# IUPTIS: A Practical, Cache-resistant Fingerprinting Technique for Dynamic Webpages

Mariano Di Martino[1], Pieter Robyns[1], Peter Quax[2] and Wim Lamotte[1]

[1]*Hasselt University/tUL - Expertise Center for Digital Media, Belgium*

[2]*Hasselt University/tUL/Flanders Make - Expertise Center for Digital Media, Belgium*

Keywords: Webpage Fingerprinting, Social Networks, Privacy, Traffic Analysis.

Abstract: Webpage fingerprinting allows an adversary to infer the webpages visited by an end user over an encrypted channel by means of network traffic analysis. If such techniques are applied to websites that contain user profiles (e.g. booking platforms), they can be used for personal identification and pose a clear privacy threat. In this paper, a novel HTTPS webpage fingerprinting method - IUPTIS - is presented, which accomplishes precisely this, through identification and analysis of unique image sequences. It improves upon previous work by being able to fingerprint webpages containing dynamic rather than just static content, making it applicable to e.g. social network pages as well. At the same time, it is not hindered by the presence of caching and does not require knowledge of the specific browser being used. Several accuracy-increasing parameters are integrated that can be tuned according to the specifics of the adversary model and targeted online platform. To quantify the real-world applicability of the IUPTIS method, experiments have been conducted on two popular online platforms. Favorable results were achieved, with a F1 scores between of 82% and 98%, depending on the parameters used. This makes the method practically viable as a means for personal identification.

## 1 INTRODUCTION

Historically, it was presumed that encryption of data communication was the ultimate protection of sensitive information against malicious adversaries. Unfortunately, in the late 90s, pioneers have shown that encrypted tunnels such as SSL can leak information that is critical for the construction of fingerprinting techniques, such as packet sizes and timing (Cheng et al., 1998; Sun et al., 2002). However, attacks based on these techniques were not feasible on a large scale. In late 2000, social networks have grown in popularity which quickly led to a collection of personal information given by the end users themselves. Due to the strict regulations in recent years (such as GDPR (European Commission, 2018)) that have been put in place, the method of extracting personal information has shifted back towards more covert approaches, such as fingerprinting, where the explicit input from the end user is not necessary (Perez, 2018). The concepts applied in fingerprinting techniques have also given rise to companies and government organizations that combine the analysis of public and intercepted data (Rao et al., 2015; Brandwatch, 2017; Gallagher, 2014). Regardless whether or not these analyses are legal, they are actively being used for predicting social media influence (Liu et al., 2016), personalized advertisements, assessing financial and health conditions (Economist, 2012) and forensic science (Ejeta and Kim, 2017). However, additional encryption mechanisms and tools such as SSH, VPN and Tor exist and can be (partially) utilized to protect the end user against these exposures. Despite their sophisticated and valuable capabilities, a considerably low number of users adopt these tools, which reminds us of the importance to not neglect the impact of HTTPS fingerprinting attacks on such scale. Moreover, several studies have explored *website fingerprinting* (WFP) over various encrypted layers and tunnels (Panchenko et al., 2016; Lu et al., 2010; Panchenko et al., 2011; Herrmann et al., 2009; Hayes and Danezis, 2016) and several countermeasures have been developed that defend the end user against such attacks (Panchenko et al., 2011; Cai et al., 2014a; Dyer et al., 2012). On the other hand, an exceeding number of the proposed WFP attacks have at least several issues that limit the ability to perform those attacks in a realistic environment (Juarez et al., 2014). Hence in this paper, we propose an HTTPS webpage fingerprinting technique

Di Martino, M., Robyns, P., Quax, P. and Lamotte, W.
IUPTIS: A Practical, Cache-resistant Fingerprinting Technique for Dynamic Webpages.
DOI: 10.5220/0007226501020112
In *Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018)*, pages 102-112
ISBN: 978-989-758-324-7

called IUPTIS [1] that infers the webpage of an individual online profile by analyzing the encrypted network traffic trace of an end user in order to identify a sequence of unique web object images. Our method improves upon the practicality of previous work by introducing parameters that can be fine-tuned according to the adversary model and the targeted online platform. Browser caching, dynamic webpages and the generation of a single fingerprint per profile for all browsers and versions is incorporated in these parameters and thus can be balanced with the precision and sensitivity of our technique. We specifically focus on a practical attack in a realistic scenario for traffic communicated over HTTPS, without additional encryption tunnels such as Tor. To quantify the value of our attack, we have conducted an experiment on two popular online platforms while accomplishing favorable results.
We summarize the key elements of our WFP attack:

- A novel HTTPS webpage fingerprinting attack that infers profile webpages of an online platform by searching for a unique sequence of images that are associated to that online profile, in a network traffic trace.
- Our attack introduces a technique that integrates the ability for the end user to enable browser caching and utilizes different browsers by establishing parameters that can be fine-tuned according to the intended model of the end user and online platform. Additionally, it also takes care of Content Distribution Networks (CDN) inserted between the targeted platform and attacker. This directly relaxes numerous of the assumptions in previous state-of-the-art methods (Juarez et al., 2014).
- We can perform our attack in an open world scenario without fingerprinting a selection of unmonitored webpages as was realized in previous work (Sun et al., 2002; Lu et al., 2010; Hayes and Danezis, 2016; Panchenko et al., 2016).
- We show that, unlike previous work has claimed (Sun et al., 2002; Panchenko et al., 2016), dynamic webpages that undergo frequent changes such as social profile pages *can* be fingerprinted with promising results.
- In the context of the adversary model, two experiments on popular online platforms are performed to demonstrate the effectiveness of our method. Experiments on online platforms with similar properties were not yet explored in previous state-of-the-art techniques.

In this paper, we first discuss the related work that has led to the development of our WFP attack. In Sect. 3, we introduce our intended adversary model in which we can succesfully perform our IUPTIS attack and compare its abilities with several state-of-the-art techniques. In the next section, we provide in-depth details of the inner workings of our WFP method. To quantify the practicality of our technique, we conduct experiments on two popular online platforms, DeviantArt and Hotels.com and discuss our findings in Sect. 4.1 and Sect. 4.2. To conclude our paper in Sect. 4.3 and Sect. 5, we briefly examine existing countermeasures that are able to mitigate our attack and discuss future work that might enhance the overall performance.

## 2 RELATED WORK

Early work have shown that it is feasible to fingerprint webpages over HTTPS by taking the size of web objects into account (Cheng et al., 1998; Sun et al., 2002). Nonetheless, some of the assumptions provided in these works, such as one TCP connection per web object, are not valid anymore in current modern browsers (Panchenko et al., 2011). The introduction of classifiers to infer webpages over the SSL protocol (Liberatore and Levine, 2006) has lead to the construction of Hidden Markov models to utilize the link structure of a website in combination with supplementary features such as the sizes and order of HTTPS web objects to deduce the browsing path of the end user (Miller et al., 2014; Cai et al., 2012). Furthermore, the ability to fingerprint webpages over encrypted tunnels such as SSH and Tor has been researched extensively (Panchenko et al., 2016; Lu et al., 2010; Panchenko et al., 2011; Herrmann et al., 2009). Recent work and currently a state-of-the-art technique to fingerprint HTTPS webpages (originally developed for Tor hidden services) is k-fingerprinting (Hayes and Danezis, 2016). Their work extends a previous approach (Kwon et al., 2015) and is also suitable over HTTPS. The classification of webpages is implemented using random forests and their experiment produces a TPR of 87% with a world size of 7000 unmonitored HTTPS webpages and 55 monitored HTTPS webpages by using the ordering, timing and size of TCP packets without the need to identify the actual web objects. Nevertheless, defenses against WFP attacks have been designed to reduce or completely nullify the precision and sensitivity of these experiments. Different padding methods have been evaluated to avoid the possibility of selecting packet size as a main feature (Dyer et al., 2012). Specifically desig-

[1] IUPTIS stands for 'Identifying User Profiles Through Image Sequences'

ned for HTTPS, HTTPOS is a defense in the form of a client-side proxy which implements several countermeasures to make it difficult for an adversary to use features such as timing, flow and size (Luo et al., 2011). Their defense uses the HTTP Range header to request parts of the HTTP content multiple times instead of requesting the entire content at once. Furthermore, it injects junk data to the content in order to cover up the real traffic data. A countermeasure named Camouflage (Panchenko et al., 2011) is a method to confuse WFP attacks by randomly requesting existing dummy webpages during the request of a legit webpage coordinated by the end user. Such mitigation has the advantage of explicitly generating false positives and is generally easy to incorporate in existing client-side proxies. Other defenses are much more deceptive such as ”Traffic Morphing” (Wright et al., 2009), in that they provide a theoretical approach to transform the distribution of packets of a traffic trace to another distribution in such manner that it resembles a different webpage. More recently, a defense called CS-BuFLO and an improved version has been devised (Cai et al., 2014a; Dyer et al., 2012; Cai et al., 2014b). This defense transforms a stream of original TCP packets to a continuous flow of fixed size packets to reduce the variance in timing and size of the original packets. Akin to the aforementioned defense, ’Walkie-Talkie’ is a similar approach (Wang and Goldberg, 2017) where they greatly improve upon bandwidth and practicality by devising a method that sends packets in short bursts and is currently regarded as a state-of-the-art defense. More recently, instead of manually selecting features to design WFP attacks, deep learning algorithms have been utilized to develop a process that allows an adversary to automatically select features (Rimmer et al., 2017). To conclude this section, we refer to an extensive and critical evaluation of the various WFP attacks and their countermeasures (Juarez et al., 2014).

# 3 IUPTIS: IDENTIFYING USER PROFILES THROUGH IMAGE SEQUENCES

## 3.1 Adversary Model

In the interest of a practical and reliable WFP attack, we would like to lay out some assumptions that are made:

- The adversary has a network traffic trace from the end user during the period in which they navigated to the webpage profile. Such traffic trace can be extracted with any passive MiTM attack.
- The recorded communication between the targeted online platform and the end user is handled by the HTTP/1.1 protocol encapsulated in TLS records. In other words, the end user is visiting the profile webpage over the HTTPS protocol.
- Each individual *profile* page may be accessed by an individual URL where distinctive and unique images are the main source of information on the webpage of that profile. A profile is associated with a person (e.g. social network pages) or unique entity (e.g. hotel pages). The images on each webpage profile have to be large enough (> 8 Kb) and usually larger than other resources (for instance, stylesheets) on the same webpage, to achieve acceptable results.
- The headers of the TCP and IP layer of the traffic trace are not encrypted and thus may be analyzed by the adversary. Background traffic (noise) from other websites or protocols is therefore trivial to filter out, since they will not match the IP or domain name of the targeted server.

Adversaries that adhere to these assumptions come in many forms. Social Wi-Fi providers and government agencies may essentially hold a passive MitM position and can therefore apply these techniques. Moreover, the introduction of WiFi4EU (WiFi4EU, 2016) will boost the number of accessible Wi-Fi access points and in turn, increase the attack surface to perform MitM attacks. In a similar fashion, social networks such as Facebook may provide VPN tools like Onavo (Perez, 2018) that still have access to HTTPS payloads with the ability to correlate the data with their own collection of online profiles.

## 3.2 Fingerprinting Images

Profile pages often contain several images that are uploaded by the owner of the page. These images are often the largest part of the page content and are most likely unique over the whole platform. The uniqueness of these images is very convenient to select as a feature for WFP attacks. When visiting such a profile page in a browser over HTTPS, the images will be downloaded in several TCP connections. As we are using HTTP over TLS, the actual content of the images is encrypted and thus not visible. However, the HTTP request and response sizes are not encrypted and can therefore be calculated easily (Lu et al., 2010; Cheng et al., 1998). Extracting the absolute raw size of each image contained in a HTTP response is not trivial due to the addition of HTTP headers, which are often dynamic in length. HTTP headers are the largest overhead in size that we have to eliminate in order to get the absolute size of each image. However, it is possible to deterministically model the appearance of these headers in each request and response. For each image contained in a HTTP response, we formulate the following equation that defines the total size of such HTTP response:

$$\text{Out}_x = w_{\text{out}} + p_{\text{out}} + i_{\text{out}} \tag{1}$$

Here, $w_{\text{out}}$ is the length of all HTTP headers (including the corresponding values) that are dependent on the webserver that issues the response to the browser. For instance, the header ”Accept-Language” or ”Server” is always added by the webserver (online platform) independent of the image that is requested or the browser that is used. Considering that we are targeting a specific webserver and leave out the presence of a *Content Distribution Network* (CDN) in the middle, the value of $w_{out}$ can be calculated easily.

$p_{\text{out}}$ is the length of all HTTP headers (including the corresponding values) that depend on the image requested. For instance, the ”Content-Type” and ”Content-Length” header can be different for each image requested from a given webserver and is independent of the browser that is used.

$i_{\text{out}}$ is the length of the complete HTTP response body. In our case, this only contains the raw data of the image requested. In a similar fashion, we also formulate an equation that defines the total size of HTTP request of a web object image:

$$\text{In}_x = p_{\text{in}} + b_{\text{in}} \tag{2}$$

Similar to the response, the variable $p_{\text{in}}$ is the total length of all HTTP headers that are dependent of each requested image. Examples are the GET path in the first request line and the ”Referrer” header.

$b_{\text{in}}$ is the length of all HTTP headers (including the corresponding values) that are dependent on the browser that issues the request. For instance, the ”DNT” or ”User-Agent” header may be different for each browser.

Since we would like to fingerprint webpage profiles based on the images that they contain, we have to determine the total size for each image. Then, based on the calculated values, we use the collection of all the images contained in a profile page to construct a fingerprint for the whole profile webpage.

We will extract the fingerprint of an image from the corresponding HTTP request and response sizes as follows:

$$\text{Img}_y = (\text{In}_x, \text{Out}_x) \tag{3}$$

To construct our fingerprint database for this preprocessing stage, we develop a fingerprint for each profile webpage $x$ with $n$ images where the approach is similar to the ordered sequence method (Lu et al., 2010):

$$\text{Profile}_z = \langle(\text{Img}_0, \text{Img}_1, \ldots, \text{Img}_{n-1}, \text{Img}_n)\rangle \tag{4}$$

Unlike previous work (Lu et al., 2010), the order in which the images are added have no impact on the experimental results of our method.

A practical consideration that arises is the fact that the variable $b_{\text{in}}$ is unknown and is most likely to be different for various browsers. It is therefore necessary to either figure out the browser that the end user is utilizing in order to estimate the variable (Husák et al., 2016; Cao et al., 2017) or to set the variable $b_{\text{in}}$ to a fixed size. The latter option will result in a trade-off with a lower precision of the WFP attack as we will show in Sect. 4. Likewise, $w_{\text{out}}$ may be hard to predict due to the variation of this variable within the same browser. More specifically, the usage of a CDN may introduce HTTP headers with irregular sizes, usually dependent on whether or not the requested image has been cached by the CDN. A possible solution for this concern is provided in Sect. 3.6.1 where we employ a single dimensional clustering method called ’Jenks optimization method’.

## 3.3 Constructing a Request/Response List

After our fingerprinting stage is finished, we have to build an ordered list of sizes that correspond to the HTTP requests and responses in our intercepted traffic trace, which we call a Request/response list (RRL). A request/response list of length $n$ is an ordered list that contains the timestamp of each HTTP request ($T_n$), the size of each HTTP request and HTTP response (request/response pair) *associated* with a web object image:

$$\text{RRL} = \langle(R_0, R_1, \ldots, R_{n-1}, R_n)\rangle \tag{5}$$

$$R_n = \langle(T_n, \text{Req}_n, \text{Resp}_n)\rangle \tag{6}$$

When the end user navigates to a profile webpage, several TCP connections to the webserver will download the resources located on that particular webpage. These resources consist of images, stylesheets, source files, etc.... Due to the encryption provided by HTTPS, an adversary cannot trivially identify the responses that contain images. Therefore, similar to the ordered sequence approach (Lu et al., 2010), we use the fair assumption that images are usually larger in size than other resources and filter out all other resources that are below a fixed threshold. Such threshold will effectively reduce the amount of noise associated with other resources. The value of this threshold is usually set to the fingerprinted image with the smallest size (*Resp*$_x$). Other images (which we will also observe as noise) such as icons or banners are usually much smaller in size than the fingerprinted images and are often downloaded at the beginning or end of the HTML page. Interference of such noise with our method is therefore minimal.

### 3.4 Building a Profile Prediction List

At this stage, we have collected the necessary fingerprints and have constructed a RRL based on the intercepted traffic trace.
Subsequently, for each request/response pair ($R$) from our RRL, we evaluate whether this pair *matches* one or more $\text{Img}_y$ fingerprints from any $\text{Profile}_z$, where $z$ denotes any fingerprinted profile and $y$ denotes all image fingerprints from $\text{Profile}_z$. The *matching* is performed i.f.f. the following equations hold:

$$\text{Img}_y = (\text{In}_w, \text{Out}_w) \tag{7}$$

$$\text{In}_\text{w} - \pi_\text{req} < \text{Req}_x < \text{In}_\text{w} + \pi_\text{req} \tag{8}$$

$$\text{Out}_w - \pi_\text{resp} < \text{Resp}_x < \text{Out}_w + \pi_\text{resp} \tag{9}$$

In the equation above, we perform the matching if the request and response sizes lie within an interval defined by 2 newly introduced parameters, $\pi_\text{req}$ and $\pi_\text{resp}$. Both parameters are defined as the statistical variance for respectively the request and response size and should be chosen with the intended platform and browser in mind. If the specific browser that is employed is known, $b_\text{in}$ can be calculated to be very accurate and thus requires a low $\pi_\text{req}$. Similarly, $\pi_\text{resp}$ depends on the accuracy of $b_\text{out}$. In the experiments discussed in Sect. 4, we show that a large $\pi_\text{req}$ and $\pi_\text{resp}$ are still sufficiently robust enough to achieve favorable results.
If the equations above (eq. 8 and 9) hold, the *matching* is then performed by passing the corresponding $\text{Req}_x$ and $\text{Resp}_x$ to construct a $P_g^n$ which is finally appended to the *Profile Prediction List* (PPL):

$$P_g^n = \langle \textit{profileName}, \varepsilon_\text{req}, \varepsilon_\text{resp} \rangle \tag{10}$$

$$\varepsilon_\text{req} = \text{Req}_x - \text{In}_\text{w} \tag{11}$$

$$\varepsilon_\text{resp} = \text{Resp}_x - \text{Out}_\text{w} \tag{12}$$

Subscript $g$ denotes the index in the second dimension and superscript $n$ denotes the index in the first dimension of the PPL. All elements in the PPL are chronologically ordered based on the timestamp of the matched request/response pair. The PPL is constructed as a 2D array with variable size in the second dimension. The first dimension defines each matched $R$ element (ordered by the timestamp) and the second dimension defines all the newly constructed $P_g^n$:

$$\text{PPL}[0] = \langle P_0^0, P_1^0, P_2^0, \dots \rangle \tag{13}$$

$$\text{PPL}[1] = \langle P_0^1, P_1^1, P_2^1, \dots \rangle \tag{14}$$

$$\text{PPL}[\dots] = \langle P_0^{\dots}, P_1^{\dots}, P_2^{\dots}, \dots \rangle \tag{15}$$

$$\text{PPL}[q] = \langle P_0^q, P_1^q, P_2^q, \dots \rangle \tag{16}$$

In other words, for each intercepted request/response pair $R_n$ (with a total of $q$ pairs), we assign all image fingerprints (associated to a profile) that might belong to that pair and construct a $P_g^n$ for each of them as shown in figure 1.

### 3.5 Finding a Valid Profile Sequence

After the creation of our PPL, we attempt to identify an uninterrupted sequence with length $\Phi$ in the PPL starting at any $X$ such that $\text{PPL}[X]$,$\text{PPL}[X+1]$,$\text{PPL}[X+\dots]$,$\text{PPL}[X+\Phi]$ in the sequence have respectively **a** $P^X$,$P^{X+1}$,$P^{X+\cdots}$,$P^{X+\Phi}$ such that they all have the same *profileName*:

$$P^X \in PPL[X] \tag{17}$$

$$P^{X+1} \in PPL[X+1] \tag{18}$$

$$P^{X+\cdots} \in PPL[X+\dots] \tag{19}$$

$$P^{X+\Phi} \in PPL[X+\Phi] \tag{20}$$

$$\textit{profileName} \in (P^X \cap P^{X+1} \cap P^{X+\cdots} \cap P^{X+\Phi}) \tag{21}$$

In other words, we have found a valid profile sequence if $\Phi$ request/response pairs in a row are all matched to at least one fingerprinted image of the same profile. We say that $P^X$,$P^{X+1}$,$P^{X+\cdots}$ and $P^{X+\Phi}$ form a valid sequence for that particular *profileName*. Multiple profile sequences may obviously exist. The introduction of the parameter $\Phi$ defines a balance between browser caching and resulting precision and sensitivity. When choosing this value, it's useful to look at the number of images that are exposed on each individual webpage. A large value for $\Phi$ will have a more accurate prediction but might reduce the effectiveness of the attack. For instance, if a profile webpage only has 2 images, then a $\Phi$ below 3 is necessary to identify that particular webpage. More importantly, the parameter is also utilized to reduce the impact of browser cached images. For instance, if the end user is visiting the webpage of a $\text{Profile}_z$ which has 10 images where 5 of those are already cached by the browser, we can still set $\Phi$ to a value below 6 in order to succesfully find a valid sequence.

### 3.6 Evaluating a Profile Sequence

For a small collection of image fingerprints, the resulting profile sequences are already a valuable prediction. However, this is insufficient if multiple sequences for the same time range exist or when the variance between the fingerprinted images is too small. Therefore, we have to exclude profile sequences that have very different values for $\varepsilon_{\text{req}}$ and $\varepsilon_{\text{resp}}$. The exclusion is accomplished by calculating the standard deviation $\sigma_{req}$ and $\sigma_{resp}$ over respectively all $\varepsilon_{\text{req}}$ and $\varepsilon_{\text{resp}}$ in that particular sequence. The mean over all $\varepsilon_{\text{req}}$ and $\varepsilon_{\text{resp}}$ for a sequence can be large if respectively $b_{in}$ and $w_{out}$ are inaccurate or unknown even though it should not influence our result and for this reason, the standard deviation is utilized. Afterwards, we evaluate whether the standard deviations are below a threshold $H_{\text{req}}$ and $H_{\text{resp}}$. I.f.f. both deviations are above the thresholds, we can assume that the error differences in that sequence are too large and thus exclude that sequence. All other sequences are said to be 'complete'. A complete sequence will establish a prediction saying that the end user has navigated to the profile corresponding to the sequence. For instance, assume we have $\Phi$ profiles in our valid sequence – $P_1$, $P_2$ , $P_{\dots}$, $P_{\Phi-1}$ and $P_\Phi$, we can then calculate the following parameters:

$$\mu_{\text{req}} = \frac{\sum_{i=1}^{\Phi} \varepsilon_{\text{req}}(P_i)}{\Phi} \tag{22}$$

$$\mu_{\text{resp}} = \frac{\sum_{i=1}^{\Phi} \varepsilon_{\text{resp}}(P_i)}{\Phi} \tag{23}$$

$$\sigma_{req} = \sqrt{\frac{\sum_{i=1}^{\Phi} (\varepsilon_{\text{req}}(P_i) - \mu_{\text{req}})^2}{\Phi}} \tag{24}$$

$$\sigma_{resp} = \sqrt{\frac{\sum_{i=1}^{\Phi} (\varepsilon_{\text{resp}}(P_i) - \mu_{\text{resp}})^2}{\Phi}} \tag{25}$$

### 3.6.1 Jenks Optimization Method

Determining the standard deviations and then comparing it to a predefined threshold is relatively robust considering that $b_{\text{in}}$ and $w_{\text{out}}$ remains constant over all images of the same profile. Although, the presence of a CDN will essentially break that assumption by appending additional proprietary HTTP headers such as 'X-Cache' or 'X-Amz-Cf-Id', in case the requested image was cached by a CDN server. In the interest of distinguishing cached images [2] from uncached images or at least reduce the effect on the standard deviations, we employ the Jenks optimization method. This optimization method (also known as '*Goodness of Variance Fit*') clusters an 1D array of numbers into several classes with minimal average deviation from each class mean. For our IUPTIS attack, all $\varepsilon_{\text{resp}}$ of each $P_g^n$ (Eq. 10) in a valid sequence will be clustered into 2 classes (CDN-cached and uncached images). The integration of this method happens immediately after finding a valid sequence. Following the clustering, we compute $\sigma_{\text{req}}$ and $\sigma_{\text{resp}}$ for each class, which makes a total of 4 standard deviations. However, if one of the calculated classes only contain 1 element, we will have to assume that the single element is a false positive and therefore, fallback to the original method of computing the standard deviations for the whole sequence. Having multiple CDN of the same provider is not an issue, due to the responses not changing in size. When validating our experiments, we did not encounter an instance where multiple CDN of different providers were utilized on the same webpage. If it nevertheless does occur, the number of classes for the optimization method can be increased to compensate for this.

## 3.7 Recap

Our IUPTIS method is composed of the following steps:

1. Intercept a network traffic trace from the end user.
2. Establish the collection of fingerprints by extracting the fingerprints of each targeted profile (Sect. 3.2).
3. Build an ordered Request/Response list (RRL) from the raw traffic trace as discussed (Sect. 3.3).
4. Construct a Profile Prediction List (PPL) by matching the elements from the RRL to one or multiple image fingerprints ($Img_x$).
5. Find a sequence of $\Phi$ elements in the PPL that all contain at least one image from the same $Profile_y$ (Sect. 3.5)
6. Evaluate the formed sequence by our IUPTIS algorithm which decides whether or not the sequence is classified as a valid profile prediction (Sect. 3.6).

Each attack is executed with the following tuneable parameters:

- $b_{\text{in}}$: The expected size of data (HTTP headers and corresponding values) in a request that is dependent on the webbrowser. If this value is unknown, an average value can be set albeit with a large $\pi_{\text{req}}$ and $\pi_{\text{resp}}$ to compensate for different browsers. It is for instance possible to extract this value by identifying the browser through the extraction

[2] Note the difference between browser-cached and CDN-cached images. We are talking about the latter here.

of the User-Agent header from an unencrypted HTTP request.

- *useJenks*: Utilized if a CDN is employed on the targeted online platform.
- $\Phi$: Minimum matching sequence or streak of images. This value can be freely set, although a long sequence results into an accurate prediction, but with a low accuracy for webpages that have cached images.
- $\pi_{\text{req}}$, $\pi_{\text{resp}}$: Request and response variance that allows matching to an image fingerprint. Calculate $\pi_{\text{req}}$ by taking the difference between $In_x$ and $In_y$ where $x$ defines the largest possible request size of any fingerprinted image and $y$ defines the smallest possible request size of any fingerprinted image. Some manual fine-tuning is necessary for $\pi_{\text{resp}}$ if a CDN is utilized. Creating a test case with several random profile fingerprints and then iteratively increasing $\pi_{\text{resp}}$ with a constant size is recommended until the preferred results are achieved.
- $H_{\text{req}}$, $H_{\text{resp}}$: The threshold of the maximum standard deviation for respectively, the requests and responses. Both parameters are fixed for each online platform. Similar to $\pi_{\text{resp}}$, both parameters require manual fine-tuning by iteratively increasing the value.

The ideal combination of parameters depends on the adversary model, such as whether he wants to allow browser caching or a high precision in trade for a lower sensitivity. Possible combinations are provided in Sect. 4. In figure 1, we show an example of the IUPTIS method consisting of the first 5 steps. Starting from the bottom, the actual images are downloaded by the browser, resulting into a request and response, each with a specific size. Subsequently, the PPL is constructed by matching the request and responses to one or more profiles from our fingerprinting database. Then, we find an uninterrupted sequence of at least length $\Phi$, which is 'Profile C' in our case. Finally, the sequence is evaluated to be fit for a valid profile prediction.

## 3.8 Comparison to State-of-the-art Techniques

Our attack differs from state-of-the-art techniques like k-fingerprinting (k-FP) (Hayes and Danezis, 2016) and the Miller method (Miller et al., 2014), in the idea that we specifically target a subset of online platforms, and decouple browser caching and dynamic webpages by introducing several parameters that can be fine-tuned according to the demands of an adversary.

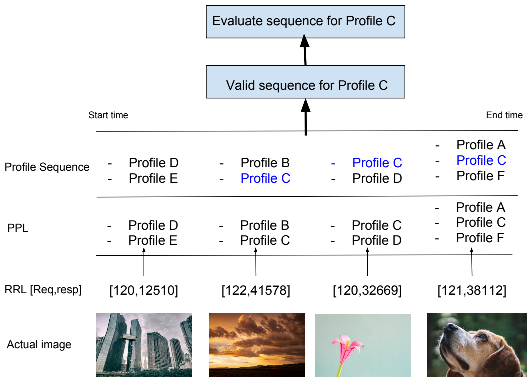

Figure 1: Toy example with a visual 5-step overview of the IUPTIS method. Assume $\Phi$ is 3 and our fingerprint database consists of Profile A to Profile G.

In comparison to machine learning (ML) attacks (Panchenko et al., 2016; Panchenko et al., 2011; Hayes and Danezis, 2016) where we have to collect several traces from page loads, our fingerprinting stage only requires one page load for each profile. Numerous strong assumptions made in state-of-the-art methods are relaxed or completely removed in our IUPTIS attack (Juarez et al., 2014). For instance, the ability to perform our attack on different browsers and devices, without the need to collect session traces from each one individually, is an approach that is rarely proposed. Moreover as we will demonstrate in the experiments, our attack does not assume that we know the end and the beginning of a page load in a given trace, which is shown to be difficult to deduce (Coull et al., 2007; Juarez et al., 2014). Although due to such valuable properties, our attack is only applicable over TLS and thus it does not support anonymization services such as Tor. On the other hand, disadvantages of our attack can be found when applying mitigations. Due to the rather deterministic nature of our algorithm, existing defenses can be very effective to mitigate our attack as discussed in Sect. 4.3. Current state-of-the-art techniques are more resistant against defenses such as padding and have even defeated more advanced defenses such as HTTPOS, CS-BufLo or Tamaraw (Hayes and Danezis, 2016; Cai et al., 2012; Juarez et al., 2016; Cai et al., 2014b). Although, as presented in the taxonomy "*I Know Why You Went To Clinic*" (Miller et al., 2014), most of these WFP methods need to fingerprint unmonitored webpages to make it feasible in an open world scenario (Cherubin, 2017; Juarez et al., 2014) and almost all WFP attacks require browser caching to be turned off, which is less feasible nowadays. Furthermore, the flexibility of our attack parameters requires manual preliminary work which involves analyzing the HTTP request/responses and then tune these parameters in pursuance of an effective attack. In addition, we address the *base rate fallacy* (Juarez et al., 2014; Wang, 2015) by carefully formulating our assumptions and adversary model and focussing on precision instead of sensitivity. Subsequently, the IUPTIS technique does not explicitly measure similarities between fingerprinted profiles and thus eliminates the necessity to create a separate collection of unmonitored webpages. As a result, our attack has the valuable property that whenever we increase the world size, only the precision will be affected and the sensitivity will remain relatively the same.

# 4 EXPERIMENTAL VALIDATION

In this section, we perform our IUPTIS attack on the social platform 'DeviantArt' and the travel booking platform 'Hotels.com'. Our experiment is simulated by randomly selecting one of the following browsers (for each test): Firefox 56.0.2 (Linux), Google Chrome 62.0 (Linux) and Google Chrome 61.0 (Android). Nevertheless, in the context of our attack, there are no notable differences between different browsers except from the change in request size. Besides the 2 platforms discussed below, we have also succesfully experimented with other platforms such as Pornhub, Pinterest and We Heart It. Unfortunately, we are not able to elaborate on these additional experiments due to page limitations.

## 4.1 DeviantArt

DeviantArt is an online art community that consists of 36 million users where artists can upload and view a substantial number of artworks. We randomly compile a list of 2150 DeviantArt profile webpages [3] that have at least 5 uploaded images. Our traffic trace is constructed by spawning each profile webpage separately after each other until all images are loaded with a minimum delay of 3 seconds before closing the previous page and opening the next webpage. *Lazy loading* is a concept that is applied on DeviantArt which means that only the images in the current viewport will be downloaded and thus visible in the traffic trace. With this generated traffic trace containing 2150 profiles, we run our IUPTIS attack and obtain the results in table 1. The first test has set the parameter $b_{\text{in}}$ which indicates that the adversary knows which browser the end user is using. Including this additional parameter has a considerable positive effect on the precision of the attack with an increase of 11% (ceteris paribus). It is also evident that a large sequence will increase the precision and decrease the sensitivity. We can attribute this due to the statistical probability that it is less likely for a profile to have the same size of several images in a row as another profile. Browser-cached images do influence the sensitivity since the request for those images will not lead to the image contained in a HTTP response. It is therefore possible that the number of images that are left on a particular profile do not meet the requirements to evolve into a valid sequence. Although, the precision is clearly not affected since browser-cached images do not generate any additional false positives due to the fact that those browser-cached images are often being requested at the start or end of the series of fingerprinted images.

Table 1: Experiment on DeviantArt with parameters ($\pi_{\text{req}}$ = 300, $\pi_{\text{resp}}$ = 40, *useJenks* = no, $b_{\text{in}} = 252$, $H_{\text{resp}}$ = 3.6 and $H_{\text{req}}$ = 0.4) and a worldsize of 2150 profiles. Parameter '*cache*=X' indicates that we pre-cache (browser based) the first X % of all the images located on the profile webpage. Sensitivity, precision and F1 score are presented as percentages.

| $\Phi$ | **Other parameters** | **Sens.** | **Prec.** | **F1** |
|---|---|---|---|---|
| 2 | $\pi_{resp}$=10, $b_{\text{in}}$=$X$ | 99 | 98 | 99 |
| 2 | $\pi_{resp}$=10, $b_{\text{in}}$=$X$, *cache*=40 | 94 | 98 | 96 |
| 2 | / | 99 | 88 | 93 |
| 3 | / | 98 | 93 | 95 |
| 4 | *useJenks*=yes | 97 | 97 | 97 |
| 5 | *useJenks*=yes | 96 | 99 | 98 |
| 5 | *useJenks*=yes, *cache*=40 | 87 | 99 | 92 |

## 4.2 Hotels.com

Hotels.com is an online travel booking platform with an average of 50 million visitors per month and currently has around 260 000 bookable properties.

We compile our profile list by randomly selecting 900 hotel profile webpages [4]. Our traffic trace is constructed by spawning each hotel webpage and then opening 75% of all the images located on the webpage. Images are not loaded automatically and thus requires the end user to click on the image in the interest of downloading the full resolution image. We argue that the average end user does not open all images when browsing through the webpage.

In table 2, we show the sensitivity, precision and F1 score based on the experiment run by altering parameter $H_{\text{resp}}$, the sequence length and whether or not we use the Jenks method. With the exception of 'Without Jenks($H_{\text{resp}}$=3.5)', a consistent F1 score between 80 - 98% is achieved. For a sequence ($\Phi$) of 8 images, 'With Jenks ($H_{\text{resp}}$=6.0,$\Phi = 8$)' yields

[3] https://www.deviantart.com/[USER_NAME]/gallery

[4] https://hotels.com/ho[NUMBER]/?[GET_PARAMS]

a F1 score of 98%. Overall, sensitivity is relatively constant in almost all tests and only decreases slightly when a longer sequence is necessary as shown in Fig. 2 and Fig. 3. On the contrary, the precision starts low and increases to a very convenient 99% in some cases. However, a low sequence length is prefered to incorporate the ability for the end user to use browser caching in trade for a lower precision. Fortunately, performing the attack without Jenks and $H_{\text{resp}}$=8.5 already attains a sensitivity and precision of respectively 99 and 92% for a sequence of 6 images. Furthermore, applying the Jenks method to model the CDN cache behavior does show major improvements in sensitivity over different $H_{\text{resp}}$ values (ceteris paribus) with only a nominal decrease in precision. For instance, 'Without Jenks ($H_{\text{resp}}$=3.5)' has inferior sensitivity (below 55%) compared to the other tests due to the fact that some images are cached by the CDN server which makes the resulting responses very different in size. On the contrary, in the DeviantArt experiment, the CDNs employed did not had a significant impact on the response size. Ultimately, we argue that 'Without Jenks ($H_{\text{resp}}$=8.5)' is ideal in this scenario due to the very advantageous precision (85% to 100%) and relatively high sensitivity (82% to 99%) over all possible sequence lengths. In conclusion, we determine that the combination of parameters to perform the attack will greatly depend on the adversary and end user model.

A timeframe of 14 days was created between generating the image fingerprints and performing our experiment to show the longevity of our fingerprints. Within this timeframe, several profiles of our targeted platforms had added and deleted several images. The results of our experiment demonstrates that the impact with such modifications is neglectable. Furthermore, we have conducted our tests on different hours and days in a week to get a decent statistical overview of all the requests. This is crucial due to the fact that a CDN is heavily dependent on the time of day which influences the responses and in turn our results.

### 4.3 Defenses

Existing WFP defenses are highly effective against an IUPTIS attack. Primarily because mitigations such as CS-BufLo (Cai et al., 2014a) and Walkie-Talkie (Wang and Goldberg, 2017) completely remove the ability for the adversary to deduce the exact size of a web object image, thus rendering our attack ineffective. Nevertheless, it is extremely important to note that the end user from the adversary model that

Table 2: Experiment on Hotels.com with a worldsize of 900 hotel profiles and fixed parameters ($b_{\text{in}} = 250$, $\pi_{\text{resp}} = 100$, $\pi_{\text{req}} = 450$ and $H_{\text{req}} = 0.2$) . Sensitivity, Precision and F1 score are presented as percentages. WOJ stands for 'Without Jenks' and WJ stands for 'With Jenks'. The underlined percentages represent the highest F1 score for that particular sequence $\Phi$.

| **Parameters** | **Sens.** | **Prec.** | **F1 score** |
|---|---|---|---|
| WOJ ($H_{\text{resp}}$=3.5,$\Phi = 5$) | 55 | 86 | 67 |
| WOJ ($H_{\text{resp}}$=3.5,$\Phi = 6$) | 27 | 97 | 43 |
| WOJ ($H_{\text{resp}}$=3.5,$\Phi = 7$) | 9 | 99 | 16 |
| WOJ ($H_{\text{resp}}$=3.5,$\Phi = 8$) | 3 | 99 | 6 |
| WOJ ($H_{\text{resp}}$=3.5,$\Phi = 9$) | 1 | 100 | 2 |
| WOJ ($H_{\text{resp}}$=8.5,$\Phi = 5$) | 99 | 85 | 92 |
| WOJ ($H_{\text{resp}}$=8.5,$\Phi = 6$) | 99 | 92 | 96 |
| WOJ ($H_{\text{resp}}$=8.5,$\Phi = 7$) | 91 | 95 | 83 |
| WOJ ($H_{\text{resp}}$=8.5,$\Phi = 8$) | 90 | 99 | 94 |
| WOJ ($H_{\text{resp}}$=8.5,$\Phi = 9$) | 82 | 100 | 90 |
| WJ ($H_{\text{resp}}$=3.5,$\Phi = 5$) | 91 | 77 | 83 |
| WJ ($H_{\text{resp}}$=3.5,$\Phi = 6$) | 91 | 91 | 91 |
| WJ ($H_{\text{resp}}$=3.5,$\Phi = 7$) | 99 | 92 | 96 |
| WJ ($H_{\text{resp}}$=3.5,$\Phi = 8$) | 83 | 99 | 90 |
| WJ ($H_{\text{resp}}$=3.5,$\Phi = 9$) | 73 | 100 | 85 |
| WJ ($H_{\text{resp}}$=6.0,$\Phi = 5$) | 97 | 71 | 82 |
| WJ ($H_{\text{resp}}$=6.0,$\Phi = 6$) | 99 | 80 | 88 |
| WJ ($H_{\text{resp}}$=6.0,$\Phi = 7$) | 99 | 85 | 89 |
| WJ ($H_{\text{resp}}$=6.0,$\Phi = 8$) | 99 | 98 | 98 |
| WJ ($H_{\text{resp}}$=6.0,$\Phi = 9$) | 82 | 99 | 89 |

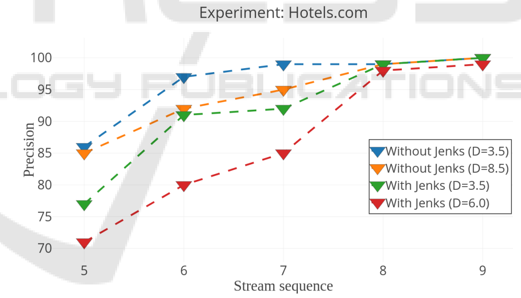

Figure 2: Hotels.com experiment consisting of various tests from table 2 with different parameters, plotting the precision on the length of a valid sequence.

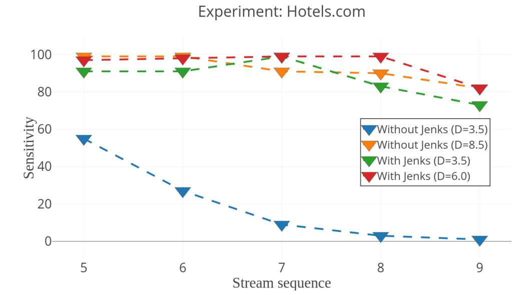

Figure 3: Hotels.com experiment consisting of various tests from table 2 with different parameters, plotting the sensitivity on the length of a valid sequence.

we are targetting often does not have the knowledge nor the ability to apply such defenses. We can argue that it is the responsibility of the online platform to protect the end users from any fingerprinting attack. Recent work has shown the demand for such a server-side countermeasure called ALPaCA (Cherubin et al., 2017). Despite their promising results on a Tor network, it is not suitable for webpages serving over HTTPS. However, recent work has also demonstrated that a proper server-side implementation of HTTP/2 does make it troublesome for an adversary to infer the exact image sizes (Morla, 2017). Even though this would defeat our current attack, it does not completely mitigate the risk for future WFP attacks that might enhance our method to discover new techniques that use a sequence of images as their main feature. To conclude, we think it is critical to educate the end user about the available tools that exist to protect themselves against fingerprinting attacks such as IUPTIS, on a wide scale.

## 5 CONCLUSION AND FUTURE WORK

We have proposed a new webpage fingerprinting technique called 'IUPTIS' that focuses on the practicality in an open world scenario. The ability to use different browser versions and enable caching is an improvement over previous state-of-the-art techniques. Our experiments have generated favorable results with F1 scores between 90% and 98% dependent on the parameters utilized and highlights the privacy impact on various online platforms. However, our attack is only applicable on the HTTP/1.1 protocol due to the assumption that we can infer the exact response size. Nonetheless, recent work (Wijnants et al., 2018) has indicated that some implementations of the HTTP/2 protocol have a rather deterministic approach in multiplexing which might make the estimation of response sizes in such protocol still relatively accurate. Furthermore, we did show the impact of our attack on a small subset of all profiles available on a platform. However, some platforms only have a small collection of online profiles (DeviantArt has 36 million users). The impact of our attack on an even larger scale is unknown and should be explored in future work.

Easing the manual fine-tuning of the parameters such as $H_{\text{req}}$ and $H_{\text{resp}}$ is also an important aspect to be improved in future work. A possible extension to this issue, would be to automatically learn these parameters by utilizing some form of machine learning. Moreover, our addition of the Jenks optimization method only shows improvements in parts of the experiments where the CDN heavily influences the response size. Further experiments on other online platforms should be performed to analyze how exactly incorporating this optimization method will improve the overall F1 score. Additionally, other metrics for calculating the error ($\varepsilon$) for each fingerprinted image in Sect. 3.4, such as the squared difference are not examined yet and might be able to generate superior results.

## ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful feedback.

## REFERENCES

Brandwatch (2017). Brandwatch Peer Index.

Cai, X., Nithyanand, R., and Johnson, R. (2014a). CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 121–130, New York, NY, USA. ACM.

Cai, X., Nithyanand, R., Wang, T., Johnson, R., and Goldberg, I. (2014b). A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 227–238, New York, NY, USA. ACM.

Cai, X., Zhang, X. C., Joshi, B., and Johnson, R. (2012). Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 605–616, New York, NY, USA. ACM.

Cao, Y., Li, S., and Wijmans, E. (2017). (cross-)browser fingerprinting via os and hardware level features. In *NDSS*.

Cheng, H., , Cheng, H., and Avnur, R. (1998). Traffic Analysis of SSL Encrypted Web Browsing.

Cherubin, G. (2017). Bayes, not Naïve: Security Bounds on Website Fingerprinting Defenses. *PoPETs*, pages 215–231.

Cherubin, G., Hayes, J., and Juarez, M. (2017). Website fingerprinting defenses at the application layer. *Proceedings on Privacy Enhancing Technologies*, pages 186–203.

Coull, S. E., Collins, M. P., Wright, C. V., Monrose, F., and Reiter, M. K. (2007). On Web Browsing Privacy in Anonymized NetFlows. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, pages 23:1–23:14, Berkeley, CA, USA. USENIX Association.

Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. (2012). Peek-a-Boo, I Still See You: Why Efficient

Traffic Analysis Countermeasures Fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 332–346, Washington, DC, USA. IEEE Computer Society.

Economist, T. (2012). Very personal finance.

Ejeta, T. G. and Kim, H. J. (2017). Website Fingerprinting Attack on Psiphon and Its Forensic Analysis. In Kraetzer, C., Shi, Y.-Q., Dittmann, J., and Kim, H. J., editors, *Digital Forensics and Watermarking*, pages 42–51, Cham. Springer International Publishing.

European Commission (2018). Data protection.

Gallagher, S. (2014). Chinese government launches man-in-middle attack against icloud [updated]. Ars Technica.

Hayes, J. and Danezis, G. (2016). k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1187–1203, Austin, TX. USENIX Association.

Herrmann, D., Wendolsky, R., and Federrath, H. (2009). Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial NaïVebayes Classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 31–42, New York, NY, USA. ACM.

Husák, M., Čermák, M., Jirsík, T., and Čeleda, P. (2016). HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting. *EURASIP Journal on Information Security*.

Juarez, M., Afroz, S., Acar, G., Diaz, C., and Greenstadt, R. (2014). A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 263–274, New York, NY, USA. ACM.

Juarez, M., Imani, M., Perry, M., Diaz, C., and Wright, M. (2016). Toward an Efficient Website Fingerprinting Defense. In Askoxylakis, I., Ioannidis, S., Katsikas, S., and Meadows, C., editors, *Computer Security – ESORICS 2016*, pages 27–46, Cham. Springer International Publishing.

Kwon, A., AlSabah, M., Lazar, D., Dacier, M., and Devadas, S. (2015). Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 287–302, Washington, D.C. USENIX Association.

Liberatore, M. and Levine, B. N. (2006). Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 255–263, New York, NY, USA. ACM.

Liu, L., Preoiuc-Pietro, D., Riahi, Z., Moghaddam, M. E., and Ungar, L. (2016). Analyzing Personality through Social Media Profile Picture Choice. In *ICWSM*.

Lu, L., Chang, E.-C., and Chan, M. C. (2010). Website Fingerprinting and Identification Using Ordered Feature Sequences. In *Proceedings of the 15th European Conference on Research in Computer Security*, ESORICS'10, pages 199–214, Berlin, Heidelberg. Springer-Verlag.

Luo, X., Zhou, P., Chan, E. W. W., Lee, W., Chang, R. K. C., and Perdisci, R. (2011). HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows. In *In Proc. Network and Distributed Systems Symposium (NDSS). The Internet Society*. 10.1.1.300.1748.

Miller, B., Huang, L., Joseph, A. D., and Tygar, J. D. (2014). I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis. In De Cristofaro, E. and Murdoch, S. J., editors, *Privacy Enhancing Technologies*, pages 143–163, Cham. Springer International Publishing.

Morla, R. (2017). Effect of Pipelining and Multiplexing in Estimating HTTP/2.0 Web Object Sizes. *ArXiv e-prints*.

Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., and Wehrle, K. (2016). Website Fingerprinting at Internet Scale. In *NDSS*.

Panchenko, A., Niessen, L., Zinnen, A., and Engel, T. (2011). Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, pages 103–114, New York, NY, USA. ACM.

Perez, S. (2018). Facebook starts pushing its data tracking onavo vpn within its main mobile app.

Rao, A., Spasojevic, N., Li, Z., and Dsouza, T. (2015). Klout Score: Measuring Influence Across Multiple Social Networks. *Conference: 2015 IEEE International Conference on Big Data (Big Data)*, pages 2282–2289.

Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., and Joosen, W. (2017). Automated Feature Extraction for Website Fingerprinting through Deep Learning. (to appear).

Sun, Q., Simon, D. R., Wang, Y.-M., Russell, W., Padmanabhan, V. N., and Qiu, L. (2002). Statistical Identification of Encrypted Web Browsing Traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, pages 19–, Washington, DC, USA. IEEE Computer Society.

Wang, T. (2015). Website Fingerprinting: Attacks and Defenses (Doctoral dissertation). University of Waterloo, Canada.

Wang, T. and Goldberg, I. (2017). Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1375–1390, Vancouver, BC. USENIX Association.

WiFi4EU (2016). Free wi-fi for europeans.

Wijnants, M., Marx, R., Quax, P., and Lamotte, W. (2018). HTTP/2 Prioritization and its Impact on Web Performance. In *The Web Conference*, WWW 2018.

Wright, C. V., Coull, S. E., and Monrose, F. (2009). Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *In Proceedings of the 16th Network and Distributed Security Symposium*, pages 237–250. IEEE.