Improving Operating System Fingerprinting using Machine Learning Techniques

Taher Al-Shehri and Farrukh Shahzad

Abstract—Operating System (OS) detection is one of the main concerns for computer security. The previous works that have been done on operating system detection, exploit some features of TCP/IP traffic based on a single packet. In this work, we built a system where TCP/IP communication is setup between machines to capture and analyze TCP/IP packets for more accurate and fine grained OS detection using our novel packet correlation approach. We used existing signature matching methods, extend it and employed machine learning techniques to detect remote operating systems with improved accuracy. We also employed mobile systems like smart phones and tablets to perform mobile OS fingerprinting. The tools we created also established encrypted communication using Secure Socket Layer (SSL) network protocol to investigate the effect of SSL communication on OS fingerprinting. The result of our experimental work showed that fine grained OS detection can be achieved for modern and mobile OSs using our approach.

Index Terms—OS fingerprinting, remote operating system detection, vulnerability assessment, mobile operating system.

I. INTRODUCTION

Today everyone is connected to the internet so the need to secure him from the intrusions is very important. What would happen if a business company that sells its goods on the internet went down for only one day? Or what happen if a bank was hacked and taken down? This external threaten for the companies trigger them to use multiple security applications like firewalls/intrusion detection systems (IDSs) in order to secure themselves from the hackers.

The operating system fingerprinting is a process of remotely detecting and determining the identity of a target system by observing the TCP/IP packets that are generated by that system. The operating system detection can be viewed from two sides. First, from the positive point of view for the hackers needs. For example, the hackers detect OS in order to exploit its vulnerabilities for their hacking purposes. Second, from the positive side for the network administrators needs because it is important for them to collect as much information as possible about their networks. It is also necessary for the system administrator to have certain statistics about the components that they have in their environment. For example, if there is a machine in the network that runs an old version of operating system which could be an easy target to be exploited by the hackers. By using OS fingerprinting, network administrators can know which machine’s OS need an upgrade. Moreover, it is very difficult for the network administrators to have full control of what are connected to the network especially for large networks. For the system administrator, it’s always important to be one step ahead of the attacker. This way, the attacker can’t make use of the latest vulnerabilities. It is also important for the network administrator to be sure that each OS in the network satisfies the applied policies. For instance, when a user formats his PC and reinstalls an old version which violates the company policies. Detecting such situation in an automated way is very important, especially for large networks. “Having access to an up-to-date network inventory could allow a company to save money by canceling the license and support service for an OS that is no longer used”[1].

Now a days, network administrators also want to know which mobile devices, like smart phones and tablets, are accessing his/her network. It may be more difficult to respond to network attacks initiated by a wireless device. In some cases, the mobile users may not be authorized and can cause network overload as network load estimation might have not included on-the-fly wireless users.

There are two basic method of performing OS fingerprinting. The active detection is achieve by sending a special packet to the target machine and get the response that can be analyzed to identify the OS type of the target machine. The main weakness of active OS fingerprinting method is that it cannot be done if the target system has firewall and intrusion detection systems (IDSs) [2]. On the other hand the passive scheme of OS fingerprinting is done by sniffing the network packets remotely instead of sending a crafted packets to a target machine [3]. The idea of passive OS fingerprinting is to analyze the headers of TCP SYN packets (or other specific packets) to determine the operating system. After the needed packets are sniffed they are compared with predefined database that contains signatures of different operating systems, and determine the type of the OS that these packets come from. It is important for network administrators to do OS fingerprinting in a passive way in order to overcome the limitation of active method due to firewalls/IDSs.

The three way handshake is the main step for the initiation of the TCP connection. First, the client initiates the connection by sending a request with SYN flag set to a server. If server is ready to open the connection, it replies with SYN+ACK packet, or if it is not ready, it replies to the initiator with RST packet. Then finally client replies with an ACK. The passive OS detection can exploit some parameters in the TCP/IP packets when SYN, SYN+ACK or RST flags are set [4]. When communication is done, client terminates the connection by sending the packet with FIN+ACK flag set.
The TCP header has multiple flags that are set indicating the TCP connection status [5]. In the passive OS detection the main focus is in the parameters of the packet headers which are time to live (TTL), window size (WS), don’t fragment bit (DF), and TCP options/flags. The main advantage of passive OS detection for the attackers is that they can detect the remote host without leaving any traces [5].

There are few tools that were developed to perform OS fingerprinting. These tools have limitations that need to be solved. For example, the active OS fingerprinting tools face a firewall or IDS in front of the target system which can be detected only using passive OS fingerprinting tools. Also passive tools have some limitations. The signature databases need to be updated continuously otherwise the newer operating systems will not be recognized on the internet any more [5]. The establishment and maintaining a good up-to-date fingerprint database requires some serious research in the area of OS security. Many performance measurements for evaluating passive and active OS fingerprinting are described by Thomas and Greenwald [6].

The rest of the paper is organized as follows. Section II presents the literature review. Section III demonstrates our proposed framework and section IV provide details of implementation. In Section V, the results of our experimental work are analyzed and compared. Finally, conclusion and future work are discussed in Section VI.

II. LITERATURE REVIEW

There is some research in the field of passive and active OS fingerprinting in the last 10-12 years. In [7] Gordon Lyon proposes several programs: checkos, sirc, and SS which are capable of fingerprinting various types of OSs by using TCP/IP traffic. The limitation of these tools is that they are not be referenced anymore because the information that is available by them is too limited.

Michal Zalewski [8] writes the first version of p0f tool for doing passive OS fingerprinting. There are four fingerprinting methods that are used in different scenarios as follows:

1) What is the system that is connecting to yours?
2) What is the system that you are connecting to?
3) What is the system that is refusing your connection?
4) What systems do you have a connection with?

Only the first one is supported well because it detects OS by analyzing the headers of the initial SYN packet.

Lanze Spitzner in [1] identifies what passive OS fingerprinting is, how it works and how to use it. He also compares between passive and active fingerprinting in terms of differences and similarities. He also talked about knowing your enemies and your assets, because when you know your enemies it is much easier to protect yourself against danger.

Gerald A. Marin in [9] looks at the general network security by covering the crucial basics of system security. He describes different attacks such as Distributed Denial of Service (DDoS) attack, land attack and Smurf attack. Several countermeasures are discussed in the paper like what IDS is and how to stop malicious code, Trojans and worms.

Authors in [10] propose a masking approach to secure systems from OS fingerprinting. The paper also discusses the main steps that the operating system fingerprinting tools go through in order to detect the remote OS. They describe some active operating system fingerprinting tools like Xprobe2 and Nmap. The paper also discusses the countermeasure for preventing operating system detection.

Greg Taleck in [3] entitled paper Ambiguity Resolution via Passive OS Fingerprinting looks at exploiting the differences in the common OSs to evade intrusion detection systems (IDSs) detection for attacking. He proposes an approach that uses passive OS detection in order to resolve the ambiguities between different networks stack implementations in a correct way. A new technique that this paper looks at is to increase the level of confidence of OS detection by looking closer at the TCP connection negotiations.

In [11] Vladimir Lifschitz identifies ASP as “representing a given computational problem by a logic program whose answer sets correspond to solutions, and then use an answer set solver to find an answer set for this program”. The author presents a scenario to claim that this approach is optimal and the test results of this ASP fingerprinting is very promising. The accuracy of recognizing 95 OSs tests is more than 80%.

Esfandiari, Bertossi, and Gagnon in [12] perform OS fingerprinting using Answer Set Programming (ASP). The main idea is that they do not consider just a single packet for determining the target OS but they analyze more packets in order to improve the accuracy of OS detection.

We found no published work that fingerprint operating systems based on correlation of multiple packets during the same communication session. Our main contribution in this work includes:

1) We build a client-server system which makes capturing, the appropriate packets for fingerprinting, simplified and automated. This is a ‘hybrid’ approach as active communication is initiated (but no special packets were injected) to perform passive fingerprinting.

2) The system also implemented packet capturing over Secure Socket Layer (SSL) encrypted communication network to analyze the effect of SSL on OS fingerprinting.

3) Due to exponential rise of mobile computing, we also captured packets from mobile devices for fingerprinting using third party socket client apps.

4) We used the latest p0f signature database [13] and convert it into a relational table to improve the performance of signature matching algorithm.

5) We found that by correlating the SYN and FIN+ACK packets during the same communication session leads to more accurate OS fingerprinting.

6) For new OS releases and Mobile OS, we employed machine learning techniques on extended p0f datasets.

III. PROPOSED FRAMEWORK

In Our framework, OS fingerprinting is achieved in multiple phases. The main components of our framework are shown in Fig. 1. The first phase is to capture relevant TCP/IP packets from network traffic. Then these samples are passed to a matching component to compare with the existing fingerprint database. If the exact match is found the process
ends. Otherwise, the data is processed using machine learning techniques by trained classifier which tries to find the closest match. In our framework, we are only interested in the SYN and FIN packets.

This classification problem can be stated as follows: Consider a set $P$ of TCP/IP packets and a set of client machines $M$, where each machine $m \in M$ has a known, labeled operating system OS$(m)$. Each machine $m$ sent SYN packet $p_{SYN} \in P$ to server machine. The data collector records the packet $p_{SYN}$, the server response $p_{SYN,ACK}$ for each $p_{SYN}$ and corresponding $p_{FIN,ACK}$ (on socket close). This yields a set of samples $S$ for the classifier $C$.

A classifier $C$ takes as input the set of samples $S$ and produces a fingerprinting detection tool $D_C$. The tool $D_C$ takes as input a sample $s$ and returns the best OS label for the sample’s machine $s(m)$. The tool is a function $f$ such that $f(s) = \text{OS}(s(m))$ for all $s \in S$.

To solve the OS classification problem, this tool $D_C$ should not only correctly return the OS of all samples in $S$, but it should also correctly return the OS of previously encountered samples not in $S$.

### D. Preprocessing Step for Data Classification

Before the classification step the data must be transformed into the format that is compatible with WEKA tool which called Attribute-Relation File Format (ARFF). This format starts with a header for its description. Then all events are stored in ARFF file with comma separated values each on their own row. The ARFF format is based on p0f format rules so for each field in p0f fingerprint, a specific attribute is defined. The order in ARFF format is not considered but it is important in TCP options so it is necessary to encode the order in ARFF file. To tackle this issue ten separate attributes are specified for each option in order to allow them to have any of the options. The result of the classification (detected OS) is the final attribute in the ARFF file which represents the target system that generates the transformed p0f fingerprints.

### E. Defining Relevant Parameters

Determining the most relevant information from TCP/IP headers is an important step for OS system detection. These relevant parameters are chosen dynamically for the classifier because there may be a new OS fingerprint contains some header fields that are not considered before in the database to be able to match it with the predefined OS classes.

In Weka, the complete set of samples is partitioned into subsets. A single subset is used to validate the model, while the other subsets are used to train the model. We choose a ten-fold cross-validation, so ten subsets are created. The complete process is repeated ten times, each time with a different subset used as the validation subset and the rest as the training data for the model.

### F. Decision Tree/C4.5 Classification Algorithm

In our experiments we select C4.5 classification algorithm [14] because it is well known with its high accuracy of classification. This algorithm goes over samples of training set many times in order to build an optimal classification model. This algorithm handles the continuous and discrete attributes where the continuous are supported by using thresholds. Furthermore, the training set with missed attribute values can be handled using this algorithm. The
algorithm goes up the tree when it counters an instance of a new class. Then the algorithm creates a decision node in the tree for the attribute that will give the highest information gain. After that it will recurs down the tree and removes the sub-trees that are not needed by replacing them with leaves. The pruning feature of this algorithm makes it possible to create the model in seconds and classifies with better accuracy.

IV. IMPLEMENTATION

To evaluate our proposed framework, we built a Java package edukfupm.csce.osfp with several classes to process and transform relevant packets from pcap format to p0f and ARFF (for Weka learning tool) format.

The p0f is one of the commonly used signature format for SYN based OS fingerprinting (Table I). We converted the most recent p0f signature data file [13] into MySQL relational table which makes matching and adding new signature easier and streamlined. We also extended p0f to include other fields as discussed in later section. The original p0f SYN signature can classify the remote OS into genre like Linux, BSD, Windows, etc. It can also distinguish some older OS versions accurately. But most of the newer OS can’t be classified at the version level. We found that our approach can lead to more accurate and fine grained OS detection.

A. Packet Capturing and Extraction

We developed two sets of client/server Java socket applications. One set used normal java socket API and other used SSL socket API. The server runs on a certain machine and multiple client applications connect to the server simultaneously (from other machines). The client application binds to the server (SYN) and then disconnects (FIN+ACK).

There was no actual data communication. Therefore only 3 types of packets were captured namely SYN, SYN+ACK and FIN+ACK as it is shown in Fig. 2. The wireshark, Windows network monitor and/or network miner tools are used to capture the packets on the server machines. The captured packets are saved in the tcpdump.pcap format. For mobile devices, third party TCP/IP client apps were utilized. Although above setup basically performed passive fingerprinting as no special packets were injected, but one can argue that only specific packets between selected machines (which are executing custom made applications) are captured. Therefore we can call it 'hybrid' fingerprinting.

To extract the right information from TCP/IP headers we build a Java application based on JnetPcap library, which is a Java wrapper for native libcap library [15], to generate fingerprint entries in p0f format.

B. C4.5 Classifier

The Weka classification tool needs the training dataset to be in the ARFF format. For this purpose, we build a converter that converts p0f fingerprints into ARFF format. This dataset is fed to the Weka application for classification. For normal p0f based classification, we have total of 31 attributes.

C. Extended P0f

Now we present how we correlate packets from same communication session to extend the p0f signature format. Basically, we link SYN packet and FIN+ACK packet using hash map. The key used for hashing is the contacted string containing source IP, destination IP, source port and destination port from the SYN packet. This key is matched with the same concatenated string from the few succeeding FIN+ACK packets. Our capturing model dictates that those two packets should not be far apart in the pcap file.

We also use Weka tool to classify OS’s using extended p0f format. The idea is that with more attributes, a more accurate classification can be achieved. Table II shows an example of extended p0f.

<table>
<thead>
<tr>
<th>OS</th>
<th>WSS</th>
<th>TTL</th>
<th>D size</th>
<th>options</th>
<th>FIN-</th>
<th>FIN-</th>
<th>FIN-</th>
<th>FIN-</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WSS</td>
<td>TTL</td>
<td>D</td>
<td>size</td>
</tr>
<tr>
<td>Win-8</td>
<td>8192</td>
<td>128</td>
<td>1</td>
<td>52</td>
<td>M*,N,W8,N,N,S</td>
<td>260</td>
<td>128</td>
<td>1</td>
</tr>
</tbody>
</table>

V. EXPERIMENTAL EVALUATION AND ANALYSIS

We use the tools and libraries listed in Table IV to test our approach.

<table>
<thead>
<tr>
<th>Tool/Library</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVA for programming</td>
<td>1.7</td>
</tr>
<tr>
<td>Wireshark for capturing</td>
<td>1.8.6</td>
</tr>
<tr>
<td>Network Miner</td>
<td>1.4.1</td>
</tr>
<tr>
<td>Microsoft network Monitor</td>
<td>3.4</td>
</tr>
<tr>
<td>WEKA for classification</td>
<td>3.6</td>
</tr>
<tr>
<td>SSL protocol for encryption</td>
<td>Java Keytool (RSA)</td>
</tr>
<tr>
<td>JnetPcap library</td>
<td>1.3</td>
</tr>
</tbody>
</table>
For our experiment, we select two different environments. In the university setting, we select few machines running Windows-8 or Windows-7 as servers and we setup our client application on few window machines, one Linux machine and one android smart phone. In home setup, we use two windows-8 machines (64 bit and 32 bit), windows XP machine, one android device, one win-CE device and an iPad (Table III). The tools used are specified in Table IV.

A. Results and Analysis

We captured TCP/IP packets on different networks for 10 days. First we ran p0f matching algorithm on some sample pcap files. Table V shows the summary of result. As discussed earlier, we utilize relational table for p0f signature matching. About 30% of packets in sample 1 were not matched to any OS. Furthermore, the matching is very coarse as the existing p0f database mapped multiple OS releases to same signature.

Next, we employed our Java application to generate p0f and ARFF files from 8 raw pcap files. These files were captured as described in Section III. Similarly, we use a separate Java application to generate extended p0f and ARFF files from same 8 raw pcap files.

Finally, we execute the Weka tool with combined ARFF dataset separately for p0f and extended p0f based instances. We use J48 (an implementation of C4.5 algorithm) with 10-fold cross-validation test mode.

The results for two classifications are compared in Table VI. With four more attributes, the extended p0f classifier creates 15 trees as compared to 13. The detection accuracy for extended p0f based classification is about 91% as compared 84% for normal p0f based classification (Fig. 2). This result shows higher accuracy when compared to related work [14, 16] especially with extended p0f based classification.

VI. CONCLUSION

In this paper, we presented a hybrid approach for automated and more accurate OS fingerprinting. Several Java tools were built to capture, process, transform, match, analyze and classify appropriate TCP/IP packets. Our research showed that by correlating packets from same TC/IP session, fine-grained OS detection can be achieved for modern operating systems and mobile devices. We also noted that SSL TCP/IP communication doesn’t show any significant differences which can effect fingerprinting.

We believe that we can achieve even finer OS detection if we have resources like computers/devices running different releases of operating system. This means we may be able to distinguish between Windows-8 64 bit and Windows-8 32 bit or iOS 5.1 and iOS 6.x. Since we have tons of smart devices in the market today, including smart phones, tablets, game consoles, consumer electronics etc., more research is needed to remotely detect the OSs running on these devices. Furthermore, new tools need to be built, if these devices use communication protocol other than TCP/IP.

REFERENCES


Taher Al-Shehari is a master student at King Fahd University of Petroleum and Minerals. He got his B.Sc. degree in Computer Science from King Khalid University in 2007. He granted the Upper Second Class Honour by the rector of the university. His research interests include plagiarism detection, wireless video streaming, website and OS fingerprinting on anonymity protocols. His 2012 paper, “Wireless video streaming over Data
Distribution Service middleware (with Al-madani, B.; Al-Roubaiey, A.), published in Software Engineering and Service Science (ICSESS), was invited for presentation in the IEEE 3rd International Conference on June 2012.

**Farrukh Shahzad** was born in Karachi, Pakistan. He did his BE(Electrical Eng.) from the NED University of Engineering & Technology, Karachi, Pakistan in 1992 and his MSEE from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in 1996. He is currently a Ph.D. student and lecturer at KFUPM.

In 1996, he moved to USA. He has 16 years field experience in product design, software development, engineering, and implementation of many M2M and satellite based remote monitoring systems. His current interests include Cloud storage security, Big Data analytics, Data sciences, Machine learning and business intelligence. He is US copyright holder of four Engineering softwares. His research activities resulted in publication of more than 10 technical papers in IEEE and other journals.
Wireless Video Streaming Over Data Distribution Service Middleware

Basem Al-madani, Anas Al-Roubaiey and Taher Al-shehari

Department of Computer Engineering
King Fahd University Of Petroleum And Minerals
Dhahran, Western Province, Saudi Arabia

{mbasem, roubaiey, g200905290}@kfupm.edu.sa

Abstract— High-quality video transmission over wireless local area networks is one of the most challenging issues nowadays. The Data Distribution Service (DDS) is a promising middleware that can be used for video distribution over WLANs. DDS has a rapid implementation in high-performance and mission-critical networks. This paper evaluates the performance parameters such as throughput and jitter when video transmission is introduced in a 802.11g networks using DDS. Furthermore, it investigates the DDS QoSs that affects video streaming over WLANs which is important when deploying end-to-end video streaming services with quality of service guarantee.

Keywords—IEEE 802.11g; WLAN; Data Distribution System; quality of service; QoS; H.264; video streaming.

I. INTRODUCTION

In wireless local area networks (WLAN), the need to video streaming nowadays is a rising tendency nowadays. However, the challenges of video streaming due to video, channel, and network characteristics should be solved to maintain a quality of service (QoS) of video transmission. The history of video over WLANs has been associated with failure to deliver true high quality of service (QoS) video due to unacceptable system latency, and insufficient wireless network bandwidth to carry real-time high bandwidth video content [6]. Due to its light weight, and it provides fine grained control over quality of services to transfer video traffic DDS is considered to be an appropriate solution for that purpose.

A. DDS Overview and Video QoS Parameters

The DDS is a worldwide data-space whose data-structures and attributes are specified by meta-information called Topic. Every topic describes a set of associated data-samples with the same data-property and data-structure. For example, a topic named “Temperature” can be used to store samples of temperature monitored by a distributed set of sensors. [7]

The entities that write and read the data-samples in the middleware are the publishers and the subscribers. A publisher consists of a set of Data Writers modules, each of which is used to write information on a particular Topic. On the other side, a subscriber reads the data samples of Topics by using its Data Readers modules. A topic is qualified by a wide set of Quality of Service parameters that manage a number of aspects of the distribution of linked data-samples.

For instance the "LIFESPAN” QoS finds out the maximum time a data-sample can stay in the system since its writing time; in real time video transmission we don't need the late frames, here we LIFESPAN QoS. Another example, the HISTORY QoS specified the maximum number of data-samples that can be stored in the middleware, if such maximum number is reached then the newest data-sample substitutes the oldest one. When an application needs to get data-samples of a particular Topic, it just feeds the DDS interface with the name of the Topic; then, the DDS takes care of setting up the underlay networking facility.

As soon as the network configuration stage terminates, the application receives a bulk of data-samples whose number is equal to the HISTORY QoS, then it will keep picking up data-samples, since they are written. This QoS is beneficial for video streaming since the late joining participant can get the previous displayed video. It is worth also mentioning that the application can determine a filtering-condition correlated to the content of data-samples e.g. temperature measure less than 20 degree. In this case, the DDS transfers only data-samples complying with the filtering condition. This filtering is very useful in many cases of video transmission; e.g. in kids safety we can use content base filter to filter the undesired frames. Best effort quality of service is used for real time applications such as video transmission. However, DDS is also introducing a reliable QoS for data sensitive applications such as FTP. The main QoS that we used in our study is the RELIABILITY QoS which indicates the level of reliability offered/requested by the Service. Possible values are:

- RELIABLE Specifies the Service will attempt to deliver all samples in its history. Missed samples may be retried. In steady-state (no modifications communicated via the Data Writer) the middleware guarantees that all samples in the Data Writer history will eventually be delivered to all the Data Reader objects. Outside steady state the HISTORY and RESOURCE_LIMITS policies will determine how samples can be discarded from it. This is the default value for Data Writers.
- BEST_EFFORT Indicates that it is acceptable to not retry propagation of any samples. Presumably new values for the samples are generated often enough that it is not necessary to resend or acknowledge any samples. This is the default value for Data Readers and Topics.[4]
B. Contributions

The main contribution of this paper is to examine the behavior of real-time video streaming over WLAN using DDS middleware. For our best of knowledge this is the first work in publications to examine such scenarios; where we concentrate on examining the total bandwidth consumed due to DDS video traffic by evaluating network throughput with different network load, number of subscribers. Furthermore, we measured the jitter to reflect the video quality performance. This paper also demonstrated the most important quality of services of DDS that can be used to improve the performance of video transmission over WLANs.

C. Organization of the Paper

This paper is organized as follows. Section II presents the literature review. Section III demonstrates the simulation model and experimental work. In section IV, simulation results are analysed. Finally, conclusions and future work are discussed in Section V.

II. LITERATURE REVIEW

A. Video Streaming previous work gnisu middleware

There is a deficiency in the literature related to video streaming over the DDS middleware. In this section we summarize the most related existing work.

Detti, Loreti and Melazzi [7] evaluated and demonstrated a technique for streaming H.264 SVC video over a DDS middleware. The structure of the DDS data-unit designed by them was able to carry H.264 SVC video-units. Also they designed a receiver-driven rate-control mechanism based on the DDS data unit and exploiting specific DDS functionality. Finally, they implemented and showed the effectiveness of their mechanism in an 802.11 wireless scenario, comparing their proposal with other solutions.

Clavijo proposed that a CORBA middleware implementation can be used to offer real-time video streaming [11]. Furthermore, in his paper [15], Karr et al stated that a CORBA based platform was introduced to respond to changing resource requirements in video applications using video streaming service over CORBA-based solution which has been the one projected for real-time environment. CORBA is a very complete technology that introduces a big number of interfaces for almost any type of required middleware functionality; however, CORBA is a complex architecture that introduces implementation overheads, in particular if compared with other lighter weight technologies such as ICE (Internet Communications Engine) [3], DDS (Data Distribution Service for real-time systems) [4], or some specific Real-Time Java based solutions [5]. Therefore, existing approaches can be improved to offer appropriate support to the real time nature of video transmission with guarantee. In addition, using new standard middleware introduces flexibility for video transmission at two approaches. First, compared to direct implementation over the network level, the utilization of a middleware is already more flexible. Second, utilizing middleware solution offers QoS management which allows to appropriately initiating real-time support to video transmission.

B. Video Streaming over wireless networks

In this approach extensive researches have been carried out. Video transmitting over wireless networks has experienced wide considerations in recent years. We reviewed the most related work. Vora and Brown [6] focused on the newer 802.11n. Their performance parameters were throughput, delay and jitter when video streaming is brought in a network carrying merely data traffic. They also gave approaching on the approximate number of users streaming high rate videos that can be supported over various environments. In [9], the author analyzed and evaluated the performance of H.264-based video streaming over multi-hop wireless local area networks (WLANs). He provided guidance on how to achieve the optimal balance for a given scenario, which is important when deploying end-to-end video streaming services with quality of service guarantee.

Chen and Zakhor proposed several TFRC connections as an end-to-end rate control solution for wireless video streaming. They showed that this approach not only avoids modifications to the network infrastructure or network protocols, but also results in full utilization of the wireless channel [10]. Stockhammer, Jenka and Kuhn proposed that the separation between a delay jitter buffer and a decoder buffer is in general suboptimal for VBR video transmitted over VBR channels [12]. They specified the minimum initial delay and the minimum required buffer for a given video stream and a deterministic VBR channel. In addition, they provided some probabilistic statements in case that they observed a random behavior of the channel bit rate.

In [14], Nassir propose a QoS adaptive multimedia service models for controlling the traffic in multimedia wireless networks (MWN) that enhances the current techniques used in cellular networks. The suggested framework is designed to take advantage of the adaptive bandwidth allocation (ABA) algorithm with new calls in order to improve the system usage and blocking probability of new calls. In his simulation the results showed that the QoS adaptive multimedia service framework outperforms the existing framework in terms of new call blocking probability, handoff call dropping probability, and bandwidth utilization.

Deer and Pan through their study in a WDS-based multi-hop wireless environment, they have found that it is likely for multi-hop wireless networks to increase the coverage and develop the video streaming performance at the same time. When they analyzed the throughput of IEEE 802.11 multi-hop wireless networks, they proposed a complete two-dimensional Markov-chain model in their paper. Their model considered the retry bound and post-back off step into account to better capture the performance of IEEE 802.11 MAC protocols in a non-ideal channel and with non-persistent traffic. The throughput analysis is validated by network simulation with extended lower and upper-layer simulation modules. The achievable throughput gives an upper bound of the video streaming performance, which is further validated by our H.264-based video streaming simulation with application-layer performance metrics. The results correspond to the observation they had on the multi-hop test bed [2].

Another study [16] highlights that since the advent of ad-hoc networks, it has been viewed as a potential multi-application technology. This paper presents a comparative study of multicasting of video and video-like data using two different ad-hoc routing protocols, viz. OLSR and PUMA. Their NS2 simulations show that OLSR produces higher throughput and lower latency.
III. EXPERIMENTAL WORK

In this section, we experimentally evaluated the performance of video streaming and data transmission over WLAN with DDS middleware. We first summarized our evaluation methodology, including simulation model and configuration. Our performance metrics were the throughput and jitter.

A. Hardware and software specifications

The experiment was carried using hardware and software tools; the measurement and monitoring tools and hardware platform specifications that was used are described in Tables 1, and 2 respectively. as specified in Table 2, The tools used are specified in Table 1.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire-shark</td>
<td>1.2.7</td>
<td>Get Throughput</td>
</tr>
<tr>
<td>Jperf</td>
<td>2.0.2</td>
<td>Get Jitter</td>
</tr>
<tr>
<td>RTI Analyzer</td>
<td>4.5f</td>
<td>QoS monitoring and network debugging</td>
</tr>
</tbody>
</table>

Table 1. Tools and Programs

<table>
<thead>
<tr>
<th>Publisher</th>
<th>Subscriber A</th>
<th>Subscriber B</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel(R) Core(TM) i5 2.40 GHZ</td>
<td>Intel Core(TM) i5 2.40 GHZ</td>
</tr>
<tr>
<td>Memory</td>
<td>1.8 GB</td>
<td>1.8 GB</td>
</tr>
<tr>
<td>Network connection</td>
<td>802.11g WLANs 54 Mbps</td>
<td>802.11g WLANs 54 Mbps</td>
</tr>
</tbody>
</table>

Table 2. Platform specifications

B. Experimental work setup

The experiment framework is shown in figure 1, to apply to the video streaming system. The backbone communication is based on P/S, publish and subscribe, DDS middleware technology over WLAN.

Our workloads that we used in this experiment to evaluate the performance were two types. The first one is the heavy load which is represented by video traffic, and the other one was the light weight traffic which is represented by data readings of locations for mobile object. We have used the RTI Granada university tool for video distribution using DDS. For video traffic the QoS that we used is the best effort as it is well known for real time applications and that is supported by DDS middleware, whereas for data readings it is possible to be reliable or best effort based on the application. In our experiment we used both QoSs parameters, reliable and best effort, for data readings workload.

IV. RESULTS AND ANALYSIS

For each result point in the chart we run the experiment 10 times each time 2 minutes long in order to examine the changes in the bandwidth accurately, and we took the average. We started by testing the throughput of the background of our WLAN to calculate the capacity left available for the video and data streams precisely. As the throughput standard of 802.11g is 54 Mbit/sec, we measured the portion that occupies by the network traffic without any load. The average background traffic was 0.000257 Mbit/s. At 128 kh/s codec bitrates and 25 fps, we streamed video in five different scenarios. These scenarios were with different number of subscribers, i.e. 2, 4, 6, 8, 10 subscribers. the simulation time was 2 minutes for each reading point.

The throughput of the two workloads, light and heavy, shown in Fig 2 increases as the number of subscribers increase. Due to the nature of video streaming, the increasing rate with number of subscribers is high comparing with light traffic, data readings. The total bandwidth consumed for video traffic in case of 10 subscribers was almost 1.8 Mbps.

Fig. 3 we measured the throughput in terms of packets per second. In case of 10 subscriber for video it was almost 510 packets/sec, whereas it was 200 packets/sec in case of data readings. The total bandwidth consumed for video traffic in case of 10 subscribers was almost 1.8 Mbps.

Fig. 3 we measured the throughput in terms of packets per second. In case of 10 subscriber for video it was almost 510 packets/sec, whereas it was 200 packets/sec in case of data readings. The total bandwidth consumed for video traffic in case of 10 subscribers was almost 1.8 Mbps.

Fig. 4 shows the jitter of both traffic types. It is clear that the jitter in case of data readings is almost stable and that is due to the very low traffic generated by the data readings; but in case of video traffic it is increased significantly with number of subscribers, it almost reaches 2.5 ms in case of 10 subscribers; and that is due to the fact of heavy load of video traffic which causes a lot of congestions and packet dropping in the network which increase the delay variations of the sent packets.

Fig. 5 shows the difference on performance for the same data type using different QoS parameters which are reliable
versus best-effort. It shows that the difference is starting to be clear from case of 4 subscribers. That is because of in case of 2 subscribers the network was not loaded and free of congestions and losing packets but starting from case of 4 subscribers where the packets start dropping which results in degrading the performance in case of best-effort. The bandwidth consumed in case of 10 subscribers is almost 0.3 Mbps.

From Figure 5. Reliability QoS parameters in reading data real time transmission over DDS on network bandwidth and jitter. From the publications that examining the real effect of video transmission over WLAN using Data Distributed Service (DDS) on network bandwidth and jitter. From our results we conclude that this technology is a promised technology for distributing video over WLANs, that is because it consumed low bandwidth, has low jitter. Furthermore, it gives more control on video streaming due to using rich set of QoSs provided by DDS. As a future work we intend to do these study on more restricted networks such as Bluetooth personal area networks, and examining the QoS parameters to come up with best adjustment for WLAN environment.

ACKNOWLEDGMENT

We would like to acknowledge RTI company for their support with all tools that we have used in this work. Also, we would like to thank COE Distributed real time system lab in computer engineering department at KFUPM.

REFERENCES

A comparative study of the resistance of top five web browsers against traffic analysis attack

Proposed by
Taher Al-shehari
g200905290@kfupm.edu.sa

Supervisor
Dr. Sami Mohamed Mahmoud Zhioua

Dec 17, 2013
Abstract

The need for fast and secured access to online resources is very significant as a lot of services become available on the internet. A high number of people are encouraged to explore and take the advantages of the World Wide Web (WWW) because of the wide growth of the internet. The web browsers are the windows to the World Wide Web so the increased of internet services contributes significantly to develop various web browsers in the market at present. In this paper we conduct comparative and analysis study to investigate the traffic analysis attack protections of the most commonly used web browsers namely Mozilla Firefox, Internet Explorer, Google chrome, Apple Safari and Opera. The main goal of the paper is to tell the browsers’ end user that which browser protects his privacy against eavesdropper attack more and which browser makes the external observer identifying his identity (e.g. the type of used browser, the name of visited website) easily based on traffic pattern analysis. The results of our experiments show that Opera and Safari have the most randomized traffic that make the traffic analysis attack much more difficult because of the root causes mentioned in the analysis section. On the other hand, Chrome and FF deal with more regular traffic pattern which make the job of an eavesdropper doing traffic analysis attack easily. IE browser is placed in between as it is shown in the experimental work and analysis section.

Index Terms—Web Browsers, Traffic Analysis Attack, Rendering Engines, Web Traffic Pattern, HTTP, TCP.
I. INTRODUCTION

We start by conducting a literature survey of the recent research articles on web browsers but we found that there is no any research that touches this topic from our point of view. In our research we investigate different aspects of browsers behavior in fetching different web objects to understand to which extent every web browser protects against website fingerprinting. Our task in this paper is to do a comparative study of the top five browsers. We select the most used browsers according to Market Share of browsers in [5], based on this usage statistics the top five used browsers in sequence are Google Chrome 52.9 %, Mozilla Firefox 27.7 %, Microsoft Internet Explorer 12.6 %, Safari 4.0 % and Opera 1.6 % as it is shown in Fig 1.

![Browser Statistics and Trends in On May, 2013][5]

Figure 1: Browser Statistics and Trends in On May, 2013 [5].

A. Web Browsers

Before conducting a comparative analysis of browsers’ properties we present an introduction of each browser which is very important to understand how each browser handles CPU usage, memory usage, load time, security…etc.

- **Google Chrome**
  
  The latest browser that is released in 2008 is Chrome which has the top market share of 52.9 % in 2013[11]. It includes many features that make it the best browser around the world which are:
  
  - Visual Browser History
  - Search option from the Address Bar
  - Task Manager for Websites
- Super Clean Contextual Menus
- Check Memory Usage
- Quick Launch Bar to launch Websites from Start Menu
- Restore website tabs if it is closed by mistake
- Developers claim that Chrome is the fastest browser, high performance, better stability, and high security.

The architecture of Chrome browser provides approaching into its security features. The open source project behind Chrome is Chromium which has two modules in separate security domain: rendering engine and browser kernel. This effective architecture assists alleviate high severity attack without compromising its compatibility.

- **Mozilla Firefox**
  Mozilla’s company develops Firefox (FF) as an open source browser. It is the second most popular browsers after Chrome with a market share of 27.7% in 2013 [11]. The internal architecture components of FF are divided into submodules and each module is managed by a specific entity. FF has many functions such as spell checker, live bookmarking, tabbed browsing, incremental find, download manager, and built in search system. Furthermore, a lot of features can be added by third party developers called add-ons such as Adblock Plus ad blocking utility, NoScript JavaScript disabling utility, StumbleUpon (website discovery), FoxyTunes media player control toolbar, download enhancer, Web of Trust security site advisor, Foxmarks Bookmark Synchronizer (bookmark synchronizer)…etc. We are expecting that these add-ons will affect the performance of the browser in website fingerprinting results because more loads will be added to the browser handlers[17].

- **Internet Explorer**
  Internet browser (IE) is a graphical web browser developed by Microsoft as a part of Microsoft Windows started in 1995. “It has been the most widely used web browser since 1999, attaining a peak of about 95% usage during 2002 and 2003 with IE 5 and IE6 and that percentage share has declined since in the face of renewed competition from other web browser developers”[6]. In 2013 its market share declines dramatically into 12.6 % because of the appearance of competing browsers Chrome and FF[11]. IE utilizes DOCTYPE sniffing to decide between
"standard mode", W3C's specifications, and "quirks mode", older versions of IE, for Cascading Style Sheets (CSS), which adds additional layout and style information, and HTML representation on screen. IE always uses standard mode for printing. It is also supported by ECMA Script, a scripting language that is used to generate web pages with dynamic behavior called Jscript.

❖ Safari

According to [5] Safari is the fourth common browser with market share of 4.0 % in 2013. It was developed by Apple in 2003 just for Mac OS X and iOS operating systems [8]. The first version for Microsoft Windows operating system was released on June 11, 2007[9].

Safari browser has the following features:

- Auto-fill feature to fill web forms automatically
- The integration of bookmark with address Book
- Pop-up ad blocking[7]
- Nitro JavaScript Engine to run JavaScript faster
- Like Chrome browser, Safari uses DNS pre-fetching to speed up page loads
- Security extensions to secure and sign extensions running on the browser
- HTML5 features that support closed-captioning on HTML5 video and full-screen video, draggable attribute, Geolocation, Ruby, AJAX History sectioning elements, EventSource, forms validation, and WebSockets[10].

Though safari browser has good features but the statistics given above the users taken chrome because chrome is the latest browser by Google and users want to know the details of the new browser.

❖ Opera

Despite its fast, good security Opera browser has a market share of 1.6 % in 2013[5]. In Opera there are standalone applications called Widgets which live outside the browser and interact with the internet. Widgets do not change the basic behavior of the browser as add-ons components in Firefox browser do. Opera has innovative functions such as BitTorrent support, BitTorrent support, RSS support, tab thumbnails, built-in BitTorrent client and it was the first browser with tabs. Moreover, fitting to window size (ERA), crash recovery, rewinding, tab closing, page zoom, instant back, and periodic reloading. It also allows the duplication of tabs and visit URL feature for un-hyperlinked web address. Furthermore, it does the best on the Acid2 web standards test [17].


B. Browsers’ functions

The main functionality of web browsers is to request web resources from servers and display them in web browsers windows. The format of requested resources is usually HTML, image, PDF......etc. the main specifications that the browser interprets and displays web contents are HTML and CSS that are maintained by the standards organization for the web World Wide Web Consortium (W3C). The latest versions for these specifications are 5 for HTML and 3 for CSS. Unlike other browser there is an important feature that Chrome browser has which runs with multiple processes of rendering engines as each tab of the browser has a separate process [12].

In this section we present a brief introduction of the main components that are commonly used by most of the browsers. The aim of this section is to give an idea of how the main components of the browsers work internally and interact with each other as it is shown in figure 2:

![Figure 2: Browsers main components [12].](image.png)

1. **User interface (UI):** which includes the main components of browser’s window like address bar, bookmarking menu ....ect.
2. **Browser engine:** which represents the interface to manipulate and query the rendering engine. It is the middle component between the UI and the rendering engine as it is shown in Fig 2.
3. **Rendering engine:** the main function of this component is to request and show the requested contents on the browser screen. It displays HTML, XML and images by default but other types of data can be displayed by plug-ins browsers like PDF viewer plug-in for displaying PDF format.
4. **Networking:** This is an independent interface which performs the network calls like HTTP and CCS requests.
5. **User interface (UI) backend:** The main function of this unit is to draw basic widgets windows and combo boxes. It is a generic interface which uses the operating system user interface methods

6. **JavaScript interpreter:** Which is used to parse and execute the JavaScript code.

7. **Data storage/ Data persistence:** It stores all sorts of requested data on the hard disk like cookies. The new HTML5 specification improves this unit and makes the database in browsers very light.

Chrome browser differs from others in that it runs multiple processes of rendering engine one process for each tab. The rendering engine gets the flow of requested contents from networking layer as it is shown in Figure 3.

![Figure 3: Basic flow of rendering engine](image)

Figure 3 shows the gradual processes for interpreting and displaying code on the screen. First, the HTML document is parsed and turned its tags to DOM tree called content tree by rendering engine. Then another render tree, nodes with visual attributes like dimensions and color to be displayed on the screen, will be created using CSSs files and visual instructions in HTML document. After that, the constructed render tree is passed through layout process to give each node its coordinates to be displayed in proper position in the screen [16]. The final step the render tree nodes are painted by UI backend layer. In the literature we found that different browsers apply various rendering engines as follow: Chrome and Safari use Webkit, Firefox uses Gecko, IE uses Trident, and Opera uses Presto render engine.
A. How Do Web Browsers Work?

After we look at the internal structure of the browsers let us see how web browsers work from protocol point of view. The browsers use the protocol layers in the process of fetching web pages from servers. The HTTP web protocol is the application layer used to fetch URL and it runs on the top of TCP/IP transport and network layer protocols. There are two protocols that depend on the network which are link and physical layer protocols. If the network is wired these two protocols are combined in the form of Ethernet as it is shown in figure 4 below.

Web browsers access World Wide Web and display web pages underlying web protocol called Hyper Text Transfer Protocol (HTTP). This protocol acts as a translator to communicate between clients and servers. It defines the format and transmission actions that the browsers and servers should take to different instructions. To make it clear assume we enter a website address (www.yahoo.com) in certain browser what happen is that a TCP three-way handshake will be established to open the connection between the browser and yahoo server. Then the first data packet (HTTP GET request) will be send to fetch and display the requested website information on the browser as it is shown in Figure 5. The exchange of data between the client and server should follow this pattern but it differs slightly if a packet was lost and must be retransmitted. The client will contact the HTTP server and exchange HTTP messages to fully download target page until the issue of FIN TCP packet.
After the browser connects to web server at resolved IP address on port 80 the requested web pages and other web contents are displayed on screen. The web pages are fetched differently on various web browsers as we will see in testing and analyzing section.

**B. Contribution**

The main contribution of this paper is to investigate the traffic pattern behaviours of the most commonly used web browsers in fetching and rendering webpage contents in order to know which browser protects the privacy of users against eavesdropper attack more and which browser makes the external observer identifying users’ identity (e.g. the type of used browser, the name of visited website) easily based on traffic pattern analysis. To the best of our knowledge this is the first work in the publications to investigate such scenarios on modern web browsers with sufficient drill-down to the component level. We did a concrete evaluation of browsers-specific factors that make browsers behave so differently by conducting multiple tests such as HTML5 standard specifications support, JavaScript and CCSs features support in terms of coarse grained features listed below.

- The incoming and outgoing packet counts of each browser
- The incoming size of retrieved web contents
- The rate of traffic pattern stability to measure the resistances of tested browsers against traffic analysis attack
- How parallel downloads behave across browsers
We evaluate also other aspects that affect the behaviour of browser in rendering websites as it is the analysis section.

II. EXPERIMENTAL WORK

In this section, we experimentally evaluated the retrieval behavior of the most commonly-used web browsers. We first summarized our evaluation methodology, including the configuration of our test bed specified in details below.

A. Hardware and software specifications

The experiments are conducted on tools listed in Table 1 and hardware specifications specified in Table 2. We used the latest stable version available of web browser.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>32-bit Windows 7 Service Pack 1, build 7601</td>
</tr>
<tr>
<td>Wireshark</td>
<td>Version 1.10.0 (SVN Rev 49790 from /trunk-1.10)</td>
</tr>
<tr>
<td>Chrome</td>
<td>Version 31.0.1650.57 m</td>
</tr>
<tr>
<td>Firefox</td>
<td>24.0</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>10.0.7</td>
</tr>
<tr>
<td>Safari</td>
<td>5.1.7</td>
</tr>
<tr>
<td>Opera</td>
<td>15.0.1147.153</td>
</tr>
</tbody>
</table>

Table 1: Software Tools

<table>
<thead>
<tr>
<th>Devices</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel(R) Core(TM) Duo CPU T9600 @ 2.80 GHZ</td>
</tr>
<tr>
<td>Memory</td>
<td>4 GB</td>
</tr>
<tr>
<td>Ethernet Card Model</td>
<td>1EECD81C-907A-475B-ACF1-D106043C1F6D</td>
</tr>
<tr>
<td>Network connection</td>
<td>2 Mbps</td>
</tr>
</tbody>
</table>

Table 2: Platform specifications

B. Experimental work setup

The architecture framework for our experiments has been set up using the above software and hardware tools. The backbone communication is based on Ethernet environment of 2Mbit/s. We believe that distinguishing between the behaviors of browser retrieval is difficult, since there are many factors that affect the pattern of traffic such as the content and structure of the websites, browser configuration, users’ browsing behavior, client hardware configuration, geographic location, and network connection. Therefore, we design and implement the
experiments in a controlled environment in order to focus on the factors that influence the behavior of browsers retrieval by eliminating the external factors such as operating system, network latency, and server load in order to get accurate results of relevant browsers. Our workloads to evaluate the behavior of web browsers are selected from the commonly browsed websites listed by ALEXA namely Yahoo, Amazon, Ebay, Wikipedia, and Microsoft. We intent to choose the websites carefully that contain various collections of web contents like text/x, CSSs, images/x, JS, etc. We start by launching each browser alone and clear its cache. All experiments are conducted with a clean profile and no add-ons/extensions installed on browsers to make sure that the observed results are accurate. Then we launch Wireshark analyzer tool, select the interface whose packets should capture traffic through, leave other options at their default values and start the capture. When the capture is started, we put the URL of websites in the tab address of browser to fetch its web contents. The ideal browsing scenario is a one website retrieval taking place at any time and each webpage is allowed to fully download before the next one. At the time of downloading, the packets are recorded by Wireshark as the contents are transmitted until the website is fully loaded then we stop the trace. We repeat these steps 5 iterations for the list of selected websites under each browser. We flush the cache before each test to insure controlling the effects of caching and we end up with 125 PCAP files. After data collection/capturing phase, we filter the traffic and extract result values for analysis phase. Finally, we average the results take into consideration the importance of standard deviation to see the confident and dispersed over time rate of each browser.

III. RESULTS AND ANALYSIS

Packets traces in Wireshark are color coded so we filter different kinds of packets using a lot of command filters based on their meaning to analyze the traffic pattern of tested browsers. We extract the relevant information from traces to determine the proposed characteristics that affect the way of browsers in retrieving the contents of websites. This section shows the analysis of experimental results with sufficient drill-down to the web components level. We conduct several tests each test resulted in 125 values and we calculate their average and S.D to investigate a specific feature of each browser. Based on the results the smallest S.D is the most regular traffic pattern of samples and the largest S.D is most dispersed pattern of
samples. We strongly believe that the browser, who retrieves packets traces in a regular pattern, is less resistance for traffic analysis attacks than browser with a high randomized traffic pattern which appears to constitute a powerful protection against traffic analysis attacks.

A. Browsers’ incoming traffic pattern

The most important features, that distinguish between browsers behaviors, are those associated with the byte and packet counts in each direction. At the beginning of webpage retrieval, the HTML document is parsed and its embedded objects are requested from their server(s) that handle the requests of the browser. As each browser has different rendering engine, Java Script interpreter, HTML5 and CSS3 features support, the number of interacted packets is different in each browser. The retrieval of webpage contents are handled by a lot of requested and responded packets associated with their Acks and Syncs packets. The results of tests show that there are differences in the number of incoming and outgoing packets that the browsers interact with. Regarding the outgoing packet counts and sizes in general, the results on Figure 6 show that FF has the largest outgoing packets followed by Opera. This is because they are the only browsers that support link pre-fetch feature that allows browsers utilizing their ideal time in loading and fetching the contents that the user may use based on hints given by websites. On the other hand, the lowest outgoing traffic is issued by Safari followed by Chrome because they have an Origin header function that mitigates cross-site request forgery attacks resulting in less outgoing traffic.

![Figure 6: Average numbers of browsers’ outgoing packets for retrieving websites contents](image)
B. Browsers’ outgoing traffic

Although the browsers retrieve the same webpage, the incoming packets counts/sizes and fetched objects sizes are also different as it is shown in Figure 7 and 8 below. The experimental results show that in general pattern, Opera retrieves the largest incoming traffic. The reason behind is that Opera is the browser that supports Iframe feature, an element inserted into the HTML document like advertising objects that are retrieved from different domains, which makes the browser to create nested browsing contents resulting in more retrieved traffic. Furthermore, it also supports Link pre-fetch feature mentioned above which further increase the incoming traffic of the browser. The second largest incoming traffic retrieves by FF because it also supports link pre-fetch function.

![Websites' incoming packets retrieved by browsers](image)

Figure 7: The average number of WebPages’ incoming packets retrieved by browsers

On the other hand, Figure 9 shows that Safari and IE retrieve the least incoming traffic as they support blocking cross site scripting (XSS) feature which blocks the execution of JS code that appears in request URL. The XSS is a browser-based event that mitigates XSS attacks which results in reducing the traffic of corresponding browsers. We notice that Safari retrieves the most traffic of WebPages that do not have Asynchronous Java Script XML (AJAX) interactions like Wkipedia and Ebay. This observation supports the HTML’5 specification features that says Safari doesn’t support Asynchronous script execution [23]. The reason behind retrieving less incoming traffic by IE is that it has TostaticHTML function that filters dynamic html elements and attributes from html fragments for security purpose. Furthermore, the incoming traffic pattern is stable in browsers for
both the incoming packets counts and the sizes of webpages contents as it is shown in Figure 8 below.

![Figure 8: Average number of kilobytes of WebPages contents sizes retrieved by browsers](image)

**Figure 8:** Average number of kilobytes of WebPages contents sizes retrieved by browsers

### C. Stability of Browses’ traffic pattern

In our experiments we calculated the Standard Deviation values for samples set of each test in order to investigate the resistance of each browser against traffic analysis attack as it is shown in Figure 9 below. During the analysis of the results we observe that Chrome has the most regular pattern followed by Safari so the attacker can infer the identity of websites retrieved by these browsers easily. On the other hand, Opera has the most randomized traffic pattern which indicates that tracking its pattern for fingerprinting purpose will be more difficult. For FF and IE browsers they have not stable pattern so they usually jump between both sides.

![Figure 9: Shows the standard deviation of traffic pattern of browsers that indicates which browsers have randomized traffic pattern and which have regular pattern](image)

**Figure 9:** Shows the standard deviation of traffic pattern of browsers that indicates which browsers have randomized traffic pattern and which have regular pattern
To investigate the root causes that make different behaviors of browsers’ traffic patterns, we make a lot of browser tests (e.g. HTM5 support, JS and CCSs support) and we found that the main factors that make these differences are the duplicated and retransmitted packets that caused by the a delay some browsers. This delay happens when the browser has some delays for acknowledging the responses coming from a server which makes the server retransmit the response packer again causing the browser to generate a duplicate ack for the retransmitted packet. Figure 10 shows the result of the test where Opera causes the largest duplicates and retransmits followed by Safari which makes them most randomized traffic pattern. On the other hand Chrome causes the least duplicates and retransmits followed by FF which make their traffic the most regular patterns as it is shown in Figure 9 above. IE can be placed between both categories with most balanced traffic pattern.

![Figure 10: The retransmitted and duplicated packets of browsers in retrieving WebPages contents](image)

How fast the browser loads a webpage is an important factor for judging the performance of the browsers but this factor is not accurate as the browsers performance can be affected by other factors such as the structure ordering of WebPages contents, etc. For example, if the Style-sheets at the top of HTML document it makes the load faster as it allows the page to render progressively [18]. Safari has an interesting behavior in retrieval duration which is contradicted with the amount of its incoming traffic. Although Safari has the least amount of websites incoming traffic, it takes the most retrieval duration in most of cases. This is because that Safari differs from other browsers in that it doesn’t have progress function that manages fetching of WebPages contents efficiently based on the results of HTML5 support test. Progress function has tow attributes: The first is the value attribute that specifies how much of task has been completed and Max attribute that specifies how much the task requires in total [24]. Therefore, the
absence of this function in Safari affects its retrieval duration poorly as it is shown in Figure 10 below.

![Retrieval duration of websites by browsers in Seconds](image)

*Figure 10: The average retrieval duration that the browsers consumed in retrieving WebPages contents*

### D. Parallel downloads of web browsers

Regarding the parallelism connection of browsers, we test tow parameters which affect the performance of the browsers substantially. First we test the number of parallel connections that each browser opens per host name. By default each browser open only two parallel connections per hostname under HTTP/1.1 specification. As the web pages contain a lot of resources so download these resources two-at-a-time makes browsing the web too slow. To overcome this issue the HTTP 1.1 protocol provides browsers with the ability to perform several HTTP requests and open multiple connections in parallel. We test how many connections that each browser opens per a single hostname as well as per multiple hosts. The results show that there aren’t fixed parallel connections that open by browsers because we do many repeated tests for the parallelism of browsers and we found it is affected by other factors (e.g the number of hosts that the webpage contents retrieved from) because the number of parallel connections can be increased as the resources are fetched from more than one hostname. This technique called “domain sharding” which is the process of spreading page contents over several hostnames. This technique used to fool browsers into multithreading because the browser look at the hostname not the ip address since different hostnames can all point to the same IP [25]. The more parallel connections, the faster the WebPages render, so improve the performance of parallel downloads the website’s owner use
an effective strategy by spreading resources across multiple hosts. Therefore, based on our test results we can say that the number of parallel downloads of web browsers is affected mainly by server side in its website contents distribution approach. Moreover, the structure of the HTML document of WebPages are also has an impact on the parallel downloads. In our experiments we test whether the browsers download scripts in parallel with other resources or whether they block other downloads until scripts are parsed and executed. We take in account this important point because we expect that parsing and executing scripts in order is important for maintaining code dependencies. The results show that all browsers download scripts in parallel with other resources but we observe an interesting performance issue. For example, in FF browser when style-sheet tag is followed by an inline script all other downloads is blocked because an inline script depends on CSS rules in the style-sheet [26]. The result show that when script is placed between style-sheet and image tag, the image is not downloaded until the style-sheet finishes Table 3 show the browser support of downloading stelle-sheets with scripts

<table>
<thead>
<tr>
<th></th>
<th>Chrome</th>
<th>FF</th>
<th>Safari</th>
<th>IE</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CSS + Inline script</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

*Table 3: Shows the CSS parallel download browser support with other resources*

The table 3 shows that in which browser support downloading style-sheet CSS with other resources in parallel. Furthermore, the table shows also which browser support downloading style-sheet followed by inline script it is faster but it could lead to unexpected results as we see in the performance issue of FF above.

IV. DISCUSSION

It is most important for the researchers who interested in any research related to web browsers to know the browsers based factors that affect the traffic pattern of each browser from different aspects. In fact, there are noticeable differences among the top five web browsers in retrieving WebPages contents across the range of considered tests. For example, in the outgoing traffic feature the largest traffic are issued by FF followed by Opera as they support Link pre-fetch function that allow the browser to utilize the ideal time for fetching more contents that the user may
used based on hints given by retrieved webpage. On the other hand, Safari and Chrome issue the least as they support Origin header feature that mitigate cross-site requests.

Regarding the browsers incoming traffic, Opera retrieved the largest incoming traffic as it has Iframe function that allows inline Iframe element to create a nested browsing context making Opera to retrieve the largest traffic. The second largest incoming traffic is retrieved by FF as it supports pre-fetch function. The least incoming traffic is retrieved by Safari as it support Origin header that mitigate cross-site responses followed by IE that support security policy 1.0 which more restrictions for incoming resources.

The main objective of this research is to investigate the resistance of the top five web browsers against traffic analysis attack. The experimental results show that Opera has the most randomized or dispersed pattern in all tests followed by Safari making the traffic analysis attack more difficult. From the other side, the results show that the traffic analysis attack on Chrome and FF can be recognized easily as they have the most regular traffic pattern in their retrieval browsing. Our testing results show that the retrieval duration (speed) and parallel downloads are not determinant by browsers as there are many factors that affect these two important performance measures mentioned in analysis section.
V. CONCLUSION AND FUTURE WORK

This paper compares the resistances of the most commonly used browsers against traffic analysis attack. We investigate different aspects that affect the traffic patterns of browsers from coarse grained point of view (e.g. like the quantity of packets in each direction with the reasoning of browser based factors that affect their traffic behaviors) to point out which browser protects the privacy of users against eavesdropper attack more and which browser makes the external observer identifying users’ identity (e.g. browser fingerprinting, website fingerprinting) easily based on traffic pattern analysis. To the best of our knowledge this is the first study in the publications that investigate the retrieval behavior of the browsers from such scenarios. From our results we conclude that Chrome and FF will secure less against traffic analysis attack and Opera and Safari will secure more with experimented causes mentioned above. As a future work we intend to do this research on the anonymity network where Tor protocol is turn on (encrypted traffic). We suspect from our testing results that the rest of browsers will be more secured on anonymity protocol against traffic analysis attack. Furthermore, a hot research issue in browsers resistance against traffic analysis attack 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 smart phones and tablets to browse internet in their daily life and their privacy can be endangered with the recent capturing tools that can eavesdrop their traffic wirelessly.

REFERENCES

[3] Boukari Souley, Amina S. Sambo “A Comparative Performance Analysis of Popular Internet Browsers in Current Web Applications” (ATBU), Bauchi, Nigeria
Web Browsers Resistance to Website Fingerprinting Attack

TaherAl-shehari and Sami Zhioua

Information and Computer Science Department
King Fahd University Of Petroleum And Minerals
Dhahran 31261, Saudi Arabia

Email: {taheralis99@gmail.com, zhioua@kfupm.edu.sa}

Abstract— 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. Although Tor is considered as the most commonly used anonymity system, an attacker can recognize visited websites 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 websites 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 paper we conducted a website fingerprinting attack over "Tor" using the most commonly used browsers to investigate their resistances against traffic analysis attack based on features inferred from their traffic patterns. To the best of our knowledge, this is the first comparative and analytical study to evaluate browsers’ resistances against website fingerprinting attack based on a deep analysis that reaches the main causes that stand behind their different resistances. The aims of applying website fingerprinting attack on top web browsers are three folds: First, to expose possible privacy vulnerabilities in (Tor and browsers) that the attacker may exploit to do traffic analysis attack in order to push Tor and browsers makers to empower their products to defeat any possible exploits by attackers. Second, to raise web users’ privacy awareness by figuring out to which extent each browser protects their privacy. Third, the outcomes of our research might be very beneficial to security forces and law enforcement of the governments to highlight what are the websites that their citizens may visit anonymously on different browsers. Our empirical results showed to which extent each browser protects against websites fingerprinting attack as well as exposing the root causes stand behind different resistances of browsers.

Index Terms— Information Privacy, Anonymity Systems, Tor, Website Fingerprinting Attacks, Web Browsers, SVM.

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 [6] and Jap [21] 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 websites 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. To the best of our knowledge, we are the first to push the boundary of knowledge from this angle.

1.1. Privacy Issues on 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 [7] 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" [8]. However, the privacy of web users is satisfied when the usage, exchange and release of their information can be one hundred percent under control [9]. Unfortunately, this is not the case as the data of web users is transmitted on cyberspace so they do not maintain full control over their information so that 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 an 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 [10]:

1. The identity of their visited websites and their locations/origins on Internet space which can be revealed using website fingerprinting attack.
2. Their E-mail addresses and contact information that can be exported for targeted advertisements and attacks.
3. Their personal information such as Passwords and Credit cards which can be misused against users' preference and interest.

These are the most important privacy issues that violate the privacy of web users which can be triggered from various sources. Figure 1 shows possible entities that may carry out such kind of privacy violations which web users should take care of.

![Figure 1. A survey conducted by Pew Research Center in July 2013 shows the percentage of possible eavesdroppers that may violate the privacy of web users [45].](image)

Therefore, users must take several precautions to protect their privacy against such privacy issues as well as designing containment strategies if their personal information has been collected.

To solve privacy issues on the internet, multiple privacy enhancing technologies have been proposed (e.g. SSL, IPSec, SSH, Tor, etc) so these 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 the identity of 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 [11], [12], [13], [14], and [15].

### 1.2. Anonymity Protocols

In this section we defined the meaning of "anonymity" and highlighted 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." [16]. They meant by the anonymity set is that 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 in traffic analysis attacks, several anonymity systems have been developed in order to enhance the security and privacy of users over the internet. So the anonymity systems provide strong privacy protection by encrypting the transmitted data ranging from simple to complicated method. For example, Chaum [17] 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 such as (e.g. padding and mixing packets) in order to make it very difficult for attackers to trace and analyze the traffic [18]. 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 the possible attacks that are based on timings such as Mixminion [9] and Mixmaster [19]. These systems employ various strategies (mixing, reordering and patching) to defend against traffic analysis attacks that are based on packet timings/delays [46]. 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) [20], Java Anon Proxy (JAP) [21], and Invisible Internet Protocol (I2P) [22]. In our experiments we have selected Tor anonymous system to evaluate different resistance levels of browsers 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 the 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 [23], [24], [25], [26], [1], and [27] with different features and encryption mechanisms have been used by web users to protect their privacy from real threats that have been increased since last decade.

### 1.3. Website Fingerprinting

Hints is the first who described the traffic analysis for identifying visited websites as a term of “Website Fingerprinting” [28]. Website fingerprinting is one kind of traffic analysis attacks that enables the adversary to infer a visited web page that the victim may visit for violating his privacy even if he uses certain anonymous system like Tor [29]. 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 is fetched with its referenced contents (e.g. CSSs, JSs, Images, Text, etc). A fetched content has specific characteristics (size, order, direction and delay). Encrypting protocols (e.g. encrypting tunnels, SSL, Tor, etc) encrypt the contents of transmitted information but they do not effectively encrypt some features such as packets sizes, directions, timing, etc [30]. Therefore, it is possible for an eavesdropper to monitor/sniff the network traffic of a victim and profile fingerprint about website 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, 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 banned websites like facebook and twitter in Iran. 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 previously recorded fingerprints of banned list of websites. Some websites are changed continuously (e.g. news websites) so the government should generate new fingerprints frequently to cope such changes [28].

Existing works in website fingerprinting attacks [2], [31], [32], [33], [3], and [5] 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 try to defeat website fingerprinting attack by using several techniques and tricks such as mixing and padding of transmitted packets. More details about different encrypting techniques for defeating traffic analysis attacks are found in [18].

In our study we have selected Tor anonymity system as it is the most popular anonymous systems in use today “currently used by around 500,000 daily clients and carrying 2000 MB of data per second [34]”.

In our experiments we carried out a website fingerprinting attack using the most commonly used browsers on real user who defends himself using Tor for the sake of investigating to which extent each browser defends against website fingerprinting attack. Our research outperforms existing ones in that it breaks Tor anonymous system using the most commonly used web browsers while all existing works without exception used only Firefox web browser.

1.4. 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 various web browsers such as Chrome, Firefox, Internet Explorer, 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 for 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) [35]. 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 functionalities to perform those tasks and protect user from any malicious content residing on the World Wide Web. The details behind different browsers’ structure and functions are described in [36]. Our focus in this study is to evaluate different resistance levels of the top five web browsers against website fingerprinting attack as they have relative importance among applications today. In this research we have selected the most commonly used web browsers namely Chrome, Firefox, Internet explorer, Safari, and Opera according to the market share statistics of browsers as it is shown in the distribution of pie chart in Figure 2.

Figure 2. Browsers' Statistics and Trends On May, 2013 [38].

1.5. 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 fingerprinting attack model to test various resistance levels of popular web browsers by conducting several experiments on tested browsers (e.g., various web technologies, Java Script APIs, parallel downloads, etc) which have a great impact on web browsers traffic patterns. Furthermore, we figured out the root causes that make each browser behave differently in term of privacy protection. Our empirical analysis 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).

1.6. Overview of contributions

The contributions of this paper are to fingerprint websites being accessed over Tor using the most five 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 in their rendering engines and fetching schemes that affect their traffic patterns substantially.
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 of the most commonly used web browsers with respect to their resistance to website fingerprinting attack as well as figuring out the root causes that stand behind various resistance levels of browsers against traffic analysis attack.
1.7. Paper organization

In this section we outlined the sequence organization of this paper. Section 1 explores the motivation and relevant background behind the topics discussed in this research. Section 2 demonstrates the existing published researches that have been proposed in website fingerprinting attack.

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.

Section 3 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 phase in Section 4.2. In Section 4.3 we demonstrate 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, CSSs, 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 rise 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.

2. Literature Review

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 a useful information pattern for training traffic instances. Then, a classification mechanism is used to identify testing traffic instances. In this section we outline the existing approaches for website fingerprinting attacks.

The first author who refers to the process of identifying websites under encrypted connection as a term of "fingerprinting" is Hintz [28]. 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 [38], 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 [33] 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 achieve 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 encryption systems.

Shi et al. [39] proposed a novel method for website fingerprinting attack by analyzing the traffic of a victim under Tor anonymity system. They divide both 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. [31] 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 by using negative and positive values. In addition, they inject some features in these sequences to rise 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 to fine-tune the SVM parameters. They evaluated their method using two scenarios: Closed-world and Open-world. In closed-world scenario they do their experiment on the same data set of Federrath et al. [33] 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 [40] in addition to 5 censored websites. The experiment results show that they improve the recognition results of previous works in Tor from 3% to 55% and in JAP from 20% to 80%.

Cai et al. [3] proposed new methods for achieving a higher accuracy than previous works in Website fingerprinting attack. They used string alignment using Damerau-Levenshtein distance algorithm to compare the previously made fingerprints and observed traffic using 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 [5] in which they proposed new techniques for website fingerprinting attack under Tor with a higher accuracy than previous techniques. 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%.

To the best of our knowledge, none of these earlier 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 used only a single web browser (Firefox). 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 encrypted traffic patterns as well as the accuracy rate of website fingerprinting attack on each browser.

3. EXPERIMENTAL EVALUATION AND ANALYSIS

In this section, 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.

3.1. Experimental Setup

In order to implement the attack, we first create a program that analyzes a tcpdump log and generates a fingerprint of the https traffic in the log.

In order to test the protection range of browsers to user privacy 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.

<table>
<thead>
<tr>
<th>Tools</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Windows 8</td>
</tr>
<tr>
<td>Tshark</td>
<td>1.10.0</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>31.0.1650.6</td>
</tr>
<tr>
<td>Firefox</td>
<td>26.0</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>10.0.9200.1638</td>
</tr>
<tr>
<td>Safari</td>
<td>5.1.7</td>
</tr>
<tr>
<td>Opera</td>
<td>12.16</td>
</tr>
<tr>
<td>Tor</td>
<td>3.5.2.1</td>
</tr>
</tbody>
</table>

Table 1: Software Tools

<table>
<thead>
<tr>
<th>Devices</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>intel(R) Core(TM) Duo CPU T9600 @ 2.80 GHZ</td>
</tr>
<tr>
<td>Memory</td>
<td>4 GB</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1EECD81C-907A-475B-ACF1-D106043C1F6D</td>
</tr>
</tbody>
</table>

Table 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 web pages, browser configuration, client hardware configuration, 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 version available of web browsers with default configuration without any installed add-on/extension to make sure that the observed results are browsers-dependent.

In order to investigate the resistance levels of browsers against website fingerprinting attack, we set up our attack platform scenario illustrated in Figure 4.1. Our attack passed through several phases until we got the website fingerprinting attack results on tested browsers.
The first phase is the data collection phase where we collected our data set consisting of 1500 traffic samples for 20 web pages 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 operations on our main data set to be prepared for classification phase: we removed TCP control packets (Acks and Syncs) because they reduce the performance of the system and don't add useful information in classification process. Then, we filtered only Tor packets from other non-Tor packets that might captured during traffic sniffing.

3.2. Data collection and Features Extraction

Tor anonymous system is a live network with millions of daily users [34] 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 collected the 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 web pages, we first established Tor network connection and configured web browsers to use Tor proxy with its default configuration.

We investigated the effectiveness of our fingerprinting approach on the main pages of the top 20 most popular websites ranked by Alexa statistics [40]. We have 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 victim visits certain website from a list of predefined/fingerprinted websites by an attacker. Therefore, when a victim visits a certain website, the attacker observes the victim's trace pattern and attempts to guess which website that the victim visits from the list of previously fingerprinted websites. Table 4.3 lists the top 20 websites that we used for our study.

To automate the browsing of websites, we have 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 websites 15 visits in a round-robin fashion within controlled environment. During the web browsing visits, we scripted tshark, the command-line version of WireShark traffic analyzer, to capture traffic packets in trace/log file for each visited websites from real traffic recorded on tested browsers. Each 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, all HTTP traffic is tunneled through default Tor configuration. We set 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 information of its visited web page (e.g. the time of sent and received packets, the lengths of packets, the order in which the packets were sent or received, etc). Then, we extracted certain features from such information to create a profile for each visited web page which called "Fingerprint".

To prepare the data for classification phase, we built scripting program to extract the sizes and the directions of retrieved packets as features for 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 objects by a web page and the order in which browser issues or retrieve them.

3.3. Classification of web browsers resistances

To prepare the data classification phase, we built scripting program to extract the sizes and the directions of retrieved packets as features for websites fingerprints. The profile/trace of each website is represented as fingerprint stored as a sequence of integers in the observed direction positive for outgoing packets while the negative for incoming packets (e.g. -1150, 1500, -638, 638, etc). These integers represent the sizes and directions of TCP packets generated to load website contents over Tor. Finally the classification phase where the website traces/fingerprints are classified using Cai et al. method described in [3].

So the similarity between website fingerprints is calculated using Damerau-Levenshtein distance algorithm. It figures out the similarity between websites fingerprints by calculating the number of operations (insertion, deletion, substitution, and transposition) required to transfer trace \( t = \{-1150, 1500, -638, 638, \text{etc}\} \) into trace \( t' = \{638, -1150, 638, 1500, \text{etc}\} \). The minimum number of operations to transfer \( t \) into \( t' \) is the more similar they are to each other and they are considered as two visits from the same website. Then, the similarity distances of websites fingerprints are classified using Support Vector Machine. As a result, we got the fingerprinting accuracies of tested browsers. To the best of our knowledge in this study we covered the most commonly
used web browsers’ data traffic by organizing and classifying each browser data-set separately. Figure 4.2 shows a graphical representation of our fingerprinting results over tested browsers. It gives an overview of different resistance levels of browsers against website fingerprinting attack.

![Figure 4. The resistance rate of web browsers against website fingerprinting attack over Tor.](image)

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 shows the different recognition rates that we achieved of websites fingerprints 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. The results show that the highest recognition rate is 74% 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 are achieved by Opera of 41.6% followed by Safari of 53.8% which indicate that the surveillance of their traffic pattern 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. We found out that in the real Tor network the accuracy results of identifying websites over Chrome and FF is approximately the same as they share the features which make them behave equally. However, the root causes behind these various resistance levels of browsers against website fingerprinting attack are explained in details in the subsequent sections.

### 4. Root Causes Behind Different Levels of Browsers’ Resistance

After various results of website fingerprinting attack came to light, we need to figure out the main reasons behind various website fingerprinting results on Tor anonymous system. In this chapter we investigate the root causes of different levels of browsers resistances 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 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 several statistical analysis tests 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 chapter 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.

#### 4.1. 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 web technology features and the reflection of these features to their behavior in retrieving various web contents.

We have 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 [42], [43], and [44]. Table 5.1 shows the results of web technologies that we compare tested browser against. It shows the various supports of tested browsers to popular web technologies. The sign ($\text{√}$) indicates the browser supports the associated feature and ($\text{×}$) sign indicates the browser doesn’t support the associated feature.

<table>
<thead>
<tr>
<th>Features</th>
<th>Chrome</th>
<th>FF</th>
<th>IE</th>
<th>Safari</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous script execution</td>
<td>$\text{√}$</td>
<td>$\text{√}$</td>
<td>$\text{√}$</td>
<td>$\text{√}$</td>
<td>$\text{×}$</td>
</tr>
<tr>
<td>Navigation</td>
<td>$\text{√}$</td>
<td>$\text{√}$</td>
<td>$\text{√}$</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
</tr>
<tr>
<td>Timing API</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
<td>$\text{√}$</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
</tr>
<tr>
<td>ActiveX</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
<td>$\text{√}$</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
</tr>
<tr>
<td>Native Flash blocking</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
<td>$\text{√}$</td>
<td>$\text{×}$</td>
<td>$\text{√}$</td>
</tr>
<tr>
<td>Cached compiled programs</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
<td>$\text{×}$</td>
<td>$\text{√}$</td>
</tr>
</tbody>
</table>

We selected these features as a significant metric for browsers investigation as they have a great impact on the shape of browser’s traffic patterns. The main functionality of these features and their impact on web browsing traffic patterns are explained in the subsequent sections. These features affect the browser-side footprint heavily when retrieving rich-contents of websites. Furthermore, web
technologies (e.g. ads filtering, cookie blocking, sandboxing, flash blocking, etc) also affect 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.

### 4.2. The impact of browsers-dependent features on their web browsing traffic patterns

When a browser sends an HTML request, the corresponding server will handle the request and deliver 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 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 next section.

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.

After we did extensive analysis on all retrieved 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 Table 5.3 we demonstrated some interesting metrics that are correlated to their browsers-dependent features shown in Table 5.1 above. 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.

#### Table 3: Characterizing Traffic Characteristics Generated by Tested Web Browsers.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Average JS data flow [KB]</th>
<th>Number of empty trace files</th>
<th>Average No. of third-party domains</th>
<th>Average loading time [Sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>6845.692</td>
<td>23</td>
<td>533</td>
<td>27.426</td>
</tr>
<tr>
<td>Firefox</td>
<td>9127.72</td>
<td>21</td>
<td>493</td>
<td>28.394</td>
</tr>
<tr>
<td>IE</td>
<td>14855.82</td>
<td>24</td>
<td>603</td>
<td>29.298</td>
</tr>
<tr>
<td>Safari</td>
<td>4172.84</td>
<td>22</td>
<td>434</td>
<td>19.322</td>
</tr>
<tr>
<td>Opera</td>
<td>517.44</td>
<td>67</td>
<td>0</td>
<td>12.028</td>
</tr>
</tbody>
</table>

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.

### 4.3. The impact of JavaScripts on web browsers traffic patterns

JavaScript is enabled by default on all major browsers, and it was reported by Alexa [25] that 98 out of 100 popular websites use JavaScript. Furthermore, Michael et al.[41] proved that JavaScript contents occupy the fraction of 25% across all downloaded web contents. Therefore, we have started to analyze the behavior of tested browsers on this large portion of web browsing content. Each browser’s JavaScript interpreter 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 which allow the scripts to communicate with remote servers [35]. 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. The various amount of JS data flow which reflects the different behaviors of browsers’ JavaScript engines.](image)

Animation on web pages is the characteristic of JavaScript behavior so the more JS data flow in a website retrieval, 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 have noticed that the browsers-dependent features reported in Table 4.4 above create JS data flow fingerprints 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 ActiveX distinctive feature 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, Flash animation contents are created by JavaScript Flash language (JSFL) so IE allows all flash contents to be loaded as it is integrated with Adobe Flash. 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" 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, 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 stuff 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 4.2. However, the results show that JS behaviors of browsers have relevant impacts on their various resistances against website fingerprinting attack.

4.4. 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. In our traffic analysis tests, we found that there are a lot of web contents retrieved from different origins. 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), track user activity by analytical services (e.g. quantserve and google-analytics), and fedded 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 stuff to be retrieved based on the variety of their security policies.

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 contacted third-party servers of 603 domains. This is because IE allows all flash Ads contents to be loaded from third-party domains as we stated earlier that Adobe Flash is integrated with IE. Chrome deals with 533 third-party servers followed by FF with 493 so the reason behind is that 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). Safari retrieves stuff from 434 domains 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 features such as Pop-ups blocking and cookie disabling which are disabled by default for the sake of more security restrictions. This is why Opera has the least recognition rate of website fingerprinting attack results 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. This is clear in IE browser so while it has the highest number of retrieved third-party websites, it also has the highest rate of website fingerprinting attack of 74% which means it has least privacy protection. Moreover, Opera browser blocks all third-party domains so it has the least 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. 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.

4.5. 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 Ack 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 precise load time management are the main source of variations in the consistent order of packets. Furthermore, the dynamism of network conditions and server state causes much random noise in the sequence order of packets as it is shown in Figure 5.3. As it is shown in the screenshots of 5 visits for the same website retrieved by the same browser so there is an inconsistency between the sequence order of requesting website objects which indicate that the sequence order of retrieved objects is not browser specific. 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.

![Figure 7. The inconsistency between the sequence order of retrieved objects for 5 visits to the same website.](image)

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. In parallel downloads we observed 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 Asynchronous 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.

![Figure 8. The impact of JavaScripts blocking on web browsing traffic pattern of Opera.](image)

Thus, Opera does not support requesting JSs with other objects in parallel for 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/consistency between the lengths of websites visits/traces. However, the recent implementation of the W3C Navigation Timing API specification 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 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 ‘cite[navigation]’. 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 9. The variations in retrieval time of browsers and their impact on the consistency of web browsing traffic patterns.](image)

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

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 visits/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.

Figure 11. The number of empty log files collected over browsers during data collection phase which affect the website fingerprinting accuracy of browsers.

The characteristic of the 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 a 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 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 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.

Figure 12. The security features that caused the most empty log traces of Opera.

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 don’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 visit rejection or bypassing the certificate warning to visit a website which the full security cannot be guaranteed. In this case there wasn’t any response to the warning dialog message triggered 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. Discussion

There is no any research in the literature to evaluate 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 web page fingerprinting attack.

5.1. Dynamic contents

The contents of web pages 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, theses dynamic changes may reduce the accuracy of web pages 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.

5.2. Localization

There is another factor that may reduce the fingerability which is known as website localization so the visits of same website are different depending on its locality. For example, the contents of “www.google.com.sa” is different from “www.google.co.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 the contents of the same website is different based on its locality. Fortunately, in the Figures tow birthdays that interest different localizations so the same locality behavior can be applied for ads that are targeted to different countries.

![Image](Image)

Figure 13. 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 of this

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.co.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 such as "www.google.com" as it is shown in Figure 5.6.

5.3. 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.

Acknowledgments

We would like to thank King Fahd University of Petroleum and Minerals for its support to conduct this research.

Conclusion and Future Work

There are relatively little research works on website fingerprinting attacks on Tor and all of them are configured over Firefox web browser only. In this paper 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 proposed attacks in two folds. First, we implement web page fingerprinting attack on the most commonly used web browsers which outperforms previously proposed attacks that use only a unique browser "Firefox”. Second, we investigate the browser-dependent features to uncover 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 their own behavior 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 different browsers 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 that each browser will protect against web page fingerprinting attack on different web pages classes (e.g. business websites, news websites, social networking sites, Forum website, Gallery Website, Gaming website, Search engine sites, etc). Furthermore, a hot research issue in browsers resistance against traffic analysis attack 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 smart phones and tablets to browse internet in their daily life so 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 browsers which fit the restriction of mobile browsers in order to raise the privacy awareness of mobile users.

REFERENCES


