Traffic Analysis of Web Browsers

Sami Zhioua* and Mahjoub Langar**

*Information and Computer Sciences Department, KFUPM
P.O. Box 958, Dhahran 31261, KSA
zhioua@kfupm.edu.sa
**Ecole Nationale des Ingenieurs de Tunis
B.P. 37, Belveder, Tunis
mahjoub.langar@gmail.com

Abstract. Tor network is currently the most commonly used anonymity system with more than 300,000 users and almost 3000 relays. Attacks against Tor are typically confirmation attacks where the adversary injects easily discernible traffic pattern and observes which clients and/or relays exhibit such patterns. The main limitation of these attacks is that they require a "powerful" adversary. Website fingerprinting is a new breed of attacks that identifies which websites are visited by a Tor client by learning the traffic pattern for each suspected website. Recent works showed that some classifiers can successfully identify 80% of visited websites. In this paper we use a classic classifier, namely, decision trees (C4.5 algorithm) and we study to which extent popular web browsers can resist to website fingerprinting attacks. Among four studied web browsers, Google Chrome offers the best resistance to website fingerprinting (5 times better than the other web browsers). Since most of existing fingerprinting techniques have been evaluated using Firefox web browser, we expect the accuracy results of existing works to be reduced in case Chrome browser is used.

1 Introduction

Anonymity systems, such as Tor [1] and Jap [2] are designed primarily to provide privacy and anonymity to Internet users living in oppressive regimes giving them the opportunity to evade censorship. These systems achieve anonymity by embedding user data inside several layers of encryption and by forwarding the traffic through a set of relay nodes/proxies. This makes the job of an eavesdropping adversary much more challenging since by just observing the traffic she cannot deduce who is communicating with whom and what is the type of traffic exchanged.

Tor [1] represents the current state-of-the-art in low-latency anonymity systems. The Tor network is currently the largest deployed anonymity network ever, consisting of almost 3000 relays and more than an estimated 300,000 users.

Most of attacks against Tor anonymity system were traffic confirmation attacks where the idea was to inject easily discernable traffic pattern and observe which potential clients are exhibiting such patterns [3–7]. Most of these attacks require a "powerful" adversary which is assumed to observe the traffic of a significant number of Tor relays and in some attacks to inject malicious relays in the Tor network. These assumptions are relatively strong and beyond the capabilities of most of attackers including totalitarian regimes. A more practical attack on Tor which does not require strong assumptions is passive *traffic analysis*. Traffic analysis consists in intercepting and analyzing the traffic messages (usually encrypted) in order to reveal information about the communication (e.g. the identities of the communicating entities, the type of data exchanged, etc.). To carry out the attack, the adversary is assumed to observe the traffic of only one side of the communication (usually the Tor client). This threat model is very common and holds particularly in presence of censorship.

Website fingerprinting [8] is a variant of passive traffic analysis that can be carried out by a local eavesdropper or by any entity observing Tor client traffic. In this attack the adversary analyzes the traffic to extract patterns that can reveal the identity of the website accessed by the client. Patterns are constructed from certain features in the traffic such as the size of transferred data, the timing, the order of packets, etc.

Website fingerprinting was first used to analyze encrypted HTTP traffic [8–11]. Most of these attacks were based on tracking the size the objects fetched by the main web page. With the migration to HTTP/1.1 which makes use of persistent connections and pipelining, it is no longer possible to easily distinguish between single objects fetching. Only few works focused on implementing website fingerprinting on anonymity systems [12–15]. It turned out that website fingerprinting is much challenging when applied on anonymity systems in particular Tor. The reason is that Tor protocol performs some structural modifications in the traffic: restructuring the traffic into fixed size cells, merging small packets together, multiplexing TCP streams, etc. However, despite these challenges, recent works showed that the precision of website fingerprinting could be as high as 80% when applied on Tor [15].

Tor protocol might be used with different web browsers[1]. Since web browsers use different user agents and process data packets differently, the choice of the web browser should have an impact on the efficiency of website fingerprinting. In this paper we study the impact of the choice the web browser on the anonymity of Tor clients with respect to website fingerprinting attacks. We consider a representative set of four popular web browsers, namely, Firefox, Chrome, Konqueror, and Internet Explorer, and we empirically analyze to which extent they resist to website fingerprinting. This is the first work in the literature that studies the efficiency of website fingerprinting while using different web browsers. All existing works were focusing on a single web browser, mainly Firefox.

The contributions of this paper are two-fold:

1. A detailed and complete survey of existing website fingerprinting approaches in particular targeting anonymity systems.
2. A comparative analysis of the most popular web browsers according to their resistance to website fingerprinting.

[1] Provided that the web browser allows to configure the socket proxy

2 Related Work

Early website fingerprinting techniques were focusing on analyzing simple encrypted HTTP traffic. Hintz [8], which is the first to use the term "fingerprinting" to refer to this type of attack, implemented a simple website fingerprinting attack targeting the SafeWeb encrypting web proxy [16]. The attack was based on tracking the size of objects fetched by a visited website. This was possible because the author did a strong assumption that every web object (image, ads, etc.) is fetched through a separate TCP connection using a different port. His experiment was a simple proof of concept distingushing only 5 websites. He achieved a detection rate between 45 and 75%.

Similarly, Sun et al. [9] based their approach on the size of fetched objects. Objects are isolated in the encrypted traffic by counting the number of packets between blocks of requests. The fingerprint is expressed as a multiset of object lengths. An unknown traffic sequence is then evaluated against website fingerprints using a measure of similarity (Jaccard's Similarity [17]). A similarity value more than a threshold c indicates a matching. In their empirical analysis, Sun et al. constructed a database of 2000 website fingerprints and then tried to distinguish these same 2000 websites out of a set of 100,000 websites. The optimal accuracy was obtained with the threshold c equal to 0.7 where 75% of the 2000 websites were correctly identified with a false positive rate of 1.5%.

The strong assumption that web objects can be distinguished by observing the different TCP connections does not hold anymore since with the migration to HTTP/1.1, no TCP connection is opened for each object as was the case in HTTP/1.0.

Bissias et al. [10] were the first to use IP packet sizes and Inter-Packet-Time (IPT) instead of the size of fetched objects to fingerprint websites. From every website visit they extract two traces: one size trace and one time trace. The size trace is the sequence of packet sizes while the time trace is the sequence of IPT times. Then all traces corresponding to a given website are merged into a website profile[2] by computing the arithmetic mean at every time step. Once a bank of profiles is constructed, an unknown network traffic is matched with each one of the constructed profiles using Cross Correlation [18]. The empirical analysis was based on the 100 websites most visited in the authors's department (University of Massachusetts) and showed that size profiles are much more efficient in identifying visited websites than time profiles. With size profiles, 20% of the analyzed traces are correctly identified after one guess and 65% after 10 guesses while with time profiles 8% of websites were correctly identified after one guess and 27% after 10 guesses. The analysis showed also that the time gap between the training phase (constructing the profiles) and the testing phase has only a small impact on the accuracy: a one hour gap is only 5% better than a 168 hours gap.

Liberatore et al. [11] obtained much better results by focusing only on packet sizes. In their work, they represented a traffic trace as a vector of packet size

[2] Actually two profiles: a size profile and a time profile

frequencies: each visit will result in a histogram of packet size frequencies. They tried two classification techniques: Jaccard's Similarity [17][3] and Naive Bayes [19]. The empirical analysis was also performed using the University of Massachusetts's typical traffic by filtering the top 2000 visited websites. Jaccard's based classification was slightly better than Naive based one with 73% of website visits correctly identified. Experiments showed also that the training set need not be very large since a training set of size 4 resulted in almost the same accuracy of training set of size 12.

All above works focused on website fingerprinting typical encrypted HTTP traffic. As anonymity systems became popular, recent website fingerprinting contributions focused on attacking those systems, in particular Tor [12–15].

Shi et al. [13] detailed a website fingerprinting attack on Tor. They adapted Hintz [8] and Sun et al. [9] techniques for Tor since instead of tracking the size of fetched objects (which is not possible in Tor), they tracked the number of packets sent or received in every interval[4]. A traffic trace is then represented by a vector specifying the number of intervals with 2 packets, the number of intervals with 3 packets, etc. Once a profile is built for a website (after several visits), the similarity between a profile and a traffic trace is computed using cosine similarity. The technique was evaluated using the traffic from the top 20 websites in Japan. They could identify successfully 50% of the visited websites.

Federrath et al. [12] used a Multinomial Naive Bayes (MNB) classifier to website fingerprint 6 anonymity systems: 4 single-hop proxy VPNs and 2 multi-hop systems: Tor and JonDonym [2]. As in Liberatore et al. [11] a traffic trace is represented as a histogram of packet size frequencies distribution without taking into consideration the packet ordering nor packet timing. They improved the efficiency of the MNB classifier by using text mining optimizations [19] such as Term Frequency Transformation, Inverse Document Frequency, etc. The evaluation was based on the top 2000 websites extracted from the log files of medium-range proxy server used by 50 schools. These 2000 websites has been filtered to 775 websites. The accuracy of their technique was very good for single-hop anonymity systems where 94% of website visits were correctly identified while it was relatively poor for multi-hop anonymity systems: 20% for JonDonym and only 3% for Tor. This shows once again that website fingerprinting is much more challenging with anonymity systems than with typical encrypted HTTP traffic.

Panchenko et al. [14] focused only on Tor and JonDonym and used SVMs (Support Vector Machines) for classification. 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 help in the classification such as size markers (whenever flow direction changes, insert the size of packets in the interval), number markers

[3] To use Jaccard's Similarity as a classifier, they turned the metric value into a class membership probability by dividing it by the sum of all metric values in the traininig set.

[4] An interval refers to the time period without packet flow change. Moving from one interval to the other happens when the direction of the flow changes.

(number of packets in every interval), total transmitted bytes, etc. They used Weka tool [20] to fine-tune the SVM parameters. The proposed technique has been evaluated using two experiments: Closed-world and Open-world. In the closed-world experiment, the same set of 775 websites of Federrath et al. [12] as well as ten-fold cross validation have been used to estimate the accuracy. As of open-world experiment, 5000 websites have been randomly chosen among the top one million websites according to Alexa [21] in addition to 5 censored websites. The closed-world experiment showed that the SVM technique resulted in an accuracy of 30% for the basic variant and 54% when all features are used[5]. The open-world experiment showed that, censored websites were successfully identified with a true positives rate between 56 and 73% while the false positives rate was less than 1%.

The most recent contribution was by Cai et al. [15]. As in Panchenko et al. [14], they represented a traffic trace as a sequence of (negative and positive) packet lengths. The training and testing is based on an SVM with a distance-based kernel. They tried several variants of parameters and distances and obtained the best accuracy with a Damerau-Levenshtein edit distance [22, 23]. The use of this distance is motivated by the fact that it is the length of the shortest sequence of character insertions, deletions, substitutions, and transpositions required to transform a trace t to t'. These operations correspond to discarding and reordering of packets in a stream. In order to compute the distance between two traces of different lengths, the Damerau-Levenshtein distance is normalized with respect to the length of the shortest trace between the two. For evaluation they used the top 1000 websites according to Alexa which are then filtered to 800 websites. Using the basic version of Tor, they could successfully identify 80% of visited websites. However, when random cover packets are added to the traffic, the accuracy falls to 50%. Decreasing the size of the training set from 36 to 4 samples decreased the accuracy with 20%.

In all aformentioned works, without exception, the website fingerprinting approaches have been evaluated using only one web browser, mainly, Firefox. Since web browser use different browser engines and user agents and consequentely fetch pages differently, we strongly think that the chosen web browser has an impact on the estimated accuracy. In this paper, we consider a representative set of popular web browsers and repeat the data collection and experiments for every one of them. Our aim is to compare the resistance of web browsers to website fingerprinting attacks.

3 Threat Model

The typical threat model for anonymity systems is a global passive adversary that can observe all the traffic of the network. However, since Tor is a low-latency anonymity system, it has been designed to protect against a weaker form of adversary. Indeed, it is assumed that the adversary can observe only

[5] The feature with the highest contribution was the total number of packets in the traffic trace.

some fraction of the network traffic; who can generate, modify, delete, or delay traffic; who can operate onion routers of his own; and who can compromise some fraction of the onion routers [1]. In this paper we assume a weaker threat model where the attacker can only access the encrypted traffic between the client and the first Tor relay. The attacker does not generate, modify, delete or delay any traffic which makes the attack completely stealth. Except the attacker own Tor node, no other Tor relay or server is compromised which makes the attack easily deployable in practice.

4 Tor Traffic Capture

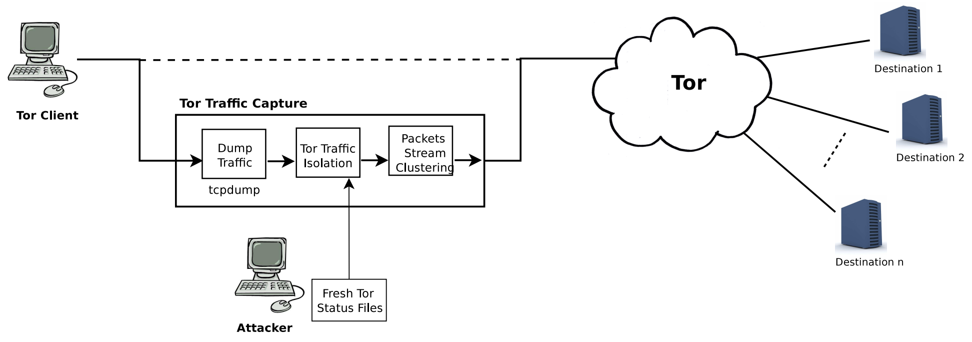


Fig. 1. Tor Traffic Capture

Traffic analysis typically starts by intercepting the traffic packets. In a typical scenario, only the victim and the gateway can capture the data flowing between them. In practice, however, several entities might have access to the traffic packets. The administrator of the LAN has access to the traffic of all endpoints in the network. The ISP (Internet Service Provider) can monitor the traffic of any of its subscribers. A Law Enforcement Agency, after approval from the ISP, can observe and record the traffic of any internet user which is referred to as lawful intercept. A censoring entity can observe the traffic of any user in its "juridiction". In addition to these entities, a malicious user in the LAN can carry out a MITM (Man-In-The-Middle) attack between the victim and the gateway and make all the traffic pass through her. The MITM attack can be easily performed using ARP spoofing/poisoning. Cain & Abel [24] and Ettercap [25] are two popular tools for ARP Spoofing/Poisoning.

Assuming that the attacker can intercept the Tor traffic packets using one of the above scenarios, the Tor traffic capture goes through three stages as shown in Figure 1. First the traffic is dumped in a file using a simple tool like tcpdump[6].

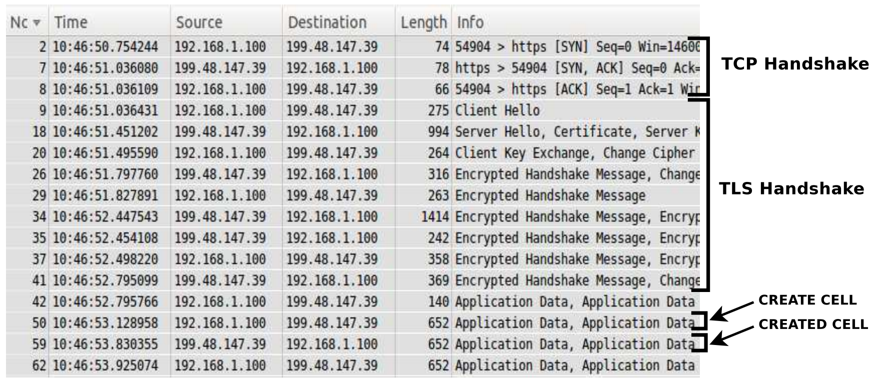
[6] Alternative tools like Wireshark/tshark or Omnipeek can be used as well.

Then the raw traffic is filtered to keep only Tor related traffic. Finally, the traffic is classified into streams.

Since the list of Tor relays is public, we use it to filter Tor traffic from the rest of the traffic. The list of Tor relays can be extracted from the Tor status files, in particular the cached-consensus file. These files can be downloaded manually from one of the authorities or they can be accessed directly in the Tor status local folder. Tor automatically updates those files once they are no more fresh. In our setting, the attacker, which is also a Tor client, uses her own Tor status files to extract the list of Tor relays.

The next step in the Tor traffic capture is to classify the packets into streams at the TCP protocol level[7]. Every stream is then tracked using IP addresses, ports, TCP flag bits, Sequence and Acknowledgment numbers. The stream is closed after a TCP connection termination (FIN, ACK-FIN and ACK).

Typically, Tor creates two types of streams: short living streams and long living streams. Short living streams are streams to download either router descriptors or directory consenus. They last around 3 minutes because the download takes a couple of seconds and then the stream stays idle until it hits the maximum stream idle period which is set to 3 minutes in Tor. Typically, only one circuit is established during a short livinig stream which is a single-hop circuit. The long living streams are for data communications and typically several 3-hop circuits are established during the lifetime of such streams. Besides, since all Tor communications are encrypted, all Tor streams initiate a TLS handshake just after the TCP three-way handshake.



Nc	Time	Source	Destination	Length	Info
2	10:46:50.754244	192.168.1.100	199.48.147.39	74	54904 > https [SYN] Seq=0 Win=1460
7	10:46:51.036080	199.48.147.39	192.168.1.100	78	https > 54904 [SYN, ACK] Seq=0 Ack=
8	10:46:51.036109	192.168.1.100	199.48.147.39	66	54904 > https [ACK] Seq=1 Ack=1 Wir
9	10:46:51.036431	192.168.1.100	199.48.147.39	275	Client Hello
18	10:46:51.451202	199.48.147.39	192.168.1.100	994	Server Hello, Certificate, Server K
20	10:46:51.495590	192.168.1.100	199.48.147.39	264	Client Key Exchange, Change Cipher
26	10:46:51.797760	199.48.147.39	192.168.1.100	316	Encrypted Handshake Message, Change
29	10:46:51.827891	192.168.1.100	199.48.147.39	263	Encrypted Handshake Message
34	10:46:52.447543	199.48.147.39	192.168.1.100	1414	Encrypted Handshake Message, Encryp
35	10:46:52.454108	199.48.147.39	192.168.1.100	242	Encrypted Handshake Message, Encryp
37	10:46:52.498220	192.168.1.100	199.48.147.39	358	Encrypted Handshake Message, Encryp
41	10:46:52.795099	199.48.147.39	192.168.1.100	369	Encrypted Handshake Message, Change
42	10:46:52.795766	192.168.1.100	199.48.147.39	140	Application Data, Application Data
50	10:46:53.128958	192.168.1.100	199.48.147.39	652	Application Data, Application Data
59	10:46:53.830355	199.48.147.39	192.168.1.100	652	Application Data, Application Data
62	10:46:53.925074	192.168.1.100	199.48.147.39	652	Application Data, Application Data

Fig. 2. Tor stream initial packets generated by Wireshark

[7] It is important to note that the classified streams at this stage are streams on top of which Tor communications are conveyed. These are different from the TCP streams tunneled through Tor. Hence, there are two levels of TCP streams: one level below the Tor protocol and one level on top of Tor protocol.

Figure 2 shows the initial packets in a Tor stream. The three first packets are for establishing the TCP connection then a TLS handshake follows. The illustrated stream is a short-living stream whose goal is to download a set of routers descriptors. Therefore, it creates a single hop circuit with the directory server which is performed using a CREATE and CREATED cells as shown in the Figure.

5 Data Collection

In order to evaluate the accuracy of the website fingerprinting for each web browser, 100 websites have been used. The set of 100 websites is composed of 90 randomly chosen websites from the top 1000 visited websites worldwide according to Alexa [21] and 10 censored websites in some countries of the Middle East. All censored websites are related to anonymizers and proxy services[8]. Website traffic traces are collected in 24 hours sessions. In each session, websites are fetched several times in a round-robin fashion. For visiting websites, we used two lab machines running Ubuntu Linux. Another Windows 7 machine is used to fetch websites through Internet Explorer. Traffic packets are dumped using tcpdump version 4.1.1 and libpcap version 1.1.1. Only packet headers are dumped in the dump file[9]. The experiments were performed using Tor version 0.2.2.39.

We wrote a python script to automate the fetching of websites. For each website, the script proceeds as follows: (1) it records the system time, (2) requests the website url, (3) waits for 50 seconds (the time to load the website[10], (4) stops the website connection, (5) records the system time, (6) waits for 10 seconds (to have a time gap between every two visits). Time snapshots are taken just before and after a website visit so that they can be intersected with the dump file in order to isolate the traffic for every website visit.

Once isolated, the taffic corresponding to each visit is represented as a sequence of packet sizes where a positive value refers to an inbound packet while a negative value refers to an outbound packet. This representation captures three features about the trace, namely, the size and direction of each packet and the order of packets[11]. The traffic representation is the same as recent works [14, 15]. For every visit, we keep only the first 500 packets of the traffic so that all obtained sequences have the same length[12].

Interestingly, parsing Tor traffic during a website visit shows that packets are flowing through several TCP streams not just one. One reason is that Tor needs to update the Tor relays status regularly by fetching fresh data from the directory servers and also to send dummy cells to keep some circuits open. TCP streams

[8] Examples of these websites include: torpoject.org, unblock-proxy.net, etc.

[9] tcpdump is launched with options -n -tttt.

[10] If a website takes more than 50 seconds to load, the sample sequence will be incomplete.

[11] As in most of related work, acknowledgement packets are ignored.

[12] This is a requirement for the classification algorithm.

used for fetching directory servers data can be easily distinguished from TCP streams used for data communication since the number of packets exchanged does not exceed 100 packets. Interestingly, ruling out these short TCP streams, the traffic resulting from visiting some websites is carried through two TCP streams with more than 200 packets each. This shows that Tor does not always multiplex the traffic of a website in a single stream. For those website visits, the corresponding traffic sequence is obtained by merging both streams into one. Similarly to normal visits, only the first 500 packets of the merged stream are kept.

In order to avoid the noise introduced by active content (Flash, etc.) and the browser cache, active content and caching are disabled. For instance, Chrome internet browser is used with the "incognito" mode while firefox is used with the "private" mode.

Four Web browsers have been used for fetching websites, namely, Chrome 18.0, Firefox 15.0.1, Internet Explorer 9.0 and Konqueror 4.8.5.

The same experiment is performed four times, each with a different browser. The experiment consists in 10 iterations. Every iteration consists in visiting all 100 websites once. Hence, every website is visited 10 times using the same web browser yielding a maximum of 10 samples for very website[13].

6 Classification Algorithm

The goal of this paper is to compare popular web browsers with respect to their capabilities to resist to web site fingerprinting. To this end, we use a classical classifier, namely, decision trees. It is important to mention that recent related work showed that better fingerprinting results can be obtained using other classifiers in particular Support Vector Machine (SVM) based. The next paragraphs give an overview of the decision tree classifier and the techniques used to evaluate the accuracy of the classifier for every web browser.

6.1 Decision Tree Classifier

Decision tree learning is a well known and classic classification technique [26]. It is very popular because it is self-explanatory, easy to understand and to use since it requires few parameter settings. It has been successfully used for classification in several diverse areas. Overall, it is well suited for exploratory knowledge discovery. In this paper we use a decision tree known as C4.5 [27] to classify website visits traffic sequences.

A decision tree can be learned typically using a top-down approach where each node corresponds to one of the input variables, each edge corresponds to possible values of each variable, and each leaf correspond to a class label. Every data set is split into subsets based on attributed values. This process is repeated

[13] Some website visits resulted in less than 500 packets. These sequences are discarded from the data set.

recursively and is called recursive patitioning. The recursion is completed when splitting adds nothing to the prediction. Inducing a decision tree using a top-down approach requires dealing with three other issues apart from selecting the best attribute to use at each node in the tree. Firstly, one has to choose a splitting threshold to form the two children for each node. Second, one needs a criterion to determine when to stop growing the tree. Thirdly, the final issue is how to decide what class label to assign for the terminal (leaf) node.

Classic strategies for splitting mainly focus on the use of impurity criteria, e.g., information gain, gain ratio and gini index. C4.5 decision tree uses gain ratio to select the best attribute and choose the optimal splitting threshold. In this approach, the attribute value that provides the best gain ratio is chosen as splitting threshold. To address the second issue discussed above, i.e., to avoid difficulties in choosing a stopping rule, most decision tree induction algorithms grow the tree to its maximum size where the terminal nodes are pure or almost pure, and then selectively prune the tree. The class label of each of the terminal nodes are typically decided based on the majority voting, i.e. the class label of data instances that are major in terms of counting compared to the other classes that contain in the respective terminal node.

6.2 Cross-Validation

To achieve a generalized performance of the decision tree used in this paper a cross-validation (CV) scheme is applied. CV is a well known method to test the performance of a classifier by varying training and test datasets [28]. CV is a standard test commonly used to test the ability of the classification system using various combinations of the testing and training data sets [29, 28, 30]. In this method, classification is measured by systematically excluding some data instances during the training process and testing the trained model using the excluded instances [31]. The process is repeated to cover all the dataset as testing dataset. In this paper, we have chosen 10-fold CV scheme where each time data in 1 fold are applied as test data and the rest 9 folds are used to train the model.

6.3 Performance Metric

Classification accuracy is one of the widely used performance metric to evaluate a classifier. Classification accuracy (ACC) is defined as the ratio of the number of all samples that are classified correctly over the total number of samples available (N).

$$ACC = (TP + TN)/N \tag{1}$$

where, TP (True Positives) = the total data instances from positive class that are classified as positive by the classifier; TN (True Negatives) = the total data instances from negative class that are classified as negative by the classifier.

7 Web Browser Resistance to Fingerprinting

Popular web browsers differ in several aspects, in particular, they use different web browser engines. The engine does most of the work of a web browser since it retrieves the document corresponding to a given URL and handles links, cookies, scripting, plug-ins loading, etc. The type of the web browser engine has an impact on the shape of the observed (encrypted) traffic. For example, some web browser engines may wait until all data is received before rendering a page while others may begin rendering before all data is received.

In order to compare the resistance of popular web browsers to website fingerprinting attacks, data is collected using different web browsers and then a C4.5 decision tree classifier is used to evaluate the accuracy of the website fingerprinting. More precisely, once the data about website visits is collected, we evaluated the accuracy of website fingerprinting in four scenarios:

- **Basic Packets Sequence**: The traffic trace is the first 500 packets of the TCP stream with the largest number of packets.
- **Merged Streams**: The traffic trace is the first 500 packets obtained by merging all TCP streams with more than 200 packets. If only one TCP stream has more than 200 packets, this scenario is the same as the first one.
- **Rounded Packet Sizes**: The same as the first scenario but the packet size values are rounded to multiples of 600. For instance, a packet size of 512 is rounded to 600 while a packet size of 743 is rounded to 1200.
- **Merged Streams and Rounded Packet Sizes**: This scenario is the combination of the two previous scenarios.

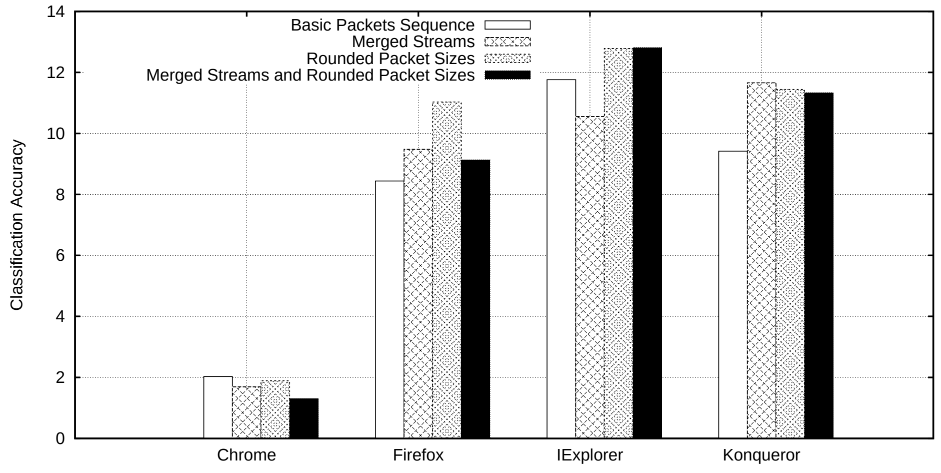


Fig. 3. Website Fingerprinting Accuracy for Common Web Browsers

Figure 3 shows the accuracy of the C4.5 classifier for each browser and for each scenario. Using Firefox, Internet Explorer and Konqueror, more than 9% of websites have been correctly identified by the classifier. With Chrome, however, only 2% of websites have been successfully identified. The histogram shows also that Rounding packet size values improves the efficiency of our classifier for all browsers. Merging TCP streams, on the other hand, improved the efficiency of the classifier only for Firefox and Konqueror. Merging streams resulted in lower accuracy than the basic scenario for Chrome and Internet Explorer. The most important result illustrated by the histogram is that Chrome browser offers a better resistance to website fingerprinting than the other studied browsers. The advantage Chrome browser has on the other studied browsers is expected to be preserved in case a more efficient classifier (e.g. [15]) is used.

8 Conclusion

Website fingerprinting is a new attack on Tor anonymity system that tries to reveal the identities of visited websites by recognizing patterns in the Tor traffic. Compared to previous attacks on Tor, in particular confirmation attacks, website fingerprinting does not require an attacker with extended capabilities. Only the ability to sniff the Tor client encryted traffic is required. In this paper we presented a detailed survey on website fingerprinting techniques which recently reached high accuracy rates (80%) [15] on Tor anonymity system. The main contribution of this paper, however, is an empirical analysis of how much resistance popular web browsers provide against website fingerprinting. Four notable web browsers have been considered, namely, Firefox, Chrome, Internet Explorer, and Konqueror. The analysis showed that the resistance of Chrome to website fingerprinting is five times better than the remaining web browsers. Since most of existing fingerprinting techniques have been evaluated using Firefox web browser [12, 14, 15], we expect the accuracy results to be reduced in case Chrome browser is used.

There are two main mitigation approaches for the website fingerprinting attack. The first and the most commonly used approach is padding where the sender appends some random (dummy) bits to the actual data to obtain, for instance, fixed size packets. It has been shown that padding reduces the accuracy of fingerprinting techniques only slightly [14, 15]. The second mitigation approach is traffic camouflage which can be implemented in two ways. The first variant is to obfuscate the actual traffic by loading several pages simultaneously. Panchenko et al. [14] load a randomly chosen page whenever an actual website is to be visited. The second variant is to disguise the actual traffic within a typical encrypted cover protocol such as Skype voice over IP. The StegoTorus proxy [32] constructs a database of pre-recorded packet traces from real sessions of a given cover protocol. Then, when a Tor client visits a website, it choses randomly a pre-recorded trace from the database which is used as a template to reshape the actual traffic. The packet sizes of the pre-recorded trace are matched exactly and the packet timings are matched “to the nearest millisecond”.

References

1. Dingledine, R., Mathewson, N., Syverson, P.: Tor : the second-generation onion router. In: Proceedings of the 13th Usenix Security Symposium. (August 2004)
2. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Proceedings of Designing Privacy Enhancing Technologies, Springer-Verlag, LNCS 2009 (July 2000) 115–129
3. Murdoch, S., Danezis, G.: Low-cost traffic analysis of Tor. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, IEEE CS (May 2005)
4. Murdoch, S.: Hot or not: Revealing hidden services by their clock skew. In: Proceedings of CCS 2006. (October 2006)
5. Hopper, N., Vasserman, E., Chan-Tin, E.: How much anonymity does network latency leak? ACM Transactions on Information and System Security **13**(2) (February 2010)
6. Evans, N., Dingledine, R., Grothoff, C.: A practical congestion attack on tor using long paths. In: Proceedings of the 18th USENIX Security Symposium. (August 2009)
7. Mittal, P., Khurshid, A., Juen, J., Caesar, M., Borisov, N.: Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In: Proceedings of the 18th ACM conference on Computer and communications security. CCS '11, New York, NY, USA, ACM (2011) 215–226
8. Hintz, A.: Fingerprinting websites using traffic analysis. In: Privacy Enhancing Technologies (PETS),LNCS. Volume 2482., Springer (2002) 171–178
9. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical identification of encrypted web browsing traffic. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. SP '02, Washington, DC, USA, IEEE Computer Society (2002) 19–
10. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy vulnerabilities in encrypted http streams. In: Proceedings of the 5th international conference on Privacy Enhancing Technologies. PET'05, Berlin, Heidelberg, Springer-Verlag (2006) 1–11
11. Liberatore, M., Levine, B.N.: Inferring the source of encrypted http connections. In: Proceedings of the 13th ACM conference on Computer and communications security. CCS '06, New York, NY, USA, ACM (2006) 255–263
12. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial na&#239;ve-bayes classifier. In: Proceedings of the 2009 ACM workshop on Cloud computing security. CCSW '09, New York, NY, USA, ACM (2009) 31–42
13. Shi, Y., Matsuura, K.: Fingerprinting attack on the tor anonymity system. In Qing, S., Mitchell, C., Wang, G., eds.: Information and Communications Security. Volume 5927 of Lecture Notes in Computer Science., Springer Berlin Heidelberg (2009) 425–438
14. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th annual ACM workshop on Privacy in the electronic society. WPES '11, New York, NY, USA, ACM (2011) 103–114
15. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: website fingerprinting attacks and defenses. In: Proceedings of the 2012 ACM conference on Computer and communications security. CCS '12, New York, NY, USA, ACM (2012) 605–616

16. Safe Web: Safeweb proxy. "`http://www.safeweb.com`"
17. Rijsbergen, C.J.V.: Information Retrieval. 2nd edn. Butterworth-Heinemann, Newton, MA, USA (1979)
18. Bracewell, R.: Pentagram Notation for Cross Correlation. The Fourier Transform and Its Application. McGraw-Hill, New York, USA (1965)
19. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Second edn. Morgan Kaufmann (June 2005)
20. Weka Tool: Weka: Data mining software in java. "`www.cs.waikato.ac.nz/ml/weka`"
21. Alexa Website: Alexa: The web information company. "`www.alexa.com`"
22. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklay (1966)
23. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys **33** (1999) 2001
24. Massimiliano Montoro: Cain & abel. "`http://www.oxid.it/cain.htm`"
25. ALor and NaGA: Ettercap. "`http://ettercap.sourceforge.net`"
26. Kotsiantis, S.B.: Supervised machine learning: A review of classification techniques. Informatica **31** (2007) 249–268
27. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
28. Barton, J., Lees, A.: An application of neural networks for distinguishing gait patterns on the basis of hip-knee joint angle diagrams. Gait & Posture **5**(1) (1997) 28 – 33
29. Ding, C.H.Q., Dubchak, I.: Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics **17** (2001) 349–358
30. Hassan, M.R., Begg, R., Taylor, S., Kumar, D.K.: Hmm-fuzzy model for recognition of gait changes due to trip-related falls. In: Proceeding of the IEEE Eng Med Biol Soc., Springer Berlin Heidelberg (2006) 1216–1219
31. Begg, R.K., Palaniswami, M., Owen, B.: Support vector machines for automated gait classification. IEEE Transactions on Biomedical Engineering **52** (2005) 828–838
32. Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Cheung, S., Wang, F., Boneh, D.: Stegotorus: a camouflage proxy for the tor anonymity system. In: Proceedings of the 2012 ACM conference on Computer and communications security, New York, NY, USA, ACM (2012) 109–120