Client Diversity Factor in HTTPS Webpage Fingerprinting*

Hasan Faik Alan
Department of Computer Science
UNC - Chapel Hill, NC, USA
alan@cs.unc.edu

Jasleen Kaur
Department of Computer Science
UNC - Chapel Hill, NC, USA
jasleen@cs.unc.edu

ABSTRACT

Webpage fingerprinting methods infer the webpages visited in a traffic trace and are serious threats to the privacy of web users. Prior work evaluates webpage fingerprinting methods using traffic samples from a single client and does not consider the client diversity factor—webpages can be visited using different browsers, operating systems and devices. In this paper, we study the impact of client diversity on HTTPS webpage fingerprinting. First, we evaluate 5 prominent fingerprinting methods using traffic samples from 19 different clients. We show that the best performing methods overfit to the traffic patterns of a single client and do not generalize when they are evaluated using the samples from a different client (even if the clients use the same browser and operating system and only differ in device). Then, we investigate the traffic patterns of the clients and find differences in the HTTP messages generated, servers communicated and implementation of HTTP/2 across the clients. Finally, we show that the robustness of the methods can be increased by training them using the samples from a diverse set of clients. This study informs the community towards a realistic threat model for HTTPS webpage fingerprinting and presents an analysis of modern HTTPS traffic.

ACM Reference format:
Hasan Faik Alan and Jasleen Kaur. 2019. Client Diversity Factor in HTTPS Webpage Fingerprinting. In *Proceedings of Ninth ACM Conference on Data and Application Security and Privacy, Richardson, TX, USA, March 25–27, 2019 (CODASPY '19),* 12 pages.
https://doi.org/10.1145/3292006.3300045

1 INTRODUCTION

Traffic analysis, which infers information from the observation of traffic flows [1], is a fairly diverse field—both in terms of the *granularity* of information inferred (such as protocols, application types, user interests, websites, and webpages) as well as in terms of the *privacy-enhancing technology* used for transmitting the observed traffic (such as HTTPS, SSH, VPN, and Tor) [2–9]. The keywords *webpages* and *HTTPS* help set the specific context for this paper—we focus on the problem of HTTPS traffic analysis for the purpose of fingerprinting the webpages being visited, with an emphasis on the *diversity of client platforms* (different browsers, operating systems and devices that can be used to visit webpages).

While there have been numerous studies on fingerprinting web traffic [4–12], our focus differs from most in three key aspects. First, a majority of work in this area studies web traffic sent over tunnels using SSH, VPN or Tor—there is surprisingly scarce work on HTTPS traffic, which is the most commonly-used privacy setting. This is perhaps due to an implicit belief that fingerprinting analysis conducted for more private VPN or Tor traffic should also translate to less private HTTPS traffic.[1] Second, most prior work is focused on fingerprinting *websites* (and not individual webpages within a website), and considers just the landing pages of different websites. In HTTPS traffic, the website domain is often retrievable from the Server Name Indication (SNI) extension of TLS [4, 13]—fingerprinting webpages within a given website, however, is challenging due to similarity of webpages within a website [5]. Third, and most relevantly to the motivation of this study, evaluations in prior HTTPS webpage fingerprinting studies were performed using webpage traffic samples from the same client platform [4, 5]. Specifically, Miller et al. collected traffic traces of webpage visits using Firefox 22 browser in a virtual machine running Linux 12.04 OS [5]. Similarly, Gonzalez et al. used Firefox browser on a PC [4]. This observation leads us to question the robustness of such fingerprinting methods in the real world, given the diversity of client platforms as well as the influence of these platforms on webpage content and traffic [14, 15].

In this paper, our main objective is to examine how client diversity impacts HTTPS webpage fingerprinting. Our first major innovation is that we evaluate 5 prominent webpage fingerprinting methods from the traffic analysis literature using webpage traffic samples collected from 19 different clients—we consider 6 different browsers (Chrome, Firefox, Edge, IE, Opera and Safari), 5 different operating systems (Android, Ubuntu, Windows 10, Windows 7 and macOS), and 6 different devices (see Table 2). We show that all 5 webpage fingerprinting methods perform the best when the samples from the same client are used for training and test—this is the scenario studied in prior work. However, the performance of the methods decreases dramatically when they are tested with the samples from a client that is different than the one used for training—the accuracy of the best performing method decreases from 94% to 55% (when the clients use the same browser but different operating systems) and to 27% (when the clients use different operating systems and browsers). Even when the training and test clients use the same browser and operating system and only differ in device, the best performing method achieves only 57% accuracy.

*This material is based upon work supported by the National Science Foundation under Grant No. CNS-1526268.

CODASPY '19, March 25–27, 2019, Richardson, TX, USA

ACM ISBN 978-1-4503-6099-9/19/03…$15.00
https://doi.org/10.1145/3292006.3300045

[1] In this paper, we show that this is not true—features and classifiers that work well for Tor traffic do not work well in the HTTPS context.

This finding implies that evaluation of the fingerprinting methods using the samples from the *same* client may overestimate the success of a webpage fingerprinting adversary in the real world.

Next, we investigate the traffic patterns of the 19 clients. We find differences in the HTTP messages generated, HTTP/2 implementation configuration, and the servers communicated with across the 19 clients. Particularly, we find that the User-Agent string lengths of the clients differ significantly, which causes outgoing packet sizes to vary greatly across the clients. We hypothesize that this is a significant factor to help explain our findings above—when samples from only one client are used for training, fingerprinting methods may over-fit to features derived from the outgoing packet sizes. This leads to very high fingerprinting accuracy when the same client is used for testing, but very low accuracy when different clients are used.

Finally, we search for a method that is robust to the impact of client diversity. First, we observe a significant increase in the accuracy of the methods when they are trained using samples from 18 different clients and tested using samples from the remaining one client. Then, we search for the main source of improvement in the accuracies. We find that it is necessary to use traffic samples from the browser of the test client during training; however, using samples of the same browser from different client devices is even more effective in increasing the robustness of fingerprinting methods.

In the rest of this paper, we summarize problem formulation in Section 2, data collection in Section 3, and our evaluations in Section 4. We investigate differences across clients in Section 5 and search for a robust model in Section 6. We summarize related work in Section 7 and our conclusions in Section 8.

2 PROBLEM FORMULATION

2.1 Importance of HTTPS Traffic Analysis

HyperText Transfer Protocol (HTTP) is used between a web browser (e.g., Google Chrome) and a web server when a webpage is visited [16]. HTTP messages are not encrypted and their integrity is not ensured. Thus, HTTP traffic is vulnerable to eavesdropping and tampering. Given the security concerns with HTTP, browsers (e.g., Chrome and Firefox) and organizations (e.g., Let's Encrypt[2]) promote HTTPS (HTTP over TLS) which verifies the identity of a website, encrypts HTTP messages and ensures data integrity. Indeed, these efforts led to the rapid adoption of HTTPS [17].

Despite the encryption of HTTP messages, HTTPS does not provide ultimate privacy when a webpage is visited. Even though the webpage URL (e.g., https://www.plannedparenthood.org/learn/abortion) is carried in an encrypted HTTP message, identity of the visited website (e.g., www.plannedparenthood.org) is often revealed (through the IP addresses of the servers, DNS queries and/or the server name in the SNI extension of TLS) [4]. More alarmingly, Miller et al. (2014) showed that the URLs of the webpages visited *within* a website can also be predicted with high accuracy using machine learning and features based on network packet sizes [5]. This finding is alarming since more fine-grained confidential information about users (e.g., health conditions and financial status) can be learned by determining the webpages visited compared to just determining the websites visited. For example, determining that a user is reading about abortion or reading about filing a bankruptcy gives more granular information than just determining that the user is browsing a health or a finance website. Such information can be used for mass surveillance, targeted advertising or disclosure of sensitive information in a targeted attack scenario that may result in severe consequences, such as embarrassment or financial loss of a web user. Thus, the extent of HTTPS traffic analysis should be studied, HTTPS protocol should be improved towards a more robust privacy enhancing technology (if necessary[3]), and the web users should be informed accordingly.

2.2 HTTPS Webpage Fingerprinting (Threat Model)

In this paper, we consider a scenario in which a user visits a webpage within an HTTPS website and an adversary eavesdropping on the HTTPS traffic of the user tries to predict the URL of the visited webpage. We assume that the user do not use any privacy enhancing technology, such as a DNS proxy, VPN or Tor. Furthermore, we assume an HTTPS only web in which the websites enforce TLS encrypted connections and the adversary cannot decrypt the payloads.[4] However, the adversary has access to the information in the TCP/IP and TLS headers, which are transmitted in cleartext, such as the IP addresses, port numbers and server names, as well as the side channel information, such as the packet size and timing. The adversary can use such information and build a statistical model to fingerprint webpages based on their traffic patterns (e.g., number of packets sent to a specific server IP address). Furthermore, server IP addresses, DNS queries and/or the server name in the SNI extension of TLS often reveal the visited website—possibly allowing the adversary to narrow down the visited webpages to those within certain websites. Entities that can employ webpage fingerprinting methods include Internet Service Providers, Network Administrators or anyone who can eavesdrop on the HTTPS traffic of a user (e.g., an adversary who sniffs the network traffic of a public WiFi connection).

2.3 HTTPS Webpage Fingerprinting as a Machine Learning Problem

We study HTTPS webpage fingerprinting problem using the same supervised machine learning setting as in prior work [4, 5]. Specifically, the machine learning setting consists of two main phases namely the data collection and evaluation. In the data collection phase, webpage URLs are visited in a browser using a browser automation script and the network traffic of each visit is captured using a tool such as tcpdump. In the evaluation phase, the dataset is split into training and test samples. A supervised machine learning method, such as Multinomial Logistic Regression or Support Vector Machines (SVMs), is trained using the features extracted from the traffic traces of training samples. For example, a commonly used

[2]https://letsencrypt.org/

[3]While prior work achieved high accuracy in HTTPS webpage fingerprinting [4, 5], it is not clear whether such accuracies can be achieved in the real world when several factors, which are often not studied in prior work, are considered. In this work, we mainly focus on one such factor namely client diversity and discuss other factors throughout the paper.

[4]If HTTPS is not used or it is compromised, the adversary can simply inspect the packet payloads and identify the webpages visited (and much more, such as online banking login credentials) from the clear text in the payloads—a technique known as deep packet inspection.

feature extraction process is to consider each unique packet size as a feature and to count how many times each packet size occurs in a traffic trace [6].[5] During training, the label of each sample (i.e., webpage URL) is also provided and the machine learning method is expected to learn a function that maps input features to the provided labels. During test, the method predicts the labels of the test samples and is evaluated using its accuracy—how many samples out of all the test samples it labels correctly.

The webpage fingerprinting methods in the traffic analysis literature, such as Liberatore and Levine [6], BoG [5], CUMUL [8], K-Fingerprinting [18], and Wfin [9] mainly differ in the features they extract from network traffic and the machine learning methods they use. We give the details of such methods in Section 4.1 before we evaluate them.

2.4 State of the Art

To the best of our knowledge, there is only limited prominent work that studies webpage fingerprinting using HTTPS traffic [4, 5]. Miller et al. study traffic traces of around 600 webpages selected using a random walk from each of 10 prominent websites [5]. They design and evaluate a fairly elaborate fingerprinting method (termed as Bag of Gaussians) as well as a Hidden Markov Model (HMM) of likely browsing sequences. The researchers achieve 76% - 96% fingerprinting accuracy across the 10 websites—compared to around 60% when they use the methods from previous studies on SSH [6] and Tor traffic analysis [19, 20].

In a somewhat related work, Gonzalez et al. show that knowing the hostname of a visited website, which is already leaked in HTTPS traffic, is enough for user profiling purposes if the content of the website is homogeneous as in the case of the websites in the games and sports categories [4]. If the content of a website is heterogeneous, the researchers use the CUMUL method [8], which was originally proposed for Tor traffic analysis, to classify traffic traces of first-level webpages[6] within that website. The researchers achieve 13% - 97% classification accuracy across the websites.

The most signification distinction of our work from the above is that prior work has evaluated HTTPS fingerprinting methods using webpage traffic samples collected from the same client platform (e.g., Firefox browser in a virtual machine running Linux 12.04 OS in Miller et al.[5]). In this paper, we investigate the robustness of such methods to client diversity—how do the webpage fingerprinting methods perform when they are evaluated with traffic samples from different browsers, operating systems, and devices?

3 DATA

We study the same 10 websites targeted by Miller et al. [5]. We crawled these websites using a breadth-first crawling algorithm. Table 1 summarizes the results of the crawls.

We randomly select 50 webpages from each of 7 out of 10 websites—3 websites did not yield consistent webpages across all clients.[7] We then visit each of these webpages 28 times using each of 19 different clients (i.e., a total of 50x7x28x19 = 186,200 webpage visits) and capture the network traffic of each visit—this dataset was collected during the period 5 - 14 July 2018.

Table 1: Websites studied.

Host	Finished Crawling	URLs from Crawl
www.aclu.org	No	25182
www.bankofamerica.com	Yes	861
healthy.kaiserpermanente.org	No	19173
www.legalzoom.com	Yes	5260
www.mayoclinic.org	No	13266
www.netflix.com	No	127487
www.plannedparenthood.org	Yes	23260
investor.vanguard.com	Yes	477
www.wellsfargo.com	Yes	5436
www.youtube.com	No	29534

Client Platforms Browser, OS and device type of each client are given in Table 2. We used four different Android devices, one Mac mini and three different virtual machines (with Ubuntu 18.04 LTS, Windows 10 and Windows 7 operating systems). In each operating system except Android OS, we considered multiple browsers. For example, in Windows 10, we used 5 different browsers namely Chrome, Firefox, Edge, IE and Opera. Note that some browsers are not available in all operating systems—Safari and Edge are only available in macOS and Windows 10, respectively. Furthermore, we excluded Opera in Windows 7 as it crashed frequently during data collection. In Android devices, we used only Chrome as, to the best of our knowledge, only Chrome has a driver to automate webpage visits in Android OS.[8]

Traffic Capture We used the Selenium browser automation framework[9] and tcpdump[10] to capture the network traffic of webpage visits. For each webpage visit, a new instance of a web browser (i.e., a browser without any user data) was used. Webpage URLs from all of the websites were visited in a round-robin manner 28 times in each of the 19 clients.

4 IMPACT OF CLIENT DIVERSITY

In this section, we evaluate the impact of client diversity on the accuracy of prominent webpage fingerprinting methods from the traffic analysis literature. First, we summarize the webpage fingerprinting methods we evaluate. Then, we describe our evaluation methodology. Finally, we discuss the results.

4.1 Fingerprinting Methods

Liberatore and Levine (LL) (2006) [6] uses only the packet size counts as features and Naive Bayes classifier for classification. LL method was introduced to fingerprint the visits to the landing pages

[5]Incoming and outgoing packets are considered separately. For example, with a maximum packet size of 1500 bytes, a sample is represented as a vector of 3000 elements—each element corresponds to a packet size with a direction and the value of each element is how many times a packet with that size and direction occurs in the traffic trace of the sample.

[6]Webpages that are linked from the landing page of a website.

[7]Netflix directed all webpage URLS to the same sign-in page; Youtube and Kaiser Permanente webpages were redirected to URLs from hostnames that we never visit in other clients—Kaiser Permanente webpages often reported an HTTP error in the Android clients. To achieve a balanced dataset for supervised machine learning, we wanted to select an equal number of samples for each webpage from each client. Thus, we excluded these three websites and found that we have at least 28 samples from 50 webpages in each of the remaining 7 websites for each of the 19 clients.

[8]While any app can be automated using Android adb utility, we are not aware of a method to determine whether a webpage is loaded in other browser apps (e.g. Firefox)—the driver of Chrome provides such events.

[9]https://www.seleniumhq.org/

[10]http://www.tcpdump.org/

Table 2: Browser, OS and device type of the clients used for webpage visits. User-Agent strings of the clients are given in Table 3.

ID	Browser	OS	Device
1	Chrome (67.0.3396.87)	Android 4.4.2	SM-T230NU
2	Chrome (67.0.3396.87)	Android 4.4.4	GT-I9195I
3	Chrome (67.0.3396.87)	Android 6.0.1	Nexus 5
4	Chrome (67.0.3396.87)	Android 6.0.1	Nexus 7
5	Chrome (67.0.3396.99)	Ubuntu 18.04 LTS	vm
6	Chrome (67.0.3396.99)	Windows 10	vm
7	Chrome (67.0.3396.99)	Windows 7	vm
8	Chrome (67.0.3396.99)	macOS 10.13.5	Mac mini
9	Edge (42.17134.1.0)	Windows 10	vm
10	Firefox (61.0)	Ubuntu 18.04 LTS	vm
11	Firefox (61.0)	Windows 10	vm
12	Firefox (61.0)	Windows 7	vm
13	Firefox (61.0)	macOS 10.13.5	Mac mini
14	IE (11)	Windows 10	vm
15	IE (11)	Windows 7	vm
16	Opera (67.0.3396.87)	Ubuntu 18.04 LTS	vm
17	Opera (67.0.3396.87)	Windows 10	vm
18	Opera (67.0.3396.87)	macOS 10.13.5	Mac mini
19	Safari (13605.2.8)	macOS 10.13.5	Mac mini

of websites in an SSH proxy channel. Miller et al. [5] considered LL as a baseline method that uses low level packet inspection and evaluated it in the context of HTTPS webpage fingerprinting.

Bag-of-Gaussians (BoG) (2014) [5] uses features based on clustering pairs of incoming and outgoing burst sizes according to the second level domain names of the servers as well as features based on packet size counts.[11] BoG uses logistic regression with L2 regularization for classification. Miller et al.[5] showed that BoG achieves substantially greater accuracy in HTTPS webpage fingerprinting compared to the methods introduced by Liberatore and Levine [6], Panchenko et al. [19] and Wang et al. [20]. The researchers also showed that a Hidden Markov Model can be used to model a sequence of webpage visits within a website, that can augment fingerprinting methods quite successfully.

CUMUL (2016) [8] uses 100 points sampled from a cumulative representation of packet sizes as well as the number of incoming/outgoing packets and the sum of incoming/outgoing packet sizes as features. CUMUL uses SVM with RBF kernel for classification. Panchenko et al. [8] introduced CUMUL for fingerprinting webpages visited in Tor network traffic. Gonzalez et al. [4] used CUMUL for HTTPS webpage fingerprinting.

K-fingerprinting (KFP) (2016) [18] uses 175 traffic features, such as the statistics based on the number of packets and packet timings. KFP uses Random Forest Classifier for classification. The authors of the method evaluated KFP in fingerprinting hidden services in Tor network traffic as well as in fingerprinting encrypted Web traffic.

[11]Burst size is defined as the total bytes in contiguous packets transmitted in one direction.

Wfin (2018) [9] Yan and Kaur identified 40 most important traffic feature categories in web traffic analysis, such as unique packet size, packet size count, and preposition of first 300 incoming packets, and introduced the Wfin method. Wfin uses Extra-Trees classifier for classification. The researchers showed that features used in Wfin yield similar performance as features used in the LL method but perform better than features from the CUMUL and KFP methods in classifying traffic traces of landing pages of 2,000 websites.

Packet Size Counts (PS), Incoming Packet Size Counts (IPS), and Outgoing Packet Size Counts (OPS) Most methods in the traffic analysis literature include packet size counts in their feature set.[12] To have a baseline view of the webpage fingerprinting accuracy achievable using only packet size counts, we evaluate three methods that just use packet size counts and differ in the direction of the packets used: both incoming and outgoing packet size counts (PS), only incoming packet size counts (IPS), and only outgoing packet size counts (OPS). We use Random Forest Classifier for classification with these methods. Note that PS differs from LL only in the choice of classifier.

4.2 Evaluation Methodology

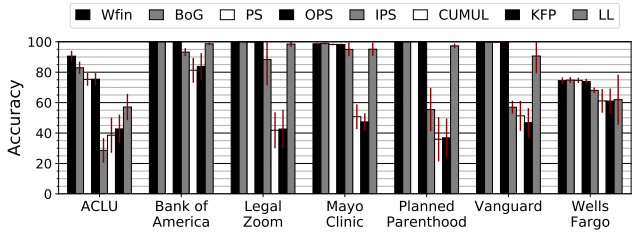
We evaluate the webpage fingerprinting methods detailed in Section 4.1, using our dataset described in Section 3. We consider five different scenarios—the training and test samples are from: (i) the same client (Scenario 1), (ii) the same browser, same OS but different device (Scenario 2)[13], (iii) the same browser but different OS (Scenario 3), (iv) the same OS but different browser (Scenario 4), and (v) different browser and different OS (Scenario 5). Note that we have 19 clients, 7 websites, 50 webpages from each website, and 28 samples from each webpage in our dataset. We perform a total of 20,216 evaluations (19 x 19 train/test client pairs x 8 methods x 7 websites). In this setting, an evaluation is a classification problem with 50 classes—given a traffic trace classify it as a trace of one of the 50 webpages within a website. When a client is used for training, we use the first 21 samples of each webpage from that client for training and use the remaining 7 samples when the client is considered for test—a total of 1050 training samples (i.e., 50 webpages x 21 samples) and 350 test samples are used in each evaluation.

4.3 Results

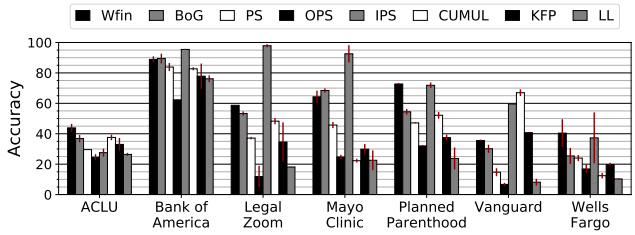
For each of the five different scenarios, Figure 1 plots the accuracy of each method (i.e., percent of the test traffic trace samples that are labeled with the correct webpage URL) for each website averaged over all training and test pairs of client platforms considered in that scenario. Figure 2 plots the accuracy in each scenario, when averaged across all websites and client pairs (and makes it easier to compare the overall performance of the methods across different scenarios). We observe that:

(1) All webpage fingerprinting methods perform their best in Scenario 1 when the training and test samples are from the same client. Recall that this is the scenario in which evaluations in all prior work are conducted. In this scenario,

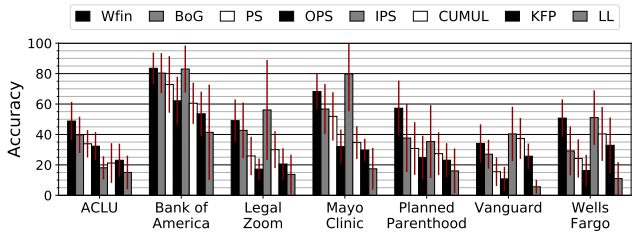
[12]We formulate HTTPS webpage fingerprinting as a machine learning problem in Section 2.3 and describe an example method that uses packet size counts as features.
[13]Note that we have only two clients namely client 3 and 4 that have the same browser and same OS but different devices.



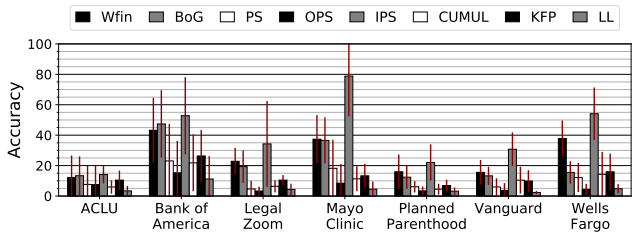
(a) Scenario 1 (Same Client)



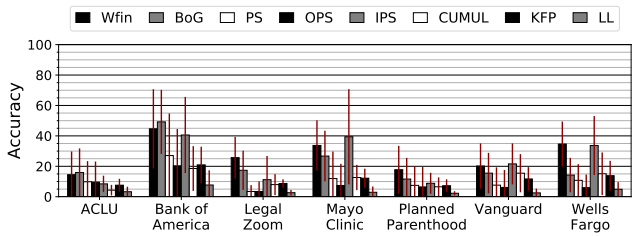
(b) Scenario 2 (Same Browser, Same OS, Different Device)



(c) Scenario 3 (Same Browser, Different OS)

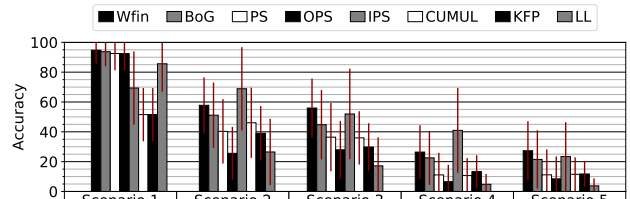


(d) Scenario 4 (Different Browser, Same OS)



(e) Scenario 5 (Different Browser, Different OS)

Figure 1: Performance of webpage fingerprinting methods in five different scenarios in which the training and test clients are varied.



	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Wfin	94	57	55	26	27
BoG	93	51	44	22	21
PS	92	40	36	11	11
OPS	92	25	27	6	8
IPS	69	68	52	41	23
CUMUL	51	46	35	10	11
KFP	51	39	29	13	11
LL	85	26	17	4	3

Figure 2: Accuracy of each method in each scenario.

the best performing methods are able to classify the traffic traces of webpages within most of the websites with high accuracy. However, as reported in prior work, the traffic traces of webpages within some websites can be classified with lower accuracy than others [4, 5]—e.g., lower accuracies are observed in ACLU and Wells Fargo websites in Figure 1a.

(2) Compared to Scenario 1, the accuracies decrease significantly in Scenario 2 when the training and test samples are from the same browser, same OS but different device (Figure 1b). The accuracies of the Wfin and BoG methods, averaged cross all websites, drop from 94% and 93% to 57% and 51%, respectively (Figure 2).
Even lower accuracies are observed in Scenario 3, when samples from the same browser but different OS are used for evaluation (Figure 1c and 2).

(3) The lowest accuracies are observed in Scenarios 4 and 5 (when samples from different browsers are used)—see Figures 1d and 1e. Compared to Scenario 1, the accuracies of the Wfin and BoG methods in Scenario 4, averaged across all websites, decrease from 94% and 93% to around 26% and 22%, respectively (Figure 2).
In a real world setting, Scenarios 2, 3, 4 or 5 are much more likely to occur than Scenario 1, if an adversary does not consider the impact of client diversity (and trains a webpage fingerprinting method using traffic samples from only a single client). Thus, *prior evaluations of fingerprinting methods using samples from the same client may significantly overestimate the success of a webpage fingerprinting adversary.*

(4) While the OPS method, which uses only the outgoing packet size counts as features, performs comparable to the best performing methods in Scenario 1, it is outperformed in other scenarios. On the other hand, the IPS method, which uses only the incoming packet size counts, is one of the best performing methods in Scenarios 2, 3, 4 and 5.

Note that features based on outgoing packet size are used in nearly all of the fingerprinting methods—indeed, IPS is the only method included in our evaluations that does not use any feature based on outgoing packet size. Our results suggest that the presence of features based on outgoing packet size may lead to "over-fitting" when evaluations consider only a single client platform.

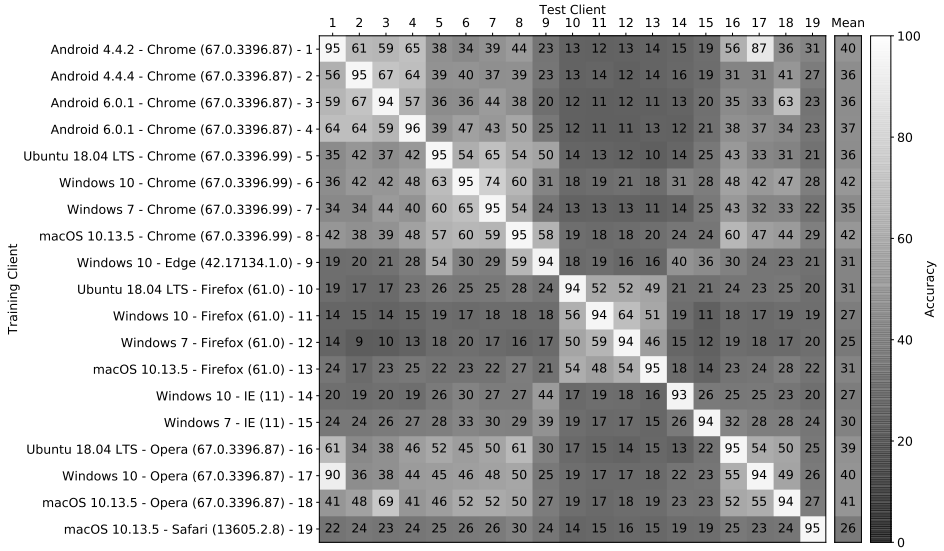


Figure 3: Accuracy of the Wfin method when the training and test clients are varied (19x19 training and test client pairs). Note that the highest accuracies are achieved when the samples from the same client are used for training and test.

In Figure 3, we plot the webpage fingerprinting accuracy (averaged across the 7 websites) of the Wfin method as a matrix for all 19x19 pairs of training and test client platforms. We find that Wfin performs better on average, when a Chrome client is used for training and Opera client is used for test (and vice versa) (e.g., using one of the clients 1-8 for training and 16-18 for test) compared to using any other client pair with different browsers. Note that the Chrome and Opera browsers are both based on the open source Chromium browser project[14]—we hypothesize that they generate similar network traffic patterns. We also find that Wfin achieves a significantly high accuracy when client 17 (Windows 10 - Opera) is used for training and client 1 (Android-Chrome) is used for test (and vice versa); and Wfin achieves the lowest accuracies when client 14 (Windows 10 - IE), 15 (Windows 7 - IE) or 19 (macOS - Safari) is used in an evaluation.

Figure 3 reports accuracy averaged across the 7 websites. We select the row for training client 8 (that gives the highest average accuracy across all test clients in Figure 3), and plot in Figure 4 the *per-website* accuracy of the Wfin method when only samples from client 8 are used for training. As before, we observe that the highest accuracies are achieved when the training and the test clients are the same. Significantly lower accuracies are observed when the browsers of the training and test clients are different (e.g., when clients 10-15 are used for test). We further investigate plausible causes of client differences in the next section.

5 CLIENT DIFFERENCES

Using data collected in 2014, prior work has shown that when different client platforms are used to access the *landing* page of popular websites, the resulting download may differ in both content as well as traffic [14, 21]. Our results so far suggest that this may be

[14]https://www.chromium.org/

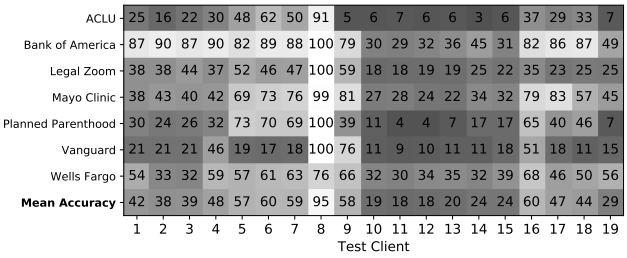


Figure 4: Accuracy of the Wfin method for each test client and website when client 8 is used for training.

true even for modern download traffic generated when webpages within websites are visited. In this section, we identify three major differences across client platforms that cause variations in web traffic patterns—these are mostly factors that influence the outgoing packet sizes in web traffic (which is likely to be a significant influence on our results in Section 4.3) .

5.1 HTTP Messages

Background When a webpage URL is entered to the address bar of a browser, the browser sends an HTTP request message to a server. The server interprets the request and returns an HTTP response. The response message is parsed by the browser and additional requests are sent and responses are received if other web resources, such as CSS, image and video files, are need to be loaded. An example HTTP message that requests the landing page of Bank of America website is given in Figure 5.

Methodology Browser developer tools provide detailed information about each HTTP message generated during a webpage visit.

Table 3: User-Agent strings of the clients.

ID	User Agent	Length
1	Mozilla/5.0 (Linux; Android 4.4.2; SM-T230NU Build/KOT49H) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36	131
2	Mozilla/5.0 (Linux; Android 4.4.4; GT-I9195I Build/KTU84P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Mobile Safari/537.36	138
3	Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5 Build/M4B30Z) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Mobile Safari/537.36	136
4	Mozilla/5.0 (Linux; Android 6.0.1; Nexus 7 Build/MOB30X) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36	129
5	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36	104
6	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36	114
7	Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36	113
8	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36	120
9	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134	129
10	Mozilla/5.0 (X11; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0	68
11	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0	78
12	Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0	77
13	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:61.0) Gecko/20100101 Firefox/61.0	82
14	Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; rv:11.0) like Gecko	89
15	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; rv:11.0) like Gecko	176
16	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36 OPR/54.0.2952.41	121
17	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36 OPR/54.0.2952.41	131
18	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36 OPR/54.0.2952.41	137
19	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1.1 Safari/605.1.15	119

```
GET / HTTP/1.1
Host: www.bankofamerica.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
    x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/67.0.3396.99 Safari/537.36
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,image/webp,
    image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 5: An HTTP request for www.bankofamerica.com, generated by the Chrome browser running on Windows 10 OS (i.e., client 6 in Table 2).

All browsers we use provide such a tool—we use these to investigate the differences in the HTTP messages generated by different clients.[15]

Results We find several practices that result in differences (and similarities) in the size of packets that carry HTTP requests from different clients:

- Clients use User-Agent strings that significantly differ in length (see Table 3 for a complete list of User-Agent strings used by the clients). For example, client 10 (Ubuntu - Firefox) and client 15 (Windows 7 - IE) have user agent strings that are 68 and 176 characters long, respectively.
- Different browsers may use different header fields (e.g., IE browser does not use "Upgrade-Insecure-Requests" header field whereas other browsers do[16]). Further, HTTP header field values may differ across different browsers (e.g., Chrome and Opera include "image/webp,image/apng" string in their default Accept value whereas other browsers do not[17]).
- Browsers use the same headers across different operating systems (e.g., Chrome on Windows 10 and Chrome on macOS generate the same headers and only differ in the User-Agent string).
- Clients that use Chrome or Opera browser (i.e., clients 1-8 and 16-18) use the same HTTP header fields and only differ in the User-Agent string. Of these, coincidentally, client 1 (Android - Chrome) and client 17 (Windows 10 - Opera) have user agent strings with the same length (131 characters). Thus, client 1 and 17 are expected to generate HTTP messages with the same length when the same URL is requested. Indeed, in our preliminary evaluation we found that when this client pair is used for evaluation, a significantly high accuracy is achieved (see Figure 3).[18]

Our observations in this section show that HTTP request sizes can differ across different client platforms, primarily due to user agent strings but also due to other header fields. Furthermore, client pairs that generate HTTP requests of the same size for a given webpage yield high fingerprinting accuracies when used for training and testing against each other.

5.1.1 User Specific Browser Configuration. Note that the HTTP request headers may change according to the configuration of a browser. For example, if a user specifies French as an additional language preference in Chrome settings, "fr;q=0.8" string will be added to the value of Accept-Language field in Figure 5; or if a user specifies sending a "do not track" request, "DNT: 1" string will be included in each HTTP request. These changes will practically have

[15]We use the Chrome remote debugging tool to investigate the HTTP headers generated by the Android clients.

[16]Indeed, IE is the only major browser that does not use this header field: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Upgrade-Insecure-Requests

[17]Note that the value of Accept field may change according to the type of resource requested (e.g., CSS, image or video): https://developer.mozilla.org/en-US/docs/Web/HTTP/Content_negotiation/List_of_default_Accept_values

[18]Note that client 4 (Android - Chrome) and client 9 (Windows 10 - Edge) also have user agent strings with the same length (129 characters long). However, these clients use different values for Accept-Language and Accept fields—Edge uses "en-US" and "text/html, application/xhtml+xml, application/xml; q=0.9, */*; q=0.8", respectively. The Chrome values for these fields are given in Figure 5.

the same impact we observed due to the variations in the length of User-Agent string across different clients. Thus, user specific browser configuration should also be considered as a part of client diversity for webpage fingerprinting purposes.

5.2 HTTP/2 Implementation

Background HTTP/2 is an optimized alternative to HTTP/1.1 [22]. HTTP/1.1 allows only one request to be outstanding at a time on a given TCP connection, suffers from head-of-line blocking, and repeats HTTP headers in each request. HTTP/2 addresses these issues and introduces several other features: request and response multiplexing over a single TCP connection, compression of HTTP header fields, request prioritization, server push and flow control. The basic HTTP/2 protocol unit is a binary *frame*. Each HTTP request/response is associated with its own *stream*. Naturally, the network traffic footprint differs when a webpage is visited over HTTP/2 versus HTTP/1.1.

Major browsers and servers support both HTTP/1.1 and HTTP/2 [23]—a client and a server negotiate which protocol to use during the TLS handshake (selected protocol is revealed in clear text). Figure 6 shows the average number of HTTP/2 and HTTP/1.1 connections used by each client during a webpage visit in our dataset. Note that three of the Windows clients (9, 14 and 17) generate more HTTP/2 connections compared to other clients whereas IE 11 on Windows 7 (client 15) does not use HTTP/2.

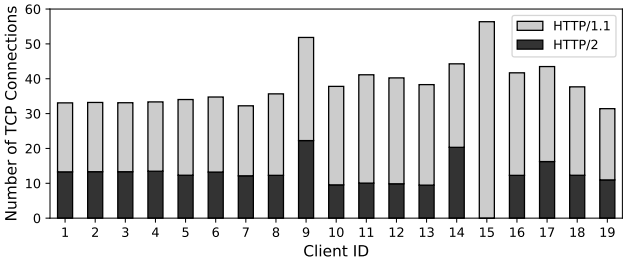


Figure 6: Average number of HTTP/2 and HTTP/1.1 connections used by each client during a webpage visit in our dataset. We have 9800 webpage visit samples (7 websites x 50 webpages x 28 samples) from each client.

Methodology Since HTTP/2 is used over TLS, HTTP/2 traffic analysis requires decryption of TLS connections. Chrome, Opera and Firefox browsers allow to generate an SSL key log file which can be used to decrypt the TLS connections whereas other browsers (IE, Edge and Safari) do not.[19] During the data collection we generated an SSL key log file for each visit performed in Chrome, Opera and Firefox browsers. After the data collection, we used tshark[20] to decrypt the TLS connections and decode the HTTP/2 frames.

Results Figure 7 shows the average number of HTTP/2 frames from each frame type generated by Chrome, Firefox and Opera browsers during a webpage visit in our dataset. We find several

[19] https://wiki.wireshark.org/SSL

[20] https://www.wireshark.org/docs/man-pages/tshark.html

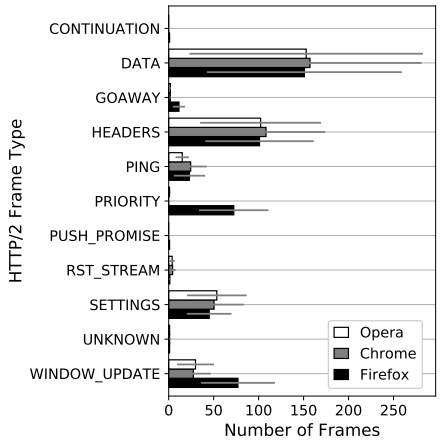


Figure 7: Average number of HTTP/2 frames from each frame type generated by Chrome, Firefox and Opera browsers during a webpage visit in our dataset.

practices that are likely to change the size of packets carrying HTTP/2 frames:

- On average, Chrome and Opera browsers generate similar number of frames. This is likely due to the fact that both Chrome and Opera are based on the Chromium open source project, and hence share the same HTTP/2 implementation.
- Firefox generates more WINDOW_ UPDATE and PRIORITY frames than Chrome or Opera. Further analysis of Firefox traffic traces reveals that WINDOW_UPDATE and HEADER frames are often found in the same network packet. A WINDOW_UPDATE frame is 13 bytes long.[21]. Thus, even if Chrome and Firefox browsers generate the same HTTP/2 HEADERS frames, most packets that contain a Firefox HEADERS frame will have 13 more bytes due to the WINDOW_ UPDATE frame in the same packet compared to the corresponding CHROME packets—significantly changing the outgoing packet sizes generated by the two browsers.
- Unlike the Chromium browsers, Firefox sends multiple PRIORITY frames in the same packet that contains the connection preface string.[22]

Our analysis in this section shows that different browsers differ in several aspects of their HTTP/2 implementations, resulting in differences in the number of packets generated as well as size of packets.

5.3 Client Specific Connections

When a webpage is visited, browsers often communicate with multiple servers to load web resources, such as HTML, CSS and image files, as well as communicate with tracking and advertisement servers. Figure 8 shows the total number of TCP connections from each client to 14 domains—we selected the domain names of the 7 websites we study and 7 additional domain names to illustrate

[21] 9-octet frame header + 4 octet payload [22].

[22] The client and server send a preface string to establish the initial settings of HTTP/2: https://tools.ietf.org/html/rfc7540#section-3.5.

the similarities and differences in the domains communicated by different clients.

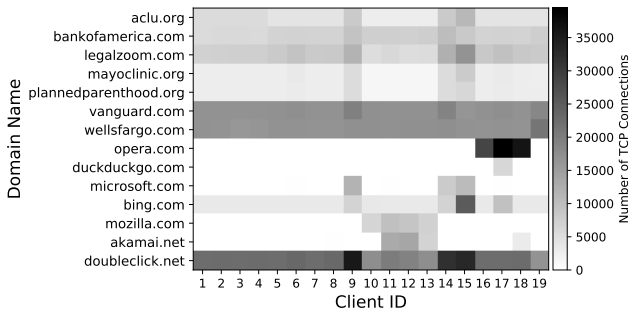


Figure 8: Total number of TCP connections from each client to 14 selected domain names in our dataset.

Similar to the observations in prior work [21], we find that some clients communicate with certain domain names significantly more than others, mainly due to browser specific communication: Opera browser (i.e., clients 16, 17 and 18) communicates with "opera.com", IE and Edge browsers (i.e., clients 9, 14 and 15) communicate with "microsoft.com" and Firefox browser (i.e., clients 11, 12 and 13) communicates with "mozilla.com". Furthermore, client 17 (Windows 10 - Opera) is the only client that communicates with "duckduckgo.com". The client specific connections significantly change the traffic features of a webpage across the clients, such as the number of incoming and outgoing packets—such features are used by several fingerprinting methods (e.g., Wfin, CUMUL and K-fingerprinting).

6 SEARCH FOR A ROBUST METHOD

In this section, we investigate whether a webpage fingerprinting method that is robust to the traffic variations across different clients can be trained through diversification of training samples.

Diversification of Training Samples We consider a scenario where a webpage fingerprinting entity, that is aware of the impact of client diversity and has sufficient computational resources, collects traffic samples from a diverse set of clients but does not consider a specific client during training—and the test samples are from that specific client. This scenario can be studied using one of the clients in our dataset as a test client and using the remaining 18 clients for training the webpage fingerprinting methods—we perform 19 such evaluations in each website with each method. As in our evaluation methodology in Section 4.2, we use the first 21 samples of a webpage from a client when the client is considered for training and use the remaining 7 samples when the client is used for test. Note that this scenario is rich in training data since we use 18 clients x 21 samples = 378 samples for each webpage during training (compared to the scenarios in Section 4 where we use only 21 samples from a single client for each webpage). Figure 9 plots the average accuracy of each method in each website in this scenario.

We observe a significant increase in the accuracy of Wfin and BoG methods compared to the scenarios in Section 4 where we evaluate the methods using samples from two different clients (Scenarios 2, 3, 4 and 5). For example, Wfin achieves around 35% accuracy in Vanguard in Scenario 2 (Figure 1b) whereas it achieves around 80% in present scenario for the same website (Figure 9).[23]

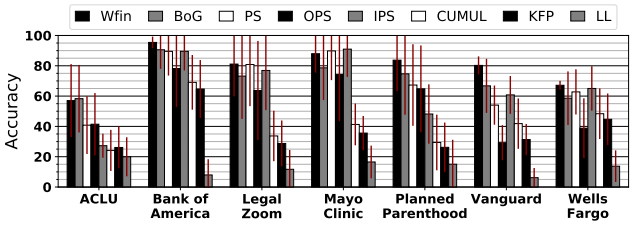


Figure 9: Webpage fingerprinting accuracy when samples from one client are used for test and samples from the remaining 18 clients are used for training—19 such evaluations are performed with each method in each website.

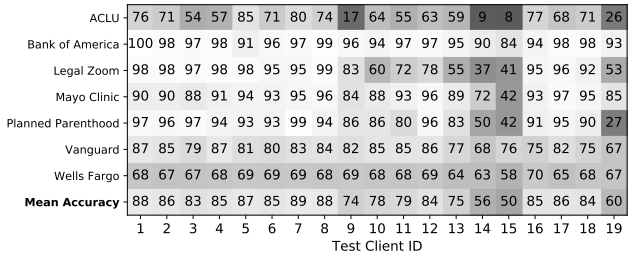
However, the fingerprinting accuracy of all methods is still lower than that achieved in Scenario 1 (Figure 1a)—even when significantly more training samples that are collected from multiple diverse clients are used. This further supports our hypothesis that when samples from only one client are used to evaluate fingerprinting methods (which is true for all prior work on HTTPS webpage fingerprinting), the fingerprinting accuracy may be significantly exaggerated (due to overfitting to specific packet sizes).

Figure 10a plots the accuracy of the Wfin method observed with each of the 19 test clients used in this scenario for each website. Compared to Figure 4, where we evaluate this method using training samples from only the single client 8, we observe higher and more uniform accuracies across the 19 test clients and the 7 websites.

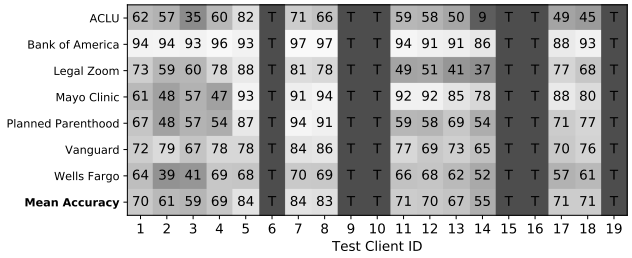
While we find that the methods perform better when they are trained using more samples from a diverse set of clients, compared to when they are trained using less samples from a single client, the main source of the improvement is not clear. Is it the diverse samples from different browsers? Is it the samples of the same browser from different clients or is it simply using more training samples? In order to answer these, we consider several other scenarios and evaluate the accuracy of the Wfin method, one of the best performing methods.

Samples From Just 6 Clients That Represent Different Browsers Figure 10b plots the accuracy of the Wfin method when the method is trained using samples from 6 different clients namely 6, 9, 10, 15, 16 and 19—these are clients that use the browsers Chrome, Edge, Firefox, IE, Opera, and Safari, respectively. Note that samples from all 6 browsers in our dataset are used during training, and 126 training samples (6 clients x 21 samples) are used for each webpage. We observe lower accuracies compared to Figure 10a where we train using samples from 18 different clients. However, compared to Figure 4, we observe much higher and more uniform accuracies for all websites and all test clients. It may be tempting to conclude that the 6 diverse clients represented in the training data are contributing to the performance improvement—however, it is important to

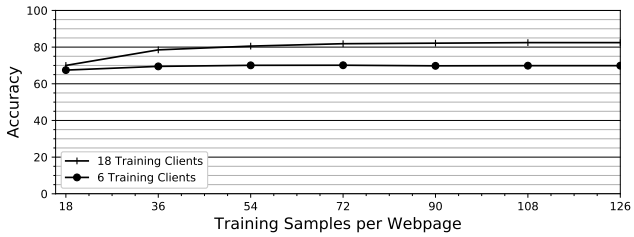
[23]Note that the average accuracy of some methods, such as PS, IPS and CUMUL, are lower in Figure 9 for some websites, compared to Figure 1b—in Figure 1b, we use only two test clients that use the same browser and OS but differ in device (clients 3 and 4), whereas in Figure 9 we average the accuracy over 19 test clients. When only the accuracies achieved in the two test clients used in Figure 1b are averaged, all methods achieve a higher accuracy when samples from 18 clients are used for training.



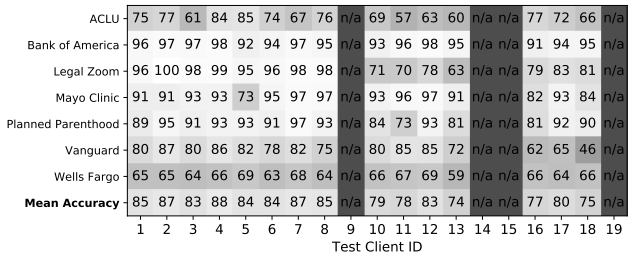
(a) When samples from one client are used for test and the samples from the remaining 18 clients are used for training.



(b) When samples from clients 6, 9, 10, 15, 16 and 19 are used for training. "T" indicates that a client is used for training.



(c) Effect of increasing the number of training samples per webpage.



(d) The clients are grouped into those that use the same browser (client ID ranges are given in parentheses): Chrome (1-8), Firefox(10-13) and Opera(16-18). One client in a group is used for test and the remaining clients in the group are used for training. "n/a" indicates that a client is not considered in this scenario.

Figure 10: Accuracy of the Wfin method in each website for each test client in different scenarios where the training samples are varied. The same 350 test samples from a test client (50 webpages within a website x 7 test samples) are used in each evaluation.

remember that the number of training samples is quite different for the evaluations summarized in each of Figures 4, 10a, and 10b. We isolate the impact of number of training samples next.

Effect of Number of Training Samples Figure 10c plots the effect of increasing the number of training samples per webpage in the scenarios of Figures 10a and 10b—for both scenarios, Figure 10c reports the average accuracy achieved for the 13 test clients that are evaluated in Figure 10b (i.e., 13 test clients x 7 websites = 91 evaluations are performed to calculate each accuracy point). Equal number of training samples are used from each of the training clients. The x-axis represents the number of training samples used per webpage—for example, when 36 samples are used per webpage, 2 samples are used from each of 18 training clients (and 6 samples are used from each of 6 training clients). We observe a significant increase in the accuracy when more samples are used from each of 18 clients—when 90 samples are used per webpage (5 samples from each of 18 clients) the accuracy increases to around 82% compared to around 70% when only 18 samples per webpage are used. However, the accuracy gain seems to saturate after 72 samples.

In the scenario using just 6 training clients, the performance gains are only slight when more samples are used from each client. Note that the 6 training clients use different browsers whereas among the 18 clients, there are multiple clients that use the same browser. We hypothesize that with 18 different training clients, Wfin mainly benefits from the larger number of samples of the same browser across different clients. We test this next.

Samples From the Same Browser Across Different Clients We divide clients into groups that use the same browser—we create 3 groups corresponding to Chrome, Firefox, and Opera. We then use one client from a group for test and the remaining clients to train the Wfin method. In order to remove the influence of number of training samples, we use around 21 samples from each webpage for training and take equal number of samples from each training client—we take 3 samples from each of 7 training clients in Chrome group, take 7 samples from each of 3 training clients in Firefox group, and take 11 samples from each of 2 training clients in Opera group. Since we have less than 3 clients that use IE, Edge and Safari browsers, we do not consider the corresponding clients here.

Figure 10d plots the accuracy observed. We observe higher and more uniform accuracies across the test clients compared to Figure 10b. The accuracies are also similar to those in Figure 10a. Note that in this scenario, we use significantly less training samples for each webpage compared to Figure 10b (21 versus 126 samples), and use similar number of training samples as the scenario in Figure 4 (in which we use 21 samples from a single client). We find that using samples of the same browser from different clients for training increases the robustness of the methods more (Figure 10d) compared to using samples from clients that use different browsers (Figure 10b). This experiment also shows that if the browser of a test client can be identified [24], a robust fingerprinting method can be trained by using samples from a diverse set of clients that use the same browser.

Based on the evaluations in this section, we conclude that:

- Even though the samples from the test clients are not considered in the evaluations during training, machine learning

methods are able to capture the variations in the traffic features across the diverse set of training clients and are now more robust to the traffic variations compared to when they are trained with samples from a single client (Figure 4).
- Significant browser differences (Section 5) necessitate that the training phase includes samples from the browser of the test client (even if the samples from the OS and device of the test client are not included).

7 RELATED WORK

Web traffic analysis research has been quite active for two decades.

HTTPS Cheng and Avnur (1998) performed the first webpage fingerprinting study in the traffic analysis literature [25]. They studied a single website and showed that most of the webpages have distinct HTML sizes which allows the visits to be identified in a traffic trace. Using HTTPS server logs, George Danezis investigated how much information can be inferred from HTTPS requests and whether a Hidden Markov Model can be used to find the most plausible explanation for the observed resource sizes [26].

In 2014, Miller et al. performed the first systematic HTTPS webpage fingerprinting study [5]. In 2016, Gonzalez et al. evaluated a method from the Tor traffic analysis domain in HTTPS webpage fingerprinting. These are discussed in Section 2.4.

Encrypted Web Proxy After the HTTPS study of Cheng and Avnur, a growing body of the traffic analysis literature investigated the feasibility of using webpage fingerprinting techniques for the web traffic protected by privacy enhancing technologies (PETs) such as SSH, VPN and Tor. One common property of these technologies is that they all use proxy tunnels to transmit web traffic and hide the identities of the web servers communicated by a client. Thus, an adversary has to consider that a client may visit any webpage on the web—the webpages cannot be narrowed down to those that are served from a single server.

Sun et al. were the first to consider such a proxy tunnel scenario [27]. They studied a scenario where a web user can visit 111,884 webpages and an adversary is interested in determining whether the user visits one of the 2191 target webpages. They argued that even though the false positive rate (i.e., the rate of predicting a visit to a webpage even though the webpage is not visited) may increase when all the webpages on the Web is considered, their methodology can be used for pruning the possibilities for a more sophisticated method. Similarly, Andrew Hintz showed that object sizes can be used to reveal the webpages visited in an encrypted web proxy named SafeWeb [28].

SSH Proxy Tunnel Bissias et al. [10] and Liberatore and Levine [6] investigated whether packet sizes can be used for fingerprinting webpages visited in an SSH proxy tunnel. Bissias et al. used cross correlation to measure similarities between packet size and inter-arrival time traces [10]. Similarly, Liberatore and Levine modeled a webpage as a multiset of packet sizes (with direction) and experimented with Jaccard Similarity and Naive Bayes classifier [6].

Anonymized NetFlow Records Coull et al. [29] and Yen et al. [24] studied fingerprinting webpages in anonymized NetFlow records (when only flow level information is available and the IP addresses are anonymized using consistent pseudonyms). Coull et al. considered the issues of network locality (collecting training and test data in different networks), browser caching, and browsing session parsing [29]. Yen et al. [24] first studied fingerprinting browsers in anonymized NetFlow records. The researchers then studied classifying traffic traces of landing pages of 52 websites as an application of browser fingerprinting. They showed that identifying the browser that generated a traffic trace first and then using a webpage fingerprinting method that was trained using the samples from that browser leads to an increase in the precision and recall (from around 25% and 5% to 32% and 15%, respectively) compared to using a generic fingerprinting method that was trained using samples from several browsers.

HTTP Maciá-Fernández et al. showed that the webpages visited within HTTP websites can be identified by matching the unique root and object file sizes of the webpages with the object sizes extracted from a traffic trace [30].

Tor Herrmann et al. investigated the feasibility of using webpage fingerprinting techniques against various other privacy enhancing technologies such as VPN and Tor and evaluated a variant of the method introduced by Liberatore and Levine [6] in this context [11]. Even though they reported less than 3% traffic trace classification accuracy for Tor, two years later Panchenko et al. [19] introduced a new set of features and used Support Vector Machines for classification and increased the accuracy to 55% in the same dataset. Since then, traffic analysis of Tor has been an active research area, with innovations mainly in the traffic features extracted and the supervised machine learning methods employed.

Notably, Cai et al. were the first to show that a sequence of webpage visits within a website can be modeled using a Hidden Markov Model to increase the accuracy of identifying a visit to a website [12]. Wang et al. [31] introduced a novel method that uses k-Nearest Neighbor to monitor the visits to 100 webpages from an open set of 5000 webpages. Hayes and Danezis used 150 different traffic features and Random Forest Classifier for fingerprinting 30 Tor hidden services [18]. Yan and Kaur [9] have shown that when an exhaustive feature selection methodology is used to find informative features, an accuracy around 92% can be achieved in classifying Tor traffic traces of landing pages of 100 websites.

Recent studies have questioned the assumptions made in the Tor traffic analysis domain [8, 32]. Panchenko et al. showed that when large number of webpages are considered, the state-of-the-art traffic analysis methods fail to identify the visited webpages in Tor network traffic. Juarez et al. [32] showed that variables such as the change of a website over time, multitab browsing, browser version, and the number of webpages considered can significantly affect the accuracy of traffic analysis methods in practice.

VPN Feghhi and Leith showed that the timing of the outgoing packets in an encrypted tunnel (e.g. VPN) can be used to predict the webpages visited [7].

Other Traffic Analysis Studies Chen et al. demonstrated that health records, tax information, investment secrets, and search queries can be leaked in network traffic due to user interactions such as keystrokes or mouse clicks [33]. Trevisan et al. showed that IP addresses and hostnames can be used to identify the traffic

of popular web services, such as facebook.com, google.com and whatsapp.net [34]. Sanders and Kaur showed that anonymized TCP/IP headers can be used to classify webpage traffic traces using several different labeling schemes, such as type of content, video vs. non-video, and mobile vs. non-mobile [35].

8 CONCLUSION

In this paper, we focus on the impact of client diversity on HTTPS webpage fingerprinting. Our analysis informs us about the differences across clients, reveals the adverse effect of client diversity on the performance of prominent webpage fingerprinting methods, and suggests using samples from a diverse set of clients for training a robust webpage fingerprinting method—even if the browser and OS of the test client are known.

Limitations While our dataset and analysis is an important step, there are several other factors—including thousands of webpages within a modern website, browser caching [5], browsing session parsing [29], multitab browsing [32], and user specific webpage content—that are expected to have a compounding adverse effect on the performance of the fingerprinting methods in the real world. We consider lack of large scale, labeled, and diverse webpage traffic trace datasets, as well as the difficulty of simulating real user webpage visits in laboratory conditions, as a major obstacle for a realistic evaluation of the webpage fingerprinting methods.

Client Diversity as a Dataset Bias Problem In the computer vision literature, dataset bias is a well known problem—object recognition methods trained using one dataset do not generalize in other datasets [36]. In future work, methods that explicitly model dataset bias (introduced by a specific web client) can be studied to approximate an unbiased traffic trace of a webpage visit (find features that are robust in webpage visits across clients) [37].

Implications for Other Traffic Analysis Studies While we focus on prior work on HTTPS webpage fingerprinting [4, 5], most prior traffic analysis studies in other settings were also performed using traffic samples from the same client [6, 7, 9, 11, 18, 25, 27, 30]. We hypothesize that the adverse effect of client diversity can also be observed in these settings. Future traffic analysis studies should collect traffic samples from a diverse set of clients to evaluate the robustness of proposed methods.

Our dataset and code have been made publicly available.[24]

REFERENCES

[1] Cooper A. et al. Privacy considerations for internet protocols. RFC 6973, RFC Editor, July 2013.

[2] T. Karagiannis et al. Blinc: multilevel traffic classification in the dark. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 229–240. ACM, 2005.

[3] W. Pan et al. Wenc: Https encrypted traffic classification using weighted ensemble learning and markov chain. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*, pages 50–57. IEEE, 2017.

[4] R. Gonzalez et al. User profiling in the time of https. In *Proceedings of the 2016 ACM on Internet Measurement Conference*, pages 373–379. ACM, 2016.

[5] B. Miller et al. I know why you went to the clinic: Risks and realization of https traffic analysis. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 143–163. Springer, 2014.

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

[7] S. Feghhi et al. A web traffic analysis attack using only timing information. *IEEE Transactions on Information Forensics and Security*, 11(8):1747–1759, 2016.

[8] A. Panchenko et al. Website fingerprinting at internet scale. In *NDSS*, 2016.

[9] Junhua Yan and Jasleen Kaur. Feature selection for website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 4:200–219, 2018.

[10] G. Bissias et al. Privacy vulnerabilities in encrypted http streams. In *International Workshop on Privacy Enhancing Technologies*, pages 1–11. Springer, 2005.

[11] D. Herrmann et al. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42. ACM, 2009.

[12] X. Cai et al. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 605–616. ACM, 2012.

[13] E Nygren. Reaching toward universal tls sni, 2017. URL https://blogs.akamai.com/2017/03/reaching-toward-universal-tls-sni.html.

[14] S. Sanders et al. The influence of client platform on web page content: Measurements, analysis, and implications. In *International Conference on Web Information Systems Engineering*, pages 1–16. Springer, 2015.

[15] Sean Sanders. *Techniques for the Analysis of Modern Web Page Traffic using Anonymized TCP/IP Headers*. PhD thesis, University of North Carolina at Chapel Hill, 2017.

[16] T. Berners-Lee et al. Hypertext transfer protocol – http/1.0. RFC 1945, RFC Editor, May 1996. URL http://www.rfc-editor.org/rfc/rfc1945.txt. http://www.rfc-editor.org/rfc/rfc1945.txt.

[17] A. Felt et al. Measuring https adoption on the web. In *26th USENIX Security Symposium*, pages 1323–1338, 2017.

[18] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security Symposium*, pages 1187–1203, 2016.

[19] A. Panchenko et al. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, 2011.

[20] Tao Wang and Ian Goldberg. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 201–212. ACM, 2013.

[21] Sean Sanders and Jasleen Kaur. On the variation in web page download traffic across different client types. In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pages 495–497. IEEE, 2014.

[22] M. Belshe et al. Hypertext transfer protocol version 2 (http/2). RFC 7540, RFC Editor, May 2015. URL http://www.rfc-editor.org/rfc/rfc7540.txt. http://www.rfc-editor.org/rfc/rfc7540.txt.

[23] M. Varvello et al. Is the web HTTP/2 yet? In *Passive and Active Measurements Conference (PAM)*, 2016.

[24] T. Yen et al. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 157–175. Springer, 2009.

[25] H. Cheng et al. Traffic analysis of ssl encrypted web browsing. *URL citeseer. ist. psu. edu/656522. html*, 1998.

[26] George Danezis. Traffic analysis of the http protocol over tls, 2009.

[27] Q. Sun et al. Statistical identification of encrypted web browsing traffic. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 19–30. IEEE, 2002.

[28] Andrew Hintz. Fingerprinting websites using traffic analysis. In *International Workshop on Privacy Enhancing Technologies*, pages 171–178. Springer, 2002.

[29] S. Coull et al. On web browsing privacy in anonymized netflows. In *USENIX Security Symposium*, pages 339–352, 2007.

[30] G. Maciá-Fernández et al. Isp-enabled behavioral ad targeting without deep packet inspection. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[31] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security Symposium*, pages 143–157, 2014.

[32] M. Juarez et al. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 263–274. ACM, 2014.

[33] S. Chen et al. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 191–206. IEEE, 2010.

[34] M. Trevisan et al. Towards web service classification using addresses and dns. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International*, pages 38–43. IEEE, 2016.

[35] Sean Sanders and Jasleen Kaur. Can web pages be classified using anonymized tcp/ip headers? In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2272–2280. IEEE, 2015.

[36] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[37] A. Khosla et al. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012.

[24] https://github.com/hfalan/codaspy19