EMPIRICAL ANALYSIS OF CENSORSHIP RESISTANCE SYSTEMS

BY

ABDULLAH BIN AYEDH ALQAHTANI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SECURITY AND INFORMATION ASSURANCE

JANUARY 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ABDULLAH BIN AYEDH ALQAHTANI** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN SECURITY AND INFORMATION ASSURANCE**.

Thesis Committee

[illegible]
Dr. Sami M. Zhioua (Adviser)

[illegible]
Dr. Farag A. Azzedin (Member)

Md. Rafiul Hasan
Dr. Mohammed Rafi Ul Hassan (Member)

[illegible]
Dr. Khalid Al-Jasser
Department Chairman

[illegible]
Dr. Salam A. Zummo
Dean of Graduate Studies

20/3/18
Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
1963

Dedicated to
my beloved parents, sisters and brothers for all your support, and the endless love.

ACKNOWLEDGMENTS

First of all, praise be to Allah, who gave me all the blessing to complete this work. Even with many challenges through the way, your grace gave me the strength to finish my thesis.

Nothing could be done without the support, prayers, and the passion of my parents thank you all from the bottom of my heart.

My gratitude and appreciation to all my brothers and sisters who supported me with every possible way to finish my studies, Thank you all.

My sincerest gratitude is extended to my mentor and advisor Dr.Sami Zhioua. Thank you for all the knowledge you gave, your patience and your guidance through my journey in the university.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AES	The Advanced Encryption Standard
BuFLO	Buffered Fixed-Length Obfuscator
CGI	Common Gateway Interface
CRS	Censorship Resistance Systems
CSS	Cascading Style Sheets
DNS	Domain Name System
DPI	Deep Packet Inspection
ESR	Extended Support Release
FF	FireFox
FTE	Format-Transforming Encryption
FTP	File Transfer Protocol
HAR	HTTP Archive format
HTTP	Hypertext Transfer Protocol
HTTPOS	HTTP Obfuscation
HTTPS	Hyper Text Transfer Protocol Secure
ISP	Internet Service Provider
JAP	Java Anon Proxy
JSON	JavaScript Object Notation
k-NN	k-Nearest Neighbours

obfs2	The Twobfuscator
obfs3	The Threebfuscator
OFB	The Output Feedback mode
OSAD	Optimal String Alignment Distance
RSA	Rivest–Shamir–Adleman (cryptosystem)
SNI	Server Name Indication
SOCKS	Socket Secure
SSH	Secure Shell
SVM	Support Vector Machine
TBB	Tor Browser Bundle
TCP	Transmission Control Protocol
TLS	Transport Layer Security
Tor	The Onion Router
Tor-PT	Tor Pluggable Transport
UDP	User Datagram Protocol
WF	Website Fingerprinting
WLLCC	Weight Learning by Locally Collapsing Classes
XML	eXtensible Markup Language

THESIS ABSTRACT

NAME:	Abdullah Bin Ayedh AlQahtani
TITLE OF STUDY:	Empirical Analysis of Censorship Resistance Systems
MAJOR FIELD:	Security and Information Assurance
DATE OF DEGREE:	JANUARY 2018

One of the most important features that characterize the Internet is freedom. However, the Internet is still censored almost everywhere by people, companies or particularly governments. There are many reasons to apply Internet censorship some of which are political, religious or moral reasons. In contrast, individuals are using censorship resistance systems (CRS) to access the banned sites in their countries. The arms race reached high levels between Internet users and censorship systems. This lead to more improvements on both sides in a way achieves both contradictory goals. In this thesis, we empirically study well-known censorship resistance systems (CRS). We conducted more than 300 experiments related to traffic analysis including statistical analysis, HTTP Archive format (HAR) analysis and Website fingerprinting attacks. To the best of our knowledge, no previous work did such an exhaustive empirical analysis. The detailed analysis allowed us *to reveal surprising results and draw very interesting conclusions in this crucial arm race of privacy and censorship.*

ملخص الرسالة

الاسم	:	عبدالله بن عائض القحطاني
عنوان الرسالة	:	تحليل تطبيقي لأنظمة تخطي الحجب
التخصص	:	امن وضمان المعلومات
تاريخ التخرج	:	يناير ٢٠١٨

الحرية هي واحدة من اهم المميزات التي تميزت بها الشبكة العنكبوتية منذ بداية ظهورها. وعلى الرغم من ذلك فإن هناك العديد من الجهات التي تقوم بفرض رقابة على استخدام الانترنت. هذه الرقابة يتم تطبيقها بواسطة أفراد او مؤسسات او حتى حكومات داخل مجال سيطرتها. هناك اسباب عديدة لدى جهات الرقابة لتبرير فرض مثل هذه القيود وتتنوع بحسب اسبابها، فمنها ماهو لأسباب اقتصادية او دينية او سياسية او اخلاقية.

في الجانب الآخر هناك مجموعة من مستخدمي الانترنت تنظر لمثل هذه الرقابة على انها تقييد لحرية استخدام الانترنت وتدخل في خصوصيات المستخدمين. من هذا المنطلق بدأت هذه المجموعة في تطوير برمجيات تهدف الى تخطي الرقابة والوصول لأي محتوى محظور على الانترنت. ولهذه المجموعة عدة اهداف منها حماية خصوصية مستخدمي الانترنت من رقابة الجهات العليا و توفيرالوصول للمحتويات المحظورة داخل الانترنت.

هذا الصراع بين كلا الطرفين نتج عنه ظهور تقنيات عديدة جعلت كل طرف يجتهد لإيجاد وسيلة لهزيمة الطرف الآخر. فعند اقتراح وتطبيق نظام لتخطي الحجب كالنظام المشهور المعروف بإسم تور (Tor)، فإن انظمة الرقابة تسارع بإيجاد طريقة في كشفه او التعرف عليه ومن ثم اغلاقه.

في هذا البحث سنقوم بتسليط الضوء على اشهر انظمة تخطي الرقابة المستخدمه بشكل فعال واشهر تلك الانظمة هو نظام تور وملحقاته ونظام اخر يطلق عليه اسم جاب (JAP) . وسنقدم في هذا البحث تطبيقات عملية وتفصيل كامل عن هذه الانظمة وكيفية عملها في تخطي انظمة الرقابة. بالإضافة الى ذلك فقد قمنا بإجراء مايزيد عن ٣٠٠ تجربة في تحليل البيانات المشفرة التي تنتجها هذه الانظمة عند استخدامها. ويمكن تصنيف هذه التجارب الى ثلاث انواع اساسية الاولى تتعلق بتحليل احصائي لحركة البيانات (Statistical analysis)، النوع الثاني تحليل حركة البيانات على مستوى المتصفح (HTTP Archive analysis)، والنوع الأخير هو تطبيق هجمات التعرف على بصمات المواقع (Website fingerprinting attack).

CHAPTER 1

INTRODUCTION

Sharing information and latest news using the Internet takes only a few seconds. Furthermore, the freedom of the Internet made it easy for anyone to publish what he or she desires. In contrast, some may think that a free Internet will make it a place to share information that might not suit everyone. For example, publishing information that may threaten national security, exposing children to inappropriate contents, or even a child using the Internet at a time prohibited by the parents. Hence, the concept of Internet censorship began as a big brother who will make the Internet suitable for everyone under his protection.

On the other hand, some Internet users consider such control as suppression of freedom or privacy violation. Such users are interested in keeping the Internet as free as possible to share information, ideas or even scandals. Moreover, they want to protect their Internet activities from any monitoring attempts. In some countries, those monitoring operations could be the beginning of series of events that will lead to imprisonment or death.

As a result of that, each side of the story will develop the tools and means to achieve its objectives. The censorship will start to use the techniques that will keep any published contents under its control. On the contrary, the anti-censorship group will seek to defeat this concept to achieve its goal of a free Internet. In the middle of this arms race, many remarkable techniques produced that are worthy of study and research.

Censorship will start to apply some statistical analysis of traffic patterns generated by censorship resistance systems. This kind of traffic analysis will help censorship to see the effects and the changes that occur in the traffic after using such systems. Moreover, censorship can apply a specific type of fingerprinting attacks which is website fingerprinting attacks. In this attack, the censorship will try to identify the website visited by the user while using any CRS. This type of attack depends on machine learning and the traffic analysis of the encrypted traffic.

In this work, we studied censorship and censorship resistance systems to see the techniques and strategies used by each party to achieve the goals. Furthermore, we did an exhaustive empirical analysis of each censorship resistance system using statistical analysis of traffic patterns and website fingerprinting attacks.

In this chapter, we introduce the motivation to conduct this research. After that, we mention our contributions that are related to the analysis of censorship resistance systems (CRS). Finally, we dedicate a section about how the thesis organized.

1.1 Motivation

Internet users around the world use censorship resistance systems to preserve their privacy from any monitoring attempts by censorship. As an illustration, Tor [1] is considered one of the most popular programs used to maintain privacy and bypass censorship. According to statistics from Tor Metrics [2], the estimated number of Tor users are around two million users between August and September 2017. These statistics, indicate that a large number of Internet users around the world want to maintain their privacy while surfing the Internet. There are also many successful attempts to limit the access to Tor network in countries such as China. Citizens of such countries are unable to maintain their privacy or accessing blocked sites.

There are numerous recent researches regarding censorship since its an important topic. Each one of them focused on some field such as censorship classification, the method of attacks or the nature of how they work. Other researchers are interested in developing new systems that will try to circumvent censorship.

The main motivation of this research is the sophisticated techniques of both parties that are significantly evolving over a short period. Moreover, most studies concentrated on the theoretical aspects, especially in censorship resistance systems. In this research, we focused on the empirical analysis aspect by deploying and studying each one of the selected systems. In addition to that, our goal is to complete the full picture and study three major topics which are censorship, censorship resistance systems and website fingerprinting attacks and defenses. During all this research, we stand on the neutrality between the parties and study them individually and practically.

1.2 Thesis Contributions

- **Real world examples of censorship techniques and implementations**: We linked the theoretical part with real-world example of censorship and Tor system to complete the picture. We dedicate a section to discuss the Internet filtering in Saudi Arabia along with the organizations and regulations that organize its work.

- **Empirical comparison of censorship resistance systems**: We focused on our research to target the well-known censorship resistance systems. Tor, Tor pluggable transports, JAP and YourFreedom are well-known and currently active systems. We did an exhaustive empirical analysis of the selected systems to compare between them using different setups.

- **The largest set of different censorship resistance systems classes covered in one research** : We targeted a set of published censorship resistance systems, and we selected eight of them. Other systems excluded due to technical issues or lack of support from the developers.

- **The largest collection of datasets for various censorship resistance systems**: We build a diversified and updated data sets of the selected systems. In our research, we collected 12 different types of data sets as 48000 files of network traces.

- **Exhaustive statistical analysis on censorship resistance systems and website fingerprinting defenses**: We conducted around 576 experiments to see how those systems change the normal traffic. To process such big set of files, we developed our tools to automate such process.

- **The impact of the browser type on traffic generated by different censorship resistance systems**: We conduct HTTP Archive (HAR) analysis using different browsers with different CRS. This experiment illustrate the effect of using different browsers with each censorship resistance systems.

- **Comparison of censorship resistance systems and different defenses regarding website fingerprinting resistance** : We apply three website fingerprinting attacks and five defenses on the collected samples. Our results are based on 321 experiments using 1284000 files that are extracted and processed from the network traces mentioned before.

- **Visual demonstrations of complex and important topics in details** : We worked hard in this research to design and represent a visual explanation by illustrating each important concept in figures. This technique is very important for the reader to understand the concept quickly.

1.3 Organization of the Thesis

In this thesis, we cover three main topics in the first three chapters which are censorship, censorship resistance systems and website fingerprinting. In the last three chapters, we discuss each type of experiment and the outcome results. In chapter 4, we explain the experiments setups along with the challenges we faced. In chapter 5, we discuss the statistical analysis of traffic patterns. In chapter 6, we discuss the resistance to website fingerprinting attacks for each of the selected CRS and defenses.

In chapter 2, we start with an overview of censorship systems in general. We explain the classification of censors to see the different types of censorship. We go more in-depth to see that each censor has capabilities and attacking techniques to impose its control. After that, we mention the censor limitations to see what the censor can and can't do. We dedicate a section for a case study of real-life example between Tor and censorship. At the end of that chapter, we describe the Internet filtering system in Saudi Arabia.

In chapter 3, we start with an overview of censorship resistance techniques in general to see the different classes used. We explain that censorship resistance systems strategies could fall into one or more of six strategies known as CORDON taxonomy. After that, we discuss different examples of censorship resistance systems (CRS) that are active and running nowadays. We explain how each selected CRS succeed to bypass the censorship.

In chapter 4, we mention a specific type of attack used on encrypted traffic which is fingerprinting attack, mainly website fingerprinting attack. We mention different types of attacks, and different type of defenses that resist those attacks.

In the chapters 6 and 7, we discuss the results of more than 800 experiments conducted in this research. The experiments are divided into two major classes which are the statistical analysis of traffic patterns and the resistance to website fingerprinting attacks. In the first class of experiments, we show how we conduct the statistical analysis and HTTP archive format (HAR) analysis along with the results. In the second class of experiments, we show the impact of website fingerprinting attacks and defenses on each one of the selected systems.

CHAPTER 2

CENSORSHIP

One of the most important features that characterize the Internet is freedom, and the Internet was born free. However, the Internet is still censored almost everywhere by people, companies and particularly governments. There are many reasons to apply Internet censorship some of which are political, religious or moral reasons. On the other hand, Internet censorship could be justified for other reasons such as solving crimes that could threaten the national security or to increase the efficiency of the system. On the other hand, Internet users will consider this act as a violation of their privacy. For more clarification, there are two concepts related to monitoring Internet users and their activities which are *Censorship* and *Surveillance.*

Censorship is the process of observing communications, deleting or blocking harmful contents based on political, economical or social point of view. Censorship could involve blocking websites, arresting or harassing people who published those

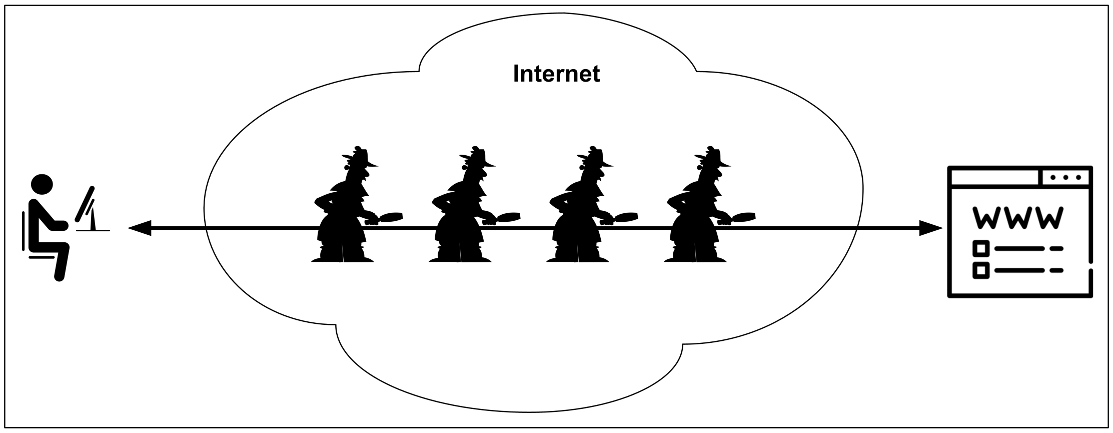


Figure 2.1: Users and Internet

harmful contents. *Surveillance* is defined as the close observation of an individual or group that may have been involved in subversive or terrorist activities. Law enforcement needs Internet surveillance to prevent terrorists or criminals from using the Internet in malicious activities [3]. Censorship is often associated with dictatorship and repressive governments in an attempt to stop certain activities. On the other side, surveillance is considered as an important aspect in democratic countries [4].

Both censorship and surveillance could interfere with the user's privacy. *Privacy* is defined as the right to left alone without having others to collect information about you. Privacy is a legitimate right for each individual unless it starts to harm others. To maintain the privacy of Internet users, information security community starts to develop solutions to circumvent censorship systems called Censorship Resistance Systems or anti-Censorship systems. At that point, the arms race began between the two parties each party will try to beat the other using new techniques in various ways.

2.1 Censor Classification

There are several motivations to carry out censorship based on the censor's goal and capabilities. For example, parents would like to prevent their children from accessing immoral contents. Companies would like to prevent access to the contents that will reduce staff's productivity or efficiency of the systems. Governments may limit the freedom of speech regarding topics that could embarrass it in front of the world. In general, we can classify the motivation of censorship to social, economic or political reasons.

Each censor has a sphere of influence where it can impose its control on it. Censors can control the access to contents that are located outside the sphere of influence, but it cannot control the content publication. For example, parents can control the access to immoral contents, but they cannot remove those contents from the Internet. Similarly, governments can block citizens from accessing opposition contents, but those contents are published outside the sphere of influence. In addition to that, each censor has technical capabilities that vary based on its goal. Governments are known to have powerful and sophisticated types of equipment to observe Internet infrastructure within its sphere of influence.

Elahi and Goldberg al. [5] categorized censors by their motivation, sphere of

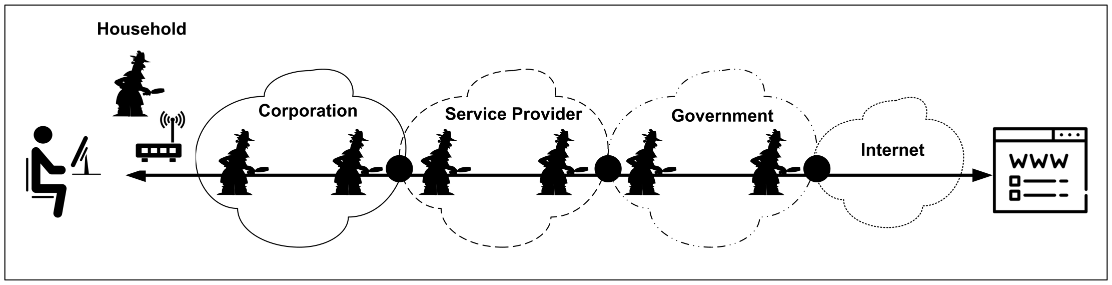


Figure 2.2: Censor Classification

influence, and technical capabilities. There are four major censorship categories which are household, corporation, service provider, and government.

2.1.1 Household Censor

This censor is motivated by social and moral reasons to limit the access to inappropriate contents on the Internet. Often it is managed by parents to control the content that their children watch. This Censor has simple technical capabilities inside a limited sphere of influence. Usually, its located inside the home Internet router to control the contents accessed by any users in that network.

2.1.2 Corporation Censor

This censor has several motivations related to productivity, efficiency, and the legal agreements between the employee and his employer. Often, It is managed by the department of information technology in any corporation. The sphere of influence for that censor will include all the infrastructure inside the corporation. The technical capabilities of this censor are more advanced than home censor to monitor the staff traffic and activities.

2.1.3 Service Provider Censor

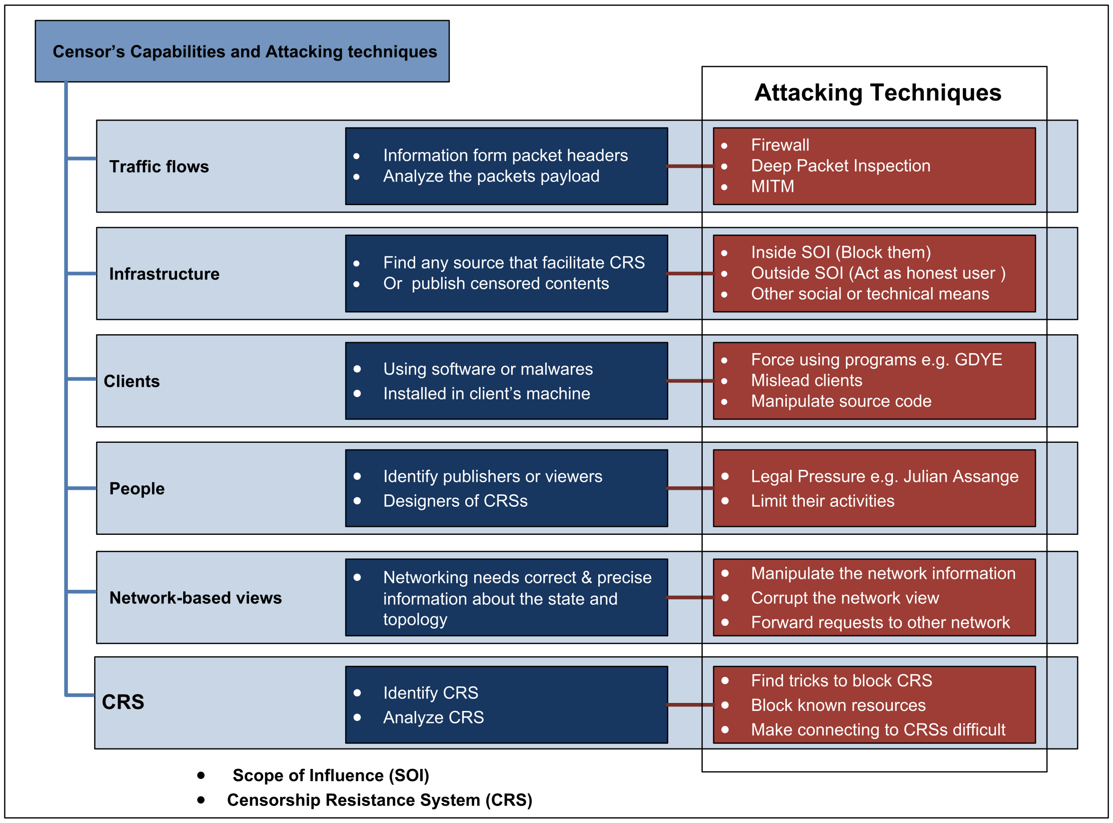
This censor is motivated by profit and economical reasons. It has the power to observe and monitor all the traffic that occurs within its infrastructure. It has the technical capabilities that can remove any unwanted contents placed inside its infrastructure, or block any inside requests to undesirable outside contents. Any Internet service provider around the world can be an example of this type of censor.

2.1.4 Government Censor

The goal of this censor is to remain in power either by good governance or persecution. The sphere of influence for that censor include all the infrastructure inside its border. It can use any social or technical means to censor all the traffic within its border. It will force the users to access the allowed contents only, and forbid any publication of censored contents. Depending on the country, it may involve arresting or charging those who violate the laws.

2.2 Censor Capabilities and Attacking Techniques

For the censor capabilities, the censor can do two techniques which are *blocking* or *detection*. The term filtering can sometimes be used to refer to blocking. Elahi and Goldberg al. [5] mentioned six entities that could be censored or attacked by the censor. In this section, we explain how the censor can apply blocking or detection.



Censor's Capabilities and Attacking techniques
Attacking Techniques
Traffic flows
Information form packet headers
Analyze the packets payload
Firewall
Deep Packet Inspection
MITM
Infrastructure
Find any source that facilitate CRS
Or publish censored contents
Inside SOI (Block them)
Outside SOI (Act as honest user)
Other social or technical means
Clients
Using software or malwares
Installed in client's machine
Force using programs e.g. GDYE
Mislead clients
Manipulate source code
People
Identify publishers or viewers
Designers of CRSs
Legal Pressure e.g. Julian Assange
Limit their activities
Network-based views
Networking needs correct & precise information about the state and topology
Manipulate the network information
Corrupt the network view
Forward requests to other network
CRS
Identify CRS
Analyze CRS
Find tricks to block CRS
Block known resources
Make connecting to CRSs difficult
Scope of Influence (SOI)
Censorship Resistance System (CRS)

Figure 2.3: Summary of censor capabilities and attacking techniques

For the attacking techniques, there are logical and physical entities that will be attacked by the censor to disrupt any unwanted communications to the Internet. In this section, we mention the censor capabilities for each one of the six entities followed by the attacking techniques regarding each entity.

2.2.1 Traffic Flows

Packet headers have essential information such as IP addresses and port numbers. Censors can use this information to detect unwanted packets, and this can be done using firewalls. Also, censors can go further by keeping track of the links state. Then the censor will apply deep packet inspection to analyze the payload of the packets. The payload could be applications, strings or other protocol patterns. Deep packet inspection requires more effort based on the censor capabilities. The censor can do the blocking at this level by dropping any unwanted packets. Also, the censor can manipulate the traffic to act as a man in the middle to break the normal flow of the traffic. For example, the Great Firewall of China places itself between two SSL end points and send reset packets (RST) to both directions. This action will prevent both sides from making any secure communication, and therefore the connection will be closed.

Attacking techniques: based on the censorship policy, the censor can monitor the traffic flows inside its network. This is an easy task for the censor to watch all the traffic if it has the needed technical capabilities.

2.2.2 Infrastructure

The censor's goal is to block any resources that facilitate censorship resistance or help the users to publish censored contents. Information that is publicly available about infrastructures such as routers, servers or hosts can be helpful for the censor to identify them. In case of secret networks, the censor can play a role of an honest user or resource to collect information and then block it. The Great Firewall of China did that by taking the publicly known information of Tor's relays and then block the access to them all. Moreover, the Great Firewall of China started to collect information about the unpublished relays (bridges) to block them as well.

In addition to that, Internet service providers can play a significant role to map Internet identities to real-world information. Therefore, censors can use any legal pressure to get this information and start the manhunt. The infrastructure could be inside the censor's sphere of influence. In that case, the censor will use its legal pressure to block them. In case that the infrastructure is outside the censor's sphere of influence, it can start attacking them if the legal methods did not work.

Attacking techniques: All the components of the censor's infrastructure network can be censored and monitored. Servers, routers, databases and even hosts could be subject to the censorship to make sure that its clear of any violated contents. The censor could apply other social or technical means to reach its goal of censoring other contents outside its infrastructure.

2.2.3 Clients

The censor can use software or malware installed on the user's machine to monitor any activities. This action could be publicly announced by the authority like the case of Green Dam Youth Escort used by the Chinese authorities in 2009 [6]. This act will give the censor the ability to start the filtering at the client level and report back to the censor.

Attacking techniques: Tor Browser Bundle is an example of a client software that helps the user to bypass censorship to view or publish contents. The censor can target the software distribution channels or the integrity of that software. All those techniques used by the censor to stop or monitor any communications between the client and the censorship resistance network.

2.2.4 People

Individuals can be targeted by the censor to reduce the publication of censored contents. Authorities can use legal pressure to stop them like the case of Julian Assange and U.S. government. The authorities will have less control over such public figures if they are outside the censor's sphere of influence.

Attacking techniques: The censor can target people such as publishers or viewers of censored information, designers of censorship resistance systems or volunteers that aim to facilitate the publication of censored contents. Besides the technical work, the censor will use other legal influence to limit their activities.

2.2.5 Network Based Views

Networking depends on correct and precise information about the state and topology of the network. The censor can use this fact to corrupt the network view and prevent the client from getting the access to censored contents or anti-censorship network. The censor can find a way to learn about censorship resistance networks whether it is publicly known or hidden from the public. Then, the censor will start to forward subsequent requests to censored and trusted network. This work can be done using many techniques like DNS poisoning and routing table changes.

Attacking techniques: The censor can target the availability of censored contents by manipulating the network information. The network communications require correct naming and routing information to reach the destination otherwise it will fail.

2.2.6 Censorship Resistance System

Censors can use the same techniques used by regular users to discover information about censorship resistance system. Therefore, the censorship resistance systems must make it harder for the censor to discover any valuable information. However, sometimes censors could find tricks and patterns that identify censorship resistance system. Short lifetimes of SSL certificates used by Tor routers is one example that can be used to differentiate from the long lifetime normal certificates.

Attacking techniques: The censor has a goal to make it difficult for the users to start connecting to the censorship resistance systems. The censor will start to attack all the important entities of the censorship resistance system. In this way, the censor will attack the availability of those systems. For example, China blocked any connection to any entry relay of Tor network to stop users from accessing Tor network.

2.3 Censor Limitations

Like any technology, many obstacles could limit the work of censorship. In this section, we discuss the most difficulties that face any censorship system.

2.3.1 Technical Limitations

Its a costly task for the censor to do a real-time filtering on each packet. If a filtering router took an extra two microseconds to process every packet, it would halve the overall throughput [7]. Some censors break this task into two stages to reduce the cost. The first stage is to decide if the vast majority of the packets are uninteresting. The second stage is to process a small set of interesting packets.

2.3.2 Sphere of Influence Limitations

The censor will face many obstacles to control something that is beyond its borders. Legal pressure or technical attacks will not always work to limit the access to censored contents. The censor can always block any user's requests to censored contents. On the other hand, the users will always find a way to get access to such contents.

2.3.3 False Positives

We discuss the false positive action from the censor's point of view. False positives happened when a legitimate traffic or request is blocked. The censor will try to keep this as minimum as possible to avoid collateral damage [8]. Censorship

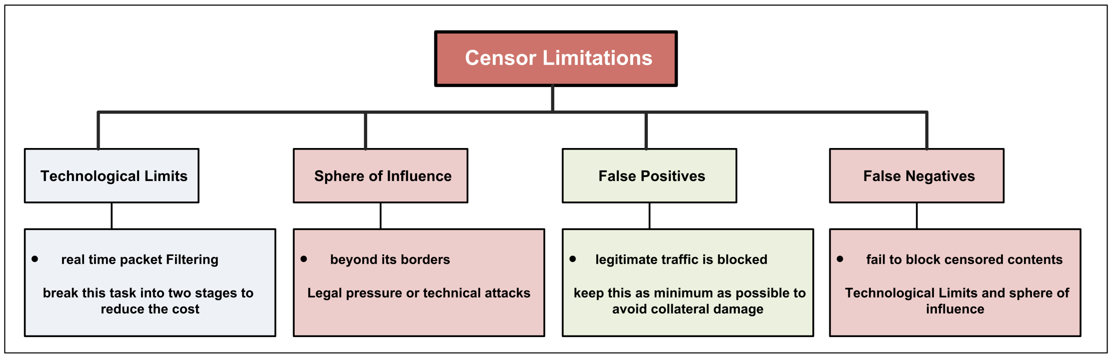

Censor Limitations
Technological Limits
Sphere of Influence
False Positives
False Negatives
• real time packet Filtering
break this task into two stages to reduce the cost
• beyond its borders
Legal pressure or technical attacks
• legitimate traffic is blocked
keep this as minimum as possible to avoid collateral damage
• fail to block censored contents
Technological Limits and sphere of influence

Figure 2.4: Summary of censor limitations

resistance systems will seek to leverage the collateral damage to develop systems that can bypass censors. SkypeMorph [9] is an example related to censorship resistance technique that uses Skype video calls as a cover to bypass censors. The censor can block all the Skype calls to prevent this technique from working which will affect other legitimate users.

2.3.4 False Negatives

Also, we discuss the false negatives action form the censor's point of view. False negatives mean that the censor is failing to block censored contents. This action will lead to information leakages and publication of censored contents. Technical limits and sphere of influence are the main reasons for false negatives to occur.

2.4 Tor and Censorship : Case Study

Tor [10] is an anonymous communication network that helps its users to use the Internet without disclosure of their identity. The user will use three Tor routers called "relays" to build a circuit to surf the Internet through it. Each relay does not have any information about the user ”the originator” except the first relay. Tor relays in one circuit does not have any information about the destination except the last relay ”Exit node”. The packet in Tor network is encrypted with three layers ”Onion packet”. This three layer of encryption will allow each relay to see only what is allowed to see.

Tor is considered as an enemy to censorship because users around the world use it to access censored contents. The war between Tor and censorship started in 2006 when some Internet service providers in Thailand blocked the access to Tor website. They used DNS filtering to redirect Tor users in Thailand to another web page. Tor network was working, but users could not get Tor software form the website. In the same year, Websense [11] and SmartFilter [12] came up with filtering technique to block Tor connections. To understand that, Tor was using two protocols HTTP to fetch directory information about Tor relays, and TLS to do the encryption part of Tor. Websense could identify the Tor HTTP directory requests then blocked it, and users could not get any information about the Tor relays. The censorship until 2009 was using filtering based on port number or other information about Tor network like Websense case.

2.4.1 Tor and The Great Firewall of China

One of the strongest opponents of Tor is the Great Firewall of China ”GFW”. At the time of writing this paper, GFW manages to block the access to all the Tor

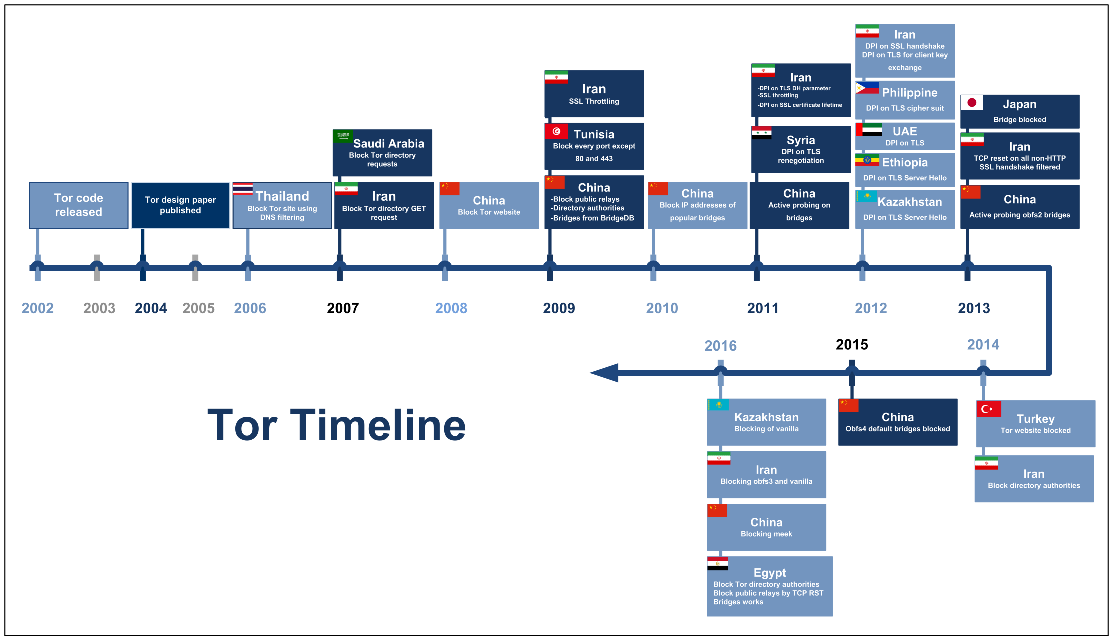


Figure 2.5: Timeline of Tor

relays on the directory authorities. In addition to that, GFW made some active probing attacks to block Tor's bridges, and it succeeded to block some of the pluggable transports bridges. First, the blocking was implemented merely by simple IP address blacklisting with the addition to filter the headers of HTTP packets. The blocking attempts at the begging were straightforward and inflexible. Currently, the blocking appears to be much more flexible and sophisticated. The GFW was able to block bridges dynamically without simple enumeration of the IP addresses.

The first step to start using Tor will require the user to connect to the directory authorities. This step is important to download all the public information of the Tor relays. After that, all public relays will be downloaded by the client. Then, the client can start creating the Tor circuits. The team of Tor project noticed that seven of all eight directory authorities were blocked on the IP layer. To do that, the GFW will let the TCP SYN pass, but it will drop the SYN/ACK sent by the bridge to the client. The same happened when a client tried to connect to a blocked bridge. However, clients in China are still able to connect to different TCP ports as well as ping the bridge. The reason for that is the GFW is blocking relays and bridges based on IP:port tuples instead of only IP addresses to minimize the collateral damage. By implementing some testing, Tor project team found that bridges are blocked for approximately 12 hours [13]. Based on the tests done by Tor team, they concluded that GFW was doing the Tor fingerprinting on the traffic going from inside China to the outside world, but not in the domestic traffic.

2.5 Internet Filtering in Saudi Arabia

The law in Saudi Arabia is based on (Sharia) or Islamic law. Therefore, some activities are prohibited by this law such as pornography, everything related to drinking alcohol, Drugs, etc. Accordingly, the use of the Internet to access such content is prohibited.

In this section, we discuss the Internet infrastructure in Saudi Arabia since the beginning. Then, we mention the content filtering regulations and the organizations responsible for it.

2.5.1 Internet Infrastructure in Saudi Arabia

In 1997, the government ordered the King Abdulaziz City for Science and Technology (KACST) [14] to provide the Internet to the residents of Saudi Arabia. KACST was responsible for building the needed infrastructure to connect to the international Internet and apply content filtering to provide suitable Internet. KACST was the only gate to the Internet, and all ISPs were connected to it. The content filtering was done by KACST, and the ISPs are not involved in this process. Figure 2.6 illustrate the Internet infrastructure in Saudi Arabia from 1997 until 2004.

In 2005, after the expansion of the Internet, it became difficult for one place to do all the work. New decisions have been issued which allow many organizations and companies to connect directly to the international Internet according to the applied regulations. The responsibility was assigned to Communications and Information Technology Commission (CITC) [15] as a supervisory organization. The content filtering task was assigned to each ISP independently based on the list provided by CITC. Figure 2.7 illustrate the Internet infrastructure in Saudi Arabia from 2005 until now.

2.5.2 Content Filtering

As we mentioned previously, the content filtering task was assigned first to KACST. As a single point of access to the Internet, KACST was applying content filtering

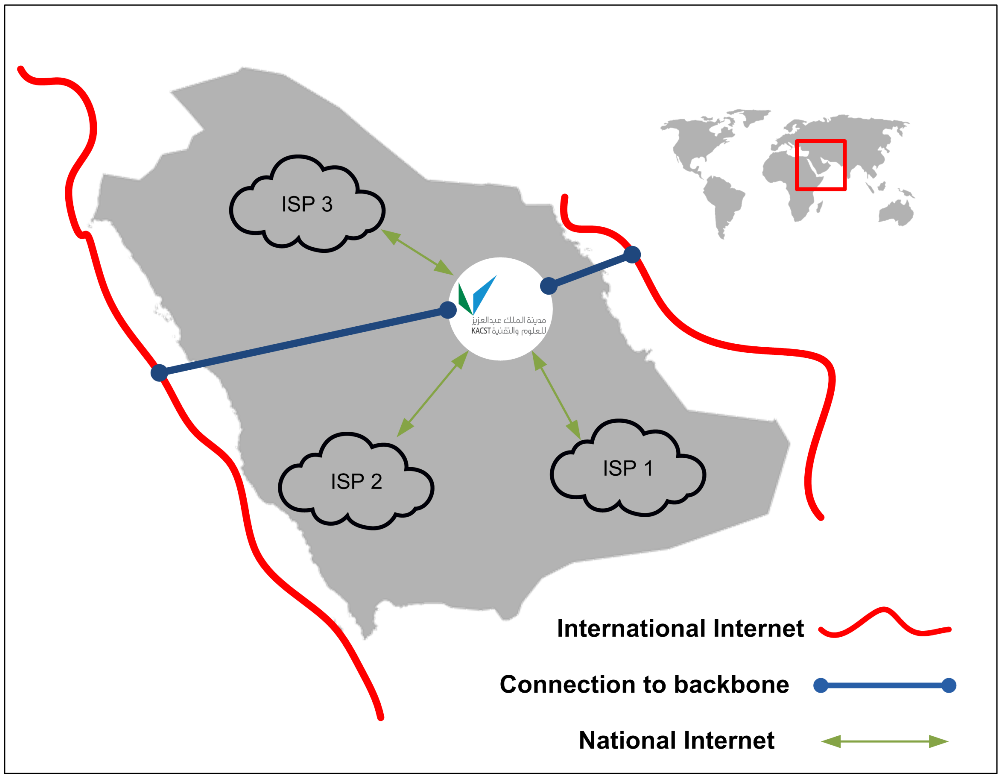


Figure 2.6: Internet infrastructure in Saudi Arabia from 1997 until 2004

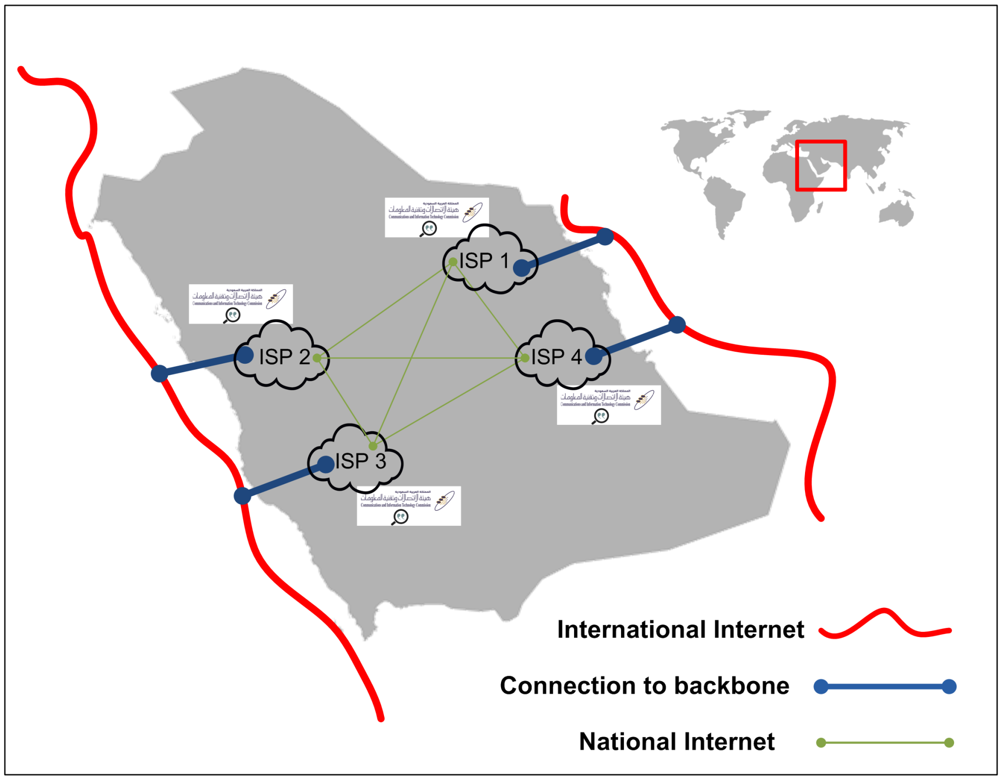


Figure 2.7: Internet infrastructure in Saudi Arabia in 2005

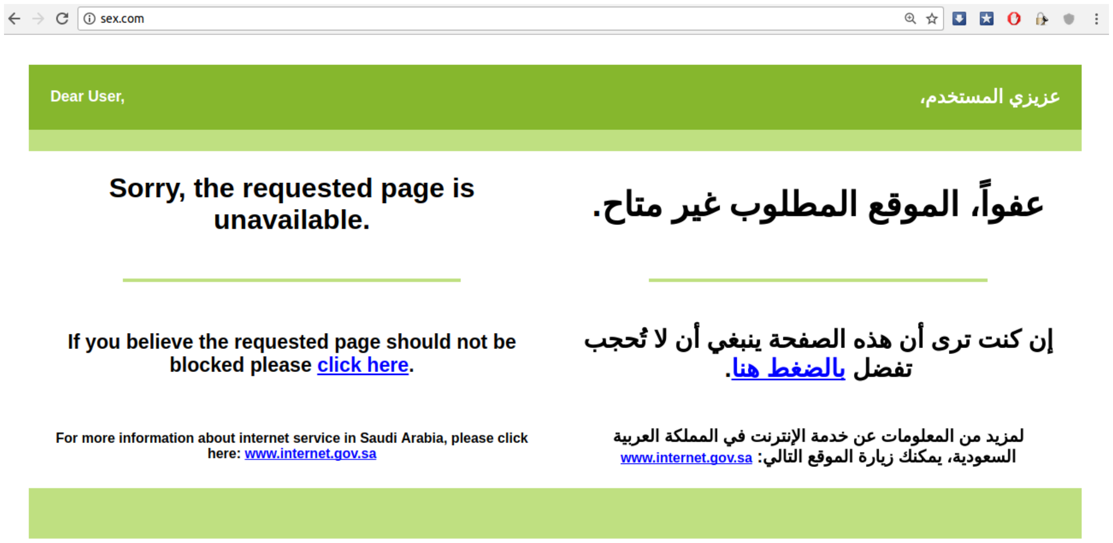
based on two type of websites lists commercial list and local list. The commercial list is generated by applications used by KACST which will classify websites based on the content. This task could be done on the web proxy level of KACST to ensure that each requested website is allowed. The Internet users can participate to provide some websites that are not included in the commercial list. The user or any organization can submit a request to block a website, the request will be evaluated, and then an action will be made. If the request accepted, the website will be added to the local list and then blocked.

Nowadays, CITC assigning the content filtering task to each ISP on their web proxy side. Each ISP is required to perform this task in a suitable way using the lists provided by CITC. CITC will process any requests for blocking a website, and if it is accepted the local list will be updated and distributed to the ISPs. The commercial list will vary from ISP to another based on the tool used on each ISP. Basically, there are certain categories that will be blocked such as pornography, gambling, drugs etc.

To request a website blocking, the user should submit a form that has detailed information about the reasons for such a request. Each request is processed independently and seriously, and if the reasons are accepted, then the website will be blocked. CITC will receive such requests, and they will evaluate it independently. In case that the website content is categorized as pornography, gambling or drugs,

Local list of banned sites	
Content type	**Percentage**
Pornographic Content	92.80%
Websites that bypass filtration systems	4.43%
Gambling, drugs etc.	2.77%

Table 2.1: The percentage of banned sites types in local list



sex.com

Dear User,

عزيزي المستخدم،

Sorry, the requested page is unavailable.

عفواً، الموقع المطلوب غير متاح.

If you believe the requested page should not be blocked please click here.

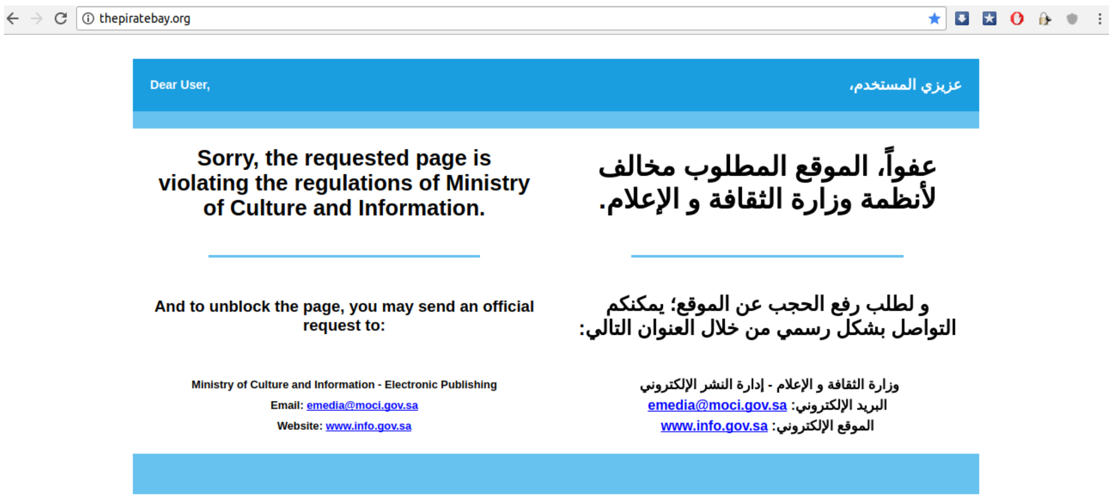
إن كنت ترى أن هذه الصفحة ينبغي أن لا تُحجب تفضل بالضغط هنا.

For more information about internet service in Saudi Arabia, please click here: www.internet.gov.sa

لمزيد من المعلومات عن خدمة الإنترنت في المملكة العربية السعودية، يمكنك زيارة الموقع التالي: www.internet.gov.sa

Figure 2.8: The blocked page when trying to open pornographic site

then they will add it to the banned local list immediately. In case that the request is related to electronic publishing and intellectual property rights, then the request will be forwarded to the ministry of culture and media for further investigation. In case that the request is related to personal affairs, then the request will be forwarded to the ministry of interior for more investigation. Table 2.1 shows the percentage the website content type in the local list. Figures 2.8 and 2.9 show the displayed page when the user is trying to access a banned site.



thepiratebay.org

Dear User,

عزيزي المستخدم،

Sorry, the requested page is violating the regulations of Ministry of Culture and Information.

عفواً، الموقع المطلوب مخالف لأنظمة وزارة الثقافة و الإعلام.

And to unblock the page, you may send an official request to:

و لطلب رفع الحجب عن الموقع؛ يمكنكم التواصل بشكل رسمي من خلال العنوان التالي:

Ministry of Culture and Information - Electronic Publishing
Email: emedia@moci.gov.sa
Website: www.info.gov.sa

وزارة الثقافة و الإعلام - إدارة النشر الإلكتروني
البريد الإلكتروني: emedia@moci.gov.sa
الموقع الإلكتروني: www.info.gov.sa

Figure 2.9: The blocked page when trying to open the famous torrent site thepiratebay

CHAPTER 3

CENSORSHIP RESISTANCE SYSTEMS

Internet users around the world started to act against censorship systems. They began to develop the systems that aim to bypass the censorship and maintain the privacy. Those types of systems are called censorship resistance systems (CRS). The studies related to this topic are under a security field called privacy enhancing technology (PET).

In this chapter, we cover the classifications of CRS in details which will help us understand the strategies and methods that are used to bypass the censorship. Next, we discuss some examples of currently active systems that are used nowadays as CRS. We explain in details each one of the selected systems to understand the applied techniques to maintain the privacy.

3.1 CRS Classification

In this section, we mention the strategies and classes of any CRS to bypass the censorship systems. The first part of this section is about the resistance strategies or CORDON. Elahi and Goldberg al. [5] presented a taxonomy of the techniques used by the censorship resistance systems.

The second part of this section discuss the classes of CRS. So, to design a CRS, the developer could apply more than one strategy to ensure the stability and the availability of the CRS. Moreover, at its core, it could belong to one or more classes.

3.1.1 Resistance Strategies CORDON

Elahi and Goldberg al. [5] provided a taxonomy of censorship resistance strategies and techniques. They categorized CRS strategies into six types which are also known as the CORDON taxonomy. In general, censorship resistance systems could fall into one or more of the CORDON taxonomy. In this section, we mention the six types of censorship resistance classes which are:

1. Collateral Damage,
2. Outside Scope of Influence,
3. Rate Limiting,
4. Decoupled Communication,

5. Overwhelm,

6. No Target,

Along with the explanation of each type, we discuss the currently available systems based on the strategies used to bypass the censorship.

Collateral Damage

It is one of the most powerful strategies used, and it is based on causing collateral damage to the censor's capital as a result of its activities. The damage can have many effects in many ways such as economic expansion, social peace, or political power. The idea is to precisely judge the censor's limitations in the technical and utility parts. To get this as an advantage, The CRS will produce a situation where collateral damage is unavoidable. For example, SkypeMorph [9] is depending heavily on Skype network, and one way to entirely stop this kind of systems is by blocking any Skype communication. This act by the censorship will affect other legitimate users that are in need of this service. The power of this types of censorship resistance techniques is to assume that the censor is unable to distinguish between censored and allowed content efficiently.

There are popular techniques fall into this class where the CRS will hide the requests among innocents looking requests. Format-Transforming Encryption (FTE) [16] is one example of such technique where the traffic is transformed into innocent looking HTTP requests. Another popular technique is to use existing and hard to block applications or services. Blocking such services will impact the censor's capital. For example, email services such as Gmail, and cloud computing platforms like Amazon Web Services. Meek [17] is one example where it uses the cloud public services as domain fronting to bypass the censorship.

Outside Scope of Influence

A major weakness that faces the censorship systems is the limitation in their scope of influence. Resistance systems will try to leverage the cloud services and other entities that are running outside the censor's political borders. Tor [10] and Tor bridges [18] are based on this class, and the Tor team is placing the relays in different countries across the world.

Rate Limiting

Many resistance systems are facing a challenge regarding how to distinguish the legitimate user from a censor that act as a user. This obstacle makes the information hiding from the censor a challenging issue. The censor may start to collect more information to launch an attack to block the CRS like the case of Tor and China [19]. This strategy is applied by Tor to slow the automated scanning for bridges by using puzzles, such as CAPTCHA and computational tasks.

Decoupled Communication

Sometimes, there is a blind spot in the censorship where the censor is unable to detect and block the censored traffic efficiently. This blind spot will classify the communication as innocent or unrelated to the usual censorship targets. The circumventor will try to leverage this blind spot by changing the signature of the bidirectional communication to beat the censor. CensorSpoofer [20] is one example of this class, where it decouples upstream and downstream data channels. The client will send the data to a CensorSpoofer proxy over a low bandwidth covert channel like email. Then, the proxy will send the data back over a UDP channel. During this process, CensorSpoofer proxy will spoof its source address, so the packets will appear as if it is originated from some other dummy host. As a result, there is no IP address for the censor to block because the proxy's true address will never appear on the wire.

Overwhelm

It is a fact that the censor has a limited time and resources in use. Therefore, the censor will have to decide which type of the traffic is allowed or blocked. This limitation could be an advantage for the CRS to use it as a strategy. The censor will not be able to deal with every single aspect, and even if it could block some, others will be allowed. An example of that is the deployment of a massive number of proxies around the world. Therefore, the censor will face a hard time to discover all of them. This will cost the censor much time and effort, and will add more effectiveness in the resistance part. Moreover, the blocking will become a serious issue when those proxies are honest nodes on the network. Blocking such resources could make a large portion of the Internet unreachable.

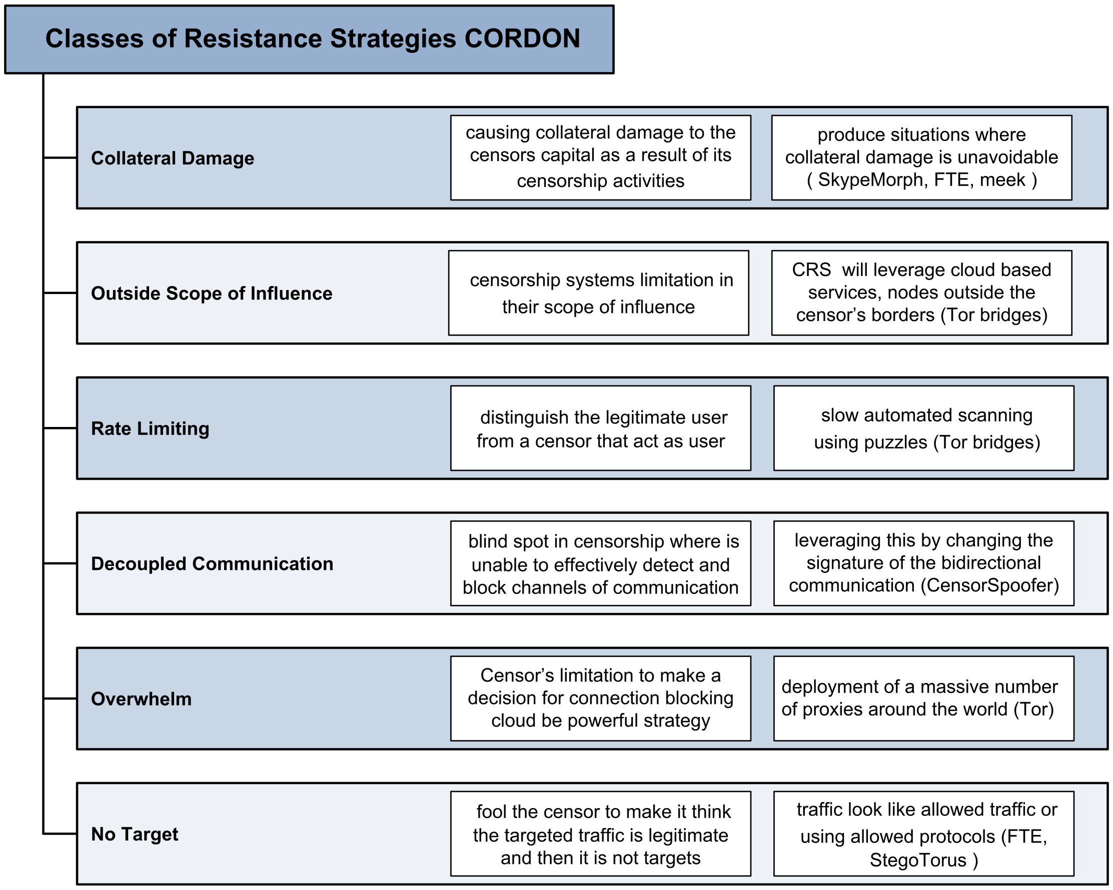

Classes of Resistance Strategies CORDON
Collateral Damage
causing collateral damage to the censors capital as a result of its censorship activities
produce situations where collateral damage is unavoidable (SkypeMorph, FTE, meek)
Outside Scope of Influence
censorship systems limitation in their scope of influence
CRS will leverage cloud based services, nodes outside the censor's borders (Tor bridges)
Rate Limiting
distinguish the legitimate user from a censor that act as user
slow automated scanning using puzzles (Tor bridges)
Decoupled Communication
blind spot in censorship where is unable to effectively detect and block channels of communication
leveraging this by changing the signature of the bidirectional communication (CensorSpoofer)
Overwhelm
Censor's limitation to make a decision for connection blocking cloud be powerful strategy
deployment of a massive number of proxies around the world (Tor)
No Target
fool the censor to make it think the targeted traffic is legitimate and then it is not targets
traffic look like allowed traffic or using allowed protocols (FTE, StegoTorus)

Figure 3.1: Summary of CORDON taxonomy

No Target

The idea of this class is to try to fool the censor to make it classify the traffic as a legitimate traffic. Resistance system will try to make the traffic look like allowed traffic or using allowed protocols. FTE [21] is a good example, it changes the shape of the traffic to HTTP traffic using port 8080. StegoTorus [7] is another example that implements this strategy by mimicking the HTTP protocol based on previously collected HTTP traffic data. Moreover, it will apply various steganographic techniques.

3.1.2 CRS Classes

Censorship resistance systems aim to obfuscate and mimic the application-layer protocols. Regarding the obfuscation technique, there are two main classes used by CRS which are traffic obfuscation and destination obfuscation [51]. First, the traffic obfuscation will aim to change the shape of the targeted traffic to another form to bypass the censorship. Second, the destination obfuscation will aim to hide the destination point from being observed by the censor to avoid any blocking action. Under each one of those classes, there are other subclasses based on the techniques used by each one of them.

For traffic obfuscation, there are four categories which are randomization, mimicry, tunneling and programmable systems. The first three types could also be known as non-programmable systems. On the other hand, for the destination obfuscation, there are two categories which are proxy and decoy routing.

In this section, we discuss each one of them along with CRS examples of such systems, see figure 3.2 which does summarize them all.

Traffic Obfuscation

As we mentioned before, the goal of the traffic obfuscation systems is to change the traffic from suspicious traffic to an innocent looking traffic. The developers of such systems could use four ways to achieve this goal. First, they can make the traffic look as random as possible by adding another layer of encryption to remove any known patterns, and this is called Randomization. Second, the developers can make the traffic look like other traffic such as VoIP or regular HTTP traffic, and this is called Mimicry. Third, they can establish a secure channel between two points and use this channel to send the traffic through it, and this is called Tunneling. Fourth, the developers can create a system that merges some of the mentioned classes to work as one system, and this called Programmable. We discuss each one of them with the examples.

1. **Randomization**: If the system is implementing the randomization approach, then the main idea is to remove any static fingerprints within the content. Moreover, it will also remove any statistical characteristics of the connection, and simply the traffic should look like "nothing". The obfs2 and obfs3 [23] are used by Tor project to remove any indicators related to Tor traffic. The idea of those two protocols introduced after the successful attempts to block Tor traffic in China. The protocol will add another layer of encryption to hide the connections between the client and the Tor bridges. After that, there are improvements related to those two protocols which introduced ScrambleSuit [24] and obfs4 protocols [23]. The Dust system [25] offers randomization both on the content and the statistical information. Dust does that on each packet instead of doing it per-connection. The previous types of mentioned protocols will work effectively if the censorship is blocking a specific kind of traffic categorized as bad traffic which is also

known as blacklist approach.

2. **Mimicry**: The developers that will apply this approach will try to change the traffic to mimic another protocol. For example, making the traffic look like a Skype video call, and this is used by SkypeMorph [9]. Blocking any Skype calls as a way to stop this protocols will lead to another problem on the censorship side which is the collateral damage. This problem will make the censorship block even the legitimate traffic using this application. StegoTorus [7] is another system used to mimic HTTP protocol based on previously collected HTTP traffic data, and using various steganographic techniques. However, there are some studies show that mimicked protocols can be distinguished from real protocols by using protocol semantics, error conditions and the dependencies among connections [26].

3. **Tunneling**: This approach depends on the potential collateral damage caused by censorship to block popular protocols in a way to avoid packet filtering. However, these systems are tunneling their data inside the payload of real requests of the target protocols. For example, Freewave system [27], uses Skypes voice channel to encode data. Facet [28] uses the Skype video channel, where SWEET [29] is using the body of email messages. JumpBox [30] is using web browsers and live web servers as a tunneling approach. CensorSpoofer [20] also tunnels the data over existing protocols, but uses a low-capacity email channel for upstream messages and a high-capacity VoIP channel for downstream. CloudTransport [31] uses another approach

by tunneling its data using hard to block cloud storage services, like Amazon S3 instead of a particular protocol. In general, tunneling approaches will add more overhead than mimicry systems because they are limited by the low capacity of the protocols used for the tunneling.

4. **Programmable**: The programmable systems are combining the advantages of both randomization and mimicry systems. This approach will allow the system to be configured to perform either strategy. Format-Transforming Encryption [21] and Marionette [22] are the only systems implementing programmable traffic obfuscation. FTE is running with Tor as pluggable transports [32] where it enables the Tor clients to connect to Tor network even if the only traffic allowed is HTTP. Since there are several attempts by governments to block clients from connecting to Tor network, the Tor team started to implement the concept of Tor bridges. Tor bridges are special relays that are not publicly published. Therefore, it will be used to bypass the censorship then connecting to Tor network. So, even if the censorship blocking the encrypted connections, then it is possible to be able to connect to Tor network. On the other hand, Marionette comes with a primary goal which is developing a system that will not depend on a single traffic obfuscation method. Instead, it will provide the user with the capability to choose the obfuscation method that will fit their needs. The user can select the method based on the target protocols, depth of the controlled traffic features, and throughput of the network [22].

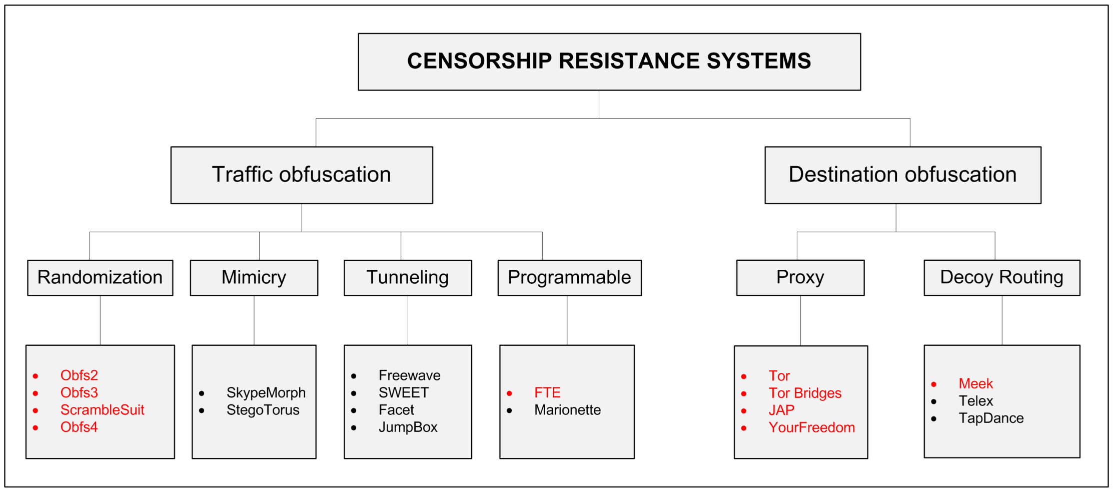


Figure 3.2: CRS Classes

Destination Obfuscation

As we mentioned at the beginning of this section, the goal of the CRS in this class is to hide the destination point from the censorship. There are two subclasses in the destination obfuscation class which are proxy and decoy routing. In the proxy approach, the CRS will use the help of other entities as a facilitator to forward the traffic instead of going directly to the destination. In the decoy routing, the goal is to fool the censor by deflecting the traffic to an innocent server.

1. **Proxy**: In this approach, other objects are included as a facilitator to forward the traffic from the CRS client to the CRS server instead of going directly to the destination. Tor and JAP are two censorship resistance systems that aim to achieve this goal to maintain the anonymity of their users. Both systems will hide the destination of the requested server for the censorship. In addition to that, both systems will hide the identity of the user from the destination server. In Tor, they apply the concept of relays and

bridges, wherein JAP they use the mixing servers. Both systems use those entities in between the client and the destination.

2. **Decoy Routing** The goal here is to deflect the traffic to an innocent server by using an intermediate agent. The CRS could use the help of some hard to block servers in the cloud such as Amazon AWS as a decoy sever. This process will start with the client connecting to this innocent server. From that, the server will forward the request to the destination or another decoy server. The censor will see a connection made to some web server that is not on the blacklist, and it will allow it. The CRS will maintain the identity of the real destination hidden from the censor. A CRS applies this method which is meek, and we will see in the next section how it work.

3.2 CRS Examples

This section is all about different types of censorship resistance systems (CRS). We cover the most known and active censorship resistance systems. We divide them into two main categories which are Tor and stand-alone systems. In Tor category, there are Tor, Tor Browser, Tor bridges and five Tor pluggable transports. In the second category we mention stand-alone systems which are JAP and YourFreedom.

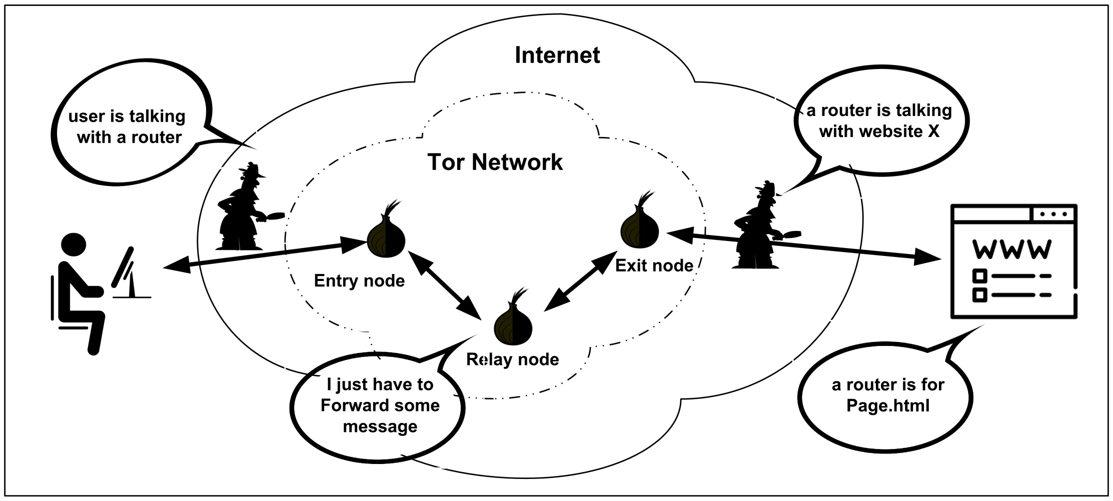

Internet
Tor Network
user is talking with a router
a router is talking with website X
Entry node
Exit node
Relay node
I just have to Forward some message
a router is for Page.html
WWW

Figure 3.3: Tor Network

3.2.1 Tor and Tor Subsystems

Tor is an anonymous communication network that helps its users to use the Internet without the disclosure of their identity [10]. The user will use three Tor routers ”relays” to build a circuit to surf the Internet through it. Each relay does not have any information about the user ”the originator” except the first relay. Tor relays in a circuit does not have any information about the destination except the last relay ”Exit node”. The packet in Tor network is encrypted with three layers ”Onion packet”. This will allow each relay to see only what is allowed to see.

Tor Browser

To use Tor, users will have to install the Tor Browser Bundle which contain all the software needed. Tor Browser is the main software in this package, and the user will use it to access the websites anonymously. While using Tor, it is recom-

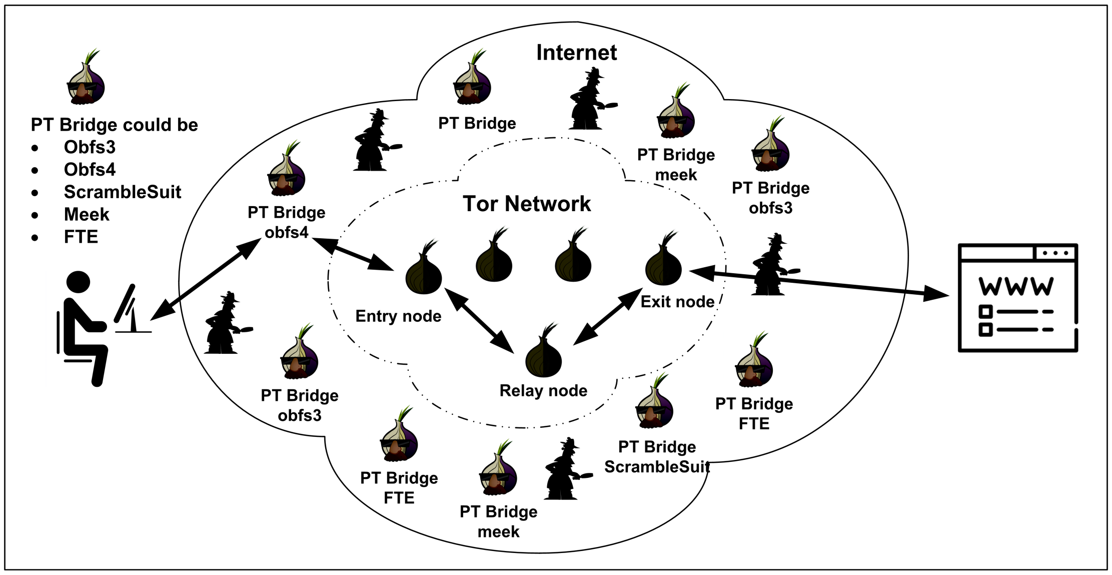


Figure 3.4: Tor and Pluggable Transport Bridges

mended by Tor project to use their browser, because they did much enhancement to support the user's privacy. The browser at its core is based on Mozilla's Extended Support Release (ESR) Firefox branch [33].

To enhance both privacy and security of this browser, Tor project did many patches with many changes in the default values of Firefox preferences. The main part of the browser is Torbutton extension which is responsible for all Tor activities. Torbutton activities including creating circuits, maintaining the communication and forwarding the traffic from the browser to Tor network. The browser is shipped with two extensions which are HTTPS-Everywhere [34] and NoScript [35]. HTTPS-Everywhere extension is a used to prevent the possibility of Tor exit node eavesdroppers. This extension will make sure that the user will request only HTTPS pages. Otherwise, it will notify the user and block it. NoScript extension is a used to allow trusted and limited websites which are selected by the user to execute JavaScript, Java, Flash and other plugins. This extension will prevent any exploitation of security vulnerabilities that are caused by such plugins.

If the public Tor network is blocked, then Tor browser can activate the pluggable transports. The browser currently supporting the following protocols obfs3, obfs4, Scramblesuit, meek, and FTE. Moreover, Tor project changed several Firefox preferences and applied many patches to the browser. The implementation phase of changing the browser has many goals to achieve. We mention some of them as following:

- **Proxy Obedience** : The goal here is to force Firefox to connect to Tor directly as a SOCKS proxy. This can be done by setting some preferences like (network.proxy.socks.X) to point to Tor application. Additionally, they also prevent proxy bypass using WebRTC by disable it at compile time and change the value of (media.peerconnection.enabled) to false. Moreover, plugins are disabled to prevent them from doing any kind of OS system calls and proxy bypassing. With each release, they perform in-depth code auditing and apply any patches.
- **State Isolation** : The state of Tor browser is separated from the state of the existing browser by using custom Firefox profile. Also, the root directory of the TBB is the home environment of the browser.
- **Avoid Writing to Disk** : The goal here is to prevent the browser from writing any records of the browsing activities to the disk. This can be done

by changing the preferences to enable Firefox private browsing, and disable any media cache.

- **Application Data Isolation** : The goal here is to prevent the browser from saving any data outside the bundle directory. This will help to prevent the users from leaving any pieces of evidence after using the browser, especially when saving the files to the local disk. Data isolation is ensured by setting the preferences of browser downloads to point to the bundle directory.

- **Browser Fingerprinting** : In Tor browser, there are some defenses applied to defeat browser fingerprinting. First, all plugins in Tor browser are disabled to prevent them from producing unique fingerprints that could identify the browser. Flash is the only one available with limited features. The rest are completely blocked. Second, Tor browser uses a predefined set of fonts shipped across platforms which will be used by the websites without using the system fonts. Third, automatically resizing the browser windows to prevent any leakage of information about the monitor size of the user. This is done by Firefox patch and preferences such as (privacy.resistFingerprinting). Fourth, they provide two Firefox patches to take care of the keyboard layout fingerprinting by providing fake properties. Fifth, they provide all websites with identical user agent and HTTP headers. This can be done by changing the preferences of the user agent. Sixth, to defeat locale Fingerprinting for non-English users, they fix all locale to windows-1252. There are around 24 browser fingerprinting defenses [33] applied in Tor browser. They are

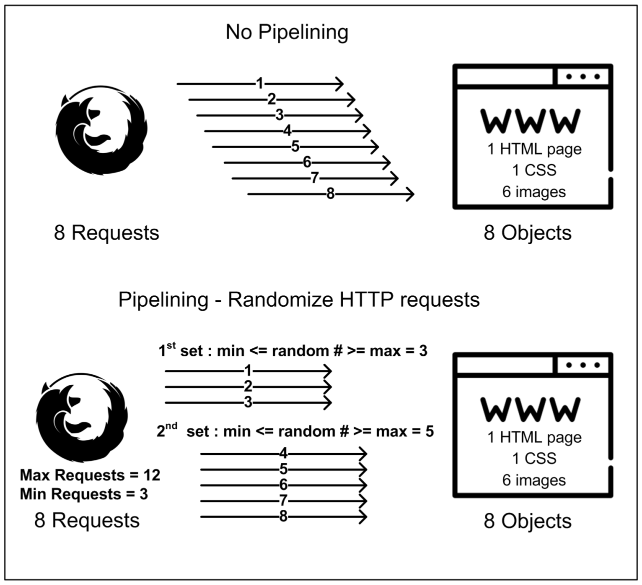


Figure 3.5: HTTP pipelining and randomized requests

applied either by changing Firefox preferences or by applying patches.

- **Long Term Unlinkability** : To achieve that, an option for a new identity is available in Torbutton menu. The browser will make sure that all data of the current state cleared before issuing the new identity.

- **Website Traffic Fingerprinting Defenses** : Currently, Tor project applies HTTP pipelining [36] as website traffic fingerprinting defenses. It is deployed as Firefox patch to enable HTTP pipelining, randomize the size of pipeline and the order of requests as in figure 3.5. Part of it could be achieved by configuring Firefox preferences (network.http.pipelining.X) to true, while the randomization is achieved by the patch. There are some WF defense considered but not deployed which are HTTPOS [37], Adaptive Padding [38] and BUFLO [39].

Tor Bridges

Tor relays are publicly published to all users through servers called directory authorities. There are ten directory authorities that will help Tor clients to learn the list of all relays to use Tor network [18]. Countries like China and Iran started to block directory authorities which prevented the users from accessing Tor network. To solve this issue, Tor introduced ”Bridges” as relays that are not listed in the main directory authorities. In case the censor starts to block all the relays in Tor network, it probably cannot block all the bridges since there is no complete public list of them. Users can use Tor directly, and they can switch to use bridges once the Tor is blocked. Tor Browser Bundle which is the client software used to connect to Tor network can perform this task quickly [18]. Tor users also can ask for the bridges using Tor bridges database [40]. There are several bridges distribution strategies used by Tor to deliver the bridges IP address. Currently, there are five strategies used by Tor which are Time-based, Location-based, Combined time and location-based, Mailing list, Emailing Tor project and Social network.

1. **Time-based**: If Tor has one hundred good bridges, then only ten of them will be available in a given hour. Next hour, it will change to next group of bridges and so on. This will make a hundred bridges hidden from the censor at a given time.

2. **Location-based** : Requests for bridges from a specific IP address will get a set of a specific bridge based on the location. This will limit some countries

from learning all the bridges unless they have censors around the world.

3. **Combined time and location based**: This method will combine the above two strategies to make it harder for the censor to learn about the bridges. At one time slot based on the location, the bridges distribution will be fixed.

4. **Mailing list**: By starting a mailing list and allowing the people to sign up as receivers. Censors could be part of that list and start to block the bridges.

5. **Emailing Tor project**: The user will send an email to Tor project asking for bridges. The user should provide legitimate Gmail or Yahoo account to receive the bridge address. Tor project will reply once only, and leave the automated account creation problem to Google and Yahoo to prevent.

6. **Social network**: Tor can send some of the bridges to trusted users in a specific country through social networks. Users then will distribute them to others in the same manner.

The arms race continued between censors and Tor, and censors could block some bridges. On the other hand, Tor will have to add more solutions to circumvent censorship, and this is the beginning of the pluggable transports.

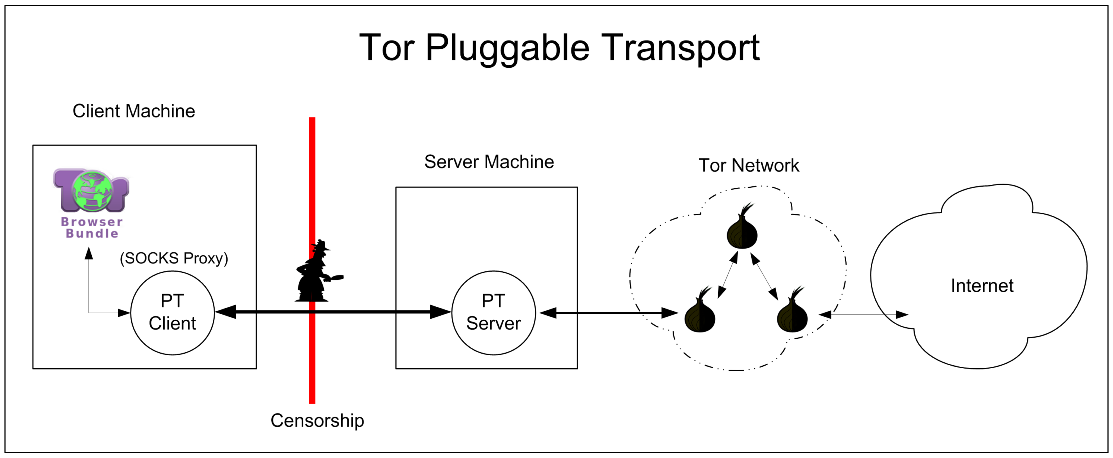


Figure 3.6: Tor Pluggable Transport

Tor Pluggable Transport

The main idea of pluggable transports (PT) [32] is to change all Tor traffic between the client and the bridge before the first relay. This action will make the traffic look like something else other than Tor. The reason behind pluggable transports is that censors started to use Deep Packet Inspection "DPI" to analyze Tor traffic flows. The censor will see innocent-looking traffic to unlisted Tor relay "bridge", and it will allow it. Any Tor user can contribute to help other users by configuring the machine to act as pluggable transport bridge server. The IP address of the PT-bridge servers will not be publicly listed on the directory authorities. Clients will use TBB as a pluggable transport client, and start to connect to Tor network through the PT-bridges as in figure 3.6. Currently, Tor Browser Bundle supports five types of pluggable transports [32] obfs3, obfs4, ScrambleSuit, meek and FTE.

Each one of the five systems is considered as a CRS. Some of them have its own implementation as stand-alone system such as FTE. Others like obfs3 and

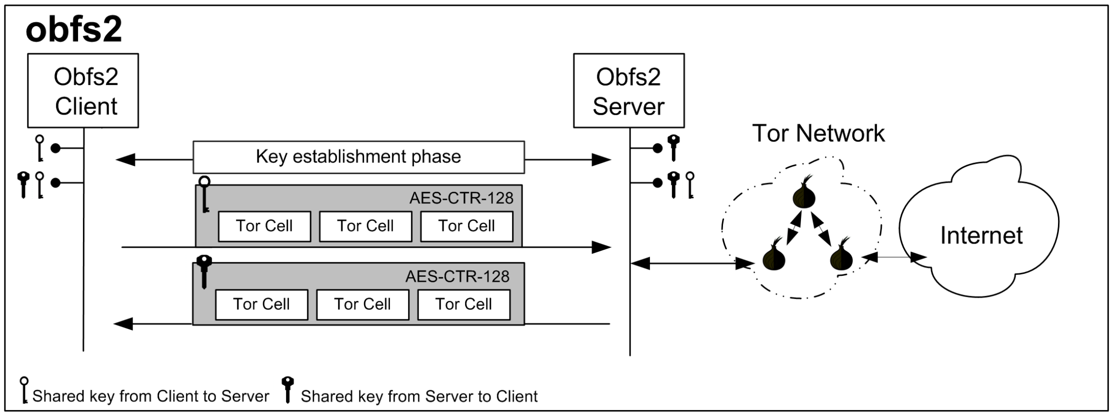


Figure 3.7: Tor Pluggable Transport obfs2

obfs4 are designed and implemented by Tor project team. We explain the idea behind each one of them.

Tor Pluggable Transport - obfsX

The first protocol used to obfuscate the traffic of Tor is the twobfuscator (obfs2) [41]. The idea started after the blocking of eight directory authorities in some countries. The obfs2 is the first pluggable transport implemented as a proof-of-concept obfuscation protocol. The protocol was simple and just protect against fingerprintable TLS content patterns. It does not provide authentication, data integrity or hide data lengths. It adds another layer of encryption using AES-CTR-128 on top of Tor packet. The problem with obfs2 is that they did not perform the key exchange by using asymmetric cryptography. Because of that, any passive attacker in the middle can observe the keys and decrypt the traffic to see the actual Tor communication.

As an improvement to obfs2, obfs3 [42] was implemented. To solve the obfs2 problem, obfs3 negotiate the keys using public key cryptography. Specifically, they used an anonymous Diffie Hellman key exchange. They used custom Diffie Hellman protocol proposed by Ian Goldberg [43]. The reason for using custom Diffie Hellman protocol is that the traditional Diffie Hellman is not suitable for their situation. This means that a passive eavesdropper will not be able to find the key by just monitoring the traffic. As in obfs2, obfs3 add another layer of encryption using AES-CTR-128 on top of Tor packet. There are two problems facing obfs3, authentication in key exchange phase and active probing attacks. For the first problem, the user and the bridge did not have any shared keys or public keys for signatures to authenticate each other before making the connection. So, an attacker can play a role of man in the middle that could fool both sides to make the traffic go through him. The second problem that obfs3 facing is active probing attacks against the bridges. The problem starts when a user connects to a bridge somewhere on the Internet, and the censor notices that connection. The censor will investigate this connection by making a connection to the bridge directly. If the bridge accepts the connection, then the censor will just block that bridge which is exactly what the Great Firewall of China did [19]. ScrambleSuit was introduced later to solve some of the obfs3 challenges as we mention later in this section.

To solve the challenges on obfs2, obfs3 and ScrambleSuit, the latest PT introduced which is obfs4 [44]. As in obfs2 and obfs3, obfs4 mainly goal is to provide an

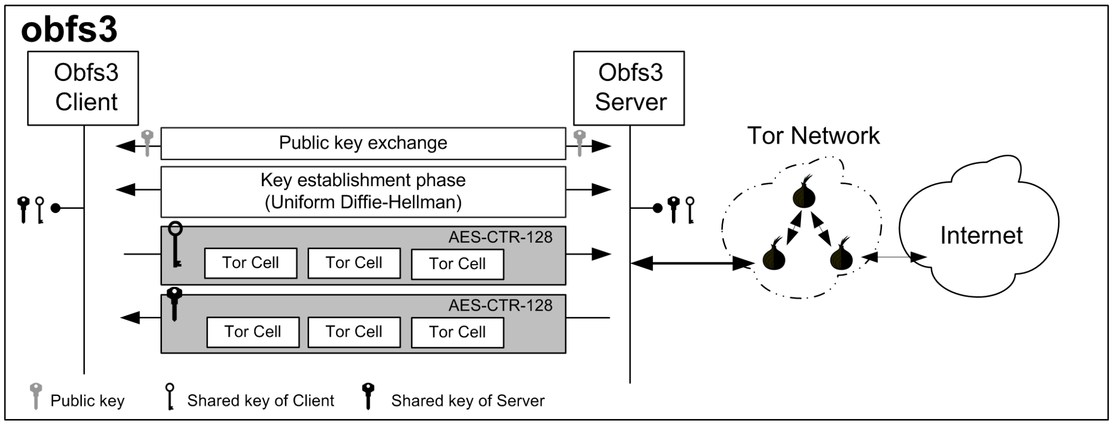


Figure 3.8: Tor Pluggable Transport obfs3

obfuscation layer on top of an authenticated protocol such as TLS or SSH. Moreover, obfs4 provide authentication, data integrity, and protection against several attacks such as passive Deep Packet Inspection and active probing. ScrambleSuit and obfs3 use custom Diffie Hellman protocol for key exchange which has several issues in performance and authentication. Instead, obfs4 uses high-speed elliptic curve cryptography specifically, Curve25519 public keys and Elligator 2 mapping [45] for transmitting data. In Deep Packet Inspection and active probing attacks, the attacker should get two type of information to proceed which are the node ID and the public key of the obfs4 server. In case the attacker wants to pretend to be an obfs4 server, he needs to have the node ID, the public, and private keys.

Tor Pluggable Transport - ScrambleSuit

To defend against active probing and some fingerprinting techniques, ScrambleSuit [24] was introduced as a thin protocol layer on top of TCP. The goal of ScrambleSuit is to obfuscate the application data transported. They use techniques such as morphing and a secret exchanged out-of-band [24]. ScrambleSuit aim to hide

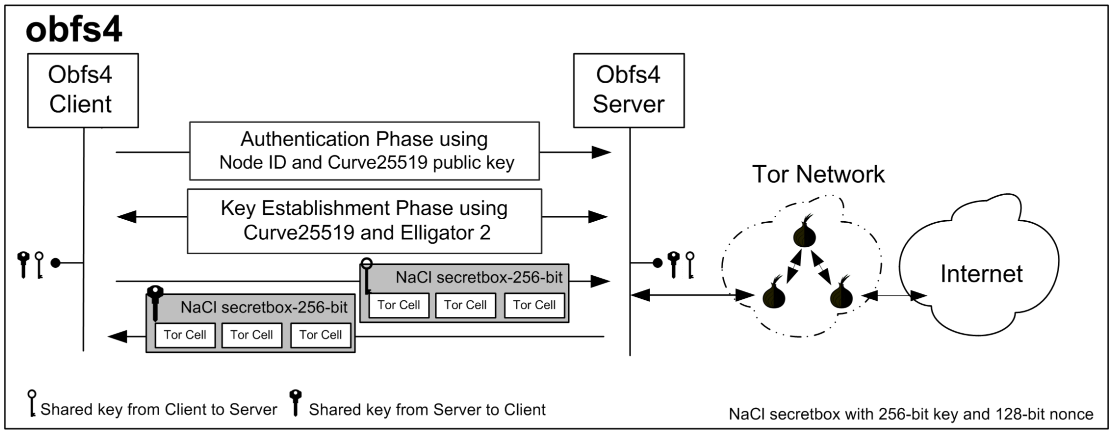

obfs4
Obfs4 Client
Obfs4 Server
Authentication Phase using Node ID and Curve25519 public key
Key Establishment Phase using Curve25519 and Elligator 2
NaCl secretbox-256-bit
Tor Cell
Tor Cell
Tor Cell
NaCl secretbox-256-bit
Tor Cell
Tor Cell
Tor Cell
Tor Network
Internet
Shared key from Client to Server
Shared key from Server to Client
NaCl secretbox with 256-bit key and 128-bit nonce

Figure 3.9: Tor Pluggable Transport obfs4

several features of Tor's communication such as payload, packet length distribution and inter-arrival times. ScrambleSuit will encrypt the traffic to conceal Tor's payload and its fingerprints such as Tor's TLS cipher list. Also, ScrambleSuit will seek to hide the well-known characteristic of Tor which the 586-byte packet size by using a randomly chosen distribution of packet sizes. ScrambleSuit will change the inter-arrival times by applying small and random sleep intervals before sending the data on the wire.

ScrambleSuit server will not accept any connection unless the client presents a secret value that is exchanged out-of-band. After the client prove that he has the secret value, the client starts to authenticate himself. Authentication phase in ScrambleSuit can be done using two methods which are session ticket or custom Diffie Hellman handshake. If the client does not has a session ticket, then a custom Diffie Hellman handshake is conducted with new session ticket issued for future connections. Client and server will agree on a 192-byte random number.

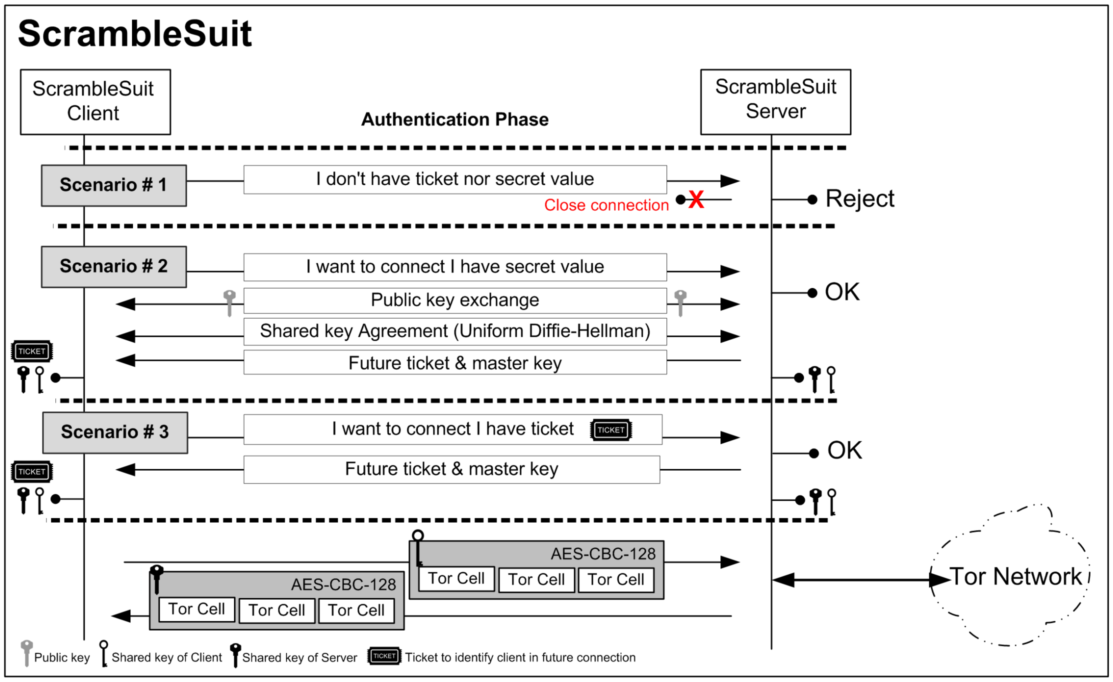


Figure 3.10: Tor Pluggable Transport ScrambleSuit

This random number will be hashed using SHA256 to get the 256-bit master key. The master key then used to derive the session keys [46]. The encryption of the traffic is done by applying AES-CBC with a 128-bit key.

ScrambleSuit and obfs3 share the concept of using custom Diffie Hellman protocol for key exchange. Moreover, the use of secret value will help to prevent active probing attacks that are done by the censors. The successor of ScrambleSuit is obfs4, which solve some issues regarding the performance of custom Diffie Hellman and the authentication of the server.

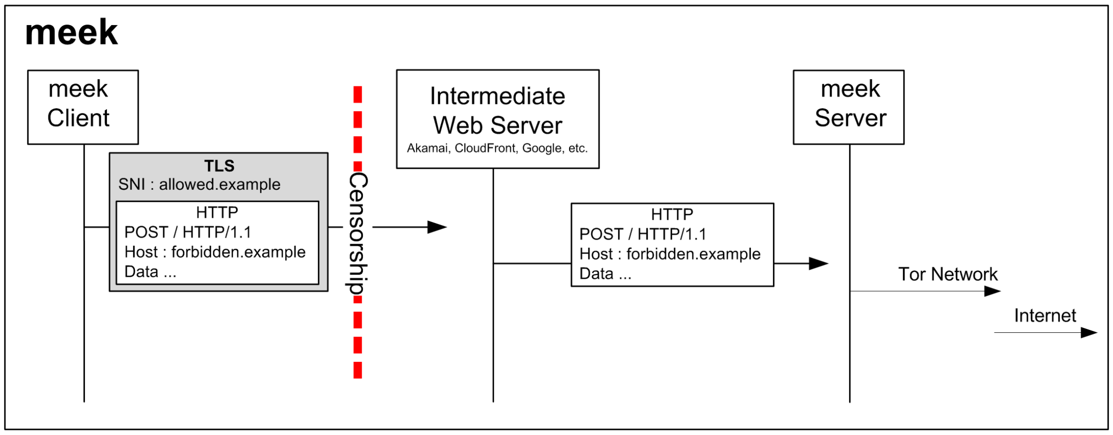


Figure 3.11: Tor Pluggable Transport meek

Tor Pluggable Transport - meek

This protocol is based on domain fronting [17] by encoding a data stream as a sequence of HTTPS requests and responses. The requests will go through a hard-to-block third-party web server to avoid talking to Tor bridge directly [17].

The idea of meek is to put the allowed domain on the outside of the request. In particular, the allowed domain is placed on the DNS query and the Server Name Indication (SNI) TLS extension. The forbidden domain will be on the inside of the request, specifically in the Host header of the HTTP request. The allowed domain is the address of the intermediate server. The forbidden domain is the address of the bridge where the meek server is placed. This deception will work when some web services ignore to check SNI and process the request based on the Host header. Amazon CloudFront and Google App Engine are used to implement this pluggable transport. Censors will see the clients communicating with the allowed domain, and it will pass it.

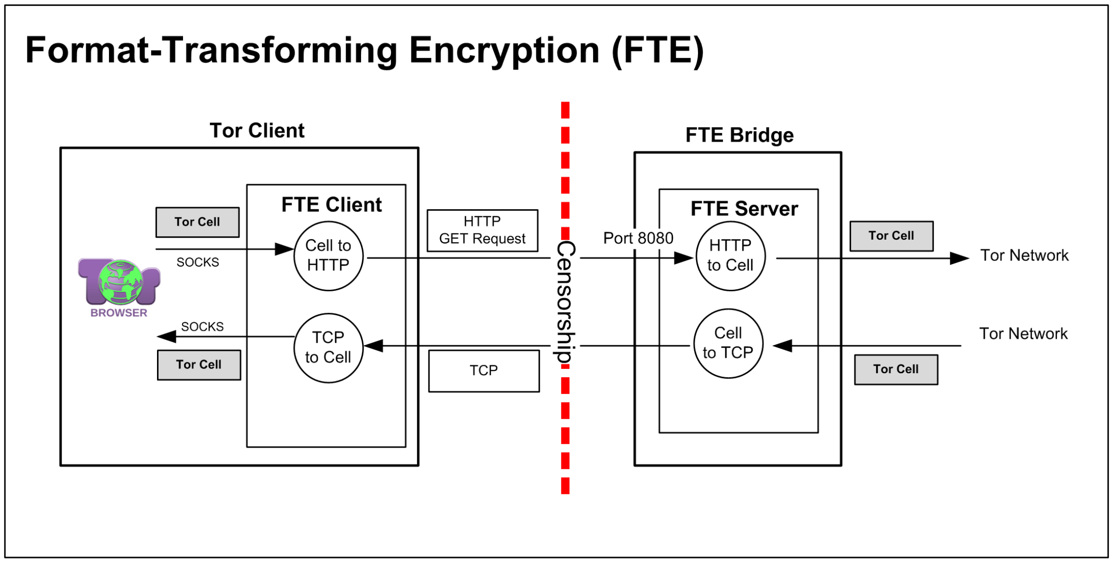


Figure 3.12: Tor Pluggable Transport FTE

Tor Pluggable Transport - FTE

FTE is introduced as a solution in case that the cryptographic protocols are blocked. Most of the deep packet inspection (DPI) systems use the regular expression and keyword search against application layer packets. FTE will make sure that every output will not raise any DPI flags especially in case of keyword search for blocked content. This can be done by encrypting and encoding the packets so that they will pass any regular expression check. The traffic will look like (mimicry) HTTP traffic and absent any flagged keywords [16].

Technically, it is called FTE proxy system [47] which consist of two components client and server. They are communicating with each other in a way that any censor in between will see the traffic as normal HTTP connection using port 8080. FTE client should be in a place where the Internet is censored, and some websites are not available in that area. On the other hand, FTE server should be on the uncensored network to make the FTE client benefit from that connection.

3.2.2 Stand alone Systems

There are some stand-alone CRS that are around. They have a different concept than Tor, but they share the concept of censorship resistance systems.

JAP JonDo

The structure of JAP system is based on the concept of web-mixing and proxy services [48]. It consists of four components which are Java Anon Proxy (JAP), mix servers, cache proxies and InfoService. JAP application is the client-side software and has only one connection to a mix server using TCP/IP Internet connections. The mix server has only one connection to one or two other mix servers. There are three types of mix servers first, middle and last mix server. The mix server which has a connection to the JAP is called the first mix. The mix server which sends the packets to the cache proxy is called the last mix. The mix server with two connections to other mix servers is called middle mix.

The first mix server will receive the traffic from different JAP users at the same time. It will scramble the order of the received data streams, and change their look by applying new encryption phase. The middle mix server job is to receive packets from a mix server and forward them to the other mix server. Cache proxies are the last parts in the JAP system which will make the connections to the Internet. The traffic will be received from the last mix to the cache proxy to communicate with servers outside JAP system. The cache proxy will receive the response, then forward it back to the client through the mixes in reverse order.

The packets that are transmitted from JAP to the cache proxy then to the Internet will go through a chain of connected mixes called MixCascade. There are different MixCascade existed at the same time, JAP client application will select only one of them. This selected MixCascade will remain active for the whole session until the client logs out.

InfoService provides meta-information regarding JAP system such as available MixCascades, information about mix servers, number of users, etc. Figure 3.13 illustrates JAP system structure and components.

To use JAP, the user will configure the browser to use JAP as a local proxy. The default settings are localhost with port 4001, but this can be editable. The client starts JAP which will connect to InfoService. This step will make sure that the software used by JAP and the mix servers are compatible. Then, JAP will register to the first mix by opening a TCP/IP connection. The first mix will reply with information about the MixCascade. It includes the public keys of the mix servers using XML format. JAP will send a special MixPacket encrypted with the public key of the first mix containing two symmetric keys. Each key size is

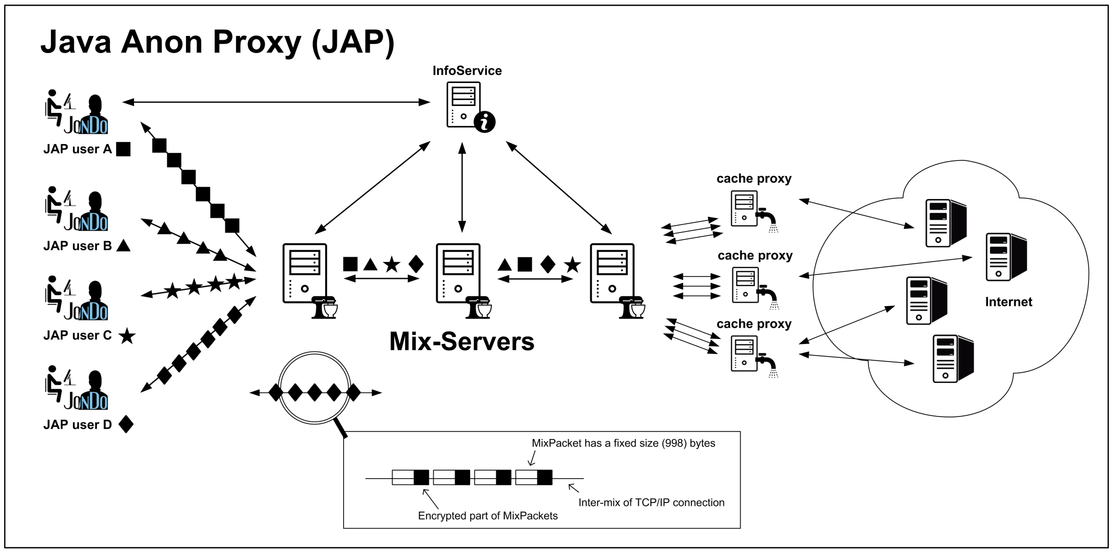


Figure 3.13: JAP System

16 random bytes. One key will be used for packets between JAP and first mix. The other key will be used for packets between first mix and JAP. The browser will be configured to send the packets through JAP. Then, JAP will encrypt the packets using the first key mentioned above to the first mix. After that, the first mix will mix the data that is coming from other clients in one MixCascade to the next mix server.

The packets used in JAP system known as MixPacket with a fixed size of 998 bytes. MixPacket is consist of channel-ID (4 bytes), flag (2 bytes) and data (992) bytes. For every mix, the channel-ID and the data will be changed because each mix will do a single encryption and decryption. The encryption methods used by JAP are RSA with 1024-bit key length and AES with 128-bit key length. The encryption methods used by mixes are AES-128/128 with the OFB-128 mode.

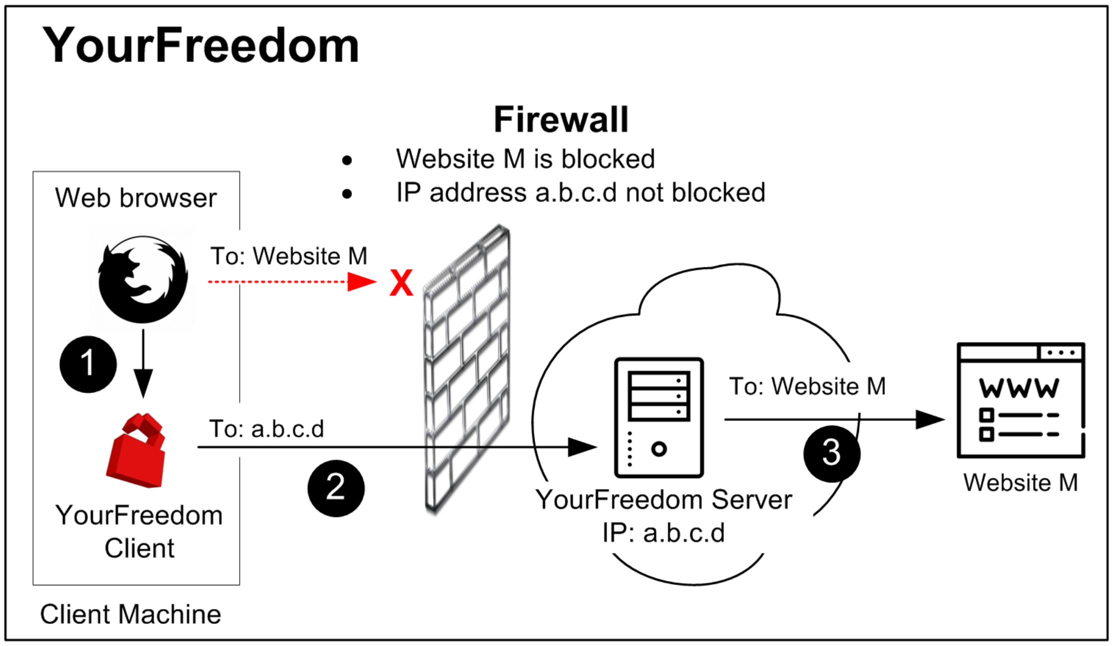


Figure 3.14: YourFreedom CRS

YourFreedom

YourFreedom [49] is one of the most popular CRS used nowadays. Besides supporting most desktop operating systems, it also has an android app with around five million downloads. Based on Freedom House [50] survey, YourFreedom is one of the top ten CRS used in Iran in 2011. They explicitly mentioned on their website [49] that YourFreedom is a connectivity service only, not a perfect anonymizer, VPN nor firewall.

Technically, YourFreedom is a java application that applies the concept of tunneling. With around 39 tunneling servers in 10 different countries that support different tunnel modes such as HTTPS, FTP, UDP, HTTP/POST/CGI and DNS. The idea behind YourFreedom is to turn the client's computer into a web proxy and a SOCKS proxy. Once the user runs the client application, it will connect to one of the available servers using one of the tunneling modes. This method will allow the local applications to use the client side of YourFreedom instated of going directly to the Internet. Then, it will communicate with the server side of the system.

One scenario of using YourFreedom starts with a user wishes to fetch Wikipedia page using Firefox browser. The user will start the client application, and he will choose the tunneling mode with the server that supports it. Once that prepared, the user should configure the browser to use SOCKS port 1080 and web proxy port 8080. After that, the request will go from the browser to YourFreedom client-side application. Then, the client-side application will tunnel the request using the chosen mode, and send it to the server side. The server will communicate with Wikipedia server to ask for the page. Then, The response will return from YourFreedom server to the web proxy on the client side.

CHAPTER 4

WEBSITE FINGERPRINTING

In chapters 2 and 3, we explained the two conflicting parties which are censorship and CRS. Censorship can use many methods to attack CRS systems, and fingerprinting attack is one of them. Fingerprinting means that an eavesdropper can apply a traffic analysis on the encrypted traffic generated by the CRS to get some information about that traffic. In general, fingerprinting could be based on the following methods which are destinations, content, flow properties or protocol semantics [51].

- **Fingerprint Destinations**, a flow can be linked with a protocol based on the destination information that is acquired from the connection tuple. The destination port is a typical target of censorship, for example, HTTP uses port 80. Also, flows with destination IP addresses that are associated with the blacklist systems could be interrupted by censorship.
- **Fingerprint Content**, flows can be fingerprinted by some strings that are specific to some protocols. For example, blacklisted keywords, domain

names and HTTP hosts that could be inside the content of the traffic. DPI systems could perform a regular expression based traffic classification to identify such strings.

- **Fingerprint Flow Properties**, packet length, and timing-related features such as inter-arrival times and burstiness could be used to identify the type of the traffic. The censor could fingerprint some protocols by creating a statistical model based on the mentioned traffic properties.
- **Fingerprint Protocol Semantics**, the censor can fingerprint the traffic based on the behavior of that protocol. The censor will trigger different types of active manipulation to see how the protocol behavior. For example, what will happen in case of dropping, injecting, modifying and delaying of the packets?

Censorship already applied some of those fingerprinting attacks against CRS. For example, Tor was attacked by blocking all the public IP addresses that were published in the directory authorities (Fingerprint Destinations).

In this chapter, we explain a different type of fingerprinting attacks which is website fingerprinting attacks. In this attack, the censorship will try to identify the website visited by the user while using any CRS. This type of attack depends on machine learning and the traffic analysis of the encrypted traffic.

4.1 Website Fingerprinting Attacks

Website Fingerprinting (WF) is a specific type of fingerprinting attacks. WF attack will help some local and passive network eavesdroppers to identify the web page accessed by a client by using traffic analysis [52]. The position of the attacker could be in any router between the user and the destination website. This powerful yet invisible threat could be implemented by several entities such as Government, ISP, network administrator or secret services. Each one of those entities may use WF attacks for different purposes such as censoring, surveillance or advertisements. Despite that the traffic is encrypted, the attacker will act as a passive network eavesdropper without the need to decrypt the traffic. The attacker will start to collect the victim's traffic, then apply machine classification to that collected information. Based on that results, the attacker can guess which web page was visited by the victim.

The following scenario is assumed by any WF methods [53]. The scenario starts with a user who wants to protect his web browsing activities. To achieve that, the user will use any CRS that will hide the traffic from any third parties. This task can be done by installing a dedicated software to establish a secure and encrypted link between his machine and a trusted server on the Internet. On the other hand, the attacker who wants to apply WF attack should be able to do three things. First, record the victim's traffic, this is essential to get the information about the victim's browsing activities. Second, the attacker's location is between the victim and the CRS trusted server. The location of the attacker will help him to get the victim's real IP address. Third, the attacker has the power to identify the victim based on the IP address.

There are two phases of WF attack which are training phase and testing phase. In the training phase, the attack will start to monitor and store a set of censored websites. The set of websites could be large or small number of websites based on what interests the attacker. In the testing phase, the attacker will record the encrypted traffic of the targeted user. Then, the attacker will use machine classification to creates the fingerprints. In the end, the attacker will try to match the results with the stored records form the training phase.

We test two types of WF attacks which are Optimal String Alignment Distance (OSAD) also know as Cai-OSAD attack et al. [54][55] and k-Nearest Neighbours (k-NN) classifier [56].

4.1.1 OSAD Attack

In 2012, Cai et al. [54] improved the accuracy of WF on Tor. They modified the kernel of Support Vector Machine (SVM) which used to classify traffic instances in classes that match the collected site. The representation of the classification instances for each traffic trace is consist of a series of (positive or negative) packet lengths. Training and testing phases are based on SVM with the

Damerau-Levenshtein edit distance. Damerau-Levenshtein distance between two traces t and t' is the minimum number of operations to transform a trace t to t'. The operations are insertions, deletions, substitutions, and transposition. The mentioned operations are corresponding to two operations that could occur on the packets inside a stream which are discarding and reordering [57].

In 2013, Wang and Goldberg et al. [55] improved the accuracy of Cai-OSAD on Tor. The attack is also known as Wa-OSAD. They made some modifications to the distance computation. Moreover, they removed the substitutions from the list of operations in Damerau-Levenshtein distance. The reason to remove such operation is that substitutions do not correspond to the different trace when they repeatedly load the same page. The second modification was increasing the cost of outgoing packet operations. That is because it is less likely to have a different number of outgoing packets than a different number of incoming packets for the same page. The third modification was making the transposition cost varied over the packet sequence. They made the transposition cost larger near the top of the packet sequence and with a lower cost at the end of the sequence. That is because packets at the beginning of the sequence will be less affected by the random changes because of network conditions.

To perform this type of attack, we need to collect samples of a set of websites using different CRS. Once the samples of each type of CRS is collected, the packet sequence of OSAD attacks could be in one of three forms as the following.

- **Raw packet sequence**: which means that the packet sequence will be without any changes.
- **Rounded packet sequence**: which means that each packet size in the raw packet sequence will be raised to the nearest multiple of 600. For example, a packet size of 512 will be rounded to 600, 1024 will be rounded to 1200.
- **Fixed length sequence**: which means taking only the first 500 of each raw packet sequence.

4.1.2 kNN attack

There are two scenarios regarding WF attacks, close world, and open world. Close world means that the training and testing phase will target a specific set of websites know as monitored pages. The attacker's goal is to be able to recognize if a client visited a monitored page included in that set. Any visits outside the monitored pages will not be recognized. In more realistic environments the client can visit a page that is not included in the selected set of monitored pages which means the open world. In open world, there is a set of monitored pages that will be used in the training phase and another set of the non-monitored page. The attacker will train the classifier to identify if the client visits a monitored page or at least give the fact that the client visited a non-monitored page.

To address the problem of the open world scenario, Wang et al. [56] designed k-Nearest Neighbours (k-NN) also know as (Wa-kNN). The k-nn classifier uses distance metric to find the similarity between two packet sequences P and P'. The distance metric function depending on two main inputs which are features F and weights W. For features, they want the results to be accurate even if the client applies some defenses that can remove some features from the available feature set. For example, if Tor removes the feature of the unique packet lengths. They have a list of 4226 features, and each feature is a function F that takes a packet sequences P [52]. For example, there are 100 features which are the length of the first 100 bursts. Another ten features for the direction of the first ten packets.

The weight learning process in k-NN is called Weight Learning by Locally Collapsing Classes (WLLCC). This process applied to reduce the weights for less interesting features information. This action will make the classifier focus on the features that are more useful for the classification process.

We collected the samples from a set of websites using different CRS to apply k-NN attack. Once the samples of each type of CRS is collected, the packet sequence of k-NN attacks could be in one of three forms.

- **Cells**: the packet sequence will consist the time arrival of each packet and (+ or -) 1 based on the direction of each packet. The number 1 here will represent the cell count in each packet of Tor traffic.
- **Time and packet size**: means that the packet sequence will consist the

time arrival of each packet and (+ or -) packet size based on the direction of that packet.

- **Packet size only**: means that the packet sequence consists of (+ or -) packet size only based on the direction of that packet.

4.2 Website Fingerprinting Defenses

WF defenses aim to defeat or at least reduces the damage of WF. Each defense follows different approach, but all of them could be in one of two main categories, limited or general [52]. Limited defenses are designed to reduce the damage of a specific type of WF attack. General defenses are more powerful to defeat different types of WF attacks. We studied and applied five types of WF defenses which are Traffic morphing, HTTP Obfuscation, Decoy pages as limited defenses. On the other hand, the general defenses are BuFLO and Tamaraw.

4.2.1 Traffic morphing

In 2009, Wright et al. [58] published Traffic morphing defense. The process of Traffic morphing is to randomly pad some unique packet lengths to the traffic to mimic another website traffic. This process will make those packet lengths look like another set of web pages packets. The goal is to change the packet sequence to mimic the packet sequences from another website, for example, google.com. Traffic morphing is designed to defeat statistical analysis attacks. Specifically, it changes the unique packet lengths of the traffic. However, Traffic morphing will

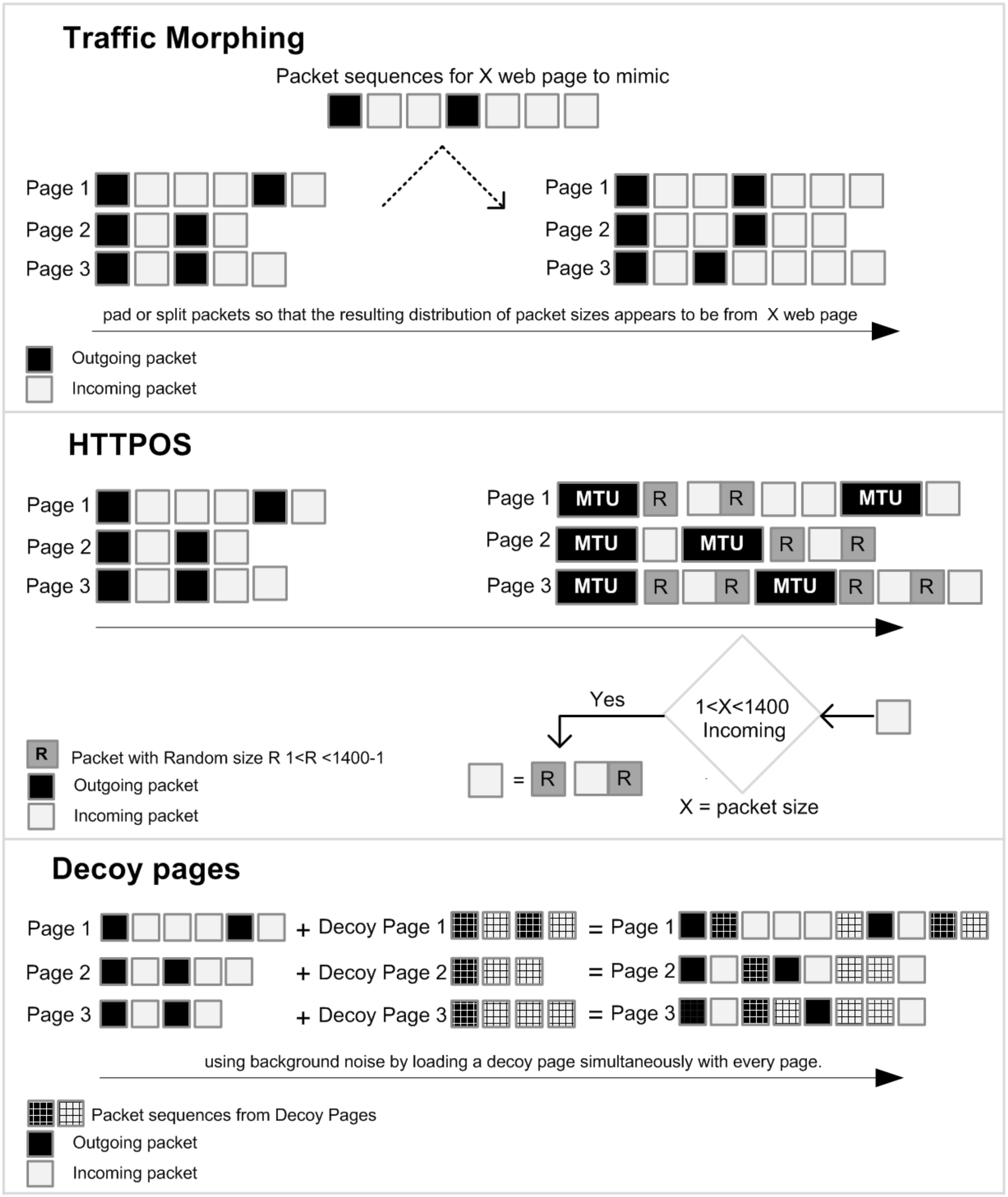


Figure 4.1: Limited WF defenses

not cover sequence length, packet order, or the timing of each packet.

4.2.2 HTTPOS split

In 2011, Luo et al. [37] published HTTP Obfuscation which also know as (HTTPOS). There is a large number of features in HTTPOS, one of them is to split or pad unique packet lengths randomly. Practically, HTTPOS defense means that if we have a positive packet (outgoing packets), then it will always pad it to the MTU which is 1500. Moreover, if we have a negative packet (Incoming packets), then we have three cases:

- if the absolute value of the packet size is more than or equal 1400, then do not change.
- if the absolute value of the packet size is between 1399 and 1, then return a random number r between 1 and (absolute value of packet size -1). Finally, add a packet with size (-r) and another packet with size (packet size + r).
- Else, leave the packet size as it is.

4.2.3 Decoy pages

In 2011, Panchenko et al. [59] proposed a defense based on adding background noise to the traffic which called Decoy pages. It was designed to defeat their WF attack which called FeaturesSVM. The defense is based on loading a decoy page once the client visits a page simultaneously. This process will make the attacker ineffectively able to recognize between the real and decoy pages.

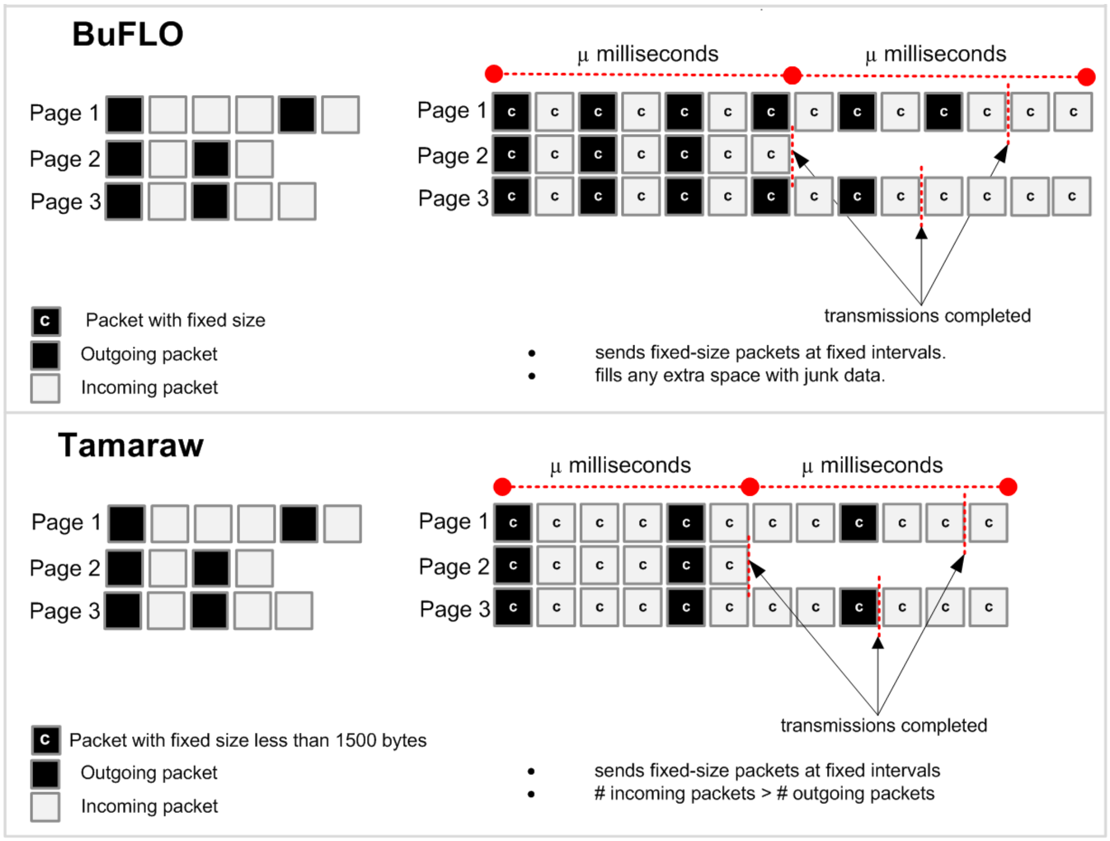


Figure 4.2: General WF defenses

4.2.4 BuFLO

In 2012, Dyer et al. [39] presented a defense called Buffered Fixed-Length Obfuscator (BuFLO) as the first general WF defense. This WF defense causes the transmission length to be extended with more dummy packets inserted to fill the gaps. BuFLO transmits fixed size packets for a fixed amount of time in both directions. If there is no application data left to send, it will fill the gap with junk data which will be removed on the other side.

4.2.5 Tamaraw

As an improvement to BuFLO, Wang and Goldberg et al. [60] presented Tamaraw in 2014. Tamaraw is similar in operations to BuFLO with more improvement in packet padding, packet scheduling, and sequence padding. One option of BuFLO is to pad all packets to the MTU which is 1500 bytes, while Tamaraw will pad the packets to a value that is less than 1500 to reduce the bandwidth overhead. Based on their study, they found that the number of incoming packets is around ten times the number of outgoing packets. Based on that results, Tamaraw will treat the incoming and outgoing packet differently.

CHAPTER 5

ENVIRONMENT SETUP AND DATA COLLECTION

In this chapter, we discuss the preparation phase for conducting the empirical analysis of censorship resistance systems. First, we will illustrate the selected systems in our work. Then, we explain how we did the environment setup with all the needed software. After that, we discuss the data collection phase with all the details. Finally, we will mention some of the challenges we faced during this part.

5.1 CRS Selection

Our goal is to conduct an empirical analysis of censorship resistance systems. To do that, we need to select a set of censorship resistance systems. There are a lot of CRS published in academic papers, but a few of them are deployed and active. Therefore, we only target the systems that are deployed and used around

#	CRS	CRS Class	Language	Stand-alone?	Selected Browser
1	Tor	Proxy	C	Yes	TBB and Firefox
2	obfs3	Randomization	Python	No- Tor PT	TBB and Firefox
3	obfs4	Randomization	Go	No- Tor PT	TBB and Firefox
4	ScrambleSuit	Randomization	Python	No- Tor PT	TBB and Firefox
5	meek	Decoy Routing	Go	No- Tor PT	Firefox
6	FTE	Programmable	Python/C++	No- Tor PT	TBB
7	JAP	Proxy	Java	Yes	Firefox
8	YourFreedom	Proxy	Java	Yes	Firefox

Table 5.1: The selected CRS on our research

the world.

We started with Tor, and it is a well-known anonymity system. Tor is active and heavily used by a large group of users around the world. The Tor project team manages the system with regular updates and patches.

In case Tor blocked, then the users will start to use the Tor pluggable transports bridges. As we mentioned before, Tor supports five pluggable transports which are obfs3, obfs4, ScrambleSuit, meek and FTE. Those five CRS are deployed as part of Tor, and they are active. Therefore, we included those five CRS in our set. In the next section, we see how to deploy each one of them.

Beside Tor, there are other stand-alone censorship resistance systems used nowadays. For example, JAP and YourFreedom are both active and deployed. Both systems are based on Java which will support the installation on different platforms. Therefore, we included them in our set of CRS.

The goal of this phase was to include as much CRS as we can to our set. In table 5.1, we summarized the selected systems that are used in our research. The table shows each system and the class that belongs to it. The last column shows the selected type of browser that each CRS will use in our research.

5.2 Environment Setup

After we selected the set of CRS, we need to deploy each one of them independently. In this phase, we deploy each CRS with different browsers. Once we deploy each one of them, then we start to collect the network traces to analyze them.

For Tor, the user can use Tor Browser Bundle (TBB) or any other browser after configuring it to use Tor. For a user who wants to use Tor with another browser, he or she can use Tor source code, compile it and run it. After that, the user will change the browser's network configuration to send the traffic through Tor application using SOCKS. We selected FireFox browser to run it with Tor and Tor-PT systems.

For Tor source code, we used tor-0.2.8.7 [1]. For Tor pluggable transports, we need to have the bridge's information for the five selected PTs. To do that we selected bridges from Tor bridges database [40]. As a common practice among the Tor community, it is not ethical to publish the address of the bridges, but the information of those bridges are with the authors. We used the same bridges to collect the traces of each Tor pluggable transport with TBB and FireFox. To do that, we edited the torrc file used in both Tor source code and TBB. The torrc file is a Tor configuration file used to guide Tor application to use bridges instead of connecting directly to Tor network.

For each Tor pluggable transports, we need the client software besides the bridge's information. The client-side software will help to encrypt and decrypt the traffic between the client and the bridge. Those programs are shipped with TBB by default because TBB is a portable software. On the other hand, using Tor by running the source code will add another step which is installing those programs manually. For obfs3 and ScrambleSuit we have to use a software called obfsproxy [61]. For obfs4 we have to install another software called obfs4proxy [62]. For FTE we have to install fteproxy [47]. Finally, the client software for meek is called meek-client [63]. Each installation process is done on Ubuntu operating system and mentioned in the appendix.

JAP [64] is a stand-alone software, and it comes without a dedicated browser like Tor. It is based on Java, therefore the user needs java of version 1.4 or above. We used JAP 00.20.001 with Firefox, and configure the browser to use the proxy with port 4001 to make the traffic go through JAP. There are two types of accounts for JAP users free and premium. The free account will allow the user to access free JAP services which are limited to a few number of free servers. We

#	Traffic type	Client Application	Browser	abbr.
1	Normal Traffic without CRS	-	Firefox	NT-FF
2	Tor only	Tor Bundle	TBB	Tor-TBB
3	Tor only	Tor source code	Firefox	Tor-FF
4	Tor with obfs3	Tor Bundle	TBB	obfs3-TBB
5	Tor with obfs3	Tor code+obfsproxy	Firefox	obfs3-FF
6	Tor with obfs4	Tor Bundle	TBB	obfs4-TBB
7	Tor with obfs4	Tor code+obfs4proxy	Firefox	obfs4-FF
8	Tor with ScrambleSuit	Tor Bundle	TBB	SS-TBB
9	Tor with ScrambleSuit	Tor code+obfsproxy	Firefox	SS-FF
10	Tor with meek	Tor code+meek	Firefox	meek-FF
11	Tor with FTE	Tor Bundle	TBB	FTE-TBB
12	JAP	JAP application	Firefox	JAP-FF

Table 5.2: The collected samples with different setups

used the free account, and it was enough to collect the needed traces. JAP software is just a jar file that any user can run it simply using any operating system.

At the end of this phase, we are successfully able to deploy 12 different setups. The next phase will be collecting the network traces of each setup. The different samples are explained in table 5.2. Notice that YourFreedom is not included, and that is due to some challenges which are explained in the next section.

5.3 Data Collection

Our goal is to collect the samples of 100 websites, and each website will be visited 40 times. For that, we selected a set which consists of the top 100 websites based on Alexa's website ranking. Then, with each setup mentioned in table 5.2, we visit each one of the 100 websites 40 times. All packets of each visit will be recorded in a separate file using tshark 2.2.6 [66]. The browser used to fetch all the pages is FireFox 50.0.2 [67], and for TBB we used TBB-linux-32-7.0.2 [1]. We used a virtual machine running Ubuntu 32-bit 16.04 LTS with a memory of 3GB. We make sure that the machine is not connected to any services that could add noise to the collected samples such as cloud services.

To make the process of collecting the samples automated, we used python [68] with several tools. We developed a python script that opens the browser using selenium-3.0.2 [69] to request the targeted website. At the same time, the script will run tshark command to capture the packets of that request. Once this process is done, the captured packets will be saved in a file with pattern website-sample. For example, the file named 10-20 means that this is for website number 10 and sample number is 20. We have to make sure that each file is valid trace sample by comparing the size of the file with the average. The file with a size less than 50% of the average is deleted and considered as bad trace, and then we redo that visit. This process will guarantee that all the samples are valid and ready for next phase.

After we collected the samples, we did another process of cleaning the data from any noises or unneeded data. First, we learn all the IP addresses on each trace and check each IP address organization. For example, If we find an IP related to Ubuntu organization we classify this IP as an unwanted address. Therefore, the packets will not be included in the process. As suggested by all the researcher, we have to ignore all the packets that are not important such as ACK, SYN, RST or FIN. This kind of packets are considered as noise, and we only focus on packets that carry the data which will be in packets of flag PSH. We implemented a python program to process all those tasks since the data set is huge.

At the end of this phase, we collected 12 different types of data sets. The total number of the network traces are 48000 files with size more than 140 GB. We use those files in our experiments after we extract the need information.

We do three different classes of experiments which are statistical analysis, HTTP archive format analysis, and website fingerprinting attacks. Each class will need a specific type of data from the collected sets. For example, WF attack requires a specific information and particular naming patterns. For instance, the k-NN attack accept traces as website-sample, while Cai-OSAD and Wa-OSAD accept website_sample.txt. Each WF attack needs a specific information from the captured files. For example, k-NN attack on cells needs two fields of each captured packet which are the time of arrival and the direction of each cell. On the other hand, OSAD attack needs only the packet size and the direction of each captured packet. We did them all using python scripts since we have 48000 files that contain the captured packets of every setup we used.

5.4 Challenges

We planned to add more CRS to the selected systems, but we were unable to run them. Tor pluggable transport obfs2 was planned to be in the set, we were able to run it at the beginning of this work, but Tor stopped it. Tor is no longer using this type of PT, and there are no bridges in the Tor bridges database for obfs2. Therefore, we excluded it from the set.

StegoTorus [7] was one of the planned CRS to be included in the set. StegoTorus is a CRS that is supposed to run as Tor PT, but Tor project does not officially support it. The designers of StegoTorus provided the source code [65] to run it with Tor. We tried to install it and run it, but it did not connect to the StegoTorus bridges. They used two bridges as StegoTorus server, and we suspected that those bridges are blocked in our region. Therefore, we tried to run StegoTorus using a dedicated virtual machine in the cloud that is outside our ISP restrictions by using Amazon AWS. Also, this attempt did not work, and the client software could not connect to the server bridge. We informed the authors about this issue, and even after a while, it did not work. Therefore, we excluded it from the set.

We planned to use FTE with two different setups which are FTE-TBB and FTE-FireFox. We successfully collected traces of 97 websites using TBB. After that, we faced a problem to connect to FTE bridges. We suspected that FTE was blocked in our region. We tried different FTE bridges from the Tor bridges database. Also, we tried different versions of TBB and Tor source codes during four months without any progress. Therefore, we used the collected samples of FTE using TBB in our set. In the same manner, we faced some challenges when we tried to run Tor at the beginning of this work due to the blocking of Tor in our region. Eventually, we managed to run it.

We were able to run some CRS successfully, but they were costly in term of collecting the traces. For example, FTE without Tor was installed and operated successfully using Amazon AWS. We used a virtual machine running Ubuntu server as FTE server. The client machine was located in Saudi Arabia, and we could successfully use this CRS to access blocked services. The problem was in the costly process of collecting 4000 traces. The connection between the server and the client was not stable for a decent period. This problem makes collecting the data a highly time-consuming process. We used FTE without Tor in other experiments, but we rejected it from the WF attacks.

We were able to run YourFreedom [49] successfully. This system is costly in term of collecting the traces. Like JAP, this software comes with two type of accounts which are free and premium. There is a limited use of the free accounts that could not help us to use it to collect the traces of 100 websites. Therefore, we included YourFreedom in other experiments, but not in WF attacks.

CHAPTER 6

STATISTICAL ANALYSIS OF TRAFFIC PATTERNS

In this chapter, we discuss the statistical analysis of traffic patterns. We conduct two classes of traffic analysis experiments. The classes are statistical analysis and HTTP Archive format (HAR) analysis. Each class has a different setup and required data. In every section, we discuss the goal of each experiment along with the results of each one.

6.1 Statistical Analysis

The goal of this experiments is to see the effects and the changes that occur in the traffic by the selected systems and defenses. In this class of experiment, we selected eight types of statistical traffic analysis tests. Each type of them is a traffic feature that is used in some of the WF attacks. Moreover, each one of those features could provide us with information regarding how the censor sees the traffic.

The normal traffic (NT-FF) will be our base to start this analysis, and each time we compare the changes in the traffic when we used any system.The results were based on the captured traffic of 12 types of systems (11 CRSs and normal traffic). As we mentioned before, with each system, we visited 100 websites with 40 visits for each. The eight statistical traffic analysis tests are:

1. Total number of packets.
2. Total number of Incoming packets.
3. Total number of outgoing packets.
4. Average packet size.
5. Time overhead
6. Data overhead
7. Incoming unique packet lengths
8. Outgoing unique packet lengths

For each experiment, we present the results as a percentage of change according to normal traffic. This will help us to understand how each system change the traffic from the normal when we apply it. Moreover, we conducted the tests to see the effect of each defense on the traffic with or without CRS. Each column in the following experiments figures is the average result of processing 4000 samples using a particular system. After the review of the following results, each CRS system developers will see the effect when they include such defenses. Therefore, they

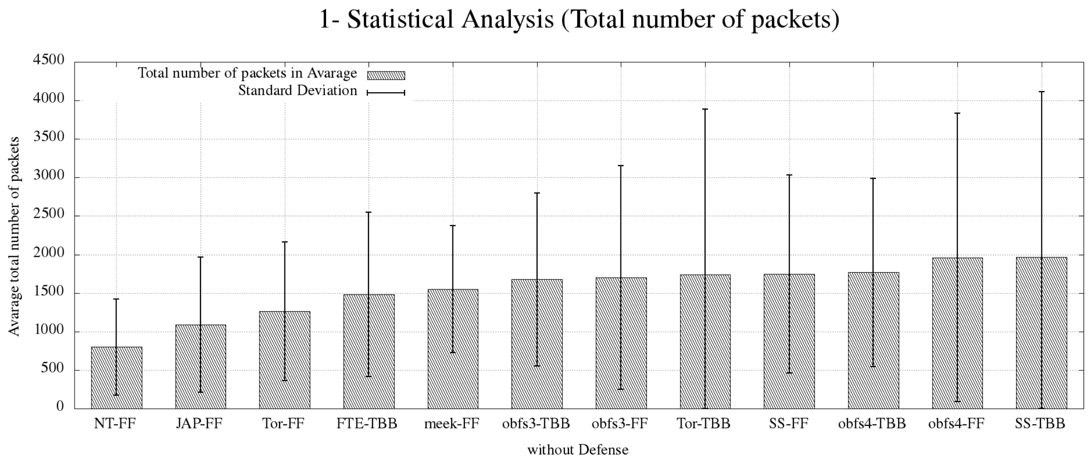


Figure 6.1: The average and the standard deviation of the total number of packets

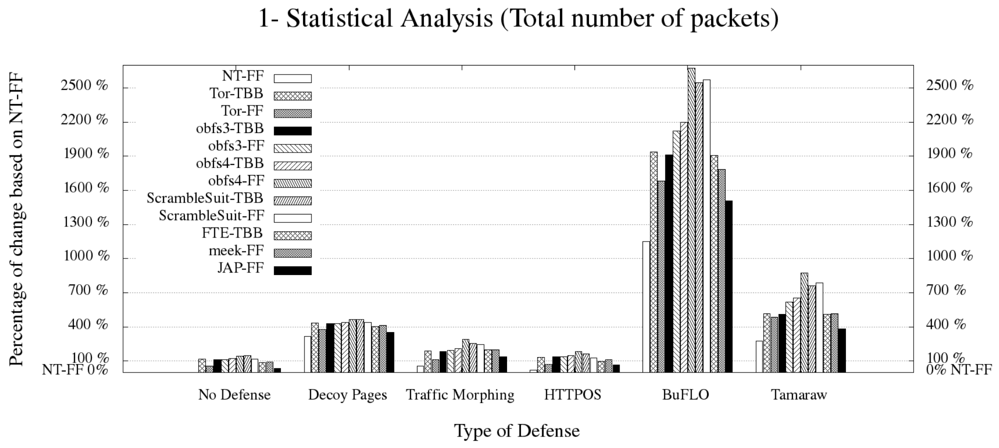


Figure 6.2: Percentage-of change in the total number of packets

can find the appropriate balance between applying any defense with the overhead that comes with it.

6.1.1 Total Number of Packets

- **The Total Number of Packets Per CRS**: In figure 6.1 we show the results of analyzing the total number of packets per CRS. In x-axis, we present each system selected, while in y-axis we show the average value of the total number of packets. The average value means that how many packets needed on average to request a webpage with each system. With each column, we added the value of the standard deviation on the positive side only. The reason for that is to limit the standard deviation to the positive side of the plot. The standard deviation will provide us with the distribution of values around the averages. In our results, all the values are positive, but for some CRS the standard deviation value is higher than the average.

In figure 6.1 we sort them from the minimum to the maximum based on the average value. The CRS with lower values means that it manages to complete the transaction using fewer packets. The lowest value is for NT-FF which is without any CRS. In NT-FF, the packet size is larger than any CRS which will help to end the transaction faster. The next is JAP-FF as the lowest value among all the CRS in this experiment. On the other hand, SS-TBB comes last with the highest value which is around 2000 packets.

- **The Total Number of Packets for all CRS and Defenses**: The results of this experiment are in figure 6.2. In x-axis, we present each type of defense, while in y-axis we show the percentage of change based on the normal traffic. The percentage of change show how much packets added to the traffic. For each type of defenses, the result of normal traffic without any CRS is in the first column of each set of columns. The first column in each set of columns represent the pure effect of each defense on the normal traffic.

The highest values presented for normal traffic are with two defenses BuFLO (1153%) and Decoy pages (319%). This means that we get 1153% more packets if we apply BuFLO without any CRS. Obviously, this percentage will be more if we applied the same defenses to any CRS. The results were expected since both defenses techniques depend heavily on injecting new packets into the traffic. Therefore, injecting more packets will increase the number of total packets for each transaction. Although those defenses give high values, but this change in the traffic will add more defense against WF attacks as we see in the next section. The highest value is 2673% which is the result of applying obfs4-FF with BuFLO. The lowest results are when we apply the HTTPOS defense, specifically Tor-FF and JAP-FF with HTTPOS defense with around 70%. For attacks that depend on traces features such as kNN, applying such defense that produces noticeable changes will be as an advantage for the defense. However, this will be an overhead that will affect the performance of any system.

6.1.2 Total Number of Incoming and Outgoing Packets

The results of this experiments are in figures 6.3 and 6.4. This experiment is more specific than the old one in which we count both incoming and outgoing packets. This type of experiment provide us with the information about the direction of packets for each system. Also, the results illustrate which direction is more important to the defense. By comparing the two results, we can see that decoy

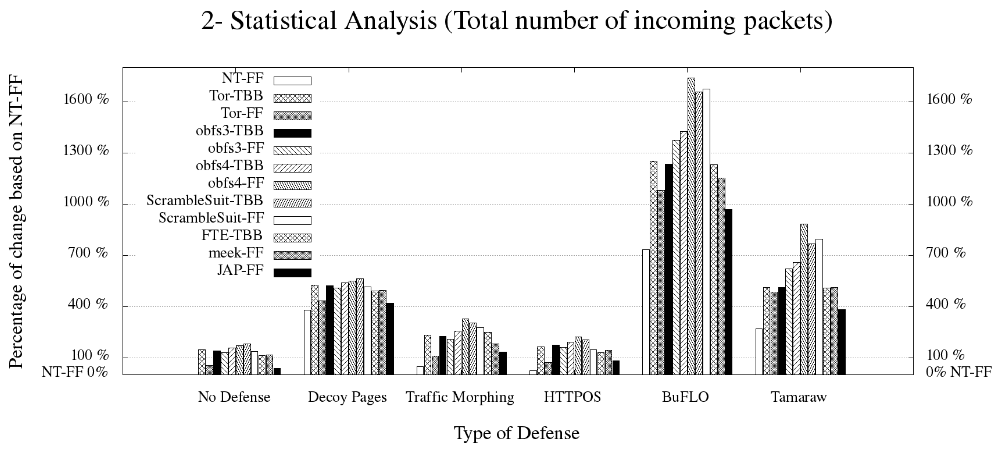


Figure 6.3: Percentage of change in total number of Incoming packets

pages defense and Tamaraw are focusing on injecting more incoming packets. In the other hand, BuFLO is doing the injection heavily in both directions a thousand times more than the normal traffic. For traffic without CRS we can see that using decoy pages defense increase the number of incoming packets by more than 300%, and 700% in BuFLO.

The number of incoming and outgoing packets are features that interest some of the WF attacks. By applying the machine learning in the training sets, the attacks could identify the patterns of those features per website. So, the WF defenses will work hard to change those patterns by injecting more junk packets in both directions. This action add more overhead to the traffic which we see in the data overhead experiments.

6.1.3 Average Packet Size

- **Average Packet Size Per CRS**: The results of this experiment are in figure 6.5. This experiment show the average packet size when we request

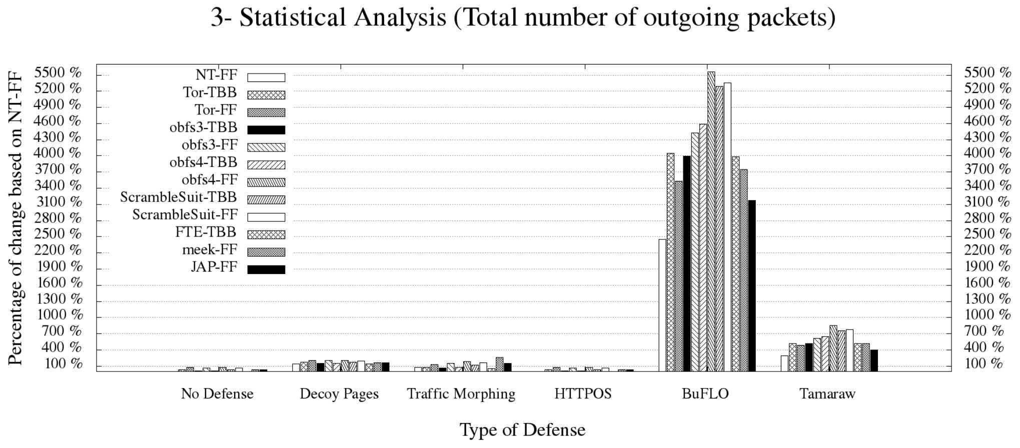


Figure 6.4: Percentage of change in total number of outgoing packets

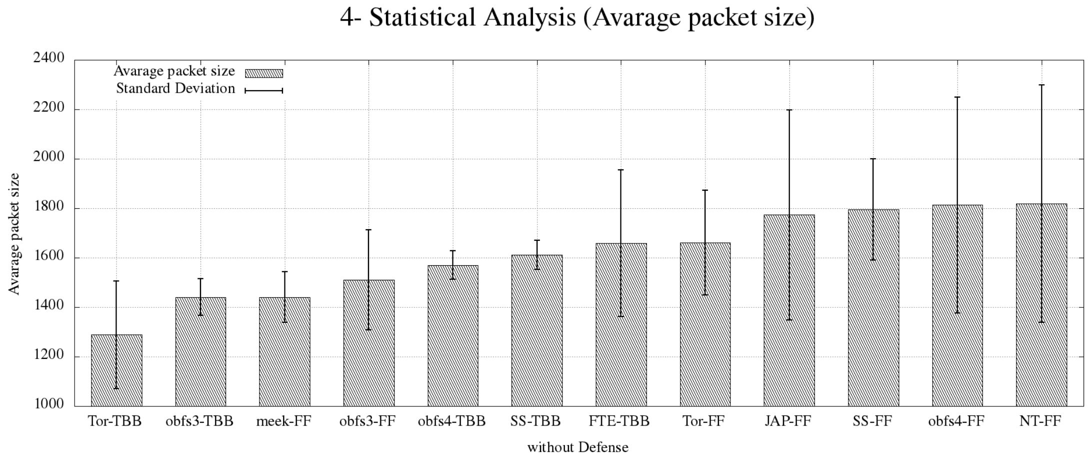


Figure 6.5: The average and the standard deviation of packet size

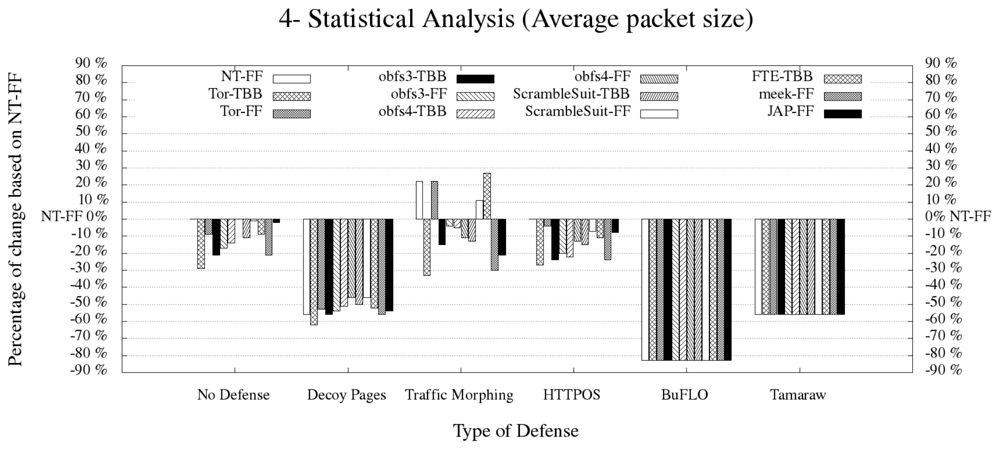


Figure 6.6: Percentage of change of the average packet size

a webpage with each system. We recall the previous results of the total number of packets experiment. If we get a lower number of packets with higher packet sizes, then this means the system will request the webpage faster. This is the case of NT-FF with the highest value of average packet size with few packets. On the other hand, notice that obfs4 is the second highest value of packet size. In addition to that, obfs4 is also the second highest value on the total number of packets experiment. As a result, obfs4 will request a webpage using large packet sizes with more number of packets which means more data overhead. This conclusion will be confirmed in the data overhead experiment.

- **Average Packet Size for all CRS and Defenses**: The results of this experiment are in figure 6.6. In this experiment, we can see the values in positive and negative sides. The reason for such results is that some systems produce packets with sizes that are less than the normal traffic. Since we used the normal traffic as our base, we notice that the normal traffic shipped the packets with the largest size available once the data is ready. Unlike CRSs, which they used fixed size packets or packets that are less than the maximum allowed size. Therefore, the results of average packet sizes are less than the normal traffic as shown in the figure. On the other hand, the defenses are all based on slicing or padding techniques which will change the average packet size of the normal traffic or any CRS traffic that

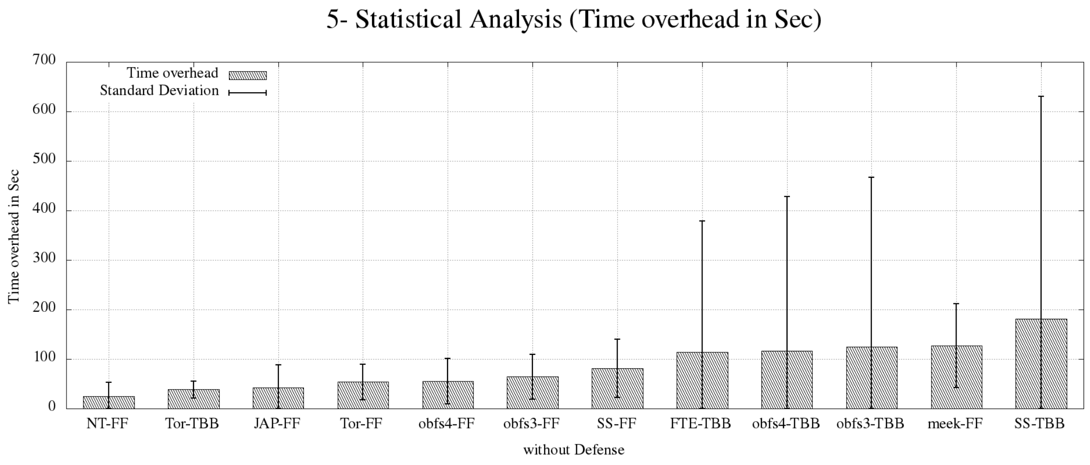


Figure 6.7: The average and the standard deviation of time overhead

apply them. For example, BuFLO source code uses fixed size packets. We selected the 300 bytes for performance reasons instead of 1500 bytes. On the other hand, Tamaraw fixes the packet sizes of all packets to 800 bytes. Both defenses will produce a percentage that is less than the normal traffic even with using different CRS. Also, we can see that both defenses show the columns as equal. That is because all the traces regardless of the CRS type, will have fixed size packets. We can also see that decoy pages defense produced less values even if we used different CRS. That is because decoy pages will inject a lot of small packets that are imported from decoy samples which will change the total average.

6.1.4 Time Overhead

In this experiment, we studied the time overhead caused by different CRS and defenses as in figures 6.7 and 6.8. The goal of this expermen is to find out the time required for each CRS to request a webpage on average. We have to men-

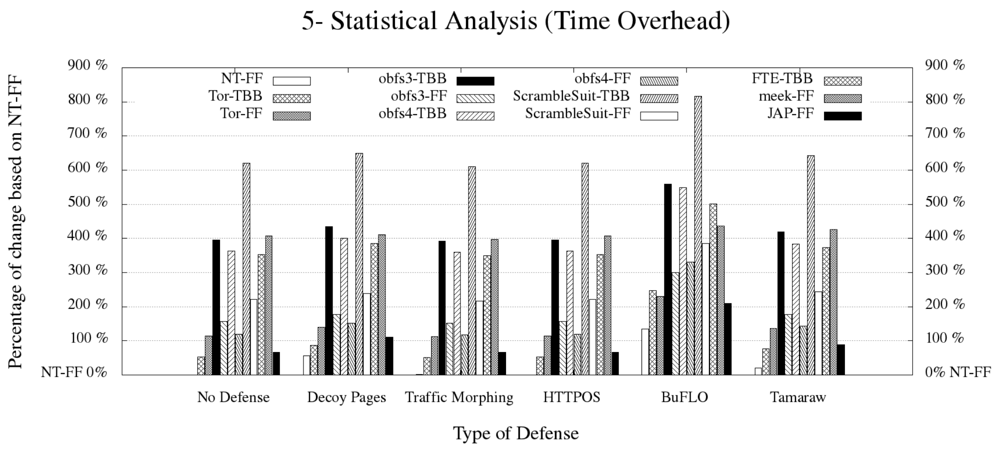


Figure 6.8: Percentage of change in Time overhead

tion here that for such experiment we should be as fair as possible. So, for Tor PT we fixed the bridge's address for all the collected samples. Furthermore, we collected the samples of all CRS at different time periods. The collected samples contain different types of websites from the top 100 websites of alexa's list. As we mentioned before, we have 4000 samples per CRS which are the result of visiting 100 web pages, and with 40 visits per each one. Moreover, Some of those websites contained more contents than others.

- **Time Overhead Per CRS**: As in figure 6.7, we sorted the systems based on the lowest average value, in other words, from the fastest to the slowest. At the x-axis, we can see the different systems, and the y-axis shows the average value in seconds.

  The lowest value is for NT-FF with around 20 seconds. So, if the user requests a webpage without using any CRS, then it will take around 20 seconds on average. This value is considered to be high for people with high-speed fiber optic connections, or when requesting a lite web page. Regardless of all that, the environment used in our experiment and the collected websites will produce such results. For the selected CRS, Tor-TBB comes first as the fastest CRS and followed by JAP-FF. For Tor-PT, the results are higher which means more time overhead to the connection. This overhead is due to the extra router and data processing for the PT-bridge before entering the Tor network.

- **Time Overhead for all CRS and Defenses**: In this experiment, we studied the time overhead caused by different CRS and defenses as in figure 6.8. The results are the percentage of change added to the traffic based on the NT-FF. The first set of columns represent the time overhead added to the normal traffic when we used each CRS without any defenses. The lowest value of time overhead in this set is 52% with Tor-TBB without any defenses. The lowest value for Tor PTs are 120% for obfs4-FF and 156% for obfs3-FF. This means that fetching the pages with Tor PTs will take twice the time of fetching the same pages without any CRS or defenses. The highest value is 816% for SS-TBB with BuFLO defense. The reason for this is that each defense will inject more packets into the original traffic. This injection and packet manipulation will be added to the time overhead.

6.1.5 Data Overhead

- **Data Overhead Per CRS**: This experiment is about the average amount of data (Kbytes) exchanged when a user requests a webpage. So, less data exchanged means less data overhead. As in figure 6.9, we sorted the systems based on the lowest average value. Based on the previous experiments, the total number of packets for NT-FF was the lowest. In addition to that, the average packet size in NT-FF was the highest. In this experiment, the data overhead in NT-FF is the lowest value. On the other hand, the total number of packets and the packet size were high in obfs4-FF which mean more data overhead. This result indicates that obfs4-FF is using a lot of big size packets for the same webpage. Hence, the relation between the packet size and the number of packets will affect the data overhead. Finally, we should mention here that this overhead of obfs4 will play a significant role to resist the WF attacks.

- **Data Overhead for all CRS and Defenses**: To compare all the selected CRS regarding the data overhead, we can focus on the first set of columns in figure 6.10. JAP-FF comes first as the lowest value with 28% overhead added to the normal traffic followed by Tor-FF with 44%. We can focus on one system for example, on Tor-TBB in figure 6.10, we can see that the best results are 63% data overhead with no defense and 77% when we apply the HTTPOS defense on Tor-TBB. Among all defenses, HTTPOS show lower results, These results show that even if HTTPOS expand only

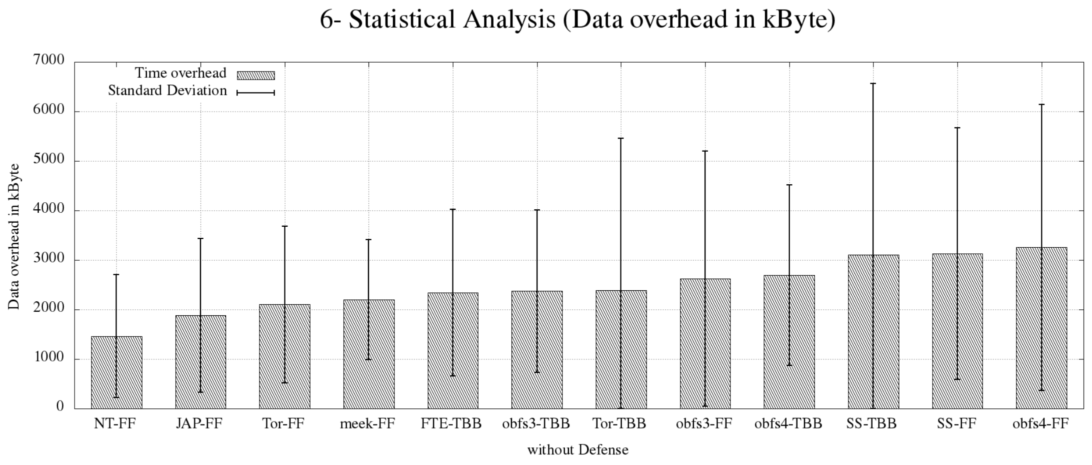


Figure 6.9: The average and the standard deviation of the data overhead

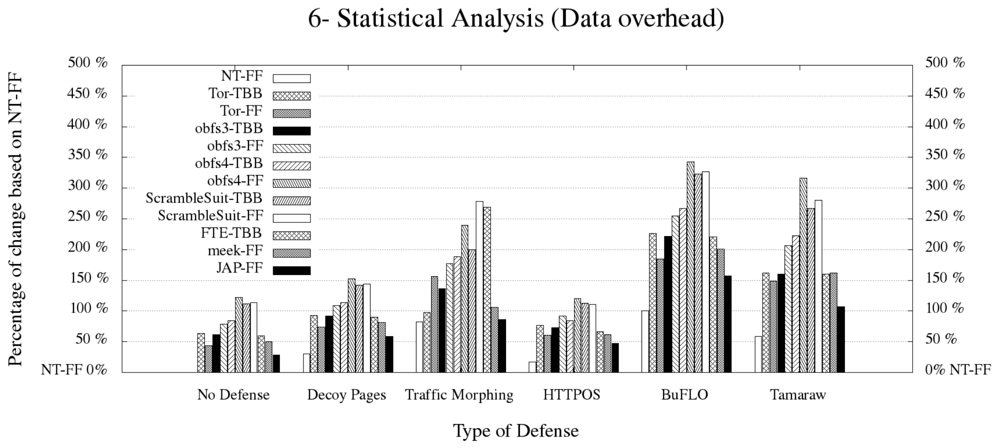


Figure 6.10: Percentage of change in data overhead

the outgoing packets to MTU, it will balance the overhead with the slicing of the incoming packets. The highest values are when we apply BuFLO defense, and it produces at least 100% data overhead.

6.1.6 Unique Packet Lengths

This statistical analysis feature is based on counting the appearance of packet sizes between 1 and 1500 which is the MTU. For example, if all the packets in

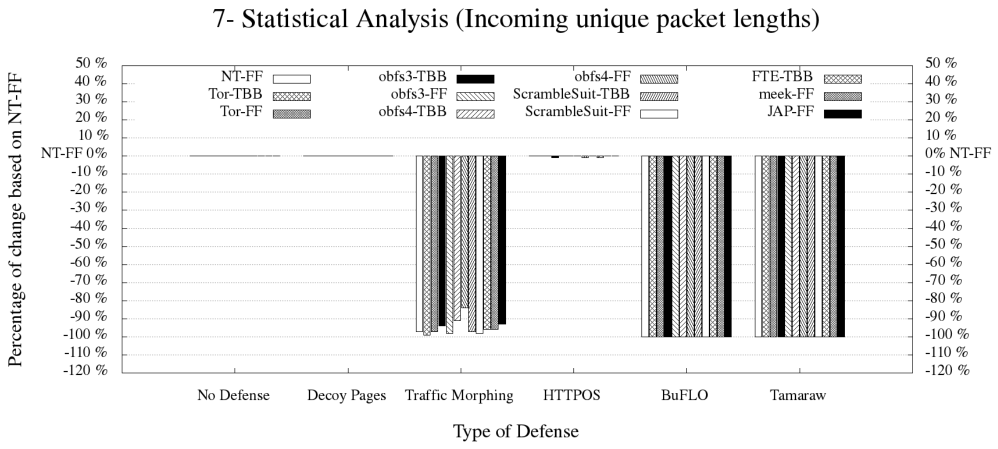


Figure 6.11: Percentage of change in Incoming unique packet lengths

the traffic are with size 1500, then the value of the unique packet lengths is 1. On the other hand, if the unique packet lengths equal 5, then the packet sizes of that traffic will have five different values between 1 and 1500. By comparing figure 6.11 and 6.12, we can see that samples of traffic without defenses, decoy pages, and HTTPOS are same in case of incoming unique packet lengths. This similarity is due to the different amount of packet sizes in the incoming traffic. In particular, the traffic samples have incoming unique packet lengths of 1495. The result in the outgoing unique packet lengths of HTTPOS, BuFLO, and Tammarw are same due to the fixed packet size which 1. For HTTPOS, it will fix the size of outgoing packets to 1500. For BuFLO, all the packets are with the size 300. Also, Tammarw fixes all the packet sizes to 800.

6.1.7 Experiment Summary

Based on all the results, we ranked the selected systems and defenses in tables 6.1 and 6.2. The first table 6.1 is divided into three parts which are time overhead,

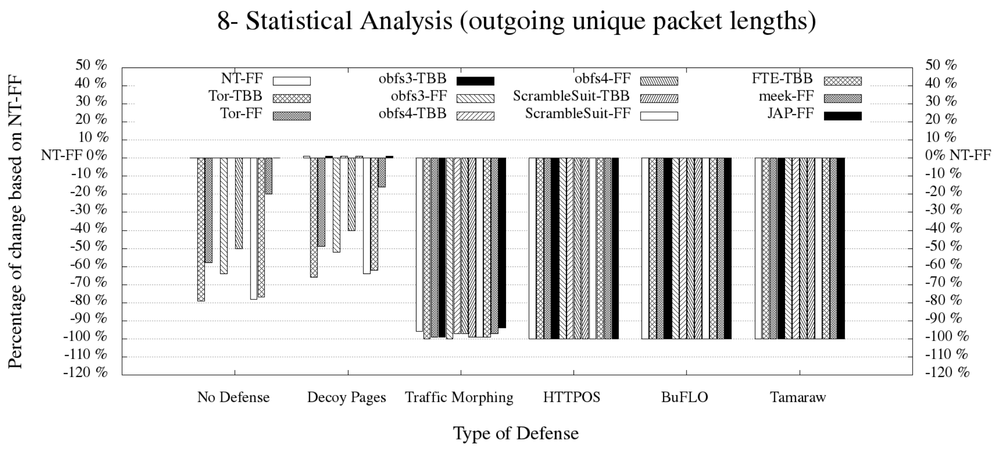


Figure 6.12: Percentage of change in outgoing unique packet lengths

data overhead and the number of packets. In each part, we ranked the systems from the lowest to the highest. For example, NT-FF comes in the first place because it provided the smallest values in time overhead, data overhead, and less number of packets. In the second table 6.2, we ranked the defenses based on the results from the lowest to the highest at the same way.

From table 6.2, we can see that HTTPOS is ranked in the first place. HTTPOS produces less data overhead and less number of packets. Those results show that HTTPOS is a lightweight defense. On the other hand, Decoy pages and Tamaraw show acceptable overhead results in the third and fourth places. As we mentioned before, the WF defenses are applied to reduce the damage of WF attacks. So, if any CRS developer wants to apply WF defenses, they should consider the overhead and the resistance to WF attacks. We see in the next chapter the results of all those CRS and defenses with the WF attacks.

	Statistical Analysis Summary					
	Time overhead		**Data Overhead**		**Number of Packets**	
#	CRS	CRS Class	CRS	CRS Class	CRS	CRS Class
1	NT-FF	-	NT-FF	-	NT-FF	-
2	Tor-TBB	Proxy	JAP-FF	Proxy	JAP-FF	Proxy
3	JAP-FF	Proxy	Tor-FF	Proxy	Tor-FF	Proxy
4	Tor-FF	Proxy	meek-FF	DR	FTE-TBB	PRGM
5	obfs4-FF	RND	FTE-TBB	PRGM	meek-FF	DR
6	obfs3-FF	RND	obfs3-TBB	RND	obfs3-TBB	RND
7	SS-FF	RND	Tor-TBB	Proxy	obfs3-FF	RND
8	FTE-TBB	PRGM	obfs3-FF	RND	Tor-TBB	Proxy
9	obfs4-TBB	RND	obfs4-TBB	RND	SS-FF	RND
10	obfs3-TBB	RND	SS-TBB	RND	obfs4-TBB	RND
11	meek-FF	DR	SS-FF	RND	obfs4-FF	RND
12	SS-TBB	RND	obfs4-FF	RND	SS-TBB	RND

Table 6.1: Statistical Analysis Summary, (RND) for Randomization, (PRGM) for Programmable and (DR) for Decoy Routing

	Statistical Analysis Summary (Defenses)			
	Data Overhead		**Number of Packets**	
#	Defense	Class	Defense	Class
1	HTTPOS	Limited	HTTPOS	Limited
2	Decoy pages	Limited	Traffic morphing	Limited
3	Tamaraw	General	Tamaraw	General
4	Traffic morphing	Limited	Decoy pages	Limited
5	BuFLO	General	BuFLO	General

Table 6.2: Statistical Analysis Summary for defenses

6.2 HTTP Archive Format Analysis

In each webpage, there are several objects such as images, scripts and cascading style sheets. When a user requests a webpage, the browser will communicate with the server to fetch all objects. The goal of such experiments is to see the impact of each selected CRS on the browser's behavior. To study the webpage fetching process, we analyze the HTTP Archive (HAR) file [70]. It is a JSON-formatted file generated by the browser that contains information about all objects in a single request.

6.2.1 Browsers and Webpage Objects

When the browser fetches an object in a webpage, then the process go through four phases. Those phases are scheduling, connecting, sending, waiting and receiving. Each stage takes a period of time, usually milliseconds. The first phase is scheduling phase, which measures the time spent for an object in a queue waiting for network connection. The second phase is connecting phase, which measures the time taken for an object to create a TCP connection. The third phase is sending phase, which is the time taken for sending the HTTP request to the server. The fourth phase is the waiting phase, which is the waiting time for the response that is coming from the server. The fifth phase is receiving phase, which measures the time for reading the whole response from the server.

Figure 6.13: Fetching Wikipedia.org using different CRSs

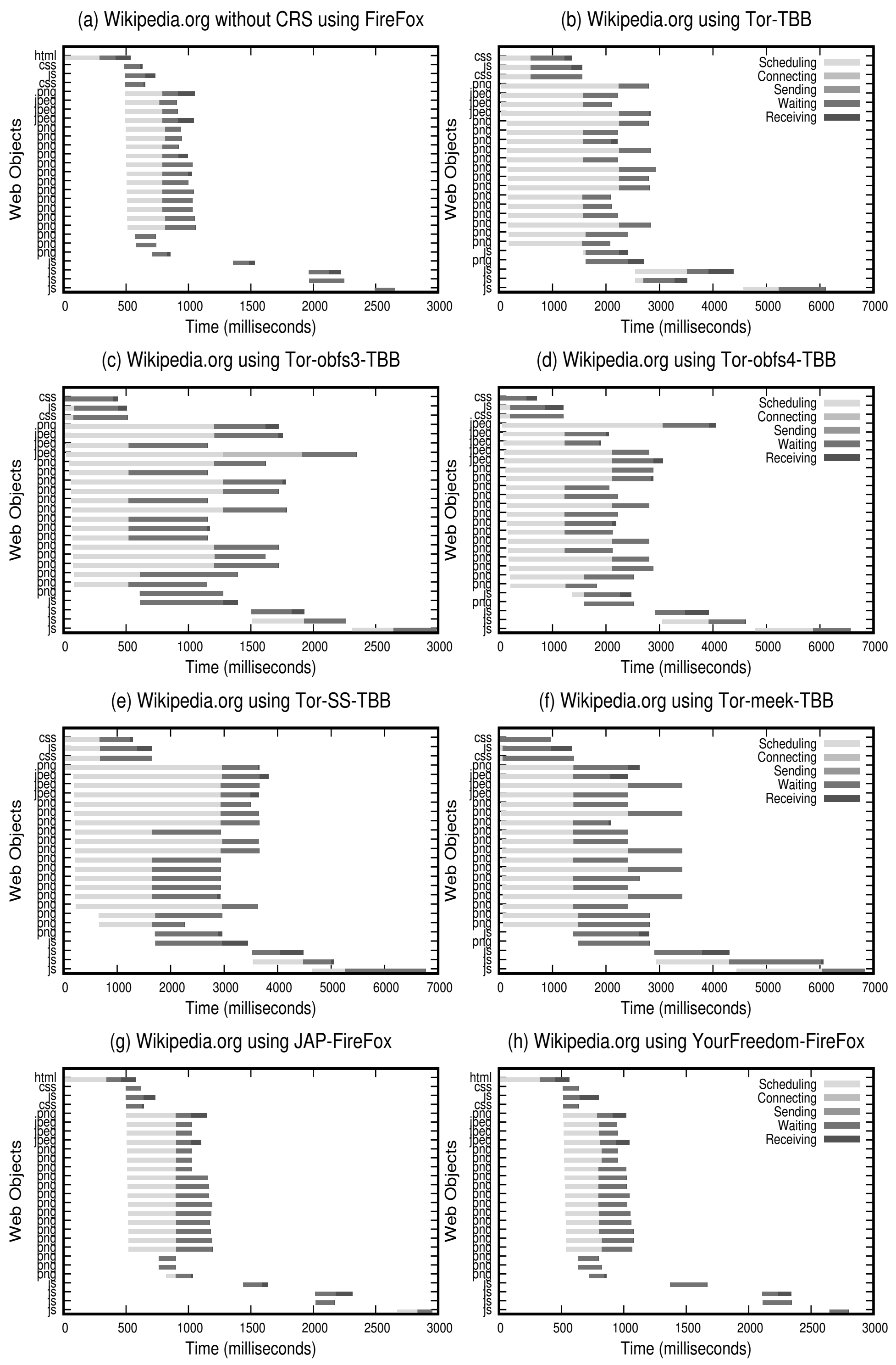
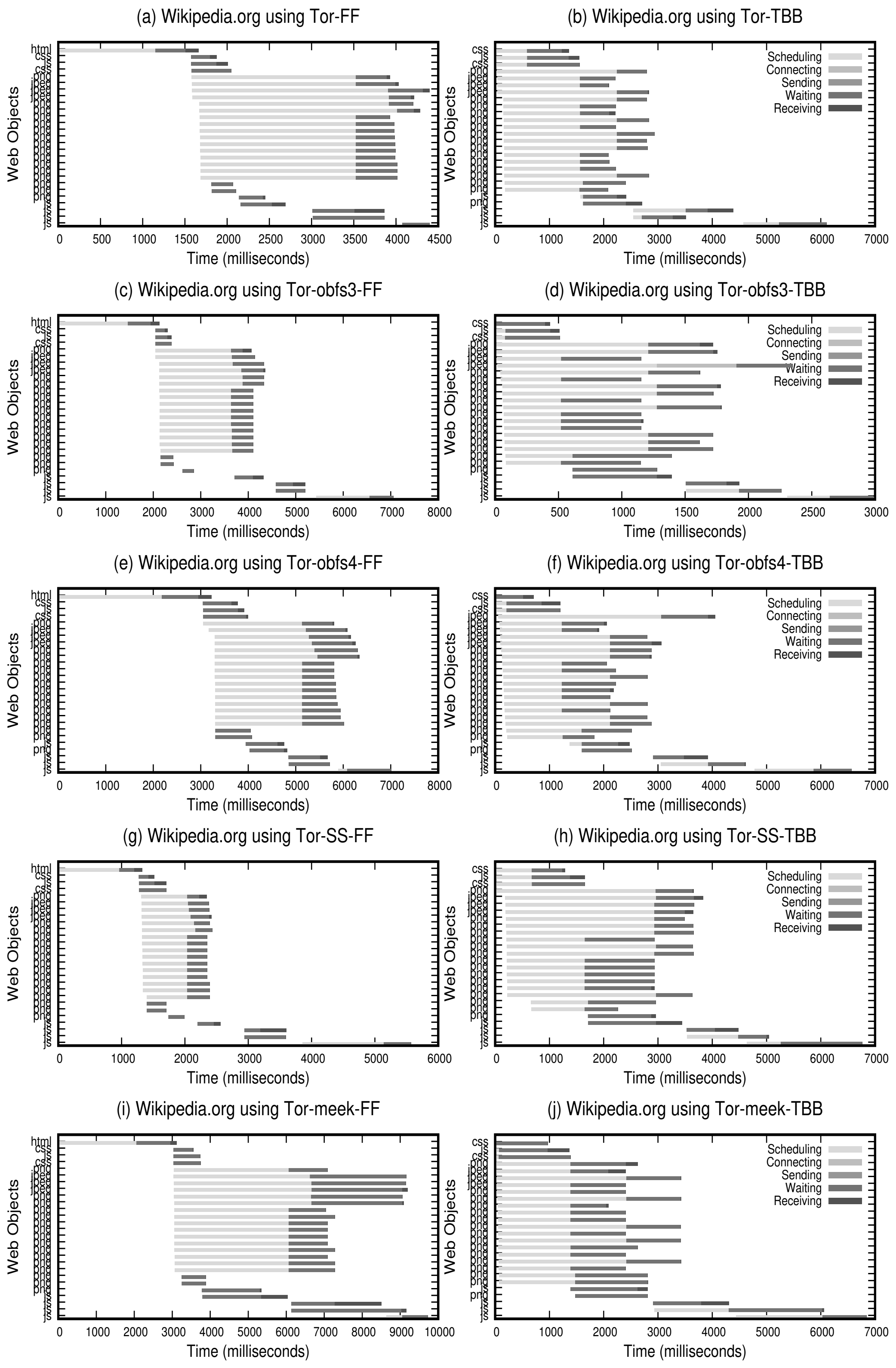


Figure 6.14: Tor and Tor PT with FireFox versus Tor Browser Bundle



6.2.2 Experiment Setup

To prepare for this experiment, we need to select a webpage target, CRS, browser type and datasets to process. We selected the Wikipedia home page as a webpage target. Then we deploy each CRS selected in our set, and request our targeted webpage. For the browser type, we selected Firefox and TBB, and they are both supporting the HAR files. For each request, we need to get the HAR file that is generated by the browser. The HAR file is a JSON-formatted file generated by the browser that includes information about each requested object.

We used the same machine and browsers versions as in 5.2. The data collection process starts by deploying the CRS, then open the browser without any saved caches or history files. After that, we request the targeted page and wait until the page is entirely loaded. Then, we can extract the HAR file from the browser. At that point, our dataset is a set of HAR files, and each file will hold the information about a visit to Wikipedia home page. We developed our python scripts that will extract the information from the HAR files and represent them as plots.

6.2.3 Experment Results

We conducted two experiments based on HAR files. The first one was comparing eight types of requests to Wikipedia homepage. The second experiment was comparing the same request using different Tor PT-bridges using Firefox and TBB. For the first experiment, we selected eight types of requests to Wikipedia homepage as following

1. Normal traffic without CRS using Firefox (NT-FF).
2. Tor only using TBB (Tor-TBB).
3. Tor with obfs3 using TBB (obfs3-TBB).
4. Tor with obfs4 using TBB (obfs4-TBB).
5. Tor with ScrambleSuit using TBB (SS-TBB)
6. Tor with meek using TBB (meek-TBB)
7. JAP using Firefox (JAP-FF).
8. Your Freedom using Firefox (YourFreedom-FF).

Wikipedia homepage at the time of this experiment has 27 objects. Those objects are one HTML page, two css, five javascript, 16 PNG and 3 JPG objects. As illustrated in figure 6.13, we can clearly see that using Firefox in three types of request are almost the same compared to other requests. On the other hand, the requests done by using TBB are more different and random. The TBB is the browser used by Tor which is a modified browser based on Firefox Extended Support Release (ESR). To investigate this results more, we conducted the second experiment.

In the second experiment, we fixed the same Tor PT-bridges and requested the same page using Firefox and TBB. As illustrated in figure 6.14, the results show that even using the same bridges but with various types of the same browser we get different results. To understand those results we have to study the browser used by Tor which is TBB.

Tor project team edited Firefox browser to make it more resistant to traffic analysis attacks. They changed the preferences by applying two features which are HTTP pipelining and randomizing number of requests [36]. HTTP pipelining as in figure 3.5 means that the client will be allowed to make multiple requests without waiting for each response [71]. As we mentioned in chapter 3.2.1, request randomization means that in each request the number of concurrently fetched objects will vary. This feature will avoid having any patterns while requesting the web page objects. Moreover, such feature will produce different patterns for the same request each time. Tor project team applied this feature by configuring Firefox preferences (network.http.pipelining.X) to true, while the randomization is achieved by a patch. On the other hand, Firefox removed this feature from the latest versions of their browsers.

CHAPTER 7

RESISTANCE TO WEBSITE FINGERPRINTING ATTACKS

We conducted 321 experiments regarding WF attacks using three major attacks Wa-kNN, Wa-OSAD and Cai-OSAD. We applied the attacks on 12 different types of traces independently. For collecting samples phase, we used the same environment setup mentioned in section 5.2. After collecting all the samples, we performed a cleaning phase as required by each type of the attacks to prepare the traces. The plan is to perform the attacks on the traces without any defenses. Then, conducting the same attacks again after applying different types of defenses to see the effects. In this chapter, we discuss all the result of WF attacks and defenses.

7.1 Wa-kNN

For Wa-kNN attack, there are three types of traces which are cells, time and packet size, packet size only. Wa-kNN attacks are targeting Tor traffic, and the best results come when the traces are in the form of cells. We conducted the other Wa-kNN attacks with different traces to see the result of changing the type of traces. For each set of traces that were collected using the same CRS, we conducted six types of experiments on cells as following:

1. Wa-kNN attack without any defense.
2. Wa-kNN with Decoy pages defense.
3. Wa-kNN with Traffic Morphing defense.
4. Wa-kNN with HTTPOS split defense.
5. Wa-kNN with BuFLO defense.
6. Wa-kNN with Tamaraw defense.

The accuracy results in Wa-kNN attack is between 0 and 1. For example, if we get a result of 0.90 that means Wa-kNN recognized 90% of the traces. Moreover, each attack will produce a result of five accuracies. The final result presented for each experiment is the average value of those five accuracies. To perform all types of Wa-kNN experiments, we selected 100 websites with 40 traces for each.

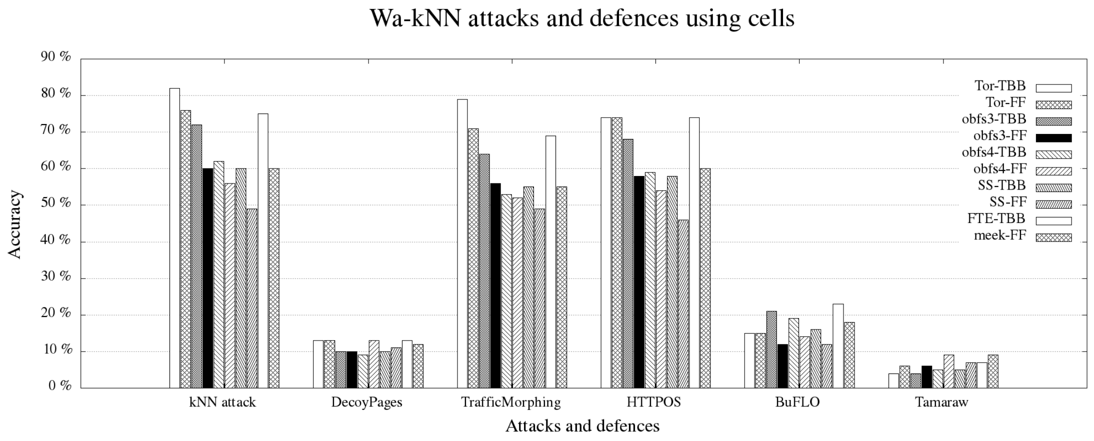


Figure 7.1: kNN attack and defenses on cell

7.1.1 Wa-kNN using Cells

When we configured k-NN to analyze traces as cells, then that means we did not include the packet size in the traces. The information about each packet will be the time of arrival and the direction of each cell. The dataset preparation process will include only the packets that carry the data. As suggested by all the researcher, we have to ignore all the packets that are not important such as ACK, SYN, RST or FIN. This kind of packets are considered as noise, and we only focus on packets that carry the data which will be in packets of flag PSH. Moreover, each packet with PSH flag will be rounded down to the nearest multiple of 512 which give us how many cells shipped inside that packet. In the end, we will represent the trace as a series of time and cell direction.

As illustrated in figure 7.1, results of traces collected using Tor-TBB and Tor-FF without any defense are the highest values among all systems. This result means that the attack could recognize around 82% of the traces. For Tor PT-bridges, SS-FF and obfs4-FF have the lowest results with less than 50%, which means they are more resistance to this attack.

In the second set of columns, we show the results after applying Decoy pages defense. Clearly, we can see the effect of that defense that minimize the results to less than 12% for all systems. This result demonstrates that applying this defense by injecting new packets as a decoy packets will make those systems more resistance to such attack.

The third and fourth sets of columns illustrate two different defenses that are nearly similar to each other which are Traffic morphing and HTTPOS. Both defenses produced results with less than 10% from the traces without defense. The results indicate that the defenses are not adding any noticeable resistance to this attack.

The fifth and sixth set of columns show better results beside decoy pages defense. BuFLO and Tamaraw defenses manage to make the attack recognize less than 20% in all the systems. This results will rank those two defenses as better defenses with decoy pages. The best results are the lowest which are 4% for both Tor-TBB and obfs3-TBB with Tamaraw defense.

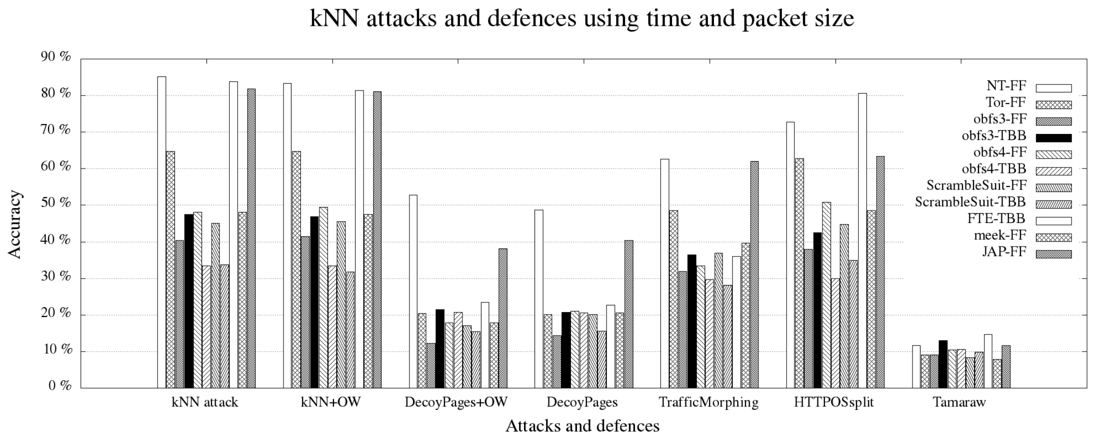


Figure 7.2: kNN attack and defenses on traces as time and packet size

7.1.2 Wa-kNN using Time and Packet Size

The traces of this part of the experiment contain the time of arrival and packet size with the direction of each packet. This type of experiments is new, and we tried to test how the attack will act when receiving a different kind of traces. Although Wa-kNN is mainly targeting Tor traffic, we did include NT-FF and JAP-FF to see how it will work. This action will add more information, but the results of this attack will be different. We included NT-FF and JAP-FF and conducted the following experiments:

1. Wa-kNN attack without any defense.

2. Wa-kNN with 1000 open world traces added.

3. Wa-kNN with Decoy pages and 1000 Open world traces (DecoyPages+OW).

4. Wa-kNN with Decoy pages defense.

5. Wa-kNN with Traffic Morphing defense.

6. Wa-kNN with HTTPOS split defense.

7. Wa-kNN with BuFLO defense.

8. Wa-kNN with Tamaraw defense.

As illustrated in figure 7.2 the attack succeed to recognize more than 80% of three types of traces which are traces without CRS, FTE-TBB, and JAP. On the other hand, the attack managed to recognize around 35% of two types of traces which are obfs4-TBB and ScrambleSuit-TBB. Based on that, we can consider them as systems that are capable of resisting this type of attack. In the second set of columns, adding 1000 open world traces did not affect the results to resist the attack.

The third and fourth sets of columns show the results after applying Decoy pages defense. Clearly, we can see the effect of that defense that minimize the results to less than 25% for all systems except traces without CRS and JAP. The attack could recognize around 15% of obs3-FF and ScrambleSuit-TBB which make them more resistance to this attack using that defense.

The fifth and sixth sets of columns illustrate two defenses which are Traffic Morphing and HTTPOS. Traffic Morphing managed to minimize the impact of the attack to recognize around 50% of the traces except for traces without CRS and JAP. While in HTTPOS, the attack could recognize more than 80% of FTE-TBB as if there is no defense.

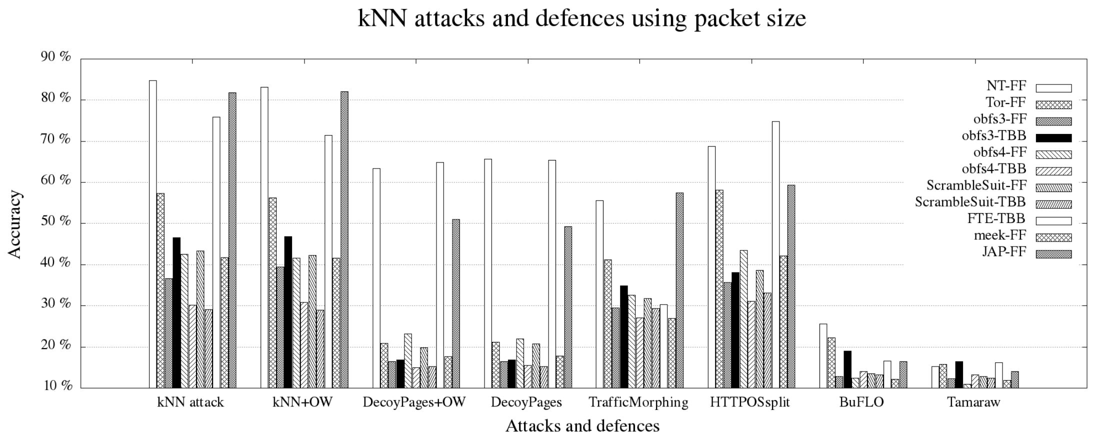


Figure 7.3: kNN attack and defenses on traces as packet size

The seventh set of columns show the best results among all defenses. Tamaraw defense managed to make the attack recognize around 10% in all the systems. This results will rank this defense as the best defense in all the experiments.

7.1.3 Wa-kNN using Packet Size

Traces used in this experiment will contain only packet sizes and the direction of each packet. As illustrated in figure 7.3, the first and second sets of columns show similar results to the previous experiment. After applying Decoy pages defense, the attack managed to recognize around 20% of the traces except for traffic without CRS, FTE-TBB, and JAP. These results indicate that Decoy pages defense is more resistance when the traces consist of packet size only.

Traffic Morphing and HTTPOS in the fifth and sixth sets of columns show better results than the previous experiment, but still higher than other defenses. As in all experiments, BuFLO and Tamaraw defenses in the seventh and eighth set of columns managed to resist the attack to recognize less than 19% of the traces.

As a summary of Wa-kNN attacks, making the traces as cells will produce higher results than other experiments. The higher results mean that the attack could recognize more traces. On the other hand, traces like normal traffic and JAP will not produce realistic results since they do not use cells as Tor. For the defenses, applying Decoy pages, BuFLO and Tamaraw defenses will be best option to resist this type of attack.

7.2 Wa-OSAD and Cai-OSAD

For Wa-OSAD and Cai-OSAD, the traces are rounded packet size. Moreover, we did not include any open world traces because both attacks are supporting only closed world traces. For Wa-OSAD we conducted five types of experiments as following:

1. Wa-OSAD attack without any defense.
2. Wa-OSAD with Decoy pages defense.
3. Wa-OSAD with Traffic Morphing defense.
4. Wa-OSAD with HTTPOS split defense.
5. Wa-OSAD with Tamaraw defense.

For Cai-OSAD we conducted 3 types of experiments as following

1. Cai-OSAD attack without any defense.
2. Cai-OSAD with Traffic Morphing defense.
3. Cai-OSAD with HTTPOS split defense.

The accuracy results in Wa-OSAD and Cai-OSAD attacks are between 0% and 100%. Each Wa-OSAD attack will produce a result of 10 accuracies. On the hand, Cai-OSAD will produce a result of 5 accuracies. In the end, the final result presented for each experiment is the average value of those accuracies. To perform Wa-OSAD and Cai-OSAD experiments, we selected 25 websites with 40 traces for each.

7.2.1 Wa-OSAD using Rounded Packet Size

Figure 7.4 illustrates the results of Wa-OSAD attack with different types of defenses. The first set of columns represent the attack without any defense. The highest value is 93% for traces of NT-FF and 89% for JAP-FF. For Tor and Tor PTs, the highest result is 87% for Tor-TBB followed by 75% for obfs3-TBB. The lowest result is when we used SS-FF and obfs4-TBB with 44%.

The second set of columns represent the attack with Decoy pages defense. The highest value is 55% for FTE-TBB followed by 51% for traces without using CRS. In general, The Decoy pages defense reduced the effect of the attack by around 30% in all system.

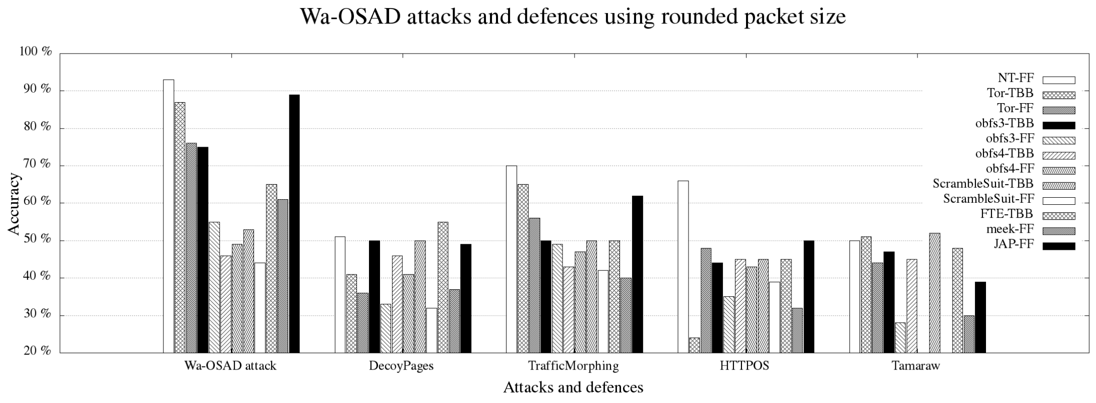


Figure 7.4: Wa-OSAD attacks and defenses on traces as rounded packet size

The third set of columns are the results of Traffic morphing defense. The highest value is 70% for traces without CRS followed by 62% for JAP. The lowest value is 40% for meek-FF, means that the defense managed to reduce the effect of the attack by 24%.

The fourth set of columns represent the attack with HTTPOS defense. In general, the result of this defense is below Traffic morphing by around 10%. Compared to the attack without defense, we can see that this defense reduced the impact of the attack by a value between 40% to 50% in all systems.

The fifth set of columns represent the attack with Tamaraw defense. The highest value is 57% when we use ScrambleSuit-TBB, where the lowest value is 28% for obfs3-TBB. Tamaraw defense showed good results to resist the attack, but not as good as the results of the Wa-kNN attack at the previous experiments. In the end, we can mention that Decoy pages and Tamaraw are the suitable defenses for this attack.

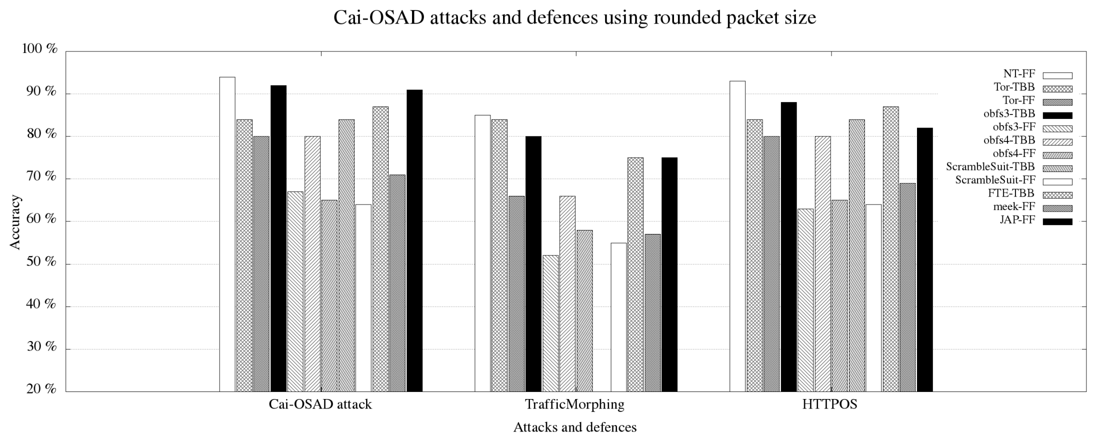


Figure 7.5: Cai-OSAD attacks and defenses on traces as rounded packet size

7.2.2 Cai-OSAD using Rounded Packet Size

The results for Cai-OSAD attacks represented as percentages from 0% to 100%. As illustrated in figure 7.5, the first set of columns represent the attack without any defense. The highest value is 94% for the traffic without any CRS and 91% for JAP-FF and obfs3-TBB. Next, there are 3 CRSs with results that are above or equal 85% which are FTE-TBB, Tor-TBB, and SS-TBB. The lowest value is 64% for SS-FF and obfs4-FF, which means that Cai-OSAD recognized more than half of the traces.

The second set of columns represent the attack with Traffic morphing defense. Traffic without CRS and Tor-TBB get the highest values with 85% followed by obfs3-TBB with 80%. The lowest value is 52% for obfs3-FF which is interesting because we used the same settings but with a different browser. The missing bar is for ScrambleSuit-FF, we tried the experiment with different environments more than four times, but each time we faced an error that stopped the experiment.

The third set of columns represent the attack with HTTPOS defense. Clearly, we can see that the defense did not succeed to reduce the effect of the attack. The results for all the system are similar to the result of the attack without any defense with around 2% less.

7.3 Number of Samples versus Accuracy

We should mention that the results of all the experiments could vary based on the selected samples. For example, if we selected more traces for each site (more than 40 traces per site) we will get higher results. Also, the training set for each experiment will affect the results, for Wa-kNN we selected 36 traces out of 40 for distance learning and four traces for testing. For Cai-OSAD the training set is 36 traces out of 40, and all the attacks applied 10-fold cross-validation.

All those variables will affect in some ways the results of the experiment. In figure 7.6 we demonstrated how Wa-kNN attack changed the accuracy based on the included samples. In that experiment, we fixed the number of websites to 100 and changed the number of samples with each attack. We can notice that the accuracy increased when we increment the number of samples. The reason for such changes is because that the training phase we include more samples. Therefore, the attack will have more information about the samples. Moreover, the chance to identify the samples will be higher.

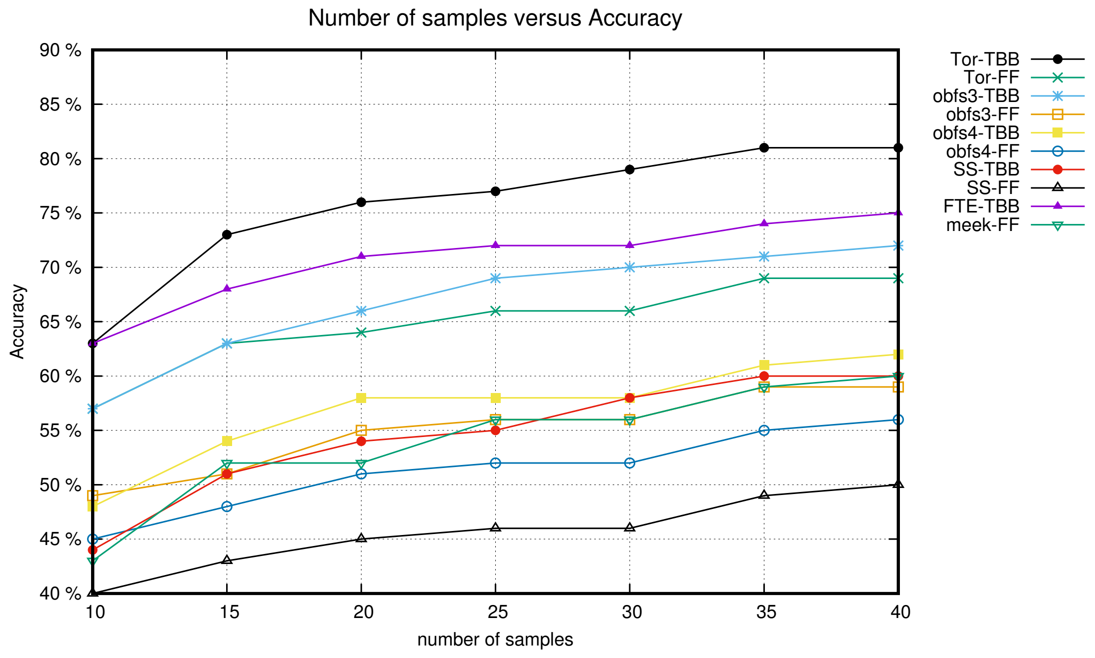


Figure 7.6: the accuracy of Wa-kNN attack versus number of samples

In figure 7.7, we demonstrate how the changing of the number of samples could produce different results for Cai-OSAD. We conduct this small experiment with ten websites and 15 traces for each. The results of the experiment are higher than the previous experiments. The set is tiny, and the chance to recognize that set is higher.

7.4 Experiment Conclusion

In the tables 7.1 and 7.2 we summarised the results of the WF attacks and defenses. In table 7.1, we ranked all the CRS used from the most resistance to the least resistance. As we can see, ScrambleSuit and obfs4 are the most resistance systems to WF attacks. Those two systems produced more overhead results in

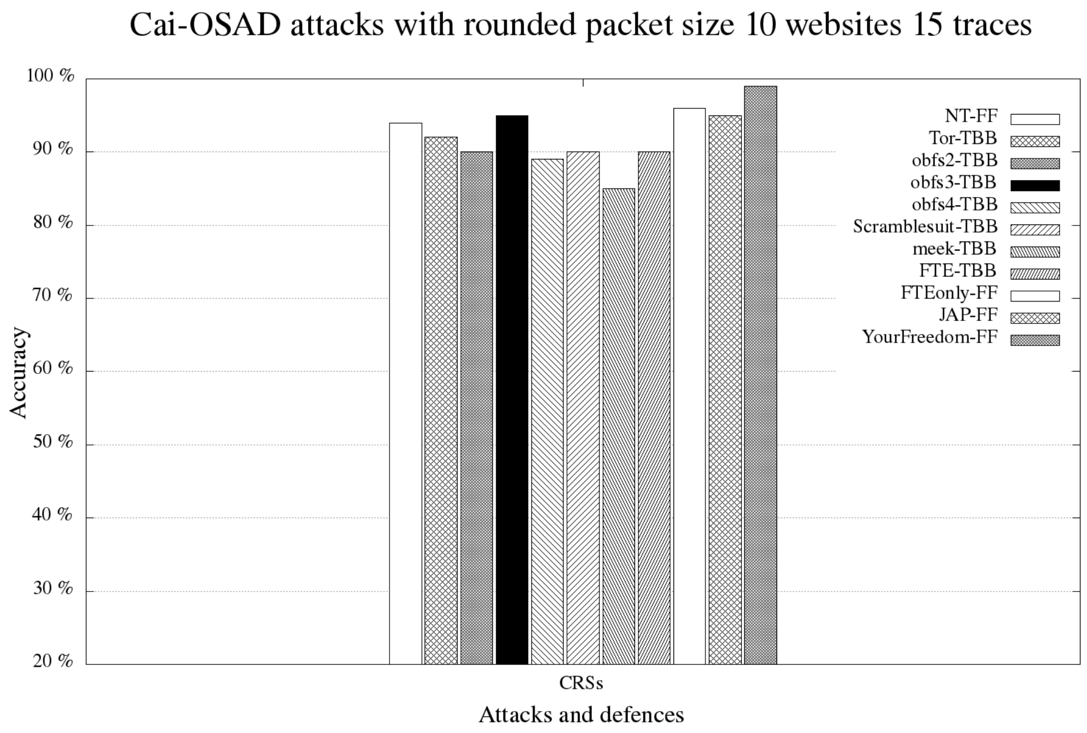


Figure 7.7: Cai-OSAD attacks on traces as rounded packet size with 10 websites 15 traces for each

the statistical analysis experiments of the previous chapter. So, the developers of those systems can tolerate the overhead if they want to reduce the damage of WF attacks. In fact, the overhead will be increased when they apply WF defenses on to on their systems.

In the table 7.2, we ranked the defenses from the most resistance to the least resistance. Tamaraw comes first as the best defense to resist WF attacks. Decoy pages defense comes in the second place as a good defense. Both defenses produced acceptable overhead results in the statistical analysis experiments of the previous chapter. As a recommendation, we suggest that applying those defenses will provide good resistance to WF attacks with acceptable overhead.

	Website Fingerprinting Attacks versus all CRS					
	Wa-kNN		**Cai-OSAD**		**Wa-OSAD**	
#	CRS	CRS class	CRS	CRS class	CRS	CRS class
1	SS-FF	RND	SS-FF	RND	SS-FF	RND
2	obfs4-FF	RND	obfs4-FF	RND	obfs4-TBB	RND
3	obfs3-FF	RND	obfs3-FF	RND	obfs4-FF	RND
4	SS-TBB	RND	meek-FF	DR	SS-TBB	RND
5	meek-FF	DR	Tor-FF	Proxy	obfs3-FF	RND
6	obfs4-TBB	RND	obfs4-TBB	RND	meek-FF	DR
7	obfs3-TBB	RND	Tor-TBB	Proxy	FTE-TBB	PRGM
8	FTE-TBB	PRGM	SS-TBB	RND	obfs3-TBB	RND
9	Tor-FF	Proxy	FTE-TBB	PRGM	Tor-FF	Proxy
10	Tor-TBB	Proxy	JAP-FF	Proxy	Tor-TBB	Proxy
11			obfs3-TBB	RND	JAP-FF	Proxy
12			NT-FF	-	NT-FF	-

Table 7.1: The resistance to website fingerprinting, summary of all CRS, (RND) for Randomization, (PRGM) for Programmable and (DR) for Decoy Routing

	Resistance to Website Fingerprinting Attacks (Defenses)			
	Wa-kNN		**OSAD**	
#	Defense	class	Defense	class
1	Tamaraw	General	Tamaraw	General
2	Decoy pages	Limited	Decoy pages	Limited
3	BuFLO	General	HTTPOS	Limited
4	Traffic morphing	Limited	Traffic morphing	Limited
5	HTTPOS	Limited	-	

Table 7.2: The resistance to website fingerprinting, summary of defenses

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this chapter, we summarize the work of this thesis. Moreover, we discuss the important results and their indications. In addition to that, we present some recommendations to both users and developers of censorship resistance systems based on our results. Then we discuss some of the limitation and validity threats we faced during work. In the end, we introduce some outlines for the future work regarding censorship resistance systems, website fingerprinting attacks and defenses.

8.1 Conclusion

In this thesis, we selected a set of 8 well-known and heavily used censorship resistance systems. We analyzed those systems empirically with more than 800 experiments. Those experiments were divided into two main classes which are the statistical analysis of traffic patterns and the resistance to website fingerprinting attacks.

For the first class of experiments, we studied the patterns appeared on the traffic for each selected CRS. Moreover, we extended the study to include five popular website fingerprinting defenses. For all the selected systems and defenses, we analyzed the changes that occurred inside the traffic. As a result of those experiments, we evaluated the selected systems based on the time and data overhead.

In case of time overhead, we found that proxy class CRS such as Tor and JAP produced less time overhead results. This conclusion means that such systems are faster than other CRS regarding web browsing. Such results are because of the less data processing involved in proxy class CRS. Also, we found that Tor pluggable transports gave more time overhead results. Those results were expected because Tor-PTs are applied on top of Tor.

In case of data overhead, we found that proxy class CRS produced less data overhead on the traffic. The packets generated by those systems are less with larger sizes. This process will allow carrying a large piece of data in fewer transactions. On the other hand, the packets generated by Tor-PTs are more with larger sizes which will add more data overhead.

For WF defenses, the developers depend on padding, slicing and time delay techniques to change the traffic shape. Such experiments will help the decision makers to select the proper defense to balance between overhead and resistance to WF attacks. We ranked the WF defenses based on the added data overhead to the traffic. HTTPOS was in the first place as a lightweight defense with less data overhead. Followed by Decoy pages and Tamaraw with acceptable results regarding data overhead. Then, Traffic morphing and BuFLO at the end of the list with more data overhead due to the massive amount of data injection.

We tested the impact of the browser type used by each CRS to see if there are any defenses applied. We found that TBB applied two types of defenses on the browser level. Such defenses on that level will change any patterns about requesting the objects inside a webpage. We found that TBB requests a set of objects at once with a pipelining method. Moreover, TBB will change the size of that set randomly in each request. Such action will produce a new pattern with each request from the same webpage. This defense is called pipelining and request randomization. We found that this defense was implemented only in TBB using patches.

For the second class of experiments, we examined the impact of website fingerprinting attacks on the selected systems and defenses. We conducted more than 300 attacks to test the resistance of each CRS with a different combination of all the defenses. At the end of those experiments, we evaluated the selected systems based on the resistance to website fingerprinting attacks. In addition to that, we ranked the defenses based on the effect of reducing the damage of website fingerprinting attacks.

For WF attacks, we found that the systems in the randomization class were the most resistance CRSs against all the attacks. Such systems are depending on changing the traffic completely by applying encryption methods. Those systems will remove any traffic features used by the classifiers of Wa-kNN, Wa-OSAD, and Cai-OSAD attacks. On the other hand, we found that proxy class systems were the least resistance CRS to such attacks.

For WF defenses, we found that Tamaraw, BuFLO and Decoy pages are the top defenses that reduced the damage of WF attacks. On the other hand, we found that the lightweight HTTPOS defense was at the end of the list after Traffic morphing. Our results provide a guideline for the developers of any CRS in case they decided to apply one of the defenses.

As a result of all the experiments combined, we proposed a list of recommendations for CRS developers and users. First, we recommended the Tor users to use TBB while accessing Tor network. We explained that TBB applied different defenses against traffic patterns fingerprinting such as request randomization and pipelining. Second, we recommend Tor users to use Tor Pluggable Transports instead of Tor only. Tor Pluggable Transports applied more advanced techniques to bypass censorship and resist website fingerprinting attacks.

Based on our results we proposed several recommendations for CRS developers. First, we noticed that the CRSs in the randomization class produced more overhead than other systems. On the other hand, those systems showed good results regarding the resistance of website fingerprinting attacks. Therefore, the developers of those systems should decide the selected website fingerprinting defense carefully. Such a decision will avoid unnecessary overhead to a good system. Second, we showed that HTTPOS defense is a lightweight defense but with less resistance to website fingerprinting attacks. On the contrary, BuFLO is heavyweight defense with more resistance to website fingerprinting attacks. While Decoy pages and Tamaraw showed an acceptable results in both overhead and attacks resistance. Therefore, those two defenses are a proper choice to balance the overhead with the security.

8.2 Threats to Validity

In this section, we mention some of the limitations we faced in our work. Moreover, we discuss some situations that could affect the validity of our work.

There are some limitations regarding the censorship research. As an example, the lack of information about the real world equipment and specifications. We tried to obtain some information from local corporations about the type of equipment used by them. Such information is considered as secret and can't be shared for security reasons.

Also, There are some limitations regarding the censorship resistance systems. For example, the ability to change the CRS server-side settings. Mainly, the server-side application is controlled by the team of each CRS. For example, if we can get access to Tor bridges or Tor entry node, then we try to apply a different mix of Tor-PTs. Moreover, we can apply the WF defenses on the Tor bridge to test how it will work in such network. It is possible to set up a local Tor network and test it, but the results will not be realistic.

Another limitation regarding the censorship resistance systems was the lack of information for the commercial application. Some closed applications like YourFreedom did not publish the exact mechanism about how it work. At that point, we only depend on some essential information and the network forensics on the traffic generated by such systems.

The results of WF attacks could vary based on the setup of each experiment as we showed in the last chapter. However, sometimes the results could be considered as not valid. The rejection will happen if the results are far from the results found in similar works. The reasons for such invalid results could be an unclean dataset, wrong setup or different parameters in the source code of the attack.

For example, Wa-kNN was applied only on Tor dataset, and there are no tests regarding Tor-PTs. To find valid results for Tor-PTs, we need to test our Tor dataset first. If we could produce similar results to others, then we are in the right direction. We rejected one dataset of Tor with 4000 samples and recollected again. The dataset was mixed with irrelevant data and added some noise to the samples.

8.3 Future Work

In this thesis, we selected eight censorship resistance systems used nowadays. This list of systems could be extended as future work to include more CRS. Furthermore, the list of web browsers used by each CRS cloud be increased to see the changes on the browser level. Moreover, the features used to analyze the traffic of each CRS could be extended to study more patterns.

This work can be extended to include more website fingerprinting attacks. There are other website fingerprinting attacks [52] such as Li-NBayes, Pa-FeaturesSVM, Wa-FLev, and others. Moreover, the designed evaluations measures of this work can be used to compare security and overhead of a larger set of systems and attacks. In addition to that, Including those attacks with more censorship resistance systems will present more empirical comprehensive studies regarding website fingerprinting attacks.

REFERENCES

[1] Torproject, "Tor source code," https://www.torproject.org/download/download.html.en, 2017.

[2] "Users-tor metrics," https://metrics.torproject.org/userstats-relay-country.html, 2017.

[3] D. R. Vukovic, Z. I. Djuric, and D. Gligoroski, "Cryptocloak - improvement proposal implementation," in *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, 2014, pp. 1067–1070.

[4] P. Winter, "Enhancing censorship resistance in the tor anonymity network," Ph.D. dissertation, 2014. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-30752

[5] T. Elahi and I. Goldberg, "Cordon–a taxonomy of internet censorship resistance strategies," *University of Waterloo CACR*, vol. 33, 2012.

[6] R. Dingledine, "Obfsproxy: the next step in the censorship arms race," *Tor Project official blog*, 2012.

[7] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "Stegotorus: a camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 2012, pp. 109–120.

[8] T. Elahi, S. J. Murdoch, and I. Goldberg, "Censorship resistance: Let a thousand flowers bloom?" *arXiv preprint arXiv:1412.1859*, 2014.

[9] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 2012, pp. 97–108.

[10] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.

[11] "Forcepoint url filtering," Aug 2017. [Online]. Available: https://www.forcepoint.com/product/web-filtering/websense-web-filter-security

[12] "Smartfilter." [Online]. Available: https://www.mcafee.com/ca/products/web-protection.aspx

[13] P. Winter, "Measuring and circumventing internet censorship," Ph.D. dissertation, 2014. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-34475

[14] "King abdulaziz city for science and technology," https://www.kacst.edu.sa/eng/Pages/default.aspx, 2017.

[15] "Communications and information technology commission," http://www.citc.gov.sa/ar/Pages/default.aspx, 2017.

[16] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Protocol misidentification made easy with format-transforming encryption," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* ACM, 2013, pp. 61–72.

[17] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 46–64, 2015.

[18] "Tor: Bridges." [Online]. Available: https://www.torproject.org/docs/bridges

[19] P. Winter and S. Lindskog, "How the great firewall of china is blocking tor." in *FOCI*, 2012.

[20] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, "Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 2012, pp. 121–132.

[21] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Format-transforming encryption: More than meets the dpi." *IACR Cryptology ePrint Archive*, vol. 2012, p. 494, 2012.

[22] K. P. Dyer, S. E. Coull, and T. Shrimpton, "Marionette: A programmable network traffic obfuscation system," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 367–382.

[23] "Tor project,obfsproxy." [Online]. Available: https://www.torproject.org/projects/obfsproxy.html.en,2015.

[24] P. Winter, T. Pulls, and J. Fuss, "Scramblesuit: A polymorphic network protocol to circumvent censorship," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society.* ACM, 2013, pp. 213–224.

[25] B. Wiley, "Dust: A blocking-resistant internet transport protocol," *Technical rep ort. http://blanu. net/Dust. pdf*, 2011.

[26] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *Security and Privacy (SP), 2013 IEEE Symposium on.* IEEE, 2013, pp. 65–79.

[27] A. Houmansadr, T. J. Riedl, N. Borisov, and A. C. Singer, "I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention." in *NDSS*, 2013.

[28] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society.* ACM, 2014, pp. 163–172.

[29] W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov, "Sweet: Serving the web by exploiting email tunnels," *arXiv preprint arXiv:1211.3191*, 2012.

[30] J. Massar, I. Mason, L. Briesemeister, and V. Yegneswaran, "Jumpbox–a seamless browser proxy for tor pluggable transports," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2014, pp. 563–581.

[31] C. Brubaker, A. Houmansadr, and V. Shmatikov, "Cloudtransport: Using cloud storage for censorship-resistant networking," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2014, pp. 1–20.

[32] "Tor: Pluggable transports." [Online]. Available: https://www.torproject.org/docs/pluggable-transports.html.en

[33] M.Perry, E.Clark, S.Murdoch, and G.Koppen, "The design and implementation of the tor browser," https://www.torproject.org/projects/torbrowser/design/, 2017.

[34] "Https everywhere—electronic frontier foundation," https://www.eff.org/https-everywhere, 2017.

[35] InformAction, "Noscript - javascript/java/flash blocker," https://noscript.net/, 2017.

[36] M.Perry, "Experimental defense for website traffic fingerprinting — tor blog," https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting, 2011.

[37] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, and R. Perdisci, "Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows." in *NDSS*, vol. 11, 2011.

[38] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 27–46.

[39] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 332–346.

[40] "Tor: Bridgesdb." [Online]. Available: https://bridges.torproject.org/

[41] "obfs2 protocol specification." [Online]. Available: https://github.com/NullHypothesis/obfsproxy/blob/master/doc/obfs2/obfs2-protocol-spec.txt

[42] "obfs3 protocol specification." [Online]. Available: https://github.com/NullHypothesis/obfsproxy/blob/master/doc/obfs3/obfs3-protocol-spec.txt

[43] "custom diffie hellman." [Online]. Available: https://lists.torproject.org/pipermail/tor-dev/2012-December/004245.html

[44] "obfs4 protocol specification." [Online]. Available: https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt

[45] D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange, "Elligator: Elliptic-curve points indistinguishable from uniform random strings," in *Pro-*

ceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013, pp. 967–980.

[46] "Scramblesuit protocol specification." [Online]. Available: https://github.com/NullHypothesis/obfsproxy/blob/master/doc/scramblesuit/scramblesuit-spec.txt

[47] "fteproxy — protocol misidentification made easy," https://fteproxy.org/, 2017.

[48] O. Berthold, H. Federrath, and S. Köpsell, "Web mixes: A system for anonymous and unobservable internet access," in *Designing privacy enhancing technologies*. Springer, 2001, pp. 115–129.

[49] "Your freedom - vpn, tunneling, anonymization, anti-censorship." https://your-freedom.net/, 2017.

[50] FreedomHouse, "Country report for end users in iran," 2011.

[51] S. Khattak, T. Elahi, L. Simon, C. M. Swanson, S. J. Murdoch, and I. Goldberg, "Sok: Making sense of censorship resistance systems," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 37–61, 2016.

[52] Wang, Tao, "Website fingerprinting: Attacks and defenses," Ph.D. dissertation, 2016. [Online]. Available: http://hdl.handle.net/10012/10123

[53] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-

bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security.* ACM, 2009, pp. 31–42.

[54] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 2012, pp. 605–616.

[55] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society.* ACM, 2013, pp. 201–212.

[56] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting." in *USENIX Security Symposium*, 2014, pp. 143–157.

[57] S. Zhioua, "The web browser factor in traffic analysis attacks," *Security and Communication Networks*, vol. 8, no. 18, pp. 4227–4241, 2015.

[58] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis." in *NDSS*, vol. 9, 2009.

[59] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society.* ACM, 2011, pp. 103–114.

[60] T. Wang and I. Goldberg, “Comparing website fingerprinting attacks and defenses,” Technical Report 2013-30, CACR, 2013. http://cacr. uwaterloo. ca/techreports/2013/cacr2013-30. pdf, Tech. Rep., 2014.

[61] “Obfsproxy source code.” [Online]. Available: https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/

[62] “Obfs4proxy source code.” [Online]. Available: https://gitweb.torproject.org/pluggable-transports/obfs4.git/

[63] “meek source code.” [Online]. Available: https://gitweb.torproject.org/pluggable-transports/meek.git/

[64] “Jap – anonymity & privacy,” https://anon.inf.tu-dresden.de/, 2017.

[65] “Overview stegotorus,” https://sri-csl.github.io/stegotorus/, 2017.

[66] “tshark - the wireshark network analyzer 2.4.2,” https://www.wireshark.org/docs/man-pages/tshark.html, 2017.

[67] Mozilla, “Internet for people mozilla,” https://www.mozilla.org, 2017.

[68] “Python.org,” https://www.python.org/, 2017.

[69] “Selenium - web browser automation,” http://www.seleniumhq.org/, 2017.

[70] J.Odvarko, A.Jain, and A.Davies, “Http archive (har) format,” https://dvcs.w3.org/hg/webperf/rawfile/tip/specs/HAR/Overview.html, 2012.

[71] "Rfc 2616 - hypertext transfer protocol – http/1.1," https://tools.ietf.org/html/rfc2616, 1999.

Appendices

Appendix 1 : Installing Tor

all this command line are applied on ubuntu 16.04 LTS

1. download the source code from Tor project it's tor-0.2.8.7.tar.gz

2. make sure you have two libraries

   ```
   $sudo apt-get install libevent-dev
   $sudo apt-get install libssl-dev
   ```

3. compile the source code run :

   ```
   $sudo ./configure && sudo make && sudo make install
   ```

Appendix 2 : Installing Tor PT

obfsproxy

To install all obfs2, obfs3 and ScrambleSuit. First install python pip and then run this commands :

```
$sudo apt-get update

$sudo apt-get install python-dev

$sudo apt-get python-pip

$sudo apt-get libgmp-dev build-essential

$pip install obfsproxy
```

obfs4proxy

To install obfs4. run the following command :

```
$sudo apt-get obfs4proxy
```

FTE

To install FTE. run the following command :

```
$sudo apt-get update

$sudo apt-get install python-dev

$sudo apt-get python-pip

$sudo apt-get libgmp-dev build-essential

$sudo pip install fteproxy
```

meek

To install meek. run the following command :

```
$sudo apt-get install meek-client
```

Appendix 3 : Configuring Tor

To update your tor configuration file and adding the bridges :

```
$sudo gedit /etc/tor/torrc
```

add the following lines based on the PT selected for example , in obfs2 it will be like :

```
UseBridges 1
ClientTransportPlugin obfs2  exec /loc/of/obfsproxy --managed
Bridge obfs2 X.X.X.X:X XX
```

Vitae

PERSONAL INFORMATION

- Name : Abdullah Bin Ayedh AlQahtani

CONTACT INFORMATION

- Email : alqahtani.sec@gmail.com
- Linkedin : https://www.linkedin.com/in/abdullahqh/

EDUCATION

- Reverse Engineering - Short Course at KFUPM
- Cybersecurity for Industrial Control Systems
- Operational Security (OPSEC) for Control SystemPUBLICATIONS
- Hands-On Penetration Testing - Short Course at KFUPM
- Web Hacking and Security - Short Course at KFUPM
- Malware Analysis - Short Course at KFUPM
- Ethical Hacking - Short Course at KFUPM
- MS in Security & Info Assurance (KFUPM)
- Cisco Certified Network Associate (CCNA) Cisco ID : CSCO11810804
- BS in Computer Sciences Imam Muhammad Bin Saud Islamic University.

SKILLS

- **Programming languages** : Assembly language, C family, objective C, Java, Python
- **Database programming** : SQL , SQL Server ,PL-SQL ,My SQL ,ORACLE, MS ACCESS ,SQLite
- **Web Programming** : JSP , ASP , PHP ,JavaScript ,CSS , XML ,HTML , AJAX , XSLT , E4X , XPath ,XQuery

INTERESTS

- Information Security : Computer, Network , Web Apps , Mobile Apps.
- Digital Forensics analysis : Malware Analysis, Network, Disks and Files forensics, Evidences handling, Anti-forensics, Encase, FTK.
- Mobile applications : iOS , android , Windows mobile.
- Web Apps and Programming

PUBLICATIONS

- AlQahtani, Abdullah and El-Sayed M. El-Alfy. "Anonymous connections based on onion routing: A review and a visualization tool." Procedia Computer Science 52 (2015): 121-128.

SEMINARS

- Anonymous Communications (King Fahd University of Petroleum and Minerals Feb 2016)