Web Browsers Resistance to Traffic Analysis Attack

BY

TAHER ALI AL-SHEHARI

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

COMPUTER SCIENCE

October, 2014

بسم الله الرحمن الرحيم

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

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **TAHER A. AL-SHEHARI** 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 DEPARTMENT OF INFORMATION AND COMPUTER SCIENCE**.

Thesis Committee

[illegible]

Dr. Sami Zhioua (Adviser)

(Co-adviser)

[illegible]

Dr. Moataz Ahmed (Member)

[illegible]

Dr. Jameleddine Hassine (Member)

(Member)

[illegible]

Dr. Adel Ahmed
Department Chairman

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

2/12/14

Date

Dedication

I dedicate this thesis to my parents, my wife and children (Ameen and Fahd).

إهـــــــــداء

أهدي هذا العمل إلى والدي ووالدتي حفظهم الله ورعاهم لصبرهم و دعائهم و تشجيعهم لي منذ الصغر ولما قدماه لي من جهد و دعم مادي ومعنوي متواصل خلال مسيرتي العلمية، و مهما عملت فلن استطيع أن أفي ولو بالجزء اليسير مما قدماه لي، أسأل الله العظيم رب العرش العظيم أن يلبسهم تاج الصحة والعافية و أن يوفقني إلى طاعتهم و برهم إنه مجيب الدعاء، كما أهديه إلى ينبوع الصبر والأمل إلى زوجتي أم أمين وإلى أبنائي الغالين أمين وفهد الذي كان لابتسامتهم دور كبير في تخفيف أعباء الدراسة، كما أهديه إلى أخواتي و إخواني لدعمهم لي بالتشجيع والدعاء.

ACKNOWLEDGMENTS

My great thanks to almighty Allah for every things on my live and for helping me to achieve this work. Acknowledgment is due to King Fahd University of Petroleum and Minerals for supporting this research. Most importantly, I would like to express my sincere gratitude to my advisor Dr. Sami Zhioua for supervising me on this thesis and helping me in every step of my work. His patience, motivation and guidance helped me immensely during my research and the writing of this thesis. I am so grateful to Dr. Moataz Ahmed and Dr. Jameleddine Hassine for their invaluable advice and interest in my research and for taking time to serve on my thesis committee. I would not come to this position without huge support from my family and friends. Thank my parents and thank my wife for your great family leader to carry all the burdens to focus on achieving my thesis. My babies Amen and Fahd who have offered their beautiful smile to comfort me during the tough journey of the thesis. I also express my sense of gratitude to all who, directly or indirectly, have lent their helping hand in this enterpriser. Great thanks due to King Fahd University of Petroleum and Minerals represented by Dr. Khaled Al-Sultan for supporting me throughout this research and during my MS study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME:	Taher A. Al-shehari
TITLE OF STUDY:	Web Browsers Resistance to Traffic Analysis Attacks
MAJOR FIELD:	Department of Information and Computer Science
DATE OF DEGREE:	November 29, 2014

Privacy enhancing technologies (PETs) is the most recent field of security focusing on protecting network communications from traffic analysis attacks. The main idea of PETs is to establish an encrypted tunnel to hide the contents and addresses of users from external observers. One of the most popular anonymity systems is The Onion Routing (Tor) with more than 4,000,000 users and around 4500 relays. The popularity of Tor is gained from protecting users' confidential activities and personal privacy by hiding their locations and traffic contents against traffic analysis attack. Although Tor is considered as the most commonly used anonymity system, an attacker can recognize visited web pages over Tor by exploiting some features inferred from observed traffic (e.g., packets sizes, directions, timings, etc). A recent and popular traffic analysis attack is called website fingerprinting. Existing website fingerprinting attacks identify visited web pages on Tor using just a *single web browser namely "Firefox". However, the web users around the world use Tor to protect their privacy over various browsers. In this thesis we setup a website fingerprinting attack on "Tor" to investigate different levels of resistance of the most commonly used browsers against traffic analysis attack based on features inferred from their traffic patterns. To the best of our knowledge, this is the first comparative analysis study involving browsers resistance against website fingerprinting attack based on a deep analysis that reaches the main causes that stand behind those different resistances. The aims of applying web page fingerprinting attack on top web browsers are three folds: First, to expose the possible vulnerabilities in (Tor and browsers) that the attacker may exploit in order to push Tor and browsers makers to empower their products to defeat any possible exploits by attackers. Second, to raise web users awareness for more privacy protection by selecting the browser with the highest privacy protection. Third, the outcomes of our research might be very beneficial to security forces and law enforcement of the governments to highlight what are the web pages their citizens may visit anonymously on different browsers. Our empirical results showed to which extent each browser protects against web page fingerprinting attack as well as exposing the root causes stand behind different resistances of browsers.*

ملخص الرسالة

"مقاومة متصفحات الإنترنت للهجوم عن طريق تحليل البيانات المارة عبرالشبكه العنكبوتية"

الاسم: طاهر علي يحيى الشهاري

القسم: علوم الحاسب الآلي والمعلومات

التخصص: علوم الحاسوب

تاريخ المناقشة: 2014\10\27

تعتبر تقنيات حماية الخصوصية من أهم المجالات الحديثة في أمن المعلومات والتي تركز على حماية الاتصالات الشبكية من الهجمات الاختراقية عن طريق تحليل خصائص البيانات المارة عبر الشبكات. إن الفكرة الرئيسية وراء تقنيات حماية الخصوصية هو إنشاء قناة اتصال مشفره لإخفاء محتويات وعناوين بيانات مستخدمي الإنترنت من المتنصتين المحتملين عبر نطاق الشبكة. يعتبر نظام التعمية تور "Tor" أكثر تقنيات حماية الخصوصية شيوعاً حيث يستخدمه حالياً أكثر من 4,000,000 مستخدم. على الرغم من أن نظام تور يعتبر من أقوى أنظمة التعمية إلا أن بيانات مستخدميه مهدده بإمكانية وجود المتنصتين المجهولين بين نقطتي الاتصال(الزبون والخادم) الذين يمكن أن يكشفوا هوية المواقع التي يتصفحونها من خلال استنتاج بعض خصائص البيانات المارة عبر الشبكة مثل (أحجام رزم البيانات، اتجاهم، التوقيت بينهم، .. إلخ). لقد طبقت الأبحاث السابقة تقنيات لكسر نظام التعمية تور عن طريق التعرف على هوية المواقع التي يتصفحها الضحية من خلال استنتاج خصائص نمط مرور البيانات باستخدام متصفح واحد فقط "فايرفوكس". إن مستخدمي الانترنت حول العالم يستخدمون متصفحات مختلفة لحماية خصوصيتهم من المتنصتين عبر نظام التعمية تور، لذا لقد قمنا في هذه الأطروحة البحثية بكسر نظام التعمية تور للتعرف على هوية المواقع التي يتصفحها الضحية باستخدام أكثر المتصفحات شيوعاً (فايرفوكس، انترنت اكسبلورر، كرووم، سفاري، أوبيرا). إن هذا البحث يتغلب على جوانب القصور في البحوث السابقة والتي استخدمت متصفح ويب واحد "فايرفوكس" من خلال قيامنا بعمل دراسة تحليليه لمدى المقاومة التي يقوم أكثر خمسه متصفحات استخداما حول العالم لحماية البيانات من المتنصتين عبرا لشبكه. بالإضافة إلى ذلك لقد قمنا بدراسة تحليليه عميقه لاستنتاج الأسباب الجذرية للمتصفحات التي تقف خلف مدى المقاومة الذي يقوم بها كل متصفح لحماية خصوصية مستخدميه. تعتبر نتائج مخرجات هذا البحث مفيدة جداً لقوات أمن المعلومات وتطبيق القانون في الحكومات من خلال وضع صوره واضحه عن مدى القوة و الضعف في حماية الخصوصية التي يمتع بها كل متصفح ضد المتنصتين وأيضا مراقبة المستخدمين الذين يستخدمون نظام التعمية تور من تصفح المواقع التي تعتبر ممنوعه في تلك الدول .

CHAPTER 1

INTRODUCTION

The World Wide Web (WWW) makes modern life much more convenient as the people relay massively on the internet in several daily activities, but there are real threats in which their privacy and anonymity might be violated. The privacy and anonymity of web users can be endangered by possible attackers using some tools to eavesdrop their activities. When internet users browse the web, their visited websites destinations are revealed to several routers along the way. External observers such as private and governmental security agencies may passively observe and collect information by monitoring and censoring users' activities on the internet. Therefore, the anonymity systems on the internet are very important to hide the privacy of people who want to surf the web for their critical needs such as sending secret E-mails and making online business. Earlier encryption techniques like secure shell (SSL) have been used for encrypting the contents of the web from being fetched by attackers but the identity of the user can be monitored using some information extracted from packets headers. To hide both the addresses of users and the contents of their traffic, advanced anonymous systems such as Tor [32] and Jap [8] are proposed to protect web users from such threats by allowing them to communicate and share information safely without hurting their privacy. Although Tor is considered as a strong anonymity system, its users can be endangered by local traffic analysis attacks that can be placed between the client and server to uncover the identity of the requested web sites even though the traffic is encrypted. The main target of our research is to investigate the behavior of the most commonly used web browsers in protecting web users against website fingerprinting attack over Tor anonymity system by conducting our proposed attack explained later. To the best of our knowledge, we are the first to push the boundary of knowledge from this angle. Furthermore, the outcomes of our research might be very beneficial to the security forces and law enforcement worldwide.

1.1 Privacy issues in Internet

The rapid advance of technology and growth of Internet raise the concern of users from possible online threats that may endanger their privacy. In order to understand privacy issues perfectly, we need to define the meaning of ”privacy”. Warren and Brandeis [83] defined privacy as the ”right to be let alone”. Roger Clarke suggested that ”Privacy is the interest that individuals have sustaining in a personal space, free from interference by other people and organizations” [24]. However, the privacy of web users is satisfied when the usage, exchange and release of their information can be one hundred percent under control [27]. Unfortunately, this is not the case as data is transmitted on cyberspace and the users do not maintain full control over their information, so the privacy attack can occur. As the individuals join to the Internet, their information may become out of their control with the presence of several threats (e.g. Malwares, traffic analysis attacks, etc). Therefore, a local observer on an ISP or attacker on a WLAN can analyze and track information sent or received to a victim efficiently, inexpensively and unconsciously. The privacy issues that most users are concerned with include [23]:

- The nature of their visited websites which can be revealed using website fingerprinting attack.
- Their E-mail addresses and contact information that can be used for targeted advertisement and other purposes.
- Their personal information that can be misused and exported against users' preference and interest.
- Their Passwords and Credit card information.
- Their locations/origin on Internet space.

These are the most important privacy issues that should raise people awareness to know possible sources that threaten their privacy. Figure 1.1 shows possible adversaries that may carry out such kind of privacy violations which web users should take care of. Therefore, users must take several precautions to protect their privacy against such privacy issues as well as designing containment strategies once their personal information has been collected.

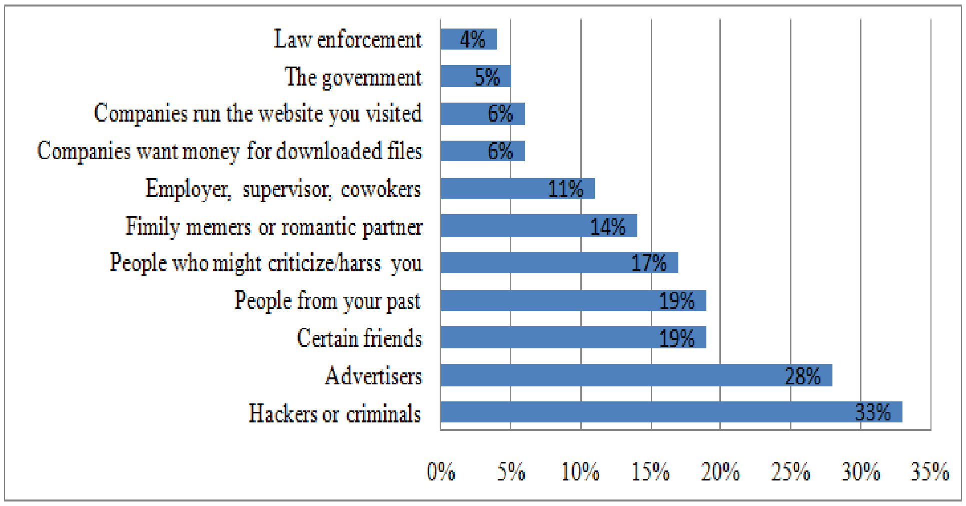


Figure 1.1: A survey conducted by Pew Research Center's internet and american life project omnibus in July 2013, shows possible eavesdroppers that may violate the privacy of web users [66].

To solve privacy issues on the internet, multiple privacy enhancing technologies have been proposed (e.g. SSL, IPSec, SSH, Tor, etc) so theses encryption mechanisms hide the content of transferred data but there are still some valuable information that can be exploited be attackers such as size, direction, order, and timing of the transmitted packets between client and server. For example, in website fingerprinting attack the attackers use this information to identify a web page that a victim visits. Several individuals such as journalists, human rights workers, the military, and ordinary citizens, employ anonymity systems to protect their identities on the Internet. Internet users' privacy issues have been highlighted in many published works such as [30], [17], [19], [55], and [75] .

1.2 Anonymity protocols

In this section we define the meaning of "anonymity" in term of our research and what are the most commonly used anonymity systems. Pfitzmann et al. defined anonymity as "the state of being not identifiable within a set of subjects, the anonymity set." [63]. They meant by the anonymity set are the set of all possible players in the system such as the sender, known as the sender anonymity set, the recipient, the recipient anonymity set, of a specific message. Because of the high increase of traffic analysis attacks, several anonymity systems have been developed in order to enhance the security and privacy of users over the internet. These anonymity tools provide privacy protection by encrypting the transmitted data ranging from simple to complicated method. For example, Chaum [18] is the first who proposed a system that provides a level of anonymity by establishing a connection that mixes certain traffic with other traffic connections. Later, several systems have been proposed which employ a wide number of sophisticated techniques that are summarized in the following points:

- Setting up various proxies through multiple number of systems worldwide in order to hide the source of the connection from its destination.
- The packets flows are mixed and reordered.
- The packets sizes are padded into a fixed length.
- The packets flow rates are controlled with any batching strategy or timer.
- The data packets are covered using some strategies (e.g. tunnels, onion

routing, etc) [25].

Different anonymity systems employ the aforementioned techniques in order to make it very difficult for attackers to trace and analyze the traffic. Some systems use timing techniques to modify the timing of packets flows which has a great effect on the system. The systems that are based on timing techniques are classified into two classes. First, High-latency systems ,which are much better at protecting possible attacks, are based on timings such as Mixminion [28] and Mixmaster [74]. These systems employ strategies such as mixing, reordering and patching to defend against traffic analysis attacks that are based on packet timings "delays" [53]. The anonymous systems that work based on timing strategies are not as widely used due to the extra delays that they add in data transmission. Second, Low-latency systems that have been used suitably for web browsing protocols such as HTTP and interactive protocols such as SSH because they do not disrupt the timing of packets during the communication. The systems that underlie this category are The Onion Routing (Tor) [32], Java Anon Proxy (JAP) [8], and Invisible Internet Protocol (I2P) [48]. In our experiments we select Tor anonymous system to evaluate different levels of browsers resistances as it is the most commonly used system of desirable Low-latency systems. Anonymity systems can be utilized by users in both an illegal and legal sides. For example, they can be misused in misappropriation of funds and terrorist actions so researches for breaking anonymous systems like ours can be much more helpful for governments to track such kinds of criminal processes. On the other hand, a lot of people employ anonymity systems in multiple legal fields such as e-banking, e-voting, e-commerce, and e-auction, etc. However, several anonymous systems [59], [99], [82], [31], [3], and [50] with different features and encryption mechanisms have been used by web users to protect their privacy from real threats that appeared in last decade.

1.3 Traffic Analysis Attacks on Anonymity Protocols

Traffic analysis term refers to the process of profiling and monitoring network traffic in order to identify the nature and behavior of parties that generate certain traffic pattern such as web browser (the client) and website (the server). Traffic analysis can be implemented equally in both encrypted and non-encrypted traffic and it is based on the observation of the traffic generated during the communication. It is easy for possible eavesdroppers to show the contents of transmitted data by capturing transmitted packets using free available capturing tool such as Wireshark, tcpdump, etc. Then, many features can be deduced from captured packets, so a potential adversary exploits those features to track and correlate a user's browsing activities such as who is talking (the source), to whom they are talking (the destination), type of visited sites, etc. Consequently, it is very important to encrypt sensitive information from being observed by eavesdroppers when it is transmitted through the internet.

However, the anonymous systems do not guarantee protecting the privacy of users completely because the eavesdroppers can exploit some information (packet sizes, timing, ordering, etc) inferred from encrypted traffic transmitted between clients and remote servers. As a result, several traffic analysis attacks can be conducted for illegal purposes (e.g., infering the identity of visited website by a victim, fingerprinting browsers to exploit their vulnerabilities, etc). On the other hand, researchers implement various traffic analysis attacks to improve the security of Privacy Enhancing Technologies by exploring their vulnerabilities in order to push anonymous systems vendors to add novel defences against such attacks.

In this section we explore the most well-known examples of traffic analysis attacks on encrypted systems. For example, Song et al. established profiles of typing characteristics of users in order to guess the keys that the user types under Secure Shell (SSH) session [70]. They used Hidden Markov model to predict the sequences of pressed keys since SSH protocol transmits each pressed key into its own separate packet across the network. They were able to measure the delays between packets then matching them to pre-determined delays of a combination of certain keys. The aim of this attack is to enhance the probability of breaking passwords over SSH protocol. Moreover, Hints [41] used traffic analysis attack to identify the website that the user may visit based of unique signature of loaded resources of a certain HTTPS website (e.g. HTML document, CSS files, JS, images, etc). He created profiles of timing signatures to detect visited websites over Security Shell Layer (SSL). More robust website fingerprinting attacks on a broader variety of encrypted systems are proposed by Bissias et al [11], Liberatore et al [49], Herrmann et al [39], Shi and Matsuura [69], Cai et al [15], and most recently Wang et al. [81].

1.4 Website Fingerprinting

Hints [41] is the first who described the traffic analysis for identifying visited websites as a term ”Web site Fingerprinting”. Website fingerprinting is one kind of traffic analysis attacks which enables the adversary to infer a visited web page that the victim may visit for violating his privacy even if the victim uses certain anonymous system like Tor [65]. It is a process of footprint information of a target web page based on inferred features from its traffic pattern.

When a user visits a certain web page, the HTML document of that page will be fetched with its referenced contents (e.g. CSSs, JSs, Images, Text, etc). Each fetched content has specific characteristics(size, order, direction, delay). Encrypting protocols (e.g. encrypting tunnels, SSL, Tor) encrypt the contents of transmitted information but they do not effectively encrypt some features such as packets sizes, directions, timing, etc[14]. Therefore, it is possible for an eavesdropper to monitor/sniff the network traffic of a victim and profile fingerprint about web page contents based on (order, direction, timing, and sizes) of the packets used to load a target web page. Thus, the set of extracted information for a given web page comprises a unique fingerprint for that page. Using such fingerprint method, a visited website can be uniquely identifiable even if the connection is encrypted using any anonymous system such as SSL, SSH, Tor, etc. For example, Website fingerprinting process on encrypted traffic can be used by governments to censor and block some web pages that they mark them as illegal websites. Using the previously mentioned web page features, a government can generate fingerprints for all banned websites. Then, they censor and sniff all traffic that matches all previously recorded fingerprints of banned list of websites. Furthermore, they can avoid monitoring a huge amount of traffic and focus only https traffic that comprises a very small portion of Internet traffic. Some websites are changed continuously (e.g. news websites) so the government should generate new fingerprints frequently to cope such changes [41].

Existing works in website fingerprinting attacks [11], [61], [11], [39], [15], and [81] show that this kind of traffic analysis attack is possible against several anonymous systems like SSH, IPSec tunnels, JAP, and Tor. Consequently, vendors of these systems tray to defeat these kind of attacks by using several techniques and tricks that are specified in Section 1.2. We select Tor as it is the most popular anonymous system in use today "currently used by around 500,000 daily clients and carrying 2000 MB of data per second"[64]. More details about website fingerprinting attack over Tor are explored in Chapter 2. In our experiments we set up a website fingerprinting attack using most commonly used browsers on real user who defends himself using Tor for the sake of investigating to which extent each browser defends against such website fingerprinting attack. Our research outperforms existing ones in that it breaks Tor anonymous system using most commonly used browsers while all existing works without exception used only Firefox browser.

1.5 Web Browsers

Today, more and more services and a variety of information contents (e.g. HTML, images, video, etc) become available on the Internet ”Web servers”. These contents are accessed and retrieved using web browsers such as Chrome, FF, IE, etc. Web browser acts as an interface between web user and web server, so the need for fast and secure browsing experience is more important to satisfy the experience of end users. Recently, the market of web browsers becomes highly competitive to fulfill the World Wide Web (WWW) demands of people securely and privately.

Internet browser is the client-side application in internet communication and its main function is to fetch the requested contents from web server and display them on browser’s window. The web contents are fetched in the form of requests and responses between web browsers and web servers by implementing Hypertext Transfer Protocol (HTTP) and its secure version (HTTPS). Before the evolution of the web, the web pages were very simple HTML pages containing simple contents (e.g. text, input boxes, and buttons) [10]. Currently, web pages contain multimedia contents (e.g. JSs, CSSs, flashes, audio, etc) so many people use different browsers to perform many tasks (e.g. access email, buy products, do research, etc). Therefore, web browsers are essential part of people daily live so they are built with a lot of functions to perform those tasks and protect user from any malicious content residing on the World Wide Web. The details behind different browsers' structure and features are described in Chapter 3. Our focus in this study is to evaluate different resistance levels of top five web browsers against website fingerprinting attack as they have relative importance among applications today. In this research we select the most commonly used web browsers namely Chrome, Firefox, Internet explorer, Safari, and Opera according to market share statistics of browsers [?] as it is shown in the distribution of pie chart in Figure 1.2.

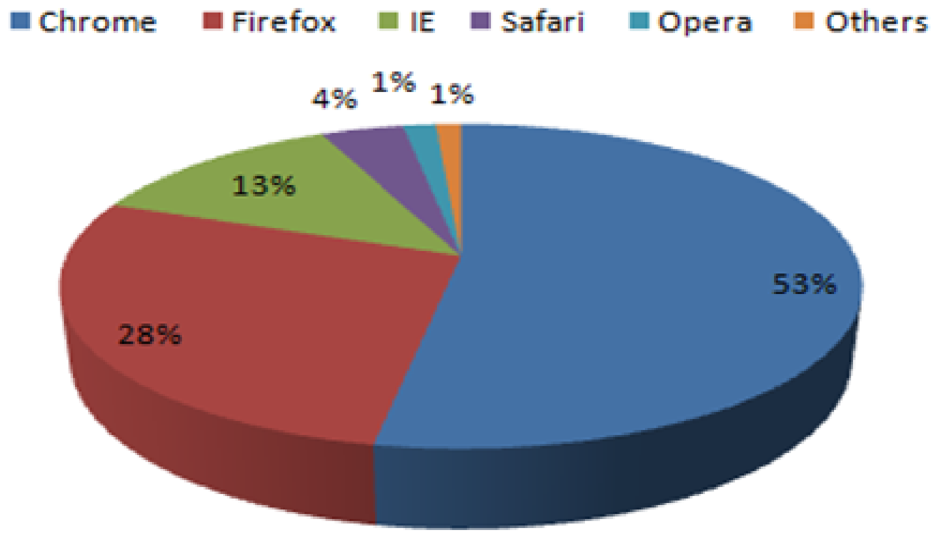


Figure 1.2: Browsers' Statistics and Trends On May, 2013. Source: http://gs.statcounter.com/

1.6 Objectives of the research

Recent attacks on data streamed over Tor identify the websites that the victim may visit just only under Firefox web browser. The question is why we do not apply traffic analysis attack on the most commonly used web browsers to see their resistances against traffic analysis attack as a lot of users browse the internet anonymously using different web browsers? To answer our research question, we build a model to test various resistance levels of popular web browsers by making a lot of experiments on tested browsers (e.g., various web technologies, Java Script APIs, parallel downloads, etc) in order to reach into the root causes that make each browser behave differently. Our approach highlights a clear picture of the resistance of each browser against website fingerprinting attack outperforming the previous approaches that share the same shortcoming which is a single web browser (Firefox).

1.7 Methodology

Our methodology in this research is to study, design, and implement a website fingerprinting attack for evaluating the resistance behavior of web browsers on Tor anonymity system. It consists of the following research phases:

1. Study and survey the existing literature of website fingerprinting approaches.

2. Carry out a technical survey on the five commonly used web browsers (Chrome, FF, IE, Safari, and Opera) to clarify their differences in terms of layout engines, Java Script engines, and HTML/CSS features, etc.

3. Set up sniffing modules/scripts for capturing web pages traffic using a strong network sniffing tool "tshark" ,the command line version of wireshark traffic

analyzer.

4. Parsing and filtering captured traffic to isolate Tor packets and browsers' relevant packets from the rest of the traffic.

5. Construct websites fingerprints based on certain features (packets sizes and directions) which serve as an input to a classification phase.

6. Carry out a comparative study of the top five web browsers to assess the variations of their resistances to website fingerprinting attack in terms of recognition accuracy of each browser.

1.8 Overview of contributions

The contribution of this thesis is to fingerprint websites being accessed over Tor using the five most popular web browsers. The main contributions of our research are as follow:

1. Carry out a deep analysis of the most commonly used web browsers to identify key differences and similarities in the rendering engines, the fetching schemes, etc.

2. Investigate website fingerprinting attack on Tor anonymity system taking into consideration the most relevant features that improve the accuracy of website fingerprinting which can be used also for ”web browser fingerprinting”.

3. Carry out a comparative analysis with respect to the resistance of the most commonly used web browsers to traffic analysis attacks on Tor.

4. Publish the outcomes of our research in reputable journals and conferences as well as patents.

1.9 Thesis organization

In Chapter 1 we outline the motivation and relevant background behind the topics discussed in this thesis. Section 1.1 explores the rapidly growing field of privacy issues on the internet and shows its importance as a field of study. Section 1.2 outlines the importance of anonymity systems and their various encryption techniques as well as their types and uses. Section 1.3 displays a brief introduction of traffic analysis attack on anonymity systems and shows the most important examples of traffic analysis attacks for improving the effectiveness of end users' privacy under various anonymity systems. A brief overview of the most commonly used browsers and their trends are explored in Section 1.4. Chapter 3 is dedicated for more details behind web browsers as they are the main topic of this research. Furthermore, Section 1.5 describes the main objectives of this research and the primary questions that we seek to answer. Section 1.6 outlines the research methodology that we followed to reach to our objectives. Section 1.7 explores our main contributions. The sequence organization of this thesis is highlighted in Section 1.8.

Chapter 2 demonstrates many concepts behind Tor anonymity system. An overview of what is Tor and its implementation fields as well as its main security features and usage statistics are explored in Section 2.1. The technique that Tor uses to establish Tor circuit for anonymous communication is explored in Section 2.2 such as how it selects the relays, associates encryption keys, etc. The summary of the existing published researches are explored in Section 2.3.

More functional details behind web browsers are illustrated in Chapter 3. An overview of typical architecture of browsers and how their internal components work are explained in Section 3.1. The core components of browsers that affect their rendering behavior are demonstrated in Section 3.2. The main task of browsers rendering/layout engines and their types are clarified in 3.2.1 Subsection. Web browsers Java Script engine/interpreters and their functionalities are explained in 3.2.2 Subsection. In 3.2.3 Subsection, the JS and CSS web contents, and web technologies that have an impact on web browsers traffic patterns are explained.

Chapter 4 demonstrates our website fingerprinting attack to investigate the various resistance levels of browsers against website fingerprinting attack. The detailed description of the experimental platform is presented in Section 4.1 followed by a presentation of the data collection and preprocessing phases that are demonstrated in Section 4.2. In Section 4.3 we explained the applied approach that we implement to validate the various resistances of browsers against website fingerprinting attack over Tor.

Chapter 5 evaluates the root causes behind various resistances of tested browsers. The comparison of browsers' web technologies is evaluated in Section 5.1. The results of the factors that affect the traffic patterns of browsers (JavaScript, Third-party domains ,parallel download and time optimization) are evaluated in 5.2, 5.2.1, 5.2.2, 5.2.3, and 5.2.4 Subsections respectively. Sections 5.3 and its related subsections we summarized the discussion of some web issues and dynamism of network condition that add a noisy on traffic patterns and reduce the privacy protections introduced by browsers. Some recommendations that raise the resistances of browsers against website fingerprinting attack are pointed in 5.3.3 Subsection.

Chapter 7 concludes the thesis by summarizing the findings and some hot research points that can be guided as hints for further future works.

CHAPTER 2

TRAFFIC ANALYSIS ON TOR ANONYMITY SYSTEM

Many governments worldwide consider viewing certain web pages by their citizens as an illegal action. For instance, in China more than 2600 websites were blocked as illegal sites and they put under the policy internet censorship of the country [95]. Furthermore, internet censorship in Iran has been increased with 50 percent of the top 500 visited websites (e.g. Facebook, Google Plus, Twitter, etc) [92]. These restrictions trigger people in such countries to use different anonymity systems (e.g. Tor, JAP, SSL, etc) to bypass online censorship and surveillance systems and browse freely. In our research we select Tor to evaluate the resistance of browsers against fingerprinting attack as it is the most commonly used anonymity system. Moreover, there are relatively a few research on website fingerprinting attacks on Tor.

2.1 Overview of Tor system

Tor is the anonymity system that was developed primarily by U.S. Navy to protect the communications of the US government. Later it has been donated to the open source community to have clients of more than 126 countries around the world [5]. It is an overlay network of virtual tunnels that is built via layered encryption mode called Onion Routing. It was built to anonymize TCP-Based services such as web browsing, instant messages and secure shell. Tor is an open network that helps users defend against a form of network surveillance that threaten their privacy, confidential business activities and relationships. Based on the statistics in [64], Tor recently is the largest deployed anonymous communication network in the world. From its first release in 2004, there are many attacks that illustrate several of its vulnerabilities. These attacks push Tor's designers to enhance its anonymity protection with several modifications and improvements. The main recognizable features of Tor protocol system are as follow:

- Renewal of circuits: Tor has perfect forward strategy in which it builds a circuit of three relays in the first of each negotiation session and this circuit is changed periodically.

- Reduce the latency: in which the TCP stream are multiplex in a circuit to which help in reduce latency.

- Reliability and anonymity: Tor makes it difficult to modify transmitted data and describe its relays because the transactions between relays are

TLS based.

- Preventing end-to-end attack: Tor has leak pipe circuit topology feature in which it is responsible for directing traffic to nodes pathway down the circuit, meaning that the traffic can exit the circuit at the middle, thereby prevent end-to-end attack.

In its first deployment, Tor has only 250 relays with just about 50,000 users, however at present there are over 4000 relays and around 4000000 daily users all around the world as it is shown in the following Figures 2.1 and 2.2.

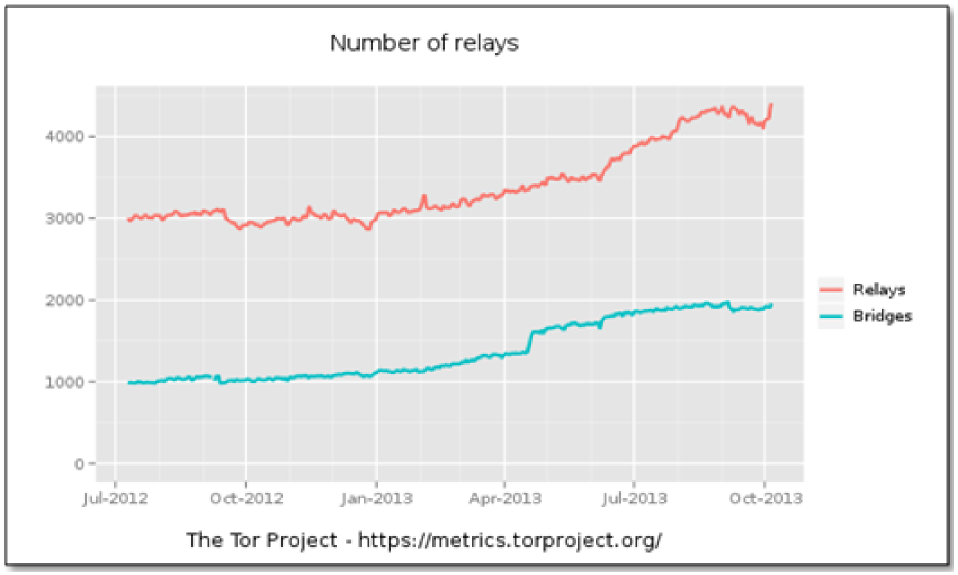


Figure 2.1: Recent statistics of the number of Tor users around the world. Source: http://www.torproject.org/about/overview.html.en.

Tor is used for different purposes; the military use Tor to hide their location and protect military operations, normal People use Tor to protect their privacy such as protecting their children from being stolen by corrupt marketers and thieves, bloggers use Tor to avoid being sued or fired for saying completely legal

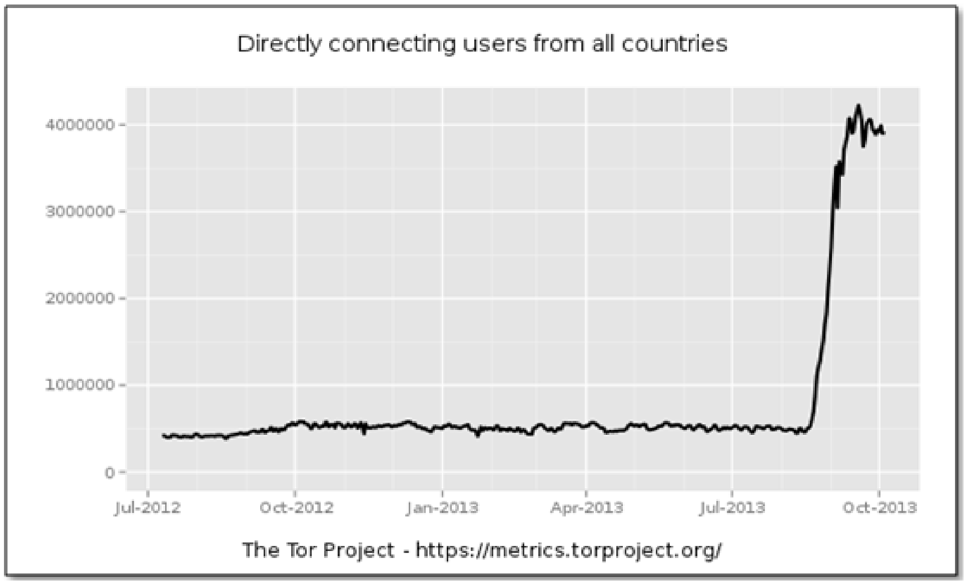


Figure 2.2: Recent trend of the number of Tor relays around the world. Source: http://www.torproject.org/about/overview.html.en.

things online, journalists use Tor to investigate state propaganda and opposing viewpoints, activists use Tor to anonymously report abuses from dangerous zones in the world, law enforcement officers use Tor for carrying undetected surveillance on questionable websites [28],[67],[72], and [73]. Furthermore, Tor has been used by political opponents in non free countries (e.g. Iran, China, etc) to access banned web pages (e.g. Facebook, Twitter, etc). Therefore, Tor encrypts the origin/place and contents of transmitted data from being exposed by possible eavesdroppers.

2.2 How does Tor work?

The main goal of Tor anonymity system is to make it very difficult for an attacker at the source point to determine the destination of a request, or an attacker at the destination point to determine the source of a request. This can be done by three components that play important roles in establishing Tor's circuit session which are the client, the server, and three Tor relays in between [65]. Tor client establishes its circuit by negotiating an individual shared secret with each node of the three circuit relays: entry/guard relay, middle relay, and exit relay as it is demonstrated in Figure 2.3.

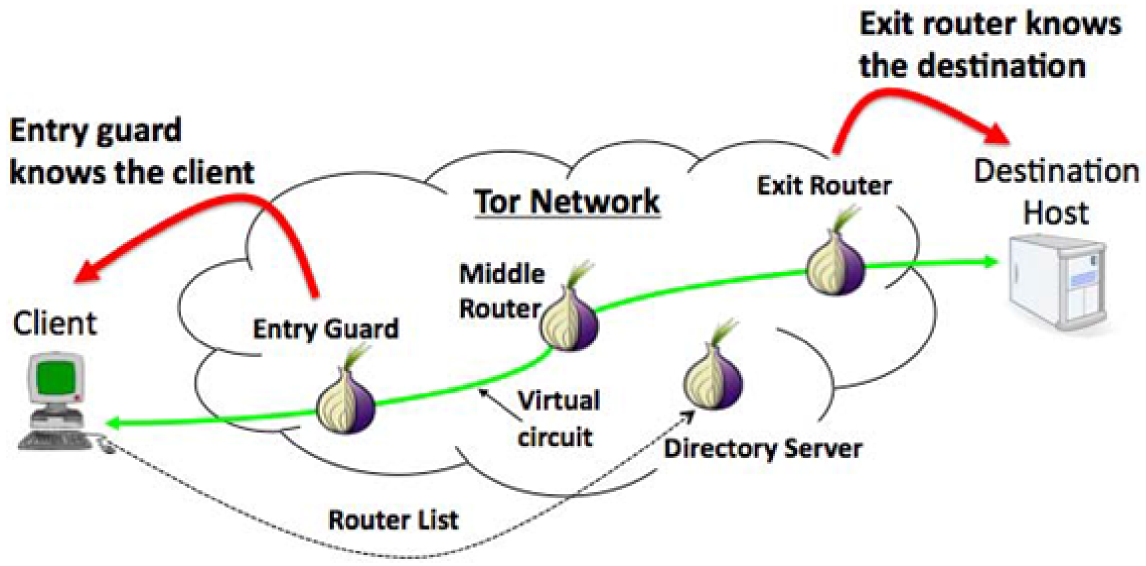


Figure 2.3: Tor circuit components: the client node, the destination node, and the three relays in between. Source: http://www.torproject.org/about/overview.html.en.

The transmission of the traffic is encrypted using three shared secret keys that are negotiated with the three relays of established Tor's circuit. Each relay in Tor network knows only its successor and predecessor, so this technique of Tor provides strong resistance against data analysis attacks of man-in-the-middle [65]. When the client establishes an anonymous connection with the server, the proxy of client starts to select three relays randomly from list of Tor relays residing in directory server as it is shown in Figure 2.4. The relays of Tor selected based on certain algorithm that depends on various Tor relays statistics that are distributed by the directory server such as some client history and preferences [1].

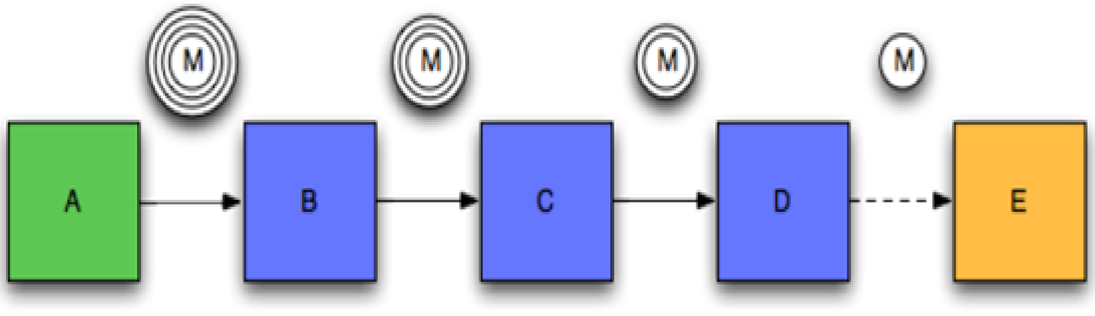
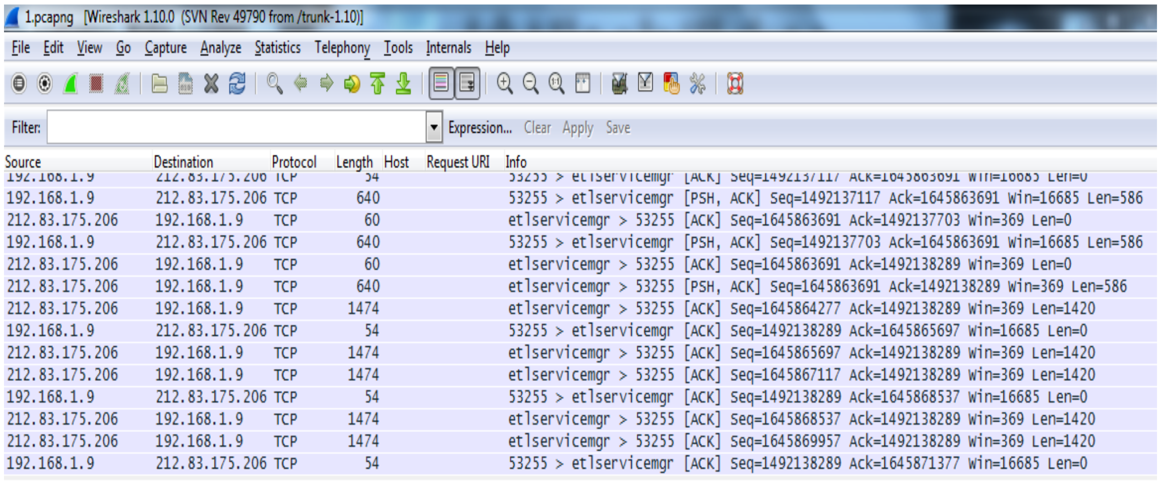


Figure 2.4: The Onion Routing technique of Tor in which the client sends a message to a server wrapped with encrypted layers[65].

The proxy of Tor client establishes session key with the first Entry relay after selecting and meeting the policy of Exit relay. Then, the tunnel is extended from the entry relay incrementally one node at a time up to the exit node [65]. After establishing Tor network, the client communicates with the server anonymously. Throughout the Tor anonymity network, the entry/guard relay knows just only the next relay in the path and this is applied for the next relay up to the exit relay. After establishing the Tor anonymity network, many virtual circuits are created periodically so transmitted traffic is routed and multiplexed to the destination. During the connection the transmitted message is sent from relay to relay along the circuit so in each relay the encryption layer of that relay/node is puled off until the original message reach to exit relay. The origin of transmitted message appears at the Tor exit relay so it will be forwarded to the distinction [94]. The middle relay knows that the message is forwarded to the exit relay, but cannot say who is sending the message.

Tor client routes incoming and outgoing traffic through a shifting and multi-hop circuit of relays. The main goal of designing such system is to make the job of eavesdropping adversary much more challenging by anonymizing who is communicating with whom and exchanged traffic. Tor system works only at the Transmission Control Protocol (TCP) stream level and it can be used by any application that supports SOCKS such as web browsers. Therefore, the web browsers can be configured to direct their traffic through Tor SOCKS interface [65, 94]. A strongest feature of Tor is that it masks its traffic to look like HTTPS normal traffic which makes Tor traffic more difficult to be identified as it is shown in Figure 2.5.


1.pcapng [Wireshark 1.10.0 (SVN Rev 49790 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
Source          Destination     Protocol Length Host Request URI Info
192.168.1.9     212.83.175.206 TCP      54                    53255 > etlservicemgr [ACK] Seq=1492137117 Ack=1645863691 Win=16685 Len=0
192.168.1.9     212.83.175.206 TCP     640                    53255 > etlservicemgr [PSH, ACK] Seq=1492137117 Ack=1645863691 Win=16685 Len=586
212.83.175.206  192.168.1.9    TCP      60                    etlservicemgr > 53255 [ACK] Seq=1645863691 Ack=1492137703 Win=369 Len=0
192.168.1.9     212.83.175.206 TCP     640                    53255 > etlservicemgr [PSH, ACK] Seq=1492137703 Ack=1645863691 Win=16685 Len=586
212.83.175.206  192.168.1.9    TCP      60                    etlservicemgr > 53255 [ACK] Seq=1645863691 Ack=1492138289 Win=369 Len=0
212.83.175.206  192.168.1.9    TCP     640                    etlservicemgr > 53255 [PSH, ACK] Seq=1645863691 Ack=1492138289 Win=369 Len=586
212.83.175.206  192.168.1.9    TCP    1474                    etlservicemgr > 53255 [ACK] Seq=1645864277 Ack=1492138289 Win=369 Len=1420
192.168.1.9     212.83.175.206 TCP      54                    53255 > etlservicemgr [ACK] Seq=1492138289 Ack=1645865697 Win=16685 Len=0
212.83.175.206  192.168.1.9    TCP    1474                    etlservicemgr > 53255 [ACK] Seq=1645865697 Ack=1492138289 Win=369 Len=1420
212.83.175.206  192.168.1.9    TCP    1474                    etlservicemgr > 53255 [ACK] Seq=1645867117 Ack=1492138289 Win=369 Len=1420
192.168.1.9     212.83.175.206 TCP      54                    53255 > etlservicemgr [ACK] Seq=1492138289 Ack=1645868537 Win=16685 Len=0
212.83.175.206  192.168.1.9    TCP    1474                    etlservicemgr > 53255 [ACK] Seq=1645868537 Ack=1492138289 Win=369 Len=1420
212.83.175.206  192.168.1.9    TCP    1474                    etlservicemgr > 53255 [ACK] Seq=1645869957 Ack=1492138289 Win=369 Len=1420
192.168.1.9     212.83.175.206 TCP      54                    53255 > etlservicemgr [ACK] Seq=1492138289 Ack=1645871377 Win=16685 Len=0
```

Figure 2.5: Tor session is decoded as it was normal HTTPS session.

Tor often establishes a connection over TCP 443 which follows spec (RFC 2246) of Transport Layer Security (TLS) protocol so most packets inspectors fail at identifying Tor traffic [42]. In our experiments we successfully filter and inspect Tor traffic using several statical analysis methods as explained later in empirical evaluation chapter. In this chapter did not go deep into Tor anonymity system as it is not the main focus of our research so more details about Tor design are given in [32].

2.3 Website fingerpinting attacks on Tor

There are few proposed papers in website fingerprinting attack over Tor all of them are relatively recent. The early techniques were focusing on analyzing the encrypted HTTP traffic by extracting certain features from traffic pattern for training traffic instances. Then, a classification mechanism is used to identify testing traffic instances. In this section we outline the existing approaches in website fingerprinting attacks.

The first author who refers to the process of identifying websites under encrypted connection as a term of ”fingerprinting” is Hintz [41]. He conducted a simple website fingerprinting attack under encrypted traffic based on features extracted from website contents such as sizes and separate TCP connections. His experiments are conducted on HTTP/1.0 version where each web content (e.g. image, JS, text, etc) is fetched using a separate TCP connection. The results show that his approach detects only 5 websites with an accuracy rate between 45 and 75 %. Later this approach becomes invalid with the presence of HTTP/1.1 version where the feature using TCP connections does not hold anymore.

Since Tor deployment in the late of 2003, few techniques have been proposed for website fingerprinting attack on Tor protocol. Liberatore et al. in [49], proposed two techniques for identifying the source of encrypted HTTP connections.

They used only one feature for their experiments which is the packets sizes of transmitted data under the cover of encrypted OpenSSH tunnels. Data mining techniques like Jaccard's coefficient and Naive Bayes (NB) classifiers are used to classify the similarities between captured traffic and predefined fingerprints of websites. The results of their experiments show that if IP packets are padded and frequencies of packet lengths are considered, the NB classifier is more robust than Jaccard's classifier. They claim that their methods are quite effective in website fingerprinting on a simple SSH tunnel with an accuracy of about 70 % in both methods.

In [39] Herrmann et al. identify websites under popular encryption methods using a text mining technique. They used Multinomial Naive Bayes (MNB) classifier for training based on the frequency distributions of IP packet lengths. They optimized their classifier by applying a set of text mining transformations so they achieve a higher accuracy than previous work under comparable conditions. Their experiments show an excellent accuracy of 96% against single-hop encryption systems (e.g. SSL, OpenSSH, etc), while they got less accuracy on multi-hop systems (e.g. Tor and JAP) with an accuracy of 20% on JAP and 2.96% on Tor. This gives a clear indication that website fingerprinting on Tor is more challenging than other encrypting systems.

Shi et al. [69] proposed a novel method for website fingerprinting attack by analyzing the traffic of a victim under Tor anonymity system. They divide both the incoming and outgoing packets into several intervals and convert these intervals into vectors. The similarities between observed vectors and predefined fingerprints are calculated by a given formula. The practical and theoretical evaluations of their results show that their method is an effective way for degrading the anonymity of users under Tor.

Panchenko et al. [61] came up with a website fingerprinting attack on Tor and JAP anonymity systems using Support Vector Machine (SVM) classifier. They represented a traffic trace as a sequence of packet lengths where input and output packets are distinguished using negative and positive values. In addition, they injected some features in these sequences to raise the accuracy of the classification such as size markers (whenever flow direction changes, insert the size of packets in the interval), number markers (number of packets in every interval), total transmitted bytes, etc. They used Weka tool [85] to fine-tune the SVM parameters. They evaluated their method using Closed-world and Open-world scenarios. In closed-world scenario they conducted their experiments on the same data set of Federrath et al. [39] with 775 websites by estimating the accuracy using a ten-fold cross validation. In open-world scenario, 5000 websites have been chosen randomly among the top one million websites listed by Alexa [2]. Their experimental results show that their approach improves the websites recognition rates from 3% to 55% and in JAP and from 20% to 80% in Tor.

Cai et al. [15] proposed a new approach for achieving the highest accuracy than previous works. They implemented string alignment using Damerau-Levenshtein distance algorithm to compare the previously made fingerprints with the observed traffic based on the features of packets' sizes and directions. They identified web pages with an accuracy of 87.3% in closed-world model. They also classified websites instead of individual web pages using Hidden Markov Models (HMMs). They claim that the recent defenses against traffic analysis over Tor are not likely to be successful.

The most recent contribution was by Wang and Goldberg [81] where they proposed new techniques for website fingerprinting attack. They enhanced the accuracy of Website fingerprinting by interpreting Tor data cells as units instead of TCP/IP packet sizes and removing Tor SENDMEs cells that provide no useful data in order to reduce the noise. They compared the similarity between the predefined fingerprint instances and observed traffic instances using new optimal string alignment distance metrics (OSAD) with limited computation resources. The results of their closed-world experiments show that their methods achieve better accuracy rate than previous works with 91%.

In our approach we conducted website fingerprinting attack using the most commonly used web browsers so they differ in their accuracy rate as they have different aspects (e.g. different rendering engines, various Java Script engines, different HTML and APIs features). Therefore, we strongly believe that these differences between tested browsers will affect the shape of their encrypted traffic patterns as well as the accuracy rate of website fingerprinting attack on each browser. To the best of our knowledge, none of the existing works have taken into consideration the fact that Tor clients may use various web browsers over

Tor, so that they share the same limitation in which they conducted the attack on a single web browser (Firefox) while in ours we take into consideration the top five web browsers. Not only that but we also investigated the root causes behind different resistance levels of browsers while others are not.

CHAPTER 3

COMMONLY USED WEB BROWSERS

The web browser becomes a crucial piece of software in modern devices from mobile phones to desktop computers. It is a client-side application and is considered as the window for internet users to the World Wide Web (WWW), so this requires a good performance, high quality, and solid reliability. The first graphical web browser that makes the WWW accessible to everyone was Mosaic [6] which was released in 1993. Later several web browsers (e.g. Internet Explorer, Firefox, Opera, Safari, Chrome, etc) join the race making what is commonly referred as (browsers war) and launch an information explosion that continues to this moment. With time, these browsers have become increasingly complex over the years, not only to parse HTML and plain text, but also to render flash, images, videos and other complex file formats and protocols. However, increasing complexity of modern web browsers have brought too many security vulnerabilities which attract Malware authors and criminals to exploit these vulnerabilities to compromise victims' systems [57]. Currently, web users can use several web browsers to browse the web expecting that browsing the web has consistent behavior across different browsers. Unfortunately, this is often not the case so browsing the web have differences which may range from minor differences to crucial functional flows [56]. The different behaviors of browsers are due to many reasons such as various web standards, accessibility tools, additional features and performance optimizing. The fact that there is no published work that figure out different levels of browsers resistances against traffic analysis attack. This indicates that the problem is more challenging and complex due to the dynamic behavior of web environment which is explained later. The main objective of this thesis is to investigate different aspects of browsers' behaviors in fetching different web objects (e.g HTML,JS , CSSs, image/x, etc) to find out to which extent each browser protects against website fingerprinting attack providing that each browser allows configuring Tor socket proxy. Therefore, in this chapter we give a detailed overview of the factors that affect the behavior of tested browsers (e.g. rendering/layout engines, Java script engines, pipelining/parallel downloads, HTML and CSSs standards) in order to reach the main differences that affect their various rates of privacy protection.

3.1 How web browsers work?

The main functionality of web browsers is to request web resources from web servers and display them in their windows. The formats of requested resources are usually HTML, image/x, JS, CSSs, etc. The main objects that the browser uses to interpret and display other web contents are the Hyper Text Markup Language (HTML) code and Cascading Style Sheets (CSSs) [71]. The HTML and CSS are maintained by the web standards organization called World Wide Web Consortium (W3C) [80]. The latest versions for these two important web objects are HTML 5 and CSSs 3 [34]. More details about HTML and CSSs are explained in subsequent sections. The core differences behind web browsers are coming up from their various rendering and Java Script engines which are reflected semantically in various supports of browsers to different web technologies. The architecture of browsers gives a clear view of how the main components of the browsers work and how they interact with each other in order to display the requested objects on their windows. A typical architecture of modern web browsers is presented in Figure 3.1. It shows the main components that are used by most common browsers.

Modern web browsers are fairly sophisticated applications comprised from several components. The following items are the browsers components with their main function:

- User interface (UI): It includes the main components of navigation controls like address bar, bookmarking menu, etc.

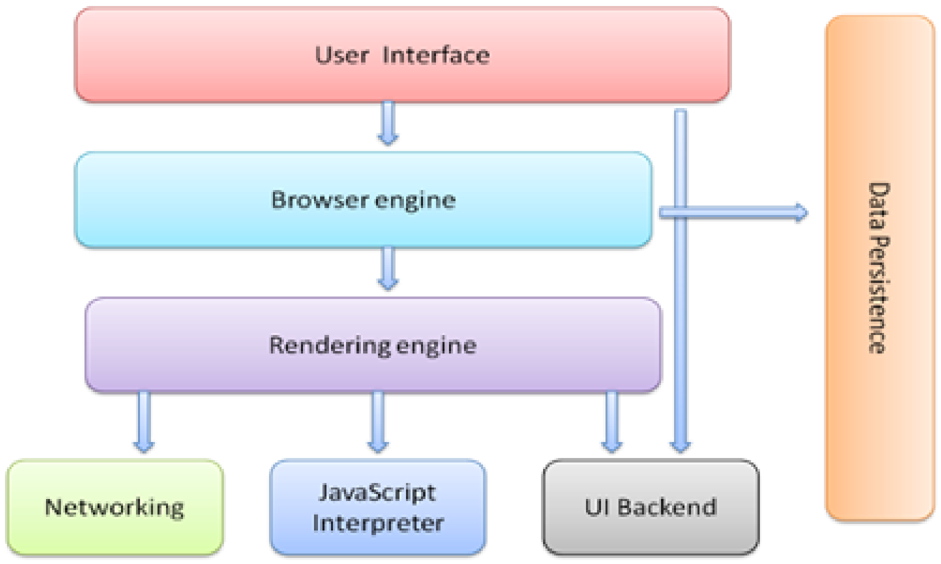


Figure 3.1: Browsers' main components [34].

- Browser engine: It represents the internal interface of browsers to manipulate and query the rendering engine. It is the middle component between the UI and the rendering engine.

- Rendering/Layout engine: The main function of this component is to parse HTML code, request web objects, and show received contents on browser window. It displays HTML, XML and images by default but in some browsers other types of data requires certain types of plug-ins like PDF viewer plug-in that displays PDF format.

- Networking: This is an independent interface that performs the network calls like web objects requests.

- User interface (UI) backend: The main function of this unit is to draw basic widgets windows like combo boxes. It is a generic interface that uses the operating system user interface methods.

- Java Script interpreter: Which is used to parse and execute JavaScript codes.

- Data storage/ Data persistence: This unit stores all sorts of requested data on the hard disk like cookies.

3.2 Core differences between web browsers

The core reasons behind different behaviors of web browsers underlie in three main sources. The first and most important one is the layout/rendering engine so it is the primary source of browsers differences [20]. Its main functions are to render and display web page objects in browser window by combining the structural information of HTML and style information of CSSs. Each browser has its own layout engine so it maintains the Document Object Model (DOM) that represents fetched web page contents in a tree representation and manages the pipelining and parallel downloads of website contents. DOM is a language-neutral platform that allows scripts and programs to dynamically access and update its tree contents so the result of this process is reflected to the presented web page contents. For more details behind Document Object Model standard definition format see [79]. The dynamism of web page elements comes from manipulating and modifying DOM tree nodes by Java Script codes. Therefore, the same HTML/DOM and CSS for the same web page can produce different view and traces in different browsers based on their various JS APIs supports. More details behind layout engines are presented in the subsequent section. The second source of browsers differences comes from Java Script engine. Each browser has its own JS engine which results in differences in browsers behaviors [20]. Interactive web pages are enriched with many Java Script programs that create dynamic web page contents. The details of Java Script APIs and their impacts on browser behaviors are explained later. The third source of a variety of browsers behavior is the user interaction (e.g., mouse click, drag and drop, mouse hover, etc). User actions perform changes in the DOM based on the DOM-APIs of the browsers which have different supports across browsers. In the next sections we explain the details behind the main sources of different traffic patterns of browsers neglecting the source caused by user interaction as the scope of our research is to test the behavior of the browser itself.

3.2.1 Web browsers' rendering/layout engines

The web developers tray to build web pages to be compatible with different browsers because the people all around the world use various web browsers. Each browser has its own rendering engine to draw HTML and CSS website contents, thus each browser renders each website differently. Wikipedia defines the layout/rendering engine as "a software component that takes marked up content (such as HTML, XML,image files, etc.) and formatting information (such as CSS, XSL, etc.) and displays the formatted content on the screen. It draws on the content area of a window, which is displayed on a monitor or a printer" [7]. When we look at a displayed web page on different web browsers, we notice that there are not major inconsistencies in observed screen but when it comes to trace level ”traffic pattern” we observe different traffic patterns as it is validated in empirical evaluation chapter. The rendering engine is the main component of a browser and it explicitly decides how to turn HTML, stylesheets, and scripts into a vibrant web page. Each rendering/layout engine behaves differently according to how it was programmed so it has everything to do with how web pages are generated and visualized [12]. The rendering engine does most of the work of website retrieval as it interacts directly with networking interface, JS interpreter, and UI backend component. Table 3.1 shows the most popular rendering engines used by tested web browsers.

Rendering/Layout engine	Browser
Blink	Chrome 31.0
Gecko	Firefox 26.0
Trident	Internet Explorer 10
WebKit	Safari 5.1
Presto	Opera 12.0

Table 3.1: The recent rendering/layout engines used by tested web browsers.

When a rendering engine parses HTML code, it calls networking layer by sending several requests to web server in order to retrieve included web page objects. Then, it gets the responded flow contents of requested objects as it is shown in Figure 3.2.

The figure shows the gradual processes of browser rendering engines for interpreting HTML code and displaying fetched web contents on the screen. First, the HTML tags is parsed and turned to Document Object Model (DOM) tree. Then,

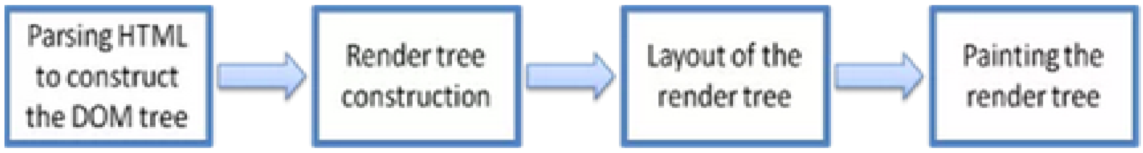

Parsing HTML to construct the DOM tree
Render tree construction
Layout of the render tree
Painting the render tree

Figure 3.2: Basic flow of browser rendering engines [34].

another render tree is built with visual attributes, dimensions, and colors specified by CSSs and visual instructions of HTML document. After that, the constructed render tree is passed through layout process to give each node its coordinates in order to be displayed in its prober position of browser window. The final step is painting the render tree nodes using UI backend component [34]. In our study we formulate the differences between browser rendering engines in terms of their support to various web features that affect browsers' trace level rather than internal presentation level because our focus is traffic analysis topic. More details on web technologies that affect the shape of web browsers' traffic patterns are explained later in Section 3.2.3.

3.2.2 Web browsers' JavaScript engines

Unlike static web pages that contain plain HTML document and static objects, dynamic web pages enrich with interactive web contents. These contents do many tasks such as an interaction between web page and users, control web browser and alter web page contents which make web pages more like an application. To build animated/flashy web pages, Java Script language is introduced for improving the user interaction experience so it is a very widely used language on the web today. JavaScript commands run automatically in web browsers to add different functionalities to web pages. Brendan Eich designed JS language in 1995 to allow non-programmers to extend web sites with client-side executable code [68]. It is an interpreted language with Object Oriented (OO) capabilities which used mostly by web browsers [100]. It has been standardized recently by European Computer Manufacturers Association (ECMA) [58]. Furthermore, it is used by 97 out of 100 most popular websites worldwide [2].

Similar to rendering engine a Java Script engine is a key component of all major web browsers and each browser has its own Java Script engine enrich with its own features support. It is a process virtual machine for interpreting and executing JS codes that are embedded in a web page. Each major browser supports JS code in order to allow client-side scripting and dynamic web pages. JS code is loaded and executed in browser side so it has a great impact in overall performance and functionality of web browsers. The differences in functionalities introduced by web pages are often due to various APIs JS supports employed by different browsers [21]. Table 3.2 shows the embedded JavaScript engines corresponded to their web browsers.

Java Script Interpreter/engine	Browser
V8	Chrome 31.0
Spidermonkey	Firefox 26.0
Chakra	Internet Explorer 10
Nitro/SquirrelFish	Safari 5.1
Carakan	Opera 12.0

Table 3.2: The recent JavaScript engines used by tested web browsers.

However, when a web browser loads a web page, an embedded JS code is interpreted and executed by browser JavaScript interpreter/engine. The executed JS codes can access a set of available Application Programming Interfaces (APIs) allowing them to do many tasks such as interacting with web page elements using "DOM APIs", accessing local browser data "cookies", communicating to remote servers using several communication APIs, and manipulating other browser events [10]. Therefore, we hypothesize that various browsers supports to JS generate different traffic patterns of browsers.

A typical JS engine contains three embedded components that execute JS codes. First, the source code is analyzed to construct a syntax tree ,that represents the flow of the code, by parser component. Next, the syntax tree is interpreted to execute the code by interpreter component. Finally, the run-time module provides JS engine with different standard objects (e.g. Array, String, Math, etc) [40]. The execution of JS codes have a great impact on both the rendering of a web page and the traffic pattern of web browsers, because the components of JS engine are often tied and interacted to its corresponding rendering engine [60]. Thus, Java Script has been used by many researchers and attackers to manipulate the pattern of the traffic for fingerprinting purposes. For example, Abbott et al. injected java script to be executed on browser side in order to change its traffic pattern as a signal that can be detected and associated for browser-based attacks on Tor [1]. Figure 3.3 shows possible interactions that Java Script behaves when browsing the web which has an impact on whole traffic pattern.

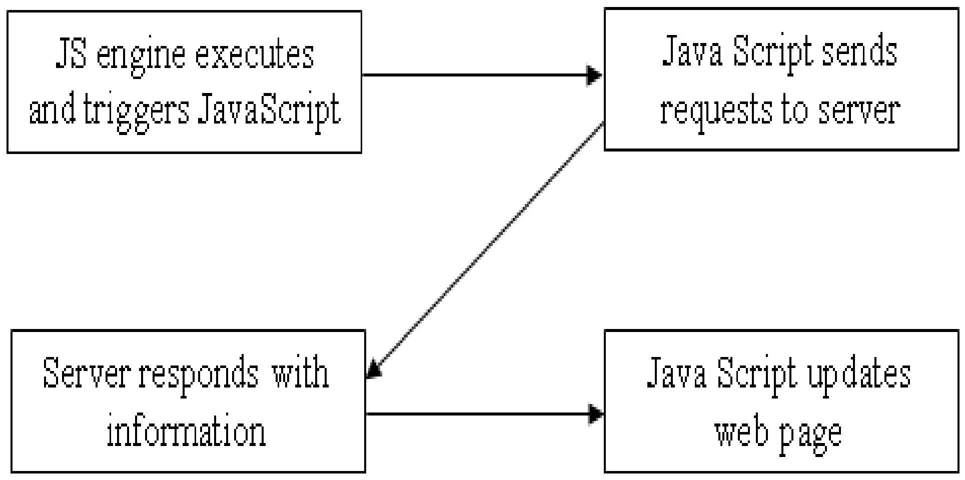

JS engine executes and triggers JavaScript
Java Script sends requests to server
Server responds with information
Java Script updates web page

Figure 3.3: The interactions between Java Script ,that is executed on browser, and web page server.

A JavaScript web application is the essence of HTML page which is associated with other resources (like CCSs, image files, etc). JS codes execution is driven by events in web browsers like timeouts occur. Most real time JS programs are embedded in the context of HTML page. When they are executed in browsers, they change the HTML DOM and access browsers' APIs which cause considerable challenges to the analysis and control of web browsing data [68]. JS has been used by large fraction of all websites which allow web applications to be more interesting, dynamic, and responsiveness. Unfortunately, these advantages of JS come with a lot of security problems that attract the attention of academicians and researchers to take care of. It has been discovered that there are a lot of attacks that exploit the JS dynamism (e.g., accessing and modifying shared objects, injecting malicious codes, etc). As a result, several researchers [96], [76], [51], [38], and [22] have beeb proposed with various approaches, to monitor and prevent JS related attacks.

Browser's Java script engine is invoked from JS code embedded within HTML documents in five primary ways [98]:

- Standalone $< SCRIPT >$ tags that enclose JS code block and running dynamically during HTML code parsing
- External/remote JS codes $< SCRIPTSRC =$ "..." $>$ which are executed based on the security context of relevant browser
- CSS Style-sheet expression(...) block that authorizes JS syntax in some browsers
- Certain URL links that are specified as target for certain JS actions
- Event handlers that are attached to HTML tags such as mouse hover

In our research we take into consideration both: first and second methods that are relevant to browser dependent actions. When JS code is called by JS engine, it has full access to DOM. The executed JS may also further invoke new JS by calling eval(), producing JS-invoking HTML, or configuring timers (e.g. setTimeout and setInterval) [46]. Moreover, at run time, new HTML elements can be created dynamically so each HTML element rises a range of JavaScript objects. Executed JS has various interactions with same-origin document data (e.g. drawing CANVASes, displaying pop-up dialog, etc). In this research we neglect JS actions that change the page look and we focus more on JS actions that have an impact on the shape of traffic pattern "level trace" as our interest is traffic analysis attack. When executing remote/external JS (e.g. $< SCRIPTSRC =$ "..." $>$, or

$< LINKREL = "Stylesheet" HREF = "..." >$), they might send requests and read responses even across domains violating Same Origin Policy(SOP). Furthermore, they may send information to third party servers by spawning objects (e.g. $< IMG >, < IFRAME >, < APPLET >$, etc) and read back the returned data [98]. The external interactions of JS depend on security policies and features supports introduced by associated browser. For example, the browser that sits more restrictions on the interaction with third-party domains, it prevents loading and executing of possible malicious JS codes that may happen like cross-site scripting (XSS) as it is shown in Figure 3.4.

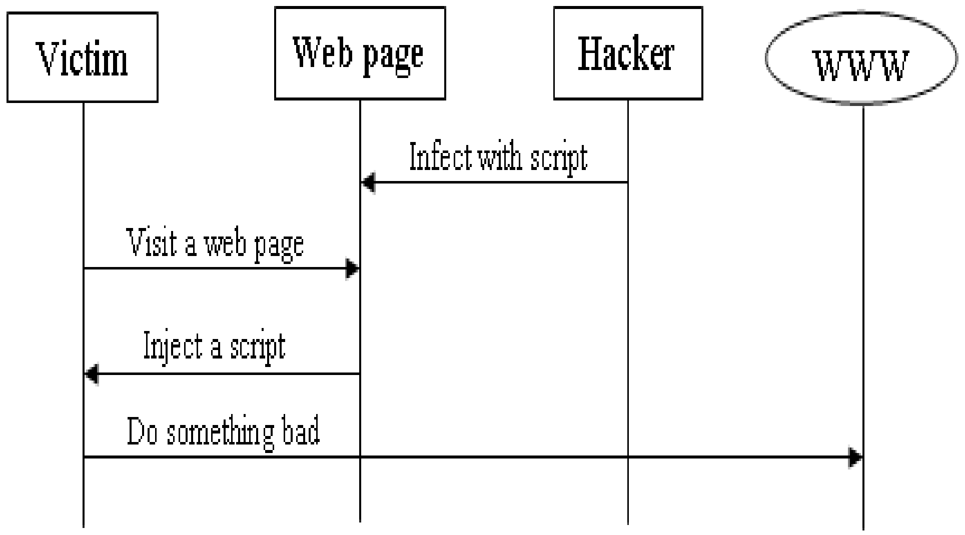


Figure 3.4: Possible scenario of XSS attack

However, it has been noted in a Benchmarking of Modern Web Browsers [4] that JavaScript and related technologies demands have increased on browsers rather than servers which of course will result of different behaviors of web browsers. A number of studies on [97] show that there are a number of users who disable scripting languages in their browsers for security reasons which results in different browser compatibilities toward several web services. Moreover, web browsers ,that is configured to use Tor system, executes the JavaScript code through Tor circuit and the browser still anonymous [1]. Therefore, Java Scripts in our research are enabled as we conducted all our experiments in default modes of tested web browsers.

3.2.3 Important web contents and web technology features

Popular websites vendors design their websites for maximum compatibilities of browsers so that the web pages are rendered correctly on modern web browsers such as Chrome, FF, IE, etc. In order to understand how a web page is loaded on different combinations of browsers, we need to investigate different features of web page contents that are browser-dependent. There are three main objects that make a web page look and behave differently across browsers which are: HTML defines web page contents and markups, CSS defines the appearance, and JavaScript defines the behavior. In this section we will discuss these important contents and focus more on web technologies the that affect the traffic pattern shapes of browsers.

First, Hypertext Markup Language (HTML) is a markup language that describes the structure and content of a document by tagging and identifying different elements embedded in the document (e.g. text, images, JS, etc)[54]. It forms the building blocks of all web pages and can be used to create interactive forms [89]. In the early years of HTML language, web developers were free to design web pages in the way they want so there wasn't any organization responsible for standardizing the language. As a result, this created too many incompatibilities between different browsers as well as making a big challenge for web developers to write HTML code that satisfies various browsers and even different versions of the same browser. To solve this issue a World Wide Web Consortium (W3C) [80] was created by a group of web developers and programmers in order to set specifications and standards to be followed by all browsers' makers. W3C maintains the specifications of HTML, CSS , and other web-related standards such as web content accessibility guidelines (WCAG) and scalable vector graphics (SVG) [43]. Although browsers' vendors use the recommended rules and guidelines of W3C specifications, they can interpret these rules as required for their own purposes. However, in our experimental work we relay on W3C standards for testing and validating a lot of features that cause different traffic pattern of tested browsers.

Many HTML versions [54] have been developed over the past years. The W3C develops HTML5 specification as a recent standard for the next generation of HTML. With the present of this new HTML standard, old versions of HTML standards are neglected by both web page owners and browsers manufacturers. Recently, all major browsers support the most recent HTML standards such as HTML 4.01, XHTML 1.1 and HTML5 [43]. The latest iteration is HTML5 and it includes many improvements to the existing features such as adding new features and scripting-based APIs. The Application Programming Interface (API) is considered as the same graphical interface for user but instead of being interface for human it is an interface for HTML code. It has been used by web developers for years to be associated with HTML5 for improving a number of techniques and put more power in developers' hands to simplify a lot of tasks. Moreover, it provides less dependence on plug-ins and third party software when browsing rich media contents like introducing HTML5-based audio and video means. Today, web pages become much more complex referred as "mashups" since there are more contents and services are fetched among them from multiple sources called third party domains. For example, some web pages present remote content from a web page that is separated from original page called iframe. Browsers have various security restrictions when they fetch web page contents from third party domains so this has an impact on the retrieval behavior of that browsers. Figure 3.5 shows a web page that combines various web contents from different domains.

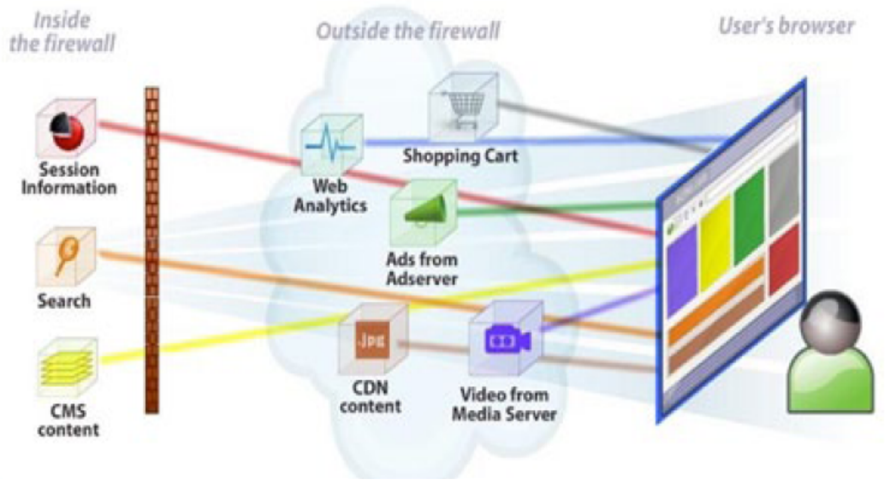


Figure 3.5: Common web contents that may be fetched from third party sites. Source: http://bharathmarrivada. blogspot.com/2010/09/browser-wars-speed-testperformance.html

Today, with the introduction of WEB 2.0, several web applications are rendered on websites from different domains in order to fulfill end user satisfaction [52]. As a result, a high competition between different browsers' vendors comes up to adopt multiple features for providing web users with better performance and rich functionality.

Web browsers makers have a security concern about malicious objects that might be injected by third party side. Thus, they support their browsers with different JavaScript security policies which of course have an impact to the shape of the traffic. There are a lot of web services introduced on web pages which depend on Java Script to be implemented such as HTML APIs that allow web developers to design much richer and interactive web pages. But more security concerns have come with the new functionality available to JavaScript programs [10]. For brevity, from our deep survey about web browsers, we hypothesize that there are important HTML APIs that may have an impact on the stability of browsers' traffic patterns which are discussed in subsequent sections.

Server-Sent Events(SSE): An API that provides real-time events which allows the server that hosts a web page to communicate back to the browser and updates a web page with some information from the server rather than repeatedly requesting it [93]. SSE was developed under HTML5 as a web pushing technology for data transmission in time interval or any time from server application to a browser that supports this API [87]. As a result, the browser that supports this API its retrieved web page gets updates automatically from server side and immediately appear on user screen (e.g. news feeds, stock price updates, Facebook/Twitter updates, etc) as a pushing scenario illustrated in Figure 3.6.

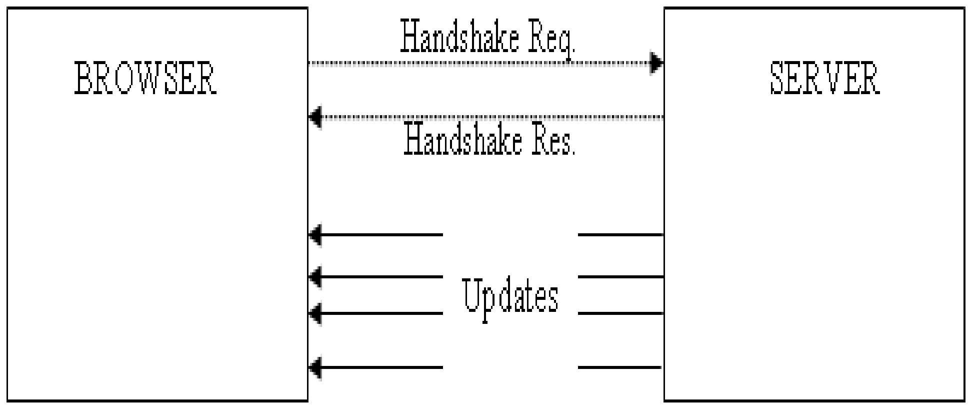


Figure 3.6: A process mechanism of Server-Sent Event between supporting web browser and a server.

The logic process of Server-Sent Event technique is shown in the figure where a handshake request is sent from a browser that support SSE to supported server. Then, a server system responses with a handshake response as text/event-stream. Finally, after the handshake committed between the browser and SSE service, the SSE service may send any amount of data at any time during a connection session [87]. However, SSE is handled directly by web browser and the user who browse the web using such supporting browser simply has to listen for coming messages from server side . More details behind Server-Sent Events API are given in [9].

Content Security Policy (CSP): The home page or root page of a website provides several links to second party HTML pages so it turns links to third party HTML pages and so on and so forth as it is shown in Figure 3.7.

A browser ,that trusts and fetches all stuff of a retrieved web page as being legitimately part of web page security region, could has a severe problem. This is because some possible attacks such as Cross-Site Scripting (XSS) bypass the same

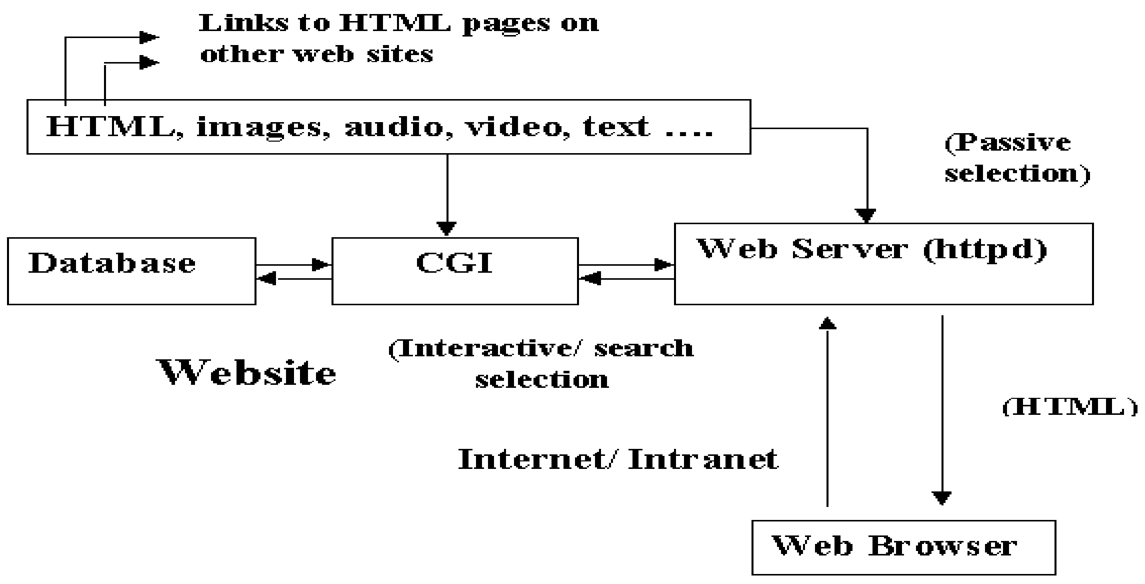

Links to HTML pages on other web sites
HTML, images, audio, video, text ....
(Passive selection)
Database
CGI
Web Server (httpd)
Website
(Interactive/ search selection
(HTML)
Internet/ Intranet
Web Browser

Figure 3.7: Accessing multi level/linked web pages over the web [45].

origin policy by tricking a web page into delivering malicious code into users along with the intended content. To solve such possible vulnerabilities, W3C proposed an important technique called Content Security Policy (CSP) [78]. This policy is implemented by a web page author which contains a list of domains for specifying from where the remote contents (e.g. scripts, CSSs, media, HTML frames etc) can be loaded and executed on a designed web page. It provides a standard HTTP header that allows web page authors to declare approved web contents that the browsers should be permitted to load on a fetched web page [90]. Therefore, CSP defense can significantly mitigates the risks and attacks that can be associated with fetched contents by whit-listing trusted sources of web page contents.

Sandbox Attribute For "iframes": Frame is a web feature that allows an HTML window to be splitted into segments each segment can show different document [91]. The iframe is an element or HTML inline frame that represents a nested browsing context embedded within a retrieved web page. The remote contents included in a retrieved web page are usually implemented by iframes that separate those contents from the main page as it is clear in Figure 3.8.

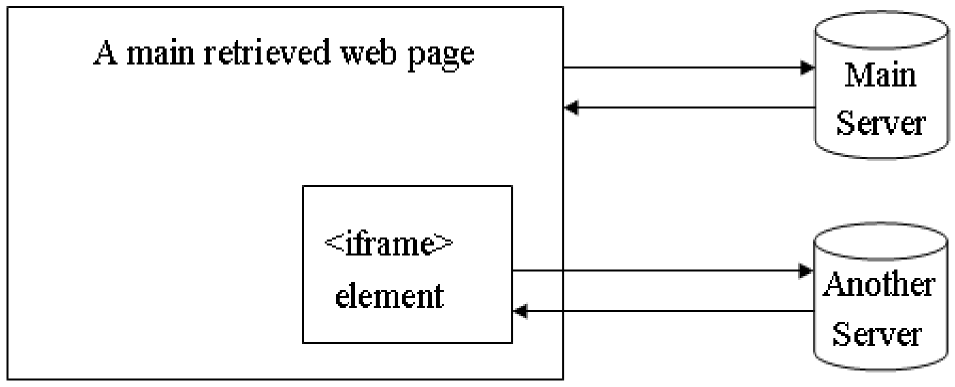


Figure 3.8: An inline iframe element embedded within the main web page.

Some web browsers allow various web contents from different domains to communicate with each other by implementing several APIs communications. An iframe element might load un-trusted contents or run malicious behavior (e.g. auto-playing video, plug-ins, and pop-ups) [88]. Consequently, a sandbox attribute is proposed which is a property that sets extra restrictions on web stuff that can appear in the inline frame [29]. So it restricts the actions of the iframe can take rather than on the resources that the iframe can load. However, a browser, that supports sandbox attribute, reduces possible risks that can be associated with iframe contents loaded from third-party and prevents a clever attacker from injecting malicious objects within fetched contents. It loads specific frame's contents in a low environment and allows only a subset of capabilities necessary to run inline iframes [88].

The discussion of content or structure layer of rendered web page is done so now it is the time to discuss what makes it pretty using Cascading Style Sheets (CSS). However, HTML marks various contents of the document, but it doesn't indicate how they are displayed. This is because the author of a web page doesn't have control on various devices and browsers that this web page will be displayed on. Therefore, the exact appearance of web page elements are described in a separate document called Cascading Style Sheet (CSS) [54]. CSS is the technology that makes how web pages look like by giving a web browser a set of instructions about the style of web page appearance (e.g. font sizes, colors, backgrounds, etc). It is a style language that describes how HTML markups are styled and presented so it describes the rendering of structured documents on various output layouts (e.g. screen, paper, speech, etc) [43]. Therefore, various style sheets are created by web developers to display their web page contents as they intent. The recent version of style-sheet is CSS3 and it has several features (e.g. animation, multiple backgrounds, transparency, specifying colors, etc). Each browser has its own internal style sheets mechanism of specifying the way that different web page elements appeared such as different font styles. For this reason, a web page author creates a separate style sheet document that is embedded within HTML document which specify the look of a web page depending on the version of a web browser that is specified in client string of requested packets. Furthermore, the author of a web page creates multiple style sheets which fit various output devices (e.g. rendering on screen, printed output, and rendering aurally) [54]. However, over time the task of figure out the differences between browsers has become more complicated because of the the large-scale usage of CSSs, the ongoing evolution of HTML standards, and the continual addition of web technologies [60]. In our research we formulate the differences between web browsers in terms of their support for web features that will be reflected to their behavior on traffic pattern. Table 3.3 shows the various web technologies that affect the trace patterns of web browsers. The browsers support for these features are tested in the empirical evaluation chapter.

Web technology	Description
Asynchronous script execution [16]	The script object has an async attribute that enables the associated elements of script to be loaded and executed asynchronously with other objects of loaded page. This web technique is a part of World Wide Web Consortium (W3C) HTML specification so the browser that supports this feature can download and execute JavaScript objects with other web pages element in parallel in order to increase the performance of page-load significantly.
Navigation Timing API [44]	It is a timing API to provide a browser with measurements related to TCP connection establishment and spent time for web pages loading. It can access to timing information related to navigation and web page elements.
ActiveX [62]	Microsoft added ActiveX into Internet Explorer to host ActiveX controls within web pages contents. It allows certain web pages to automatically download scripts, execute small applications, and embed animations in a web page such as banner ads. This makes the retrieved web page richer but that is why IE is more vulnerable to security threats.

Table 3.3: The web technologies that have an impact the web browsing traffic pattern.

The former sections demonstrated the importance of JavaScript, CSS, and web technologies for creating highly interactive web pages. We will validate their support and impact on browsers' traffic patterns experimentally in the next chapter. Furthermore, we will show how they make the resistances of browsers against traffic analysis attack distinguishable from one another.

CHAPTER 4

WEBSITE FINGERPRINTING ATTACK

In this chapter, we detail and show the results of a number of experiments to assess the efficiency and utility of our approach. Our empirical evaluation addresses the following research questions:

RQ1: Are there different resistance levels of website fingerprinting attack conducted by popular web browsers?

RQ2: Which browser protects web users' privacy against website fingerprinting attack more and which browser protects less?

RQ3: What are the root causes that stand behind various resistance levels of browsers against website fingerprinting attack?

To answer and validate these research questions empirically, we follow the following evaluation methodology including the experimental setup, data collection, applied method, results and analysis, and discussion.

4.1 Experimental Setup

In order to investigate the protection range of browsers to the privacy of web users over Tor, we have implemented a website fingerprinting attack. To do so, we have collected the traffic traces and conducted several experiments using multiple scripting codes and tools. The architectural framework for our experiments has been set up using the software and hardware tools specified in Tables 4.1 and 4.2 below.

Tools	Version
OS	Windows 8
tshark	1.10.0 (SVN Rev 49790)
Google Chrome	31.0.1650.6
Firefox	26.0
Internet Explorer	10.0.9200.1638
Safari	5.1.7
Opera	12.16
Tor	3.5.2.1

Table 4.1: Software Tools.

Devices	Specifications
CPU	intel(R) Core(TM) Duo CPU T9600 @ 2.80 GHZ
Memory	4 GB
Ethernet Card Model	1EECD81C-907A-475B-ACF1-D106043C1F6D

Table 4.2: Specifications of hardware platform.

The communication backbone of the experiments is based on Ethernet environment of ADSL 3MB. We believe that the distinction between the retrieval behaviors of browsers is difficult, since there are many factors that may affect the shape of their traffic patterns (e.g. the content and structure of the retrieved web pages, browser configuration, the configuration of client hardware, and network conditions). Therefore, we designed and conducted our experiments in a controlled environment in order to focus on browser-dependent factors that affect the retrieval behavior of browsers by eliminating external factors as they will be explained later. The experiments have been conducted on the latest stable and available version of web browsers with their default configuration without any installed add-ons/extensions to make sure that the observed results are browsers-dependent. In order to evaluate the resistance levels of browsers against website fingerprinting attack, we set up our attack platform as it is illustrated in Figure 4.1 scenario. Our attack passed through several phases to figure out the website fingerprinting attack results on tested browsers.

The first phase is the data collection phase where we sniffed the encrypted data trace of websites retrieved over the top five browsers based on the method specified in Section 4.2. Second phase is the preprocessing phase where we conducted some preprocessing steps on captured data set (e.g. remove noisy data, extract features, etc) to be prepared for classification phase. The phases that our attack passed through are explained in subsequent sections.

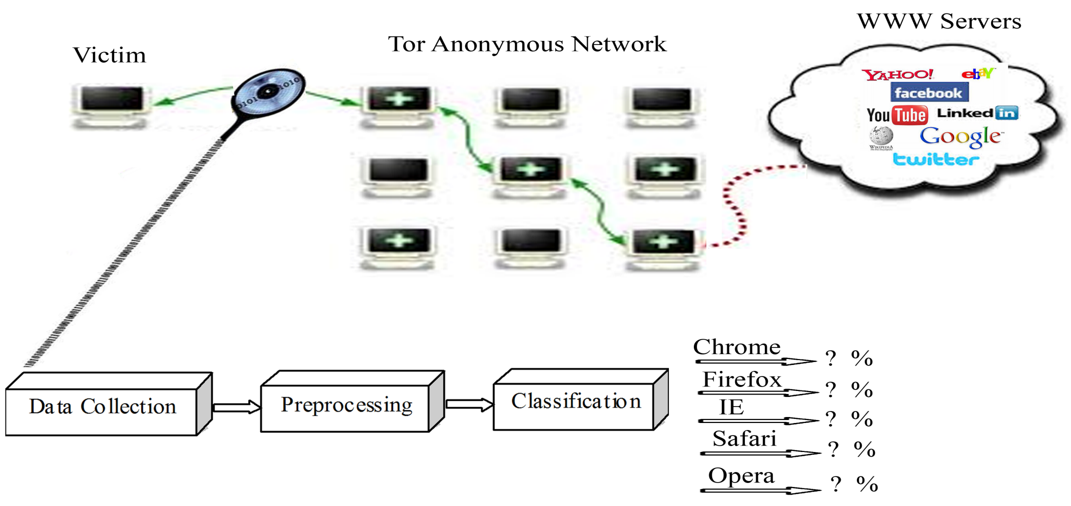


Figure 4.1: The various phases and an overview of our web page fingerprinting attack scenario over Tor.

4.2 Data collection and Preprocessing Phases

Tor anonymous system is a live network with millions of daily users [64] who may have entirely different browsers. Previous researches in website fingerprinting attacks generally did not consider websites fingerprinting on different web browsers. In this section we demonstrate how we sniffed/collected Tor anonymous data traffic on the most commonly used web browsers for comparing different protections of tested browsers against traffic analysis attack (website fingerprinting attack). In order to create fingerprints of websites, we first established Tor network connection and configured web browsers to use Tor proxy with its default configuration so all HTTP traffic is tunneled through default configuration of Tor. We investigated the effectiveness of our fingerprinting approach on the main pages of the top 20 most visited websites ranked by Alexa statistics [2]. We selected these websites/dataset because of their global popularity as well as their representative to diverse activities on the WWW such as e-commerce (amazon and ebay), search engines (google and ask), social networking (facebook and twitter), etc. The data-set of our experiments were collected during Feb. 2014 under a closed-world scenario. In such scenario, the attacker creates fingerprints for a list of websites so when a victim visits certain website from a list of predefined/fingerprinted websites, the attacker observes the victim's trace pattern and matches it with the list of previously fingerprinted websites in order to guess which website that the victim visits for the sake of testing browsers protection. To automate the browsing of websites, we wrote a python code to simulate a typical user action like typing a URL into the address bar of the browsers. It automates each web browser for browsing the list of 20 websites so each browser visits each website 15 times in a round-robin fashion. During the web browsing visits, we scripted tshark, the command-line version of WireShark traffic analyzer, to capture the real traffic packets of each visited website to be recorded in trace/log files under each browser. Each trace/log file is labeled with the browser name, visited web page, and the number of visit for further analysis. We recorded 15 log files (packets traces) for each loaded website in the list. We repeated the automation of websites visits on each browser and removed its cache after each website visit. Through browsing automation process, we sat 25 seconds as a time out for each loaded page to assure the loading of each website completely as well as for the sake of consistency between tested browsers. We ended up with 1500 log trace files for the five browsers, with (20 web pages $*$ 15 visits) per each browser. Each trace file contains the data traffic of its visited web page (e.g. the time of sent and received packets, the sizes of packets, the order in which the packets were sent or received, etc). Table 4.3 lists the top 20 websites that we used for our study.

Websites' Data Set			
1	http://www.google.com	2	http://www.facebook.com
3	http://www.youtube.com	4	http://www.yahoo.com
5	http://www.baidu.com	6	http://www.en.wikipedia.org
7	http://www.ebay.com	8	http://www.live.com
9	http://www.taobao.com	10	http://www.linkedin.com
11	http://www.sina.com.cn	12	http://www.twitter.com
13	http://www.amazon.com	14	http://www.hao123.com
15	http://www.google.co.in	16	http://www.blogspot.com
17	http://www.weibo.com	18	http://www.tmall.com
19	http://www.wordpress.com	20	http://www.ask.com

Table 4.3: List of used websites for investigating the resistance levels of browsers to website fingerprinting attack.

To prepare the data for classification phase, we did some preprocessing steps: we removed TCP control packets (Acks and Syncs) because they reduce the performance of the system and don't add useful information in classification phase. Moreover, we filtered only Tor packets from other non-Tor packets that might captured during traffic sniffing. Then, we extracted certain features from collected traces to create a profile for each visited web page which called "Fingerprint". To extract the features for classification, we built scripting program to extract the sizes and the directions of retrieved packets that represent web pages fingerprints. The traffic packets are stored as integers in the observed direction and recorded as positive for outgoing packets and negative for incoming ones (e.g. 1150, −52, 1500, −638, 52, and 638 bytes). We have selected these packets features because they reveal information about the sizes of referenced packets by a web page and the order in which the browser issues or retrieve them.

4.3 Classification of web browsers resistances

In classification phase, the profile/trace of each website is represented as fingerprint stored as a sequence of positive and negative integers. These integers represent the sizes and directions of TCP packets generated to load website contents over Tor. Then, the website traces/fingerprints are classified using Cai et al. method described in [15]. So we calculated the similarity between websites fingerprints using Damerau-Levenshtein distance algorithm. It figures out the similarity between websites fingerprints by calculating the number of operations (insertion, deletion, substitution, and transposition) that are required to transfer trace t= (−1150, 1500, −638, 638, etc) into trace t' = (−638, −1150, 638, 1500, etc). Thus, the minimum number of operations to transfer trace t into trace t' is the more similar they are to each other so they are considered as two visits from the same website. After the similarity distances between websites fingerprints are calculated, they are classified using Support Vector Machine. As a result, we got the different recognition rates of websites under tested browsers. Figure 4.2 shows our fingerprinting results over tested browsers. It gives an overview of different resistance levels of browsers against website fingerprinting attack.

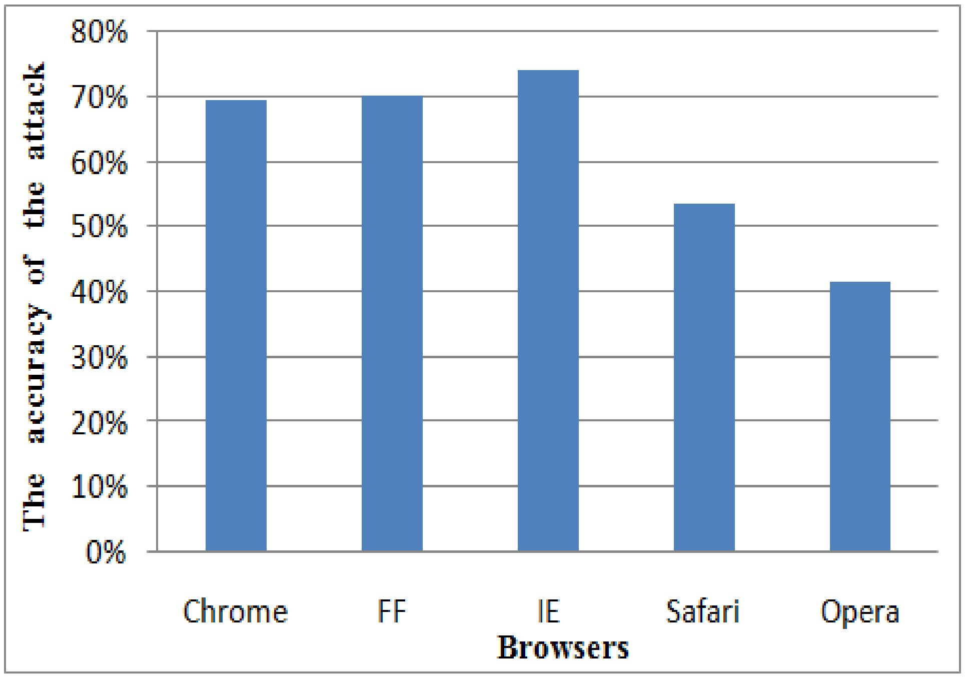


Figure 4.2: The resistance rate of web browsers against website fingerprinting attack over Tor.

Our metric in this study is the success rate of identifying websites fingerprints or the percentage achieved to guess the identity of a website that the victim visits correctly over popular browsers. Figure 4.2 shows the different recognition rates of websites fingerprints that we achieved over tested browsers. The highest recognition rate achieved by a browser is the least privacy protection that this browser introduces, while the lowest recognition rate achieved by a browser is the highest protection it introduces against website fingerprinting attack. So the highest recognition rate is 74% that is achieved by IE which indicates that it has the least protection against website fingerprinting attack so the shape of its traffic pattern can be monitored with the highest accuracy rate compared to other browsers. On the other hand, the lowest recognition rate is achieved by Opera of 41.6% followed by Safari of 53.8% which indicate that the surveillance of their traffic patterns by attackers is more difficult so they protect more against website fingerprinting attack compared to other browsers. The recognition results of Firefox and Chrome are 70.4% and 69.6% respectively. However, the root causes behind these various resistance levels of browsers against website fingerprinting attack are explained in details in the subsequent sections.

CHAPTER 5

4. ROOT CAUSES BEHIND DIFFERENT RESISTANCE LEVELS OF BROWSERS

After we detected various recognition rates of browsers in website fingerprinting attack, we figured out the main reasons behind various website fingerprinting results on Tor anonymous system. In this chapter we investigate the root causes of different resistance levels of browsers against website fingerprinting attack. However, the differences between web browsers span a wide range of features from visual/look level to traffic/trace level. These variations are caused by various functionalities that are supported by different browsers. Our approach doesn't target internal browsers differences that don't have an impact on the shape of their traffic patterns such as DOM manipulation, but it targets the browser-dependent features that affect the shape of its traffic pattern (e.g. JavaScripts, third-party ads domains, performance optimization, parallel downloads, etc). Because the web browsing traffic under Tor is anonymous/encrypted so we can't see packets data and their related information fields. Furthermore, we can't observe the behavior of browsers on retrieving different web browsing contents. Therefore, we turned off Tor then we carried out deep analytical study on web browsing traffic by sufficient drill down to web pages contents level. We have done the experiments on the same platform and data-set of previous experiments but on normal traffic instead of anonymous/encrypted traffic. In this section we conducted several fine-grained tests on browsers-dependent features and their impacts on web browsing contents to catch the main causes that they make browsers behave differently. The experiments are conducted based on the stable versions of browsers, no add-ons/extensions, and without any external programs that may affect the dependent behavior of tested browsers.

5.1 Browsers' web technology features

Each web browser has its own associated features and traffic pattern, which make it distinguishable from other browsers. To find out the main causes behind different behaviors of web browsers which resulted in various accuracies of website fingerprinting attack, we have done several experiments to test browsers' differences in both: their support to various web technology features and the reflection of these features to their behavior in retrieving various web contents. We started by testing the support of browsers to various web technologies that have an impact on the shape of their traffic patterns using a couple of standard tools and scripts [35], [26], and [36]. Table 5.1 shows the results of web technologies that we compare tested browsers against. It shows the various supports of tested browsers to popular web technologies. The sign ($\surd$) indicates that the browser supports the associated feature while the sign ($\times$) indicates that the browser doesn't support the associated feature.

Features	Chrome	FF	IE	Safari	Opera
CSS Filter Effects	$\surd$	$\times$	$\times$	$\times$	$\times$
Asynchronous script execution	$\surd$	$\surd$	$\surd$	$\surd$	$\times$
Navigation Timing API	$\surd$	$\surd$	$\surd$	$\times$	$\times$
ActiveX	$\times$	$\times$	$\surd$	$\times$	$\times$
Native Flash blocking	$\times$	$\times$	$\times$	$\times$	$\surd$
Cached compiled programs	$\times$	$\times$	$\times$	$\times$	$\surd$

Table 5.1: Web technologies that affect the shape of traffic patterns of web browsers.

We selected these features as a significant metric for browsers investigation as they have a great impact on their traffic patterns. The main functionality of these features and their impact on web browsing traffic patterns are explained in the subsequent sections.

The "Asynchronous script execution" feature [16] allows the JavaScript to be loaded and executed asynchronously with other objects of a loaded page. So the Opera browser that doesn't support this feature, block all other downloads when it retrieves and executes JavaScript file. Therefore, this behavior has a great impact on the sequence order of retrieved objects and the performance of page-load significantly as it is shown in results of section 5.2.3.

The ”Navigation Timing API” feature [44] provides a browser with accurate measurements related to the establishment of TCP connection and the spent time for web pages contents retrieval. It can also access to timing information related to navigation and web page elements. Thus, Opera and Safari don’t support this timing feature which has an impact on their traffic patterns as it is investigated in section 5.2.3.

ActiveX feature [62] is supported by Microsoft in its Internet Explorer browser. It hosts ActiveX controls within web pages contents. So it allows certain web pages to automatically download scripts, execute small applications, and embed animations in web pages such as banner ads. This allows a rich retrieval of web pages contents but our attack shows that the highest recognition rate is achieved under IE because of its behavior in allowing rich contents to be retrieved which make its traffic pattern the most distinguishable than other browsers as it is shown in the results of next sections. Native Flash blocking feature that is supported by Opera browser which blocks flash animation contents as a result its traffic pattern is affected as it is shown in the results. Furthermore, the ”Cached compiled programs” feature that is supported by Opera which allows it to cache JS libraries in its internal cache to be used again in JS contents retrieval without reloading.

Therefore, these features affect the browser-side footprint heavily when retrieving rich-contents of websites. Moreover, web technologies (e.g. ads filtering, Flash blocking, third-party domains' restrictions, etc) affect also the traffic patterns of browsers in various levels based on their support by browser. The various browser-dependent features and their impact on the retrieval behavior of various web contents that are investigated in the subsequent sections.

5.2 The impact of browsers-dependent features on their web browsing traffic patterns

When a browser sends an HTML request, the corresponding server handles the request and delivers an HTML document to the browser. Then, the rendering engine of the browser parses HTML doc so the embedded objects within HTML code (e.g., images, JSs, flashes, etc) are fetched from their referenced servers. Each web page has its own fingerprint in a term of number of various web objects. Each browser retrieves website contents differently based on its support to different web technology features so each browser has its own website fingerprint. The characterization of browser features essentially involves the characterization of various website objects (e.g. application/JavaScript, application/x-shockwave-flash, etc) retrieved by a corresponding browser. Therefore, we have matched each browser feature mentioned above with its relevant behavior in websites retrieval to see its impact on browsers traffic pattern. We have done many tests to reach the main causes behind web browsers that influence their traffic/trace patterns so we have proved their impact experimentally as it is shown in the subsequent sections.

We conducted a deep traffic analysis and aggregated analytics on real browsing data (number of retrieved resources, content types, and other metadata) of our fingerprinting data-set that posted in Table 4.3. All possible web content-types that we have analyzed are posted in Table 5.2 below.

Various web pages content-types			
1	text/html	2	text/css
3	text/javascript	4	text/plain
5	application/x-javascript	6	application/javascript
7	application/x-chrome-extension	8	application/json
9	application/x-shockwave-flash	10	application/x-www-form-urlencoded
11	application/ocsp-response	12	application/ocsp-request
13	application/octet-stream	14	application/x509-ca-cert
15	image/x-icon	16	image/png
17	image/webp	18	image/jpeg

Table 5.2: List of all possible web browsing content-types that we analyzed on Web pages' Data Set.

Thus, after we did extensive analysis on all retrieved web contents-types, we figured out the contents-types that draw a certain traffic pattern for each browser. From analyzed data we have characterized web browsing traffic to a number of metrics. In this section we demonstrate some interesting metrics that are correlated to different browsers-dependent features which affect the shape of their traffic patterns. The experimental results in Table 5.3 are conducted by several statical analysis tests on our data-set which will be explained in details in the subsequent sections.

Browser Name	Average JS data flow[KB]	Number of empty files	Average No. third-party domains	Average loading time [$Sec.$]
Chrome	6845.692	23	533	27.426
Firefox	9127.72	21	493	28.394
Internet Explorer	14855.82	24	603	29.298
Safari	4172.84	22	434	19.322
Opera	517.44	67	0	12.028

Table 5.3: Characterizing traffic characteristics generated by tested web browsers.

From our experiments, we observed that each browser exhibits different pattern on the same browsing data set. This is because each browser retrieves web pages contents differently based on its support to various web technologies. Below are the web content results that exhibit the variations on traffic patterns of different browsers.

5.2.1 The impact of JavaScripts on web browsers' traffic patterns

JavaScript is enabled by default on all major browsers, and it was reported by Alexa that 98 out of 100 popular websites use JavaScript [84]. Furthermore, Michael et al. proved that JavaScript contents occupy the fraction of 25% across all downloaded web contents [13]. Therefore, we started to analyze the behavior of tested browsers on this large portion of web browsing content. Each browser has its own JavaScript interpreter so it behaves differently in executing JavaScript programs and it can access a set of available APIs implemented by its browser. JavaScript code is loaded and executed on browser side so there are APIs that JavaScripts deal with allow the scripts to communicate with remote servers [10].As a result, loading and executing JavaScript contents have a great impact on the shape of browsers' traffic patterns. Our results show that different browsers' JavaScript engines load and execute various amounts of JavaScript data as it is shown in Figure 5.1.

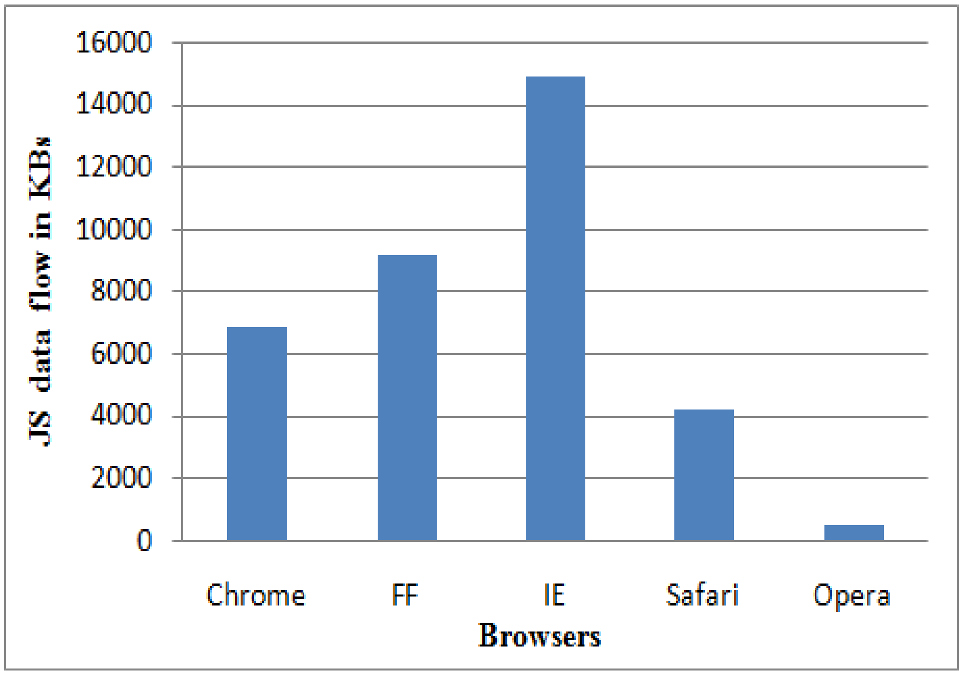


Figure 5.1: The various amount of JS data flow which reflects the different behaviors of browsers' JavaScript engines.

Animation on web pages is the characteristic of JavaScript behavior so the more JS data flow in website retrieval is the larger the chance of website fingerprint/identity to be identified uniquely. As it is clear in the experimental results in

Figure 5.1 that IE has the largest amount of JS data flow of 14855.82 KB followed by FF of 9127.72 KB and Chrome of 6845.692 KB. On the other hand, the least amount of JS data is fetched by Opera of 517.44 KB followed by Safari of 4172.84 KB. During the experiments we noticed that the different supports of browsers to various web features create different JS data flow illustrated in Figure 5.1. The results show that IE has the largest amount of JS data flow. This is because Microsoft provides its IE browser with the distinctive feature called (ActiveX) to host ActiveX controls within websites contents. Thus, it allows certain web pages to automatically execute small applications and download scripts/animations in order to enhance user browsing experience. Moreover, IE allows all flash animation contents that are created by JavaScript Flash language (JSFL) to be loaded as it is integrated with Adobe Flash by default. This makes retrieved websites by IE more richer but rises possible security vulnerabilities especially the recognition rate of website fingerprinting attack as we got in our experimental results shown in Figure 4.2. So the websites fingerprints on IE are distinguished with the highest recognition rate of 74% compared to other browsers. The behavior of FF and Chrome in JS data flow is also reflected directly on their recognition rate of 70.40% for FF and 69.60% for Chrome. They are approximately with the same resistance level to website fingerprinting as they share the same security mechanism called "Safe Browsing" [37] that blocks all suspected JSs in order to provide more phishing and malware protection. Safari deals with less JS data flow because Apple maintains an updated blacklist for malicious JSs and Flashes so that Safari blocks versions of JS and Flashes provided by certain websites which place it on the second least recognition rate of website fingerprinting with 53.80%. The least JS data flow is rendered by Opera. The reason behind is that Carakan [47],the JS engine of Opera, brings an internal caching for compiled JS programs so this technique is quite effective feature in typical scenario where the same JS libraries can be reused internally without reloading it again such as a very large JS library. Furthermore, Opera has a Flash blocking feature so these features affect the traffic pattern of Opera and make it with the least recognition rate in website fingerprinting attack of 41.60% as it is shown in Figure 4. However, the results show that JS behaviors of browsers have relevant impacts on their various resistances against website fingerprinting attack.

5.2.2 The impact of third-party loaded contents on traffic patterns of browsers

With the development of the web and fast growing of on-line business, the web sites appeared to be as a mixed of various web services provided from multiple sources [86]. In our traffic analysis tests, we found that there are a lot of web contents retrieved from different origins shown in Figure 5.2. These contents are collected from third-party servers that are correlated to the traffic of target/first-party server. For example, the visited web page can host several services: advertisement services from popular third-party ads servers (e.g. googleadservices and doubleclick), analytical services that track user activity (e.g. quantserve and google-analytics), and feed with contents from content distribution networks (e.g. Limelight and Akamai). Therefore, the amount of data traffic coming from third-party servers comprise a considerable fraction of retrieved data which have an impact on web browsing traffic patterns of browsers. The results in Figure 5.2 shows that different numbers of third-party servers are contacted by browsers that allow various amount of services to be retrieved based on the variety of their security policies.

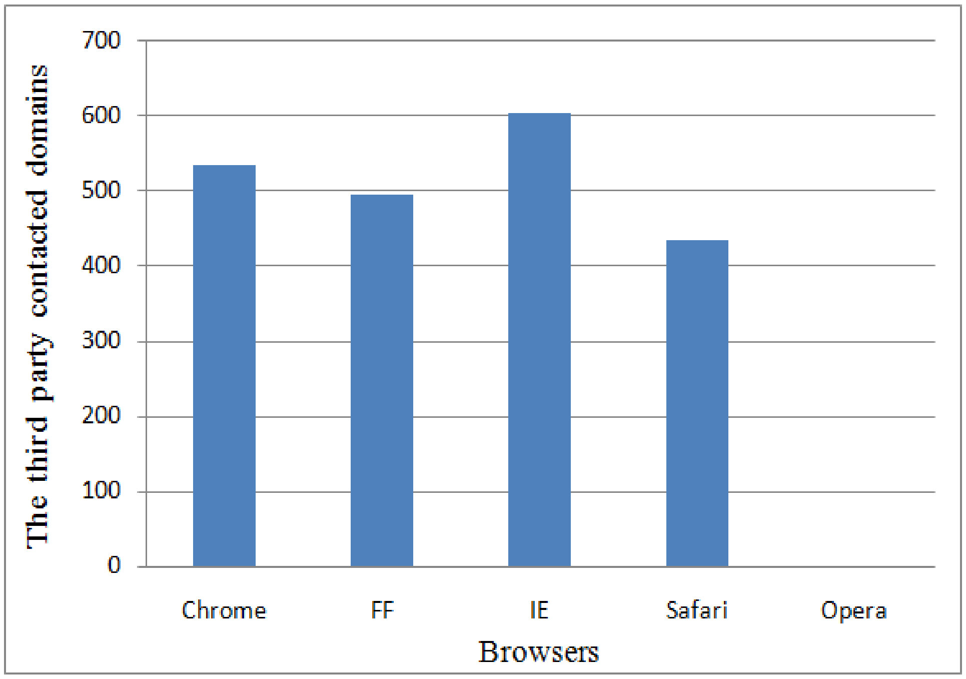


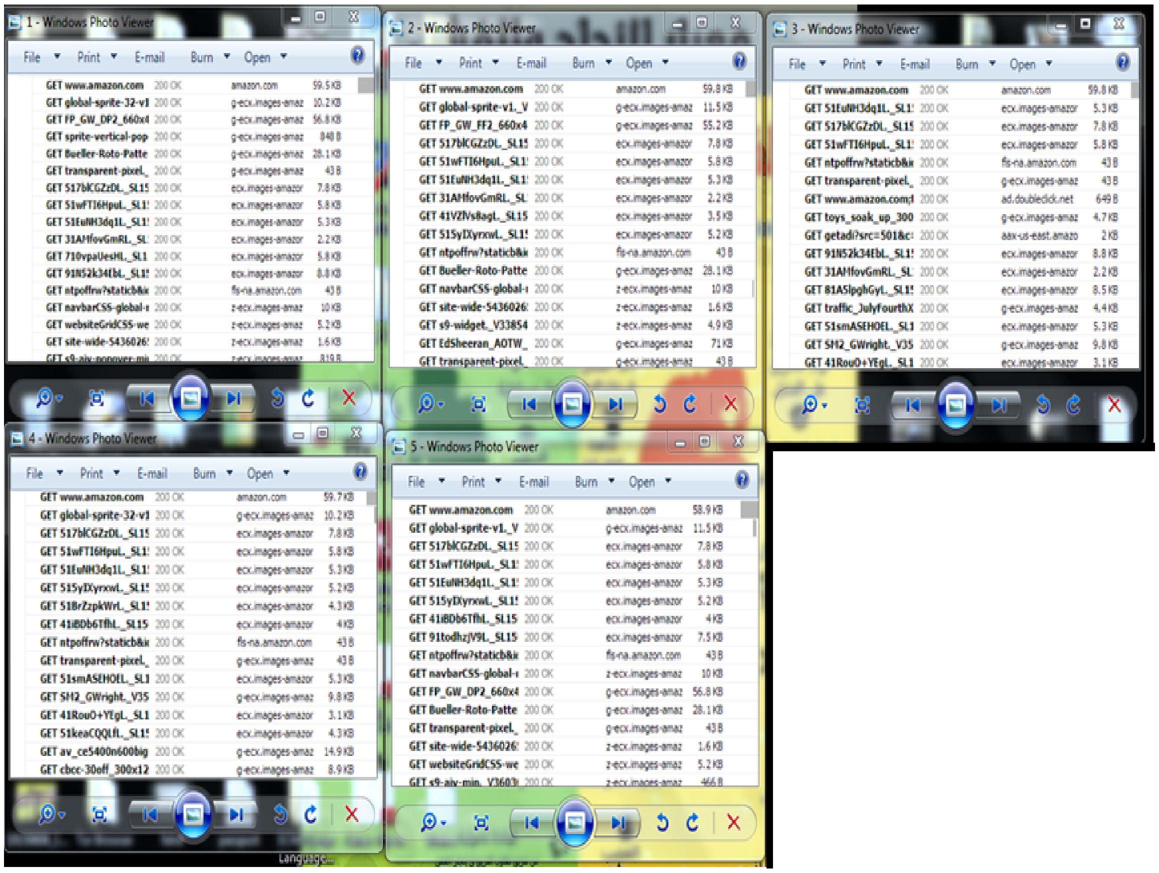
Figure 5.2: The various number of third-party servers retrieved by web browsers which affect their web browsing traffic patterns.

The browser, that allows all data coming from third-party servers, brings a high-value in its traffic pattern. As it is shown in Figure 5.2, that IE has the highest portion of the number of 603 contacted third-party servers. This is because IE allows all flash Ads contents to be loaded from third-party servers as we stated earlier that Adobe Flash is integrated by default with IE so it doesn't restrict flash ads coming from third-party servers. That is why IE the largest target to attackers who exploit any security vulnerabilities behind IE behavior. This factor affects the recognition rate of IE on website fingerprinting attack. This is clear in IE browser so while it has the highest number of retrieved third-party websites, it also has the highest recognition rate of website fingerprinting attack of 74% which means it has the least privacy protection compared to other browsers. Chrome filters pop-up ads while FF has sandbox security model to limit accessing data from other websites/third-party websites based on Same-Origin Policy (SOP) as it is shown in Figure 5.2 that Chrome retrieves web contents from 533 third-party servers followed by FF from 493. Safari retrieves stuff from 434 third-party services as it blocks third-party cookies so the third-party websites that require cookie to be enabled will be restricted. We noticed that Opera browser blocks all data coming from third-party websites as it has strong security restrictions such as Pop-ups blocking and cookie disabling for the sake of more security perspective. So Opera browser blocks all third-party domains so it has the least recognition rate of website fingerprinting attack of 41.60% which means it has the highest privacy protection against website fingerprinting attack as it is shown in Figure 4.2. The results prove that the largest number of third-party domains retrieved by a browser is the least privacy protection that this browser introduces. On the other hand, the least number of third-party domains fetched by a browser is the more privacy protection it has. We investigated the browsers' content features that affect the pattern of retrieved contents so the next subsections will evaluate browsers' performance features that have a great impact in the consistency of browsers' website visits.

5.2.3 The impact of retrieval aspects of browsers on the consistency of their traffic patterns

When a browser requests the URL visible in its address bar, the HTML document is retrieved with its embedded sub-resources (e.g. images, scripts, style-sheets, flashes, etc). However, requesting each element individually by establishing separated HTTP requests causes the retrieval process to be slow and accompanied with much traffic of TCP Acks and Syncs. To eliminate these performance issues a parallel download technique was proposed to optimize the retrieval time of websites so all browsers are permitted to open several simultaneous connections to load website contents in parallel.So the behavior of browsers in parallel downloads and loading time management is the main source of variations in the consistent order of packets as it is shown in Figure 5.3. In this section we evaluated the consistency between browsers' website visits/traces/fingerprints by visiting Amazon website 5 times with each browser and took the Average and Standard Deviation.

However, the web browsers may differ in their strategies of how they parallelize the retrieval of website contents and how they optimize the loading time. Therefore, these two features have a great impact on the stability/consistency between the lengths of website fingerprints/traces which affect the recognition rate of website fingerprinting on different browsers. In parallel downloads we observed



```
1 - Windows Photo Viewer
GET www.amazon.com       200 OK   amazon.com            59.5 KB
GET global-sprite-32-v1  200 OK   g-ecx.images-amaz     10.2 KB
GET FP_GW_DP2_660x4      200 OK   g-ecx.images-amaz     56.8 KB
GET sprite-vertical-pop  200 OK   g-ecx.images-amaz      848 B
GET Bueller-Roto-Patte   200 OK   g-ecx.images-amaz     28.1 KB
GET transparent-pixel_   200 OK   g-ecx.images-amaz       43 B
GET 517bICGZzDL._SL15    200 OK   ecx.images-amazor      7.8 KB
GET 51wFTI6HpuL._SL1!    200 OK   ecx.images-amazor      5.8 KB
GET 51EuNH3dq1L._SL1!    200 OK   ecx.images-amazor      5.3 KB
GET 31AMfovGmRL._SL'     200 OK   ecx.images-amazor      2.2 KB
GET 710vpaUesHL._SL1     200 OK   ecx.images-amazor      5.8 KB
GET 91N52k34EbL._SL1!    200 OK   ecx.images-amazor      8.8 KB
GET ntpoffrw?staticb&a   200 OK   fls-na.amazon.com       43 B
GET navbarCSS-global-i   200 OK   z-ecx.images-amaz       10 KB
GET websiteGridCSS-we    200 OK   z-ecx.images-amaz      5.2 KB
GET site-wide-5436026!   200 OK   z-ecx.images-amaz      1.6 KB
GET s9-aiv-popover-mi    200 OK   z-ecx.images-amaz      819 B
```

```
2 - Windows Photo Viewer
GET www.amazon.com       200 OK   amazon.com            59.8 KB
GET global-sprite-v1._V  200 OK   g-ecx.images-amaz     11.5 KB
GET FP_GW_FF2_660x4      200 OK   g-ecx.images-amaz     55.2 KB
GET 517bICGZzDL._SL15    200 OK   ecx.images-amazor      7.8 KB
GET 51wFTI6HpuL._SL1!    200 OK   ecx.images-amazor      5.8 KB
GET 51EuNH3dq1L._SL1!    200 OK   ecx.images-amazor      5.3 KB
GET 31AMfovGmRL._SL'     200 OK   ecx.images-amazor      2.2 KB
GET 41VZlVs8agL._SL15    200 OK   ecx.images-amazor      3.5 KB
GET 515yIXyrxwL._SL1!    200 OK   ecx.images-amazor      5.2 KB
GET ntpoffrw?staticb&a   200 OK   fls-na.amazon.com       43 B
GET Bueller-Roto-Patte   200 OK   g-ecx.images-amaz     28.1 KB
GET navbarCSS-global-i   200 OK   z-ecx.images-amaz       10 KB
GET site-wide-5436026!   200 OK   z-ecx.images-amaz      1.6 KB
GET s9-widget._V33854    200 OK   z-ecx.images-amaz      4.9 KB
GET EdSheeran_AOTW_      200 OK   g-ecx.images-amaz       71 KB
GET transparent-pixel_   200 OK   g-ecx.images-amaz       43 B
```

```
3 - Windows Photo Viewer
GET www.amazon.com       200 OK   amazon.com            59.8 KB
GET 51EuNH3dq1L._SL1!    200 OK   ecx.images-amazor      5.3 KB
GET 517bICGZzDL._SL15    200 OK   ecx.images-amazor      7.8 KB
GET 51wFTI6HpuL._SL1!    200 OK   ecx.images-amazor      5.8 KB
GET ntpoffrw?staticb&a   200 OK   fls-na.amazon.com       43 B
GET transparent-pixel_   200 OK   g-ecx.images-amaz       43 B
GET www.amazon.com;l     200 OK   ad.doubleclick.net     649 B
GET toys_soak_up_300     200 OK   g-ecx.images-amaz      4.7 KB
GET getadi?src=501&c:    200 OK   aax-us-east.amazo        2 KB
GET 91N52k34EbL._SL1!    200 OK   ecx.images-amazor      8.8 KB
GET 31AMfovGmRL._SL'     200 OK   ecx.images-amazor      2.2 KB
GET 81ASlpghGyL._SL1!    200 OK   ecx.images-amazor      8.5 KB
GET traffic_JulyFourthX  200 OK   g-ecx.images-amaz      4.4 KB
GET 51smASEHOEL._SL1     200 OK   ecx.images-amazor      5.3 KB
GET SH2_GWright._V35     200 OK   g-ecx.images-amaz      9.8 KB
GET 41RouO+YEgL._SL1     200 OK   ecx.images-amazor      3.1 KB
```

```
4 - Windows Photo Viewer
GET www.amazon.com       200 OK   amazon.com            59.7 KB
GET global-sprite-32-v1  200 OK   g-ecx.images-amaz     10.2 KB
GET 517bICGZzDL._SL15    200 OK   ecx.images-amazor      7.8 KB
GET 51wFTI6HpuL._SL1!    200 OK   ecx.images-amazor      5.8 KB
GET 51EuNH3dq1L._SL1!    200 OK   ecx.images-amazor      5.3 KB
GET 515yIXyrxwL._SL1!    200 OK   ecx.images-amazor      5.2 KB
GET 51BrZzpkWrL._SL1!    200 OK   ecx.images-amazor      4.3 KB
GET 41iBDb6TfhL._SL15    200 OK   ecx.images-amazor        4 KB
GET ntpoffrw?staticb&a   200 OK   fls-na.amazon.com       43 B
GET transparent-pixel_   200 OK   g-ecx.images-amaz       43 B
GET 51smASEHOEL._SL1     200 OK   ecx.images-amazor      5.3 KB
GET SH2_GWright._V35     200 OK   g-ecx.images-amaz      9.8 KB
GET 41RouO+YEgL._SL1     200 OK   ecx.images-amazor      3.1 KB
GET 51keaCQQLfL._SL1!    200 OK   ecx.images-amazor      4.3 KB
GET av_ce5400n600big     200 OK   g-ecx.images-amaz     14.9 KB
GET cbcc-30off_300x12    200 OK   g-ecx.images-amaz      8.9 KB
```

```
5 - Windows Photo Viewer
GET www.amazon.com       200 OK   amazon.com            58.9 KB
GET global-sprite-v1._V  200 OK   g-ecx.images-amaz     11.5 KB
GET 517bICGZzDL._SL15    200 OK   ecx.images-amazor      7.8 KB
GET 51wFTI6HpuL._SL1!    200 OK   ecx.images-amazor      5.8 KB
GET 51EuNH3dq1L._SL1!    200 OK   ecx.images-amazor      5.3 KB
GET 515yIXyrxwL._SL1!    200 OK   ecx.images-amazor      5.2 KB
GET 41iBDb6TfhL._SL15    200 OK   ecx.images-amazor        4 KB
GET 91todhzjV9L._SL15    200 OK   ecx.images-amazor      7.5 KB
GET ntpoffrw?staticb&a   200 OK   fls-na.amazon.com       43 B
GET navbarCSS-global-i   200 OK   z-ecx.images-amaz       10 KB
GET FP_GW_DP2_660x4      200 OK   g-ecx.images-amaz     56.8 KB
GET Bueller-Roto-Patte   200 OK   g-ecx.images-amaz     28.1 KB
GET transparent-pixel_   200 OK   g-ecx.images-amaz       43 B
GET site-wide-5436026!   200 OK   z-ecx.images-amaz      1.6 KB
GET websiteGridCSS-we    200 OK   z-ecx.images-amaz      5.2 KB
GET s9-aiv-min._V3603    200 OK   z-ecx.images-amaz      466 B
```

Figure 5.3: The inconsistency between the sequence order of retrieved objects for 5 visits to the same website.

a significant parallel download issue which affect on the consistency of browser traffic pattern extremely. The parallel download behavior of Opera differs from other browsers because it doesn't support || Script Image, || Script Stylesheet, and Asyncronous script execution features explained in Table 3.3. Therefore, when Opera retrieves an external script, it blocks all other downloads until the script is loaded, parsed and executed. The waterfall chart in Figures 5.4 and 5.5 show the staircase pattern where there are some intervals of JSs that block Opera from requesting website objects in parallel.

Thus, Opera does not support requesting JSs with other objects in parallel for

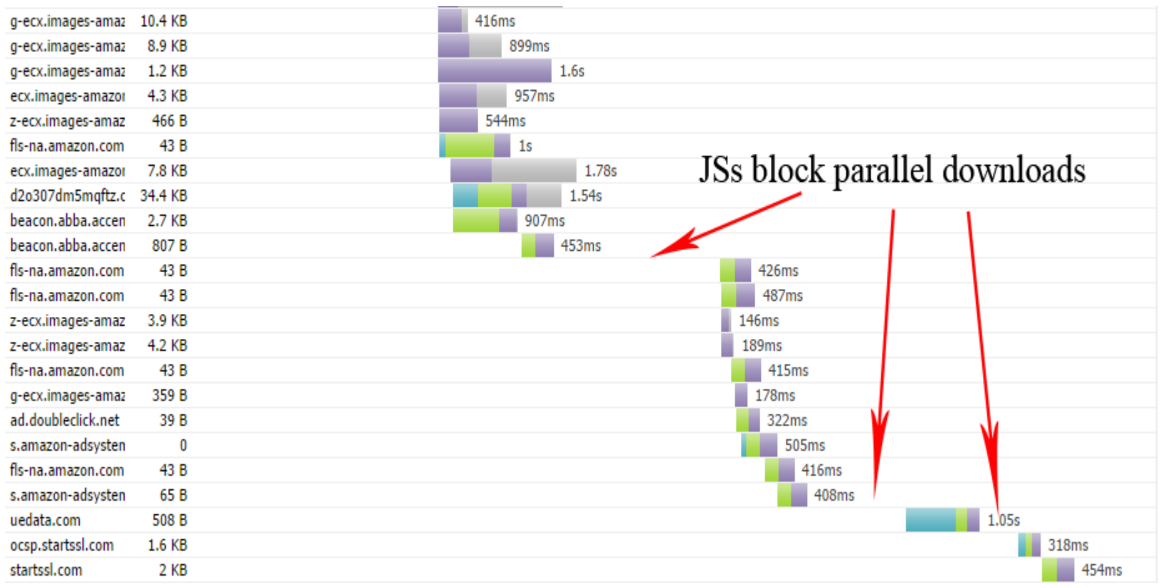


Figure 5.4: The impact of JavaScripts blocking on web browsing traffic pattern of Opera.

the sake of security concern so this parallel download issue has a great impact on the stability of its traffic pattern. Therefore, the misbehaving behavior of Opera that is caused by its parallel download makes its traffic patterns the most diversity compared to other browsers as it is shown in Standard Deviation of its websites visits in Figure 5.6. As a result, Opera has the least recognition rate in website fingerprinting attack compared to other browsers as shown in Section 4.3, Figure 4.2.

Different browsers implement different logic of retrieval optimization such as when the individual requests are dispatched so this will be reflected to the stability/consistancy between the lengths of websites visits/traces. However, the recent implementation of the W3C Navigation Timing API specification [77] that is supported by Chrome, FF and IE add a great performance optimization to their retrieval time. What can't be measured it can't be optimized so the browser that

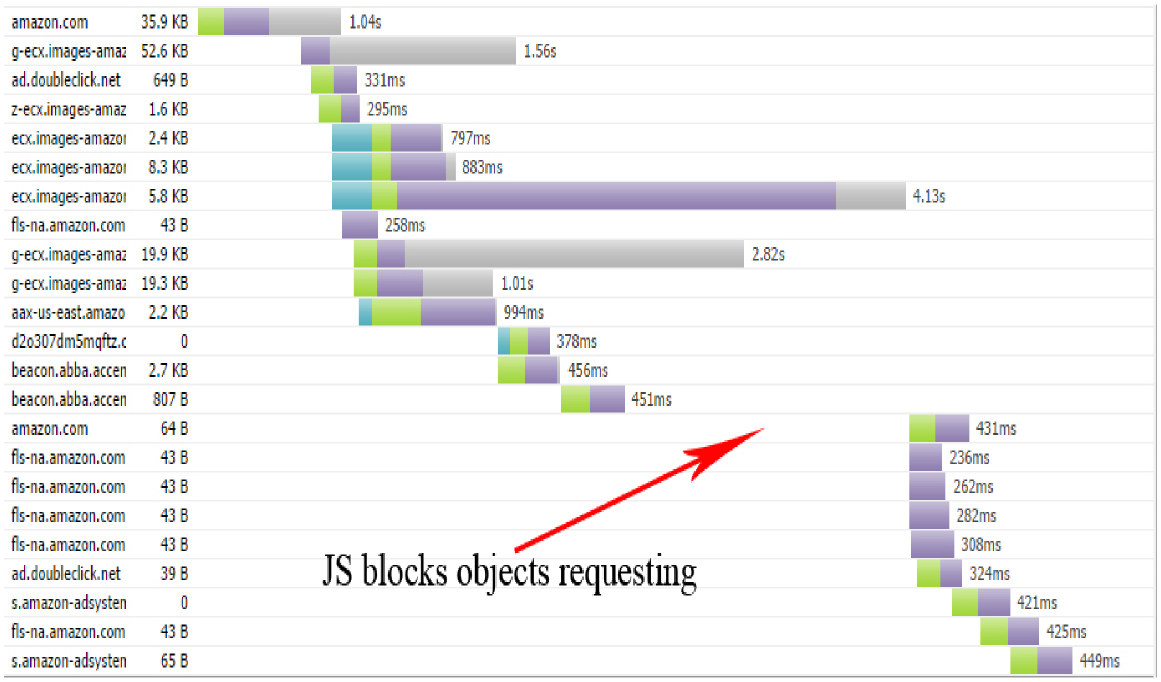


Figure 5.5: The impact of JavaScripts blocking on web browsing traffic pattern of Opera.

supports Navigation Timing feature, a major development will be added to its functionalities. This feature provides browsers with fine-grained measurements about real browsing timings such as (e.g. TCP connection, timing information related to loaded elements, etc) further information behind this feature found in [33]. The performance metrics supported by Navigation Timing optimize the retrieval behavior of browsers that support it which is reflected clearly to the shape of their traffic patterns as it is illustrated in Figure 5.6. The results show that the browsers have various average loading times and different inconsistencies between the retrieval times of their website visits. We evaluated the consistency between the retrieval periods of website visits by calculating the Standard Deviation of several website visits.

Figure 5.6 shows that Opera and Safari have the largest Standard Deviation

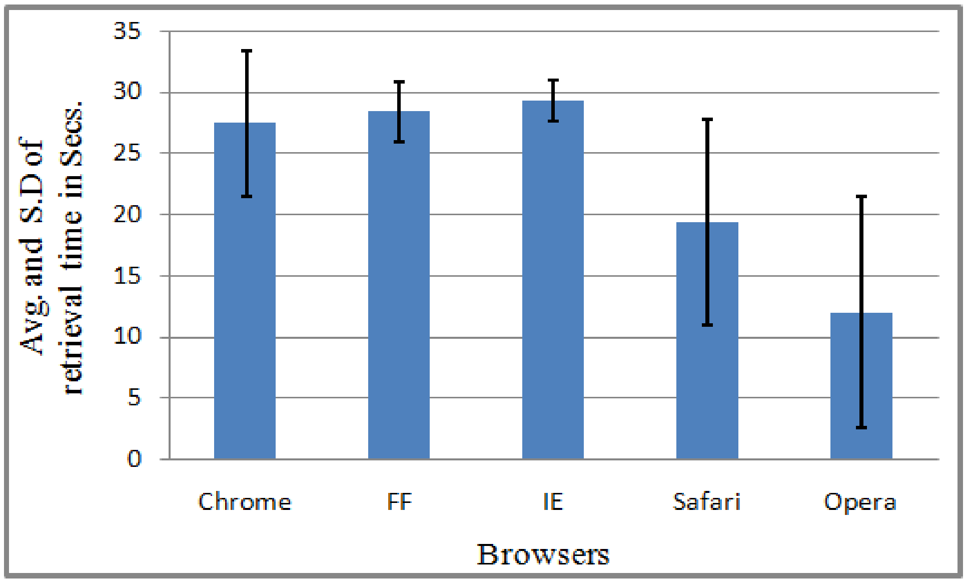
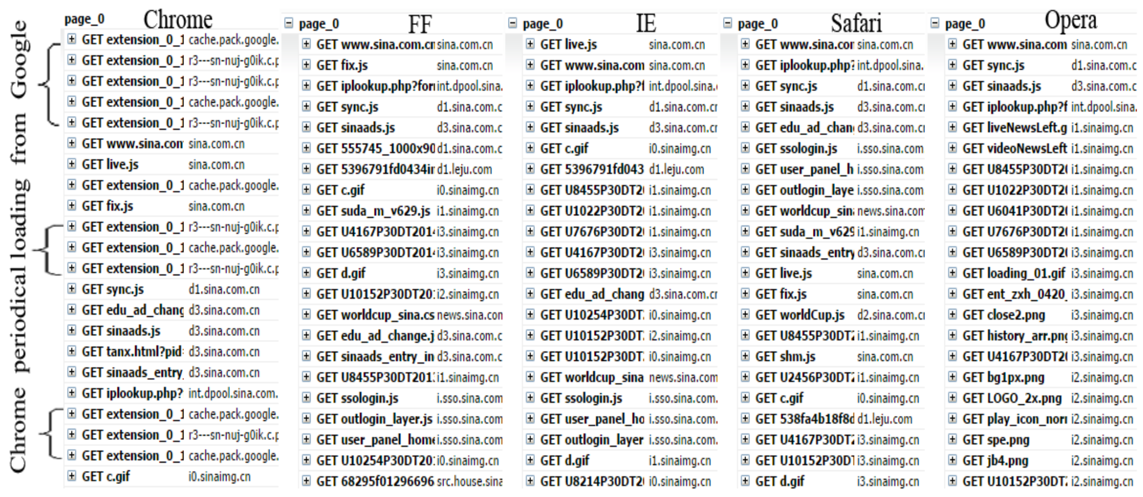


Figure 5.6: The variations in retrieval time of browsers and their impact on the consistency of web browsing traffic patterns.

across browsers as they don't support Navigation Timing feature. So the lack of Opera and Safari to the efficiency of this timing feature is reflected to the consistency of their website traces as it is shown in Figure 5.6. Furthermore, this is reflected also to their accuracy on website fingerprinting attack as it is shown in Figure 4.2. Beside the lack of Navigation Timing feature, Opera also has the parallel download issue mentioned above. So Opera has the highest inconsistency between the length of its website traces compared to other browsers because of its largest Standard Deviation value shown in Figure 5.6 which means the largest diversity between the lengths of its website traces/fingerprints. As a result, Opera has the least recognition rate of website fingerprinting attack compared to other browsers as it is shown in Figure 4.2.

The consistency between website traces on IE and FF is approximately the same as they share the support of Asynchronous script execution and Navigation

Timing features. For further information behind these features see Table 3.3. IE has the most regularity between the lengths of its website traces so this behavior is reflected to its resistance to traffic analysis attack where IE has the largest recognition rate in website fingerprinting compared to other browsers as it is shown in Figure 4.2. The consistency between website visits over Chrome is more diversity than FF and IE. Although, they share the same features (Asynchronous script execution and Navigation Timing) but we have noticed that in some website visits Chrome downloads stuff from Google's servers which are maintained periodically (e.g. the most recent Safe Browsing list maintained by Google related to blacklist of websites that Chrome must avoid and it is updated continuously, apps and themes) in a form of "application=x-chrome-extension" content-type. This behavior of Chrome is illustrated website retrieval in the most left part of Chrome in waterfall chart in Figure 5.7. This behavior of Chrome makes its website visits more inconsistency than FF and IE.



Chrome

```
page_0
GET extension_0_1 cache.pack.google.
GET extension_0_1 r3---sn-nuj-g0ik.c.p
GET extension_0_1 r3---sn-nuj-g0ik.c.p
GET extension_0_1 cache.pack.google.
GET extension_0_1 r3---sn-nuj-g0ik.c.p
GET www.sina.con  sina.com.cn
GET live.js       sina.com.cn
GET extension_0_1 cache.pack.google.
GET fix.js        sina.com.cn
GET extension_0_1 r3---sn-nuj-g0ik.c.p
GET extension_0_1 cache.pack.google.
GET extension_0_1 r3---sn-nuj-g0ik.c.p
GET sync.js       d1.sina.com.cn
GET edu_ad_chang  d3.sina.com.cn
GET sinaads.js    d3.sina.com.cn
GET tanx.html?pid d3.sina.com.cn
GET sinaads_entry d3.sina.com.cn
GET iplookup.php? int.dpool.sina.com.
GET extension_0_1 cache.pack.google.
GET extension_0_1 r3---sn-nuj-g0ik.c.p
GET extension_0_1 cache.pack.google.
GET c.gif         i0.sinaimg.cn
```

FF

```
page_0
GET www.sina.com.cn  sina.com.cn
GET fix.js           sina.com.cn
GET iplookup.php?for int.dpool.sina.
GET sync.js          d1.sina.com.c
GET sinaads.js       d3.sina.com.c
GET 555745_1000x90   d1.sina.com.c
GET 5396791fd0434ir  d1.leju.com
GET c.gif            i0.sinaimg.cn
GET suda_m_v629.js   i1.sinaimg.cn
GET U4167P30DT201    i3.sinaimg.cn
GET U6589P30DT201    i3.sinaimg.cn
GET d.gif            i3.sinaimg.cn
GET U10152P30DT20    i2.sinaimg.cn
GET worldcup_sina.cs news.sina.con
GET edu_ad_change.j  d3.sina.com.c
GET sinaads_entry_in d3.sina.com.c
GET U8455P30DT201    i1.sinaimg.cn
GET ssologin.js      i.sso.sina.com
GET outlogin_layer.js i.sso.sina.com
GET user_panel_home  i.sso.sina.com
GET U10254P30DT20    i0.sinaimg.cn
GET 68295f01296696   src.house.sina
```

IE

```
page_0
GET live.js        sina.com.cn
GET www.sina.com   sina.com.cn
GET iplookup.php?f int.dpool.sina.
GET sync.js        d1.sina.com.cr
GET sinaads.js     d3.sina.com.cr
GET c.gif          i0.sinaimg.cn
GET 5396791fd043   d1.leju.com
GET U8455P30DT2    i1.sinaimg.cn
GET U1022P30DT2    i1.sinaimg.cn
GET U7676P30DT2    i1.sinaimg.cn
GET U4167P30DT2    i3.sinaimg.cn
GET U6589P30DT2    i3.sinaimg.cn
GET edu_ad_chang   d3.sina.com.cr
GET U10254P30DT.   i0.sinaimg.cn
GET U10152P30DT.   i2.sinaimg.cn
GET U10152P30DT.   i0.sinaimg.cn
GET worldcup_sina  news.sina.com
GET ssologin.js    i.sso.sina.com.
GET user_panel_ho  i.sso.sina.com.
GET outlogin_layer i.sso.sina.com.
GET d.gif          i1.sinaimg.cn
GET U8214P30DT2    i0.sinaimg.cn
```

Safari

```
page_0
GET www.sina.cor   sina.com.cn
GET iplookup.php?  int.dpool.sina.
GET sync.js        d1.sina.com.cı
GET sinaads.js     d3.sina.com.cı
GET edu_ad_chan    d3.sina.com.cı
GET ssologin.js    i.sso.sina.com
GET user_panel_h   i.sso.sina.com
GET outlogin_laye  i.sso.sina.com
GET worldcup_sin   news.sina.com
GET suda_m_v629    i1.sinaimg.cn
GET sinaads_entry  d3.sina.com.cı
GET live.js        sina.com.cn
GET fix.js         sina.com.cn
GET worldCup.js    d2.sina.com.cı
GET U8455P30DT     i1.sinaimg.cn
GET shm.js         sina.com.cn
GET U2456P30DT     i1.sinaimg.cn
GET c.gif          i0.sinaimg.cn
GET 538fa4b18f8d   d1.leju.com
GET U4167P30DT     i3.sinaimg.cn
GET U10152P30D     i3.sinaimg.cn
GET d.gif          i3.sinaimg.cn
```

Opera

```
page_0
GET www.sina.com   sina.com.cn
GET sync.js        d1.sina.com.c
GET sinaads.js     d3.sina.com.c
GET iplookup.php?f int.dpool.sina.
GET liveNewsLeft.g i1.sinaimg.cn
GET videoNewsLeft  i1.sinaimg.cn
GET U8455P30DT2    i1.sinaimg.cn
GET U1022P30DT2    i1.sinaimg.cn
GET U6041P30DT2    i1.sinaimg.cn
GET U7676P30DT2    i1.sinaimg.cn
GET U6589P30DT2    i3.sinaimg.cn
GET loading_01.gif i3.sinaimg.cn
GET ent_zxh_0420_  i3.sinaimg.cn
GET close2.png     i3.sinaimg.cn
GET history_arr.pn i3.sinaimg.cn
GET U4167P30DT2    i3.sinaimg.cn
GET bg1px.png      i2.sinaimg.cn
GET LOGO_2x.png    i2.sinaimg.cn
GET play_icon_nor  i2.sinaimg.cn
GET spe.png        i2.sinaimg.cn
GET jb4.png        i2.sinaimg.cn
GET U10152P30DT    i2.sinaimg.cn
```

Figure 5.7: The automatic periodical loading of Chrome from Google's servers.

The figure shows the automatic periodical updates of Chrome by Google which occur continually. This distinctive behavior of Chrome affect the website fingerprinting accuracy so these updates (e.g. Safe Browsing List and themes ”CSS filters”) are triggered in some website visits randomly.

In website fingerprinting attack the consistency or inconsistency between website fingerprints/log traces have a great impact on the accuracy of website fingerprinting attack. When it comes to the classification phase the similarity between website fingerprints is calculated using Damerau-Levenshtein distance algorithm as it is described in Section 4.3. So this algorithm depends on matching the integers of the traces (-1150, 1500, - 638, 638, etc) to calculate the similarity between websites fingerprints. Therefore, the variations between the lengths of traces/fingerprints has a great impact on the accuracy of website fingerprinting.In data collection phase of our website fingerprinting attack we automated the browsers for websites visits and we noticed that there are several number of websites which were not visited by browsers so their log files were left empty. Figure 5.8 shows the various number of empty log files that are collected over browsers.

The characteristic of internet network is not stable as there are several network problems that can be happened in any second either in communication medium or server side such as DNS resolving, Server loading, etc. Because our attack is real scenario so we left everything as it is by considering the empty log files to evaluate the real browsing behavior of browsers. As it is shown in Figure 5.8 that there are approximately 20 log files that are left empty under Chrome, FF, IE and Safari. The sharing of the four browsers to this behavior indicate that it is caused by the

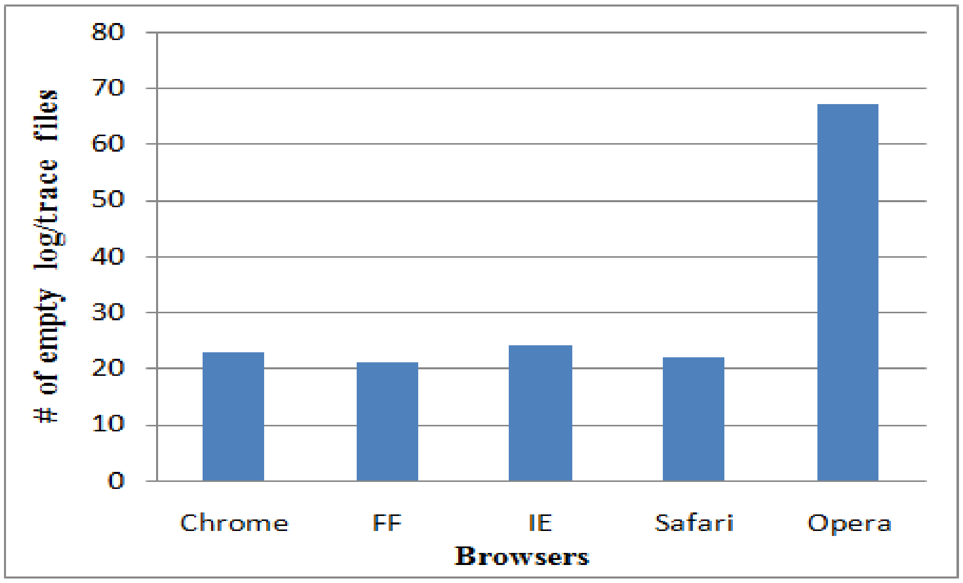
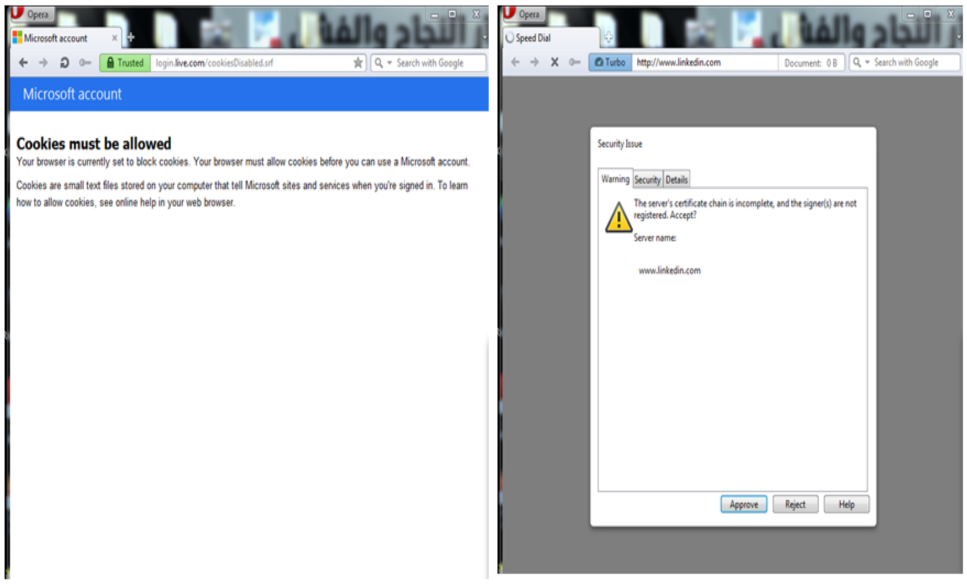


Figure 5.8: The number of empty log files collected over browsers during data collection phase which affect the website fingerprinting accuracy of browsers.

variation of internet network conditions. Opera has a distinctive number of 67 empty log traces so this is not comparable to other browsers which we collected their log traces on the same platform conditions and same data set. The reason is that the results show that Opera has two significant security features that are enabled by default which stand behind the unique behavior of Opera. The screen shots that are captured during the automated data collection of Opera are shown in Figure 5.9.

As it is shown in the figure that the left screen shot presents the blocked cookie of Opera which is disabled by default. Some websites doesn't allow to be browsed without the enabled cookie feature of the browser so those websites were not visited and cause the empty log files of Opera. The right screen shot illustrates the security certificate issue which is caused by strict security behavior of Opera to verify trusted websites. The warning dialog shows a question about website



Cookies must be allowed

Your browser is currently set to block cookies. Your browser must allow cookies before you can use a Microsoft account.

Cookies are small text files stored on your computer that tell Microsoft sites and services when you're signed in. To learn how to allow cookies, see online help in your web browser.

The server's certificate chain is incomplete, and the signer(s) are not registered. Accept?

Server name:

www.linkedin.com

Figure 5.9: The security features that caused the most empty log traces of Opera.

visit rejection or bypassing the certificate warning to visit a website which the full security cannot be guaranteed. In this case there wasn't not any response to the warning dialog message trigged by Opera so the 25 seconds interval that we set for each website visit was fired and the website wasn't visited as a result the log file trace left empty. To sum up, in this chapter we have conducted extensive analysis tests to figure out the root causes that stand behind different resistance levels of browsers against website fingerprinting attack. There are some external factors that may affect the behavior of web browsers which will be discussed in the subsequent sections.

5.3 Discussion

There is no any research in the literature that evaluates the behaviors and functionalities of the most commonly used web browsers from traffic analysis perspective. The traffic patterns of browsers may change due to some network conditions and dynamic behavior of server-side. The following subsections discuss the factors that affect the recognition rate of web browsers in website fingerprinting attack.

5.3.1 Dynamic contents

The contents of websites may change over time so the same web page would not have the same objects every day. For example, video website updates itself constantly based on the most popular viewed videos that are recommended for its visitors. Moreover, news websites change their contents more than daily. Therefore, these dynamic changes may reduce the recognition rate of websites fingerprints across browsers. This impact is limited because the change will be in some contents of a web page rather than its template/structure which may be changed infrequently.

5.3.2 Localization

There is another factor that may reduce the recognition rate of websites fingerprints which is known as website localization so the visits of same website may result in different contents depending on its locality. For example, the contents of "www.google.com.sa" is different from "www.google.com.in" contents depends on specified localization so in our website fingerprinting attack we have considered localization factor. As it is shown in Figure 5.6 that the contents of the same website is different based on its locality. Fortunately, in the Figures tow ads from

Google that are provided based on the interest of different localizations.

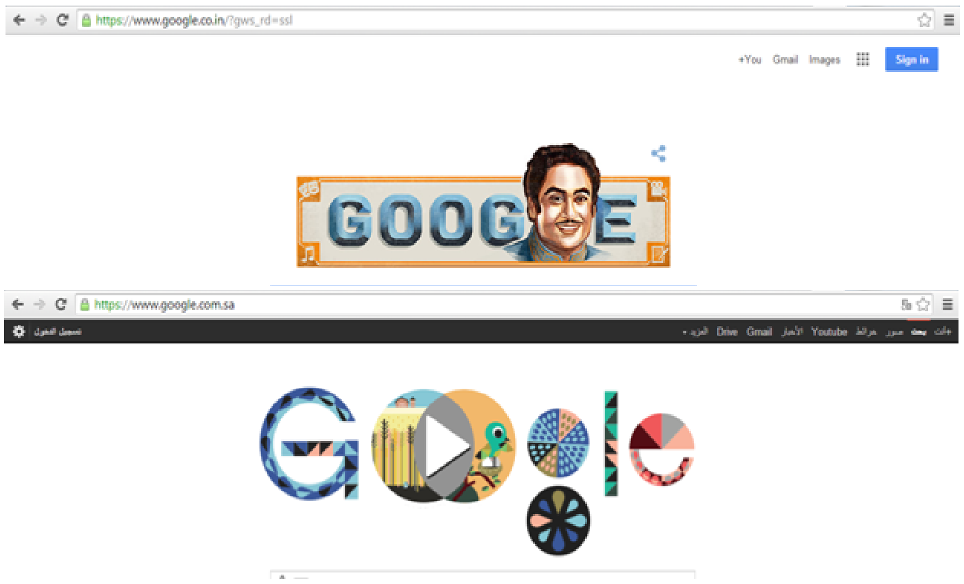

https://www.google.co.in/?gws_rd=ssl
+You
Gmail
Images
Sign in
https://www.google.com.sa
Drive
Gmail
Youtube

Figure 5.10: Different contents for the same website that shows website localization.

It is worth mentioning that when Tor is used the locality is determined by the exit relay of Tor which is selected randomly rather than by the client's location for the sake privacy protection (Anonymity). Therefore, when Tor is used the locality approach affects the website fingerprinting attack results. This happens when the locality of a target website is not specified exactly in the address bar of the browser such as www.google.com without specifying if it is "www.google.com.sa" or "www.google.com.in". As a result, we argue that the accuracy of website fingerprinting attack on Tor will be affected on websites that don't specify the locality of a target domain.

5.3.3 Recommendation

Our results show that the active contents such as JavaScript and CSSs features make the web pages fingerprints more recognizable. Their impact is more obvious in IE so it has the highest recognition rate of website fingerprinting compared to other browsers. So we argue that the most obvious resistance against website fingerprinting attack is to disable all active contents in web browsers. The disadvantage of this defense method is that many web services will be disabled but it will protect web user privacy against traffic analysis attack substantially. These active contents are real threat to web user privacy in anonymous web-browsing thus anonymous systems like Tor warn its users to disable active contents in their browsers.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis we investigated our web page fingerprinting attack to study various resistances of popular web browsers by detecting their network traffic patterns using traffic analysis. Our attack significantly outperforms previous attacks in tow folds: First, we implemented web page fingerprinting attack on the most commonly used web browsers which outperforms previously proposed attacks that used only a single browser "Firefox". Second, we investigated the browser-dependent features to uncover the underlying causes that stand behind different resistances of the top five browsers. Our website fingerprinting attack can determine which web page a victim may visit with a success rate of 74%, 70.4%, 69.6%, 53.8% and 41.6% by IE, FF, Chrome, Safari and Opera respectively. From our results we conclude that the least privacy protection can be introduced by IE as we got the highest recognition rate compared to other browsers. FF and Chrome approximately have the same resistance. The highest protection against traffic analysis attack is introduced by Opera followed by Safari because of their own dependent causes mentioned above. The aim of applying website fingerprinting attack using top browsers is to empower web users with the awareness of privacy protection that would provide them with the feedback about the level of anonymity that can be introduced by each browser and which browser that can add adequate anonymity. In future work we plan to further investigate on more fine-grained web pages dataset to evaluate to which range each browser protects against websites fingerprinting attack on different websites classes (e.g. business websites, news websites, social networking sites, Forum websites, Gallery Websites, Gaming websites, Search engine sites, etc). Furthermore, a hot research issue is that this research can be applied on more restricted mobile browsers (e.g. Android, Dolphin, etc.) since a lot of people today use their smart devices like mobile phones and tablets to browse internet in their daily life. Thus, their privacy can be endangered with the recent capturing tools that can eavesdrop their traffic wirelessly. Future work should include additional features other than of desktop browses which fit the restriction of mobile browsers in order to raise the privacy awareness of mobile users.

REFERENCES

[1] Timothy G Abbott, Katherine J Lai, Michael R Lieberman, and Eric C Price. Browser-based attacks on tor. In *Privacy Enhancing Technologies*, pages 184–199. Springer, 2007.

[2] Amazon. *"Alexa website: The web information company"*. http://www.alexa.com/topsites, 2014.

[3] Erik Archambault and Craig Shue. Understanding new anonymity networks from a user's perspective. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 995–997. ACM, 2012.

[4] Jordan Nielson Carey Williamson Martin Arlitt. Benchmarking modern web browsers.

[5] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM, 2007.

[6] Tim Berners-Lee. *"WWW FAQs: What was the first Web browser?"*. http://www.w3.org/TR/html5/forms.html, 2006.

[7] Tim Berners-Lee. *"Web browser engine"*. http://en.wikipedia.org/wiki/Layout_engine, 2014.

[8] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web mixes: A system for anonymous and unobservable internet access. In *Designing Privacy Enhancing Technologies*, pages 115–129. Springer, 2001.

[9] Eric Bidelman. *"Stream Updates with Server-Sent Events"*. http://www.html5rocks.com/en/tutorials/eventsource/basics/, 2010.

[10] Nataliia Bielova. Survey on javascript security policies and their enforcement mechanisms in a web browser. *The Journal of Logic and Algebraic Programming*, 82(8):243–262, 2013.

[11] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *Privacy Enhancing Technologies*, pages 1–11. Springer, 2006.

[12] Tim Brown. *"Type rendering: web browsers"*. http://blog.typekit.com/2010/10/21/type-rendering-web-browsers/, 2013.

[13] Michael Butkiewicz, Harsha V Madhyastha, and Vyas Sekar. Understanding website complexity: measurements, metrics, and implications. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 313–328. ACM, 2011.

[14] Xiang Cai, Rishab Nithyanand, and Rob Johnson. New approaches to website fingerprinting defenses. *arXiv preprint arXiv:1401.6022*, 2014.

[15] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 605–616. ACM, 2012.

[16] Internet Explorer Dev Center. *"Asynchronous script execution"*. http://msdn.microsoft.com/en-us/library/ie/hh673524(v=vs.85).aspx, August 2014.

[17] Darren Charters. Electronic monitoring and privacy issues in business-marketing: The ethics of the doubleclick experience. *Journal of business ethics*, 35(4):243–254, 2002.

[18] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[19] Hyunyi Cho and Robert LaRose. Privacy issues in internet surveys. *Social Science Computer Review*, 17(4):421–434, 1999.

[20] Shauvik Roy Choudhary, Mukul R Prasad, and Alessandro Orso. Cross-check: Combining crawling and differencing to better detect cross-browser incompatibilities in web applications. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, pages 171–180. IEEE, 2012.

[21] Shauvik Roy Choudhary, Husayn Versee, and Alessandro Orso. Webdiff: Automated identification of cross-browser issues in web applications. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[22] Ravi Chugh, Jeffrey A Meister, Ranjit Jhala, and Sorin Lerner. Staged information flow for javascript. In *ACM Sigplan Notices*, volume 44, pages 50–62. ACM, 2009.

[23] Winnie Chung and John Paynter. Privacy issues on the internet. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2002.

[24] Roger Clarke. *"Introduction to Dataveillance and Information Privacy, and Definitions of Terms"*. http://www.anu.edu.au/people/Roger.Clarke/DV/Intro.html, 2013.

[25] Ryan Michael Craven. *Traffic analysis of anonymity systems*. PhD thesis, Clemson University, 2010.

[26] creativecommons. *"community-driven project for profiling web browsers"*. http://www.browserscope.org/, 2014.

[27] Mary J Culnan. " how did they get my name?": An exploratory investigation of consumer attitudes toward secondary information use. *Mis Quarterly*, 17(3), 1993.

[28] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[29] Delapouite. *"<iframe> element"*. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe, 2014.

[30] Gurpreet S Dhillon and Trevor T Moores. Internet privacy: Interpreting key issues. *Advanced topics in information resources management*, pages 52–61, 2003.

[31] Claudia Díaz. *Anonymity metrics revisited.* Internat. Begegnungs-und Forschungszentrum für Informatik, 2006.

[32] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[33] Sam Dutton. *"Measuring Page Load Speed with Navigation Timing"*. http://www.html5rocks.com/en/tutorials/webperformance/basics/, October 2013.

[34] Tali Garsiel and Paul Irish. How browsers work: Behind the scenes of modern web browsers. *Slate, 5th August*, 2011.

[35] Henrik Gemal. *"community-driven project for profiling web browsers"*. http://browserspy.dk/, 2014.

[36] StatCounter GlobalStats. *"Browsers Statistics"*. http://gs.statcounter.com/, May 2013.

[37] StatCounter GlobalStats. *"Compatibility tables for support of HTML5 and JSs in browsers"*. https://http://caniuse.com/, 2014.

[38] Google.

[39] Arjun Guha, Shriram Krishnamurthi, and Trevor Jim. Using static analysis for ajax intrusion detection. In *Proceedings of the 18th international conference on World wide web*, pages 561–570. ACM, 2009.

[40] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. 2009.

[41] Ariya Hidayat. *"JavaScript Engines: How to Compile Them"*. http://www.sencha.com/blog/javascript-engines-how-to-compile-them/, 2010.

[42] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies*, pages 171–178. Springer, 2003.

[43] Erik Hjelmvik. *"Detecting TOR Communication in Network Traffic"*. http://www.netresec.com/?page=Blog&month=2013-04&post=Detecting-TOR-Communication-in-Network-Traffic, 2013.

[44] Brian P Hogan. Html5 and css3. *Develop with Tomorrow's Standards Today. The Pragmatic Programmers*, 2010.

[45] Colin Ihrig. *"Profiling Page Loads with the Navigation Timing API"*. http://www.sitepoint.com/profiling-page-loads-with-the-navigation-timing-api, September 2012.

[46] Francis Jayakanth. Web servers, browsers, server - browser interaction, web surfing. National Centre for Science Information (NCSI),Indian Institute of Science Bangalore - 560 012, 2011.

[47] Simon Holm Jensen, Magnus Madsen, and Anders Møller. Modeling the html dom and browser api in static analysis of javascript web applications. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 59–69. ACM, 2011.

[48] JENSL. *"Carakan"*. http://web.archive.org/web/20090206133936/http://my.opera.com/core/ February 2009.

[49] jrandom. *"I2P: A scalable framework for anonymous communication"*. http://geti2p.net/en/docs/how/tech-intro, 2014.

[50] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263. ACM, 2006.

[51] Feng Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[52] Sergio Maffeis, John C Mitchell, and Ankur Taly. Isolating javascript with filters, rewriting, and wrappers. In *Computer Security–ESORICS 2009*, pages 505–522. Springer, 2009.

[53] By BHARATH MARRIVADA. *"Browser wars and End user performance, content display impact"*. http://bharath-marrivada.blogspot.com/2010/09/browser-wars-speed-test-performance.html, 2014.

[54] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Privacy Enhancing Technologies*, pages 17–34. Springer, 2005.

[55] Florence Maurice. *HTML und CSS*, volume 24578. Pearson Deutschland GmbH, 2010.

[56] Carlo Maria Medaglia and Alexandru Serbanati. An overview of privacy and security issues in the internet of things. In *The Internet of Things*, pages 389–395. Springer, 2010.

[57] Ali Mesbah and Mukul R Prasad. Automated cross-browser compatibility testing. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 561–570. ACM, 2011.

[58] Bharat Mishra, Harish Singh Baghel, Manoj Patil, and Pramod Singh. Study & analysis of various protocols in popular web browsers. *Interna-*

tional Journal of Advancements in Research and Technology, vol. 1, no. 3, p. 8-14., 1(3):7, 2012.

[59] Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, and FH Campus Wien. Fast and reliable browser identification with javascript engine fingerprinting. In *Web 2.0 Workshop on Security and Privacy (W2SP).*

[60] Steven J Murdoch and Robert NM Watson. Metrics for security and performance in low-latency anonymity systems. In *Privacy Enhancing Technologies*, pages 115–132. Springer, 2008.

[61] Jordan Nielson, Carey Williamson, and Martin Arlitt. Benchmarking modern web browsers. In *2nd IEEE Workshop on Hot Topics in Web Systems and Technologies*. Citeseer, 2008.

[62] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, 2011.

[63] Paul Pardi. *"Internet Explorer Security Settings - ActiveX Controls"*. http://www.brighthub.com/internet/security-privacy/articles/1968.aspx, May 2011.

[64] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymitya proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

[65] Tor project. *"Tor Metrics Portal"*. https://metrics.torproject.org/, 2012.

[66] Tor Project. *"Tor: Overview"*. http://www.torproject.org/about/overview.html.en, 2014.

[67] Lee Rainie, Sara Kiesler, Ruogu Kang, Mary Madden, Maeve Duggan, Stephanie Brown, and Laura Dabbish. Anonymity, privacy, and security online. *Pew Research Center*, 2013.

[68] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[69] Gregor Richards, Sylvain Lebresne, Brian Burg, and Jan Vitek. An analysis of the dynamic behavior of javascript programs. In *ACM Sigplan Notices*, volume 45, pages 1–12. ACM, 2010.

[70] Yi Shi and Kanta Matsuura. Fingerprinting attack on the tor anonymity system. In *Information and Communications Security*, pages 425–438. Springer, 2009.

[71] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.

[72] Boukari Souley and Amina S Sambo. A comparative performance analysis of popular internet browsers in current web applications. *West African Journal of Industrial and Academic Research*, 4(1):62–68, 2013.

[73] Paul F Syverson, David M Goldschlag, and Michael G Reed. Anonymous connections and onion routing. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 44–54. IEEE, 1997.

[74] taligarsiel. *"The anonymizer"*. http://www.anonymizer.com/, February 2013.

[75] P. Palfrader U. Moller, L. Cottrell and L. Sassaman. *"Mixmaster protocol version 2"*. https://tools.ietf.org/html/draft-sassaman-mixmaster-03, 2004.

[76] Ellen Vanderhoven and Tammy Schellens. The role of parents, media, teachers and peers in raising the awareness of privacy-issues on social networks. In *Designing Learning Futures: Digital Media & Learning Conference (DML-2011)*, 2011.

[77] Philipp Vogt, Florian Nentwich, Nenad Jovanovic, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. In *NDSS*, 2007.

[78] W3C. *"Navigation Timing"*. https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/NavigationTiming/Overview.html, January 2013.

[79] W3C. *"Content Security Policy"*. http://dev.w3.org/html5/spec/single-page.html#history, 2014.

[80] W3C. *"Document Object Model (DOM)"*. http://www.w3.org/DOM/, May 2014.

[81] W3C. *"World Wide Web Consortium"*. http://www.w3.org, May 2014.

[82] Tao Wang and Ian Goldberg. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 201–212, New York, NY, USA, 2013. ACM.

[83] Wei Wang, Mehul Motani, and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 323–332. ACM, 2008.

[84] Samuel D Warren and Louis D Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[85] Shiyi Wei and Barbara G Ryder. A practical blended analysis for dynamic features in javascript. 2012.

[86] Weka. *"Weka 3: Data Mining Software in Java"*. http://www.anonymizer.com/, 2014.

[87] H Joseph Wen, Houn-Gee Chen, and Hsin-Ginn Hwang. E-commerce web site design: strategies and models. *Information management and computer security*, 9(1):5–12, 2001.

[88] Mike West. *"HTML 5 Server Sent Events on Glassfish 4"*. http://en.kodcu.com/2013/11/jaxrs-2-html-5-server-sent-events-on-glassfish-4/, 2013.

[89] Mike West. *"Play safely in sandboxed IFrames"*. http://www.html5rocks.com/en/tutorials/security/sandboxed-iframes/, 2013.

[90] Wikipedia.

[91] wikipedia. *"Content Security Policy"*. http://en.wikipedia.org/wiki/Content_Security_Policy, May 2014.

[92] wikipedia. *"HTML element"*. http://en.wikipedia.org/wiki/HTML_element#Frames, 2014.

[93] wikipedia. *"Internet censorship in Iran"*. http://en.wikipedia.org/wiki/Internet_censorship_in_Iran, May 2014.

[94] wikipedia. *"Server-sent events"*. http://en.wikipedia.org/wiki/Server-sent_events, 2014.

[95] wikipedia. *"Tor (anonymity network)"*. http://en.wikipedia.org/wiki/Tor_(anonymity_network), 2014.

[96] wikipedia. *"wikipedia List of websites blocked in China"*. http://en.wikipedia.org/wiki/List_of_websites_blocked_in_China, May 2014.

[97] Dachuan Yu, Ajay Chander, Nayeem Islam, and Igor Serikov. Javascript instrumentation for browser security. In *ACM SIGPLAN Notices*, volume 42, pages 237–249. ACM, 2007.

[98] Nicholas C. Zakas. *"How many users have JavaScript disabled?"*. https://developer.yahoo.com/blogs/ydn/many-users-javascript-disabled-14121.html, 2010.

[99] Michal Zalewski. *"Browser Security Handbook, part 1"*. https://code.google.com/p/browsersec/wiki/Part1, 2009.

[100] Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 514–524. IEEE, 2005.

[101] Xiaoyu Zhuang. "interaction between web browsers and script engines". 2012.

Vitae

- Name: Taher Ali Yahya Al-shehari
- Email: *taherali599@gmail.com*

Taher Al-Shehari was born in 1982 at Amran, Yemen. He obtained his Bachelor of Science (BS) degree with honors in computer science from King Khalid University (KKU), Abha, KSA in 2007. He granted the Upper Class Honour by the rector of KKU. He joined KFUPM as a full time graduate student to pursue the master's degree. He received Master of Science (MS) degree in computer science from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia in 2014. He worked in several research projects include Plagiarism detection, Code similarity detection, Geographic information system, Arabic computing, Wireless video streaming over DDS, and Operating system fingerprinting. His research interests include Plagiarism detection, Wireless video streaming, Network security and privacy (website fingerprinting attack, traffic analysis attack, privacy protection on Tor anonymity). His research activities resulted in publications of two conference papers one of them is IEEE international conference and other journal paper.