad: December 20, 2021]
- [7] Whonix, “Tor vs. proxies, proxy chains and vpns.” [Online]. Available: https://www.whonix.org/wiki/Comparison_Of_Tor_with_CGI_Proxies,_Proxy_Chains,_and_VPN_Services [Kontrollerad: December 20, 2021]
- [8] T. Project, “Tor project:faq.” [Online]. Available: <https://2019.www.torproject.org/docs/faq.html.en#Proxychains> [Kontrollerad: January 3, 2022]