Authors: I. Goldberg
         University of Waterloo
         T. Wang   C.A. Wood
         HK University of Science and Technology   Apple, Inc.
                Network-Based Website Fingerprinting

Abstract

   The IETF is well on its way to protecting connection metadata with
   protocols such as DNS-over-TLS and DNS-over-HTTPS, and work-in-
   progress towards encrypting the TLS SNI. However, more work is
   needed to protect traffic metadata, especially in the context of web
   traffic. In this document, we survey Website Fingerprinting attacks,
   which are a class of attacks that use machine learning techniques to
   attack web privacy, and highlight metadata leaks used by said
   attacks. We also survey proposed mitigations for such leakage and
   discuss their applicability to IETF protocols such as TLS, QUIC, and
   HTTP. We endeavor to show that Website Fingerprinting attacks are a
   serious problem that affect all Internet users, and we pose open
   problems and directions for future research in this area.

Note to Readers

   Source for this draft and an issue tracker can be found at https://
   github.com/chris-wood/ietf-fingerprinting.

Status of This Memo

Table of Contents

1. Introduction

Internet protocols such as TLS 1.3 [RFC8446] and QUIC [I-D.ietf-quic-transport] bring substantial improvements to end-users. The IETF engineered these with security and privacy in mind by encrypting more protocol messages using modern cryptographic primitives and algorithms, and engineering against flaws found in previous protocols, yielding several desirable security properties, including: forward-secure session key secrecy, downgrade protection, key compromise impersonation resistance, and protection of endpoint identities. Combined, these two protocols are set to protect a significant amount of Internet data. However, significant metadata leaks still exist for users of these protocols. Examples include plaintext TLS SNI and application-specific extensions (ALPN), as well as DNS queries. This information can be used by a passive attacker to learn information about the contents of an otherwise

encrypted network connection. Recently, such information has also been studied as a means of building unique user profiles [li2018can]. It has also been used to build flow classifiers that aid network management [foremski2014dns].

In the context of Tor, a popular low-latency anonymity network, a common class of attacks that use metadata for such inference is called Website Fingerprinting (WF). These attacks use machine learning techniques built with features extracted from metadata such as traffic patterns to attack web (browsing) privacy. Miller et al. [miller2014know] show how these attacks can be applied to web browsing traffic protected with HTTPS to reveal private information about users. Pironti et al. [pironti2012identifying] use similar attacks based on data sizes to identify individual social media clients using encrypted connections. Fingerprinting attacks using encrypted traffic analysis are also applicable to encrypted media streams, such as Netflix videos. (See work from Reed et al. [reed2017identifying] and Schuster et al. [schuster2017beauty] for examples of these attacks.) WF attacks have also been applied to other IETF protocols such as encrypted DNS, including dnscrypt, DNS-over-TLS, and DNS-over-HTTPS [siby2018dns][shulman2014pretty]. In the past, they have also been conducted remotely [gong2010fingerprinting], using buffer-based side channels in a victim's home router.

Protocols such as DNS-over-TLS and DNS-over-HTTPS [RFC8484], and work-in-progress towards encrypting the TLS SNI extension [I-D.ietf-tls-esni], help minimize metadata sent in cleartext on the wire. However, regardless of protocol and even network-layer fingerprinting mitigations, application layer specifics, e.g., web page sizes and client request patterns, reveal a noticeable amount of information to attackers. We argue that much more work is needed to protect encrypted connection metadata, especially in the context of web traffic.

In this document, we describe WF attacks in the context of IETF protocols such as TLS and QUIC. We survey WF attacks and highlight metadata features and classification techniques used to conduct said attacks. We also describe proposed mitigations for these attacks and discuss their applicability to IETF protocols. We conclude with a discussion of open problems and directions for future research and advocate for more work in this area.

2. Background

In this section we review how most secure Internet connections are made today. We omit custom configurations such as those using VPNs and proxies since they do not represent the common case for most Internet users. The following steps briefly describe the sequence of

events that normally occur when a web client, e.g., browser, curl, etc., connects to a website and obtains some resource. First an unencrypted DNS query is sent to an untrusted DNS recursive resolver to resolve a name to an IP address. Upon receipt, clients then open a TCP and TLS connection to the destination address. During this stage, metadata such as the TLS SNI and ALPN values are sent in cleartext. The SNI is used to denote the destination application or endpoint to which clients want to connect. Servers use this for several purposes, including selecting an appropriate certificate (one with the SNI name in the SubjectAlternativeName list) or routing to a different backend terminator. ALPN values are used to negotiate which application-layer protocol will be used on top of the TLS connection. Common values include "http/1.1", "h2", and (soon) "h3". Upon connection, clients then send HTTP messages to obtain the desired resource.

Connections look different (on the wire) with TLS 1.3, encrypted DNS via DNS-over-TLS or DNS-over-HTTPS, and encrypted SNI. DNS queries are encrypted to a (trusted) recursive resolver and TLS metadata such as SNI are encrypted in transit to the terminator. Despite the reduction in cleartext metadata sent over the wire, there still remains several sources of information that an adversary may use for malicious purposes, including: size and timing of DNS queries and responses, size and timing or application traffic, and connection attempts induced while loading a web resource, e.g., Javascript files. So while technologies such as Encrypted SNI, DoT, and DoH help protect some metadata, they are not complete solutions to the larger problem. In the following section, we discuss this overarching problem in detail.

3. Website Fingerprinting

Website Fingerprinting (WF) is a class of attacks that exploit metadata leakage to attack end-user privacy on the Internet. In the WF threat model, Adv is assumed to be a passive and local attacker. Local means that Adv can associate traffic with a given client. Examples include proxies to which clients directly connect. Passive means that Adv can only view traffic in transit. It cannot add, drop, or otherwise modify packets between the victim client and server(s). Use of reliable and encrypted transport protocols such as TLS limit on-path attackers to eavesdropping on encrypted packets. (In QUIC, however, reordering packets is possible.)

Traffic features used for classification include properties such as packet size, timing, direction, interarrival times, and burstiness, among many others [wang2016website]. Normally, features are restricted to those which are extractable as a passive eavesdropper, and not those which are viewable by modifying client or server behavior. Specifically, this means that attacks such as CRIME

[CRIME] and TIME [TIME], which rely on an attacker abusing TLS-layer compression to leak contents of an encrypted connection, are out of scope.

Website Fingerprinting attacks have evolved over the years through three phases: (1) Closed-world WF on SSL/TLS, (2) Closed-world WF on Tor, and (3) Open-world WF on Tor.

1. In the closed-world model, clients are assumed to only visit a small set of pages monitored by Adv. This is less realistic but easier to analyze than the open-world model discussed below, and so the earliest results achieved success on SSL/TLS in this model. (For a realistic attack, Adv would need to monitor every possible page of interest to each client, which is impractical.) Attacks against proxy-based privacy technologies such as VPNs and SSH tunneling, which has almost no effect on the network, falls under this category as well.

2. Tor, an anonymity network built on onion routing, is harder to attack than SSL for several reasons; successful results on Tor thus came later. First, Tor pads all cells (Tor's application-layer datagrams) to the same constant size, removing unique packet lengths as a powerful feature for the attacker. Second, Tor imposes random network conditions upon the client due to random selection of proxies, so packet sequences are less likely to be consistent.

3. In the open-world model, Adv wishes to learn whenever a victim client visits one of a select number of monitored pages [wang2016website]. Adversaries train classifiers in this model using monitored and non-monitored websites of their choosing. By definition, Adv cannot train using client-chosen pages. Clients then visit pages at will and Adv attempts to learn whenever a monitored page is visited, if any are at all. This is a realistic model capturing the fact that the set of pages any attacker would be interested in must necessarily be a small subset of the set of all pages. As this is a harder model to attack, successful results on this model came later.

4. **Attacks**

   1. Closed-world WF on TLS: WF attacks date back to applications on SSL first inspired by Wagner and Schneier [wagner1996analysis], in which the authors observed that packet lengths reveal information about the underlying data. Subsequent attacks carried out by Cheng et al. [cheng1998traffic], Sun et al. [sun2002statistical], and Hintz [hintz2002fingerprinting] continued to show access. These attacks assume Adv has

knowledge of the target resource length(s), which is not always possible with techniques such as padding.

Bissias et al. [bissias2005privacy] use cross correlation of inter-packet times in one second time windows as an WF attack. Danezis [danezis2009traffic] model websites using a Hidden Markov Model (HMM) and use it, along with TLS traffic traces revealing only approximate lengths, to identify requested resources on a page. Their results vary the amount of information available to an adversary when building the HMM. Even in cases where resource popularity is omitted, which reflects the case where an adversary scrapes static websites, resource recall was high (86\%). Liberatore and Levine [liberatore2006inferring] proposed two WF attacks using the Jaccard coefficient and the Naive Bayes classifier. Herrmann et al. [herrmann2009website] extended the work of Liberatore and Levine with a multinomial Naive Bayes classifier computed using three input frequency transformations. Results yielded higher accuracy than that of Liberatore and Levine. Herrmann's attack is the best in this category, but the authors assume packets which do not fill a MTU represent packet trailers. Therefore, uniqueness is only accurate modulo the MTU. Efficacy is limited if endpoints pad packets to the MTU or another fixed length. Modern protocols such as HTTP/2, QUIC, and TLS 1.3 all provide some form of application-controlled padding. (Note: These attacks are not successful on Tor.)

1. Closed-world WF on Tor: Shmatikov and Wang [shmatikov2006timing] presented a WF attack that exploits cross correlation of arrival packet counts in one second time windows. Lu et al. [lu2010website] developed a classifier based on the Levenshtein distance between ingress and egress packet lengths extracted from packet sequences. Distance is computed between strings of ingress and egress packet lengths. The training packet sequence with the closest distance to the testing packet sequence is deemed the match. Dyer et al. [dyer2012peek] used a Naive Bayes classifier trained with a reduced set of features, including total response transmission time, length of packets (in each direction), and burst lengths. (Wang [wang2016website] notes that measuring burst lengths in Tor is difficult given the presence of SENDME cells for flow control.) This approach did not yield any measurable improvements over the SVM classifier from Panchenko et al. Cai et al. [cai2012touching] extend the work of Lu et al. by adding transpositions to the Levenshtein distance computation and normalizing the result, yielding what the authors refer to as the Optimal String Alignment Distance (OSAD). Before feature extraction, the authors round TCP packet lengths to the nearest multiple of 600B as an estimate of the number of Tor cells.

Wang et al. [wang2013improved] tuned the OSAD-based attack to improve its accuracy. Specific changes include use of Tor cells instead of TCP packets for packet and burst lengths, as well as heuristics to remove SENDME cells (those not carrying application data) from flows to recover true burst lengths. The authors also modified the distance computation by removing substitutions, increasing the weight for egress packets, and varying the transposition cost across the packet sequence (large weights at the beginning of a trace, and smaller weights near the end, where variations are expected across repeated page loads.) Wang et al. also developed an alternate classifier with lower accuracy yet superior performance (quadratic to linear time complexity). It works by minimizing the sum of two costs: sequence transpositions and sequence deletions or insertions. These two costs are computed separately, in contrast to the first approach which computes them simultaneously.

Hayes et al. [hayes2016k] developed an attack called k-fingerprinting, which uses a k-NN classifier with features ranked by random decision forests. Their feature set includes timing information, e.g., statistics on packets per second, among the higher ranked features. (Higher ranked features have more weight in the classification phase.) Yan et al. [yan2018feature] used similar (manually curated) features with a CNN-based classifier. Time-based features were among the more effective features identified. Rahman et al. [rahman2019tik] improved time-based features by focusing on bursts, e.g., burst length, variance, inter-burst delay, etc., rather than more granular per-packet statistics. (The latter tend to vary for inconsistencies across packet traces for websites.) This improved accuracy of existing Deep Learning attacks from Sirinam et al. [sirinam2018deep], especially when coupled with packet direction information.

1. Open-world WF on Tor and TLS: Panchenko et al. [panchenko2011website] were the first to use a support vector machine (SVM) classifier trained with web domain-specific features, such as HTML document sizes, as well as packet lengths. Wang et al. [wang2014effective] also developed an attack using a k-Nearest Neighbors (k-NN) classifier, which is a supervised machine learning algorithm, targeting the open world setting. The classifier extracts a large number of features from packet sequences, including raw (ingress and egress) packet counts, unique packet lengths, direction, burst lengths, and inter-packet times, among others. (There are 4226 features in total.) The k-NN distance metric is computed as the sum of weighted feature differences.

Kota et al. [abe2016fingerprinting] were the first to use Deep Learning (DL) methods based on Stacked Denoising Autoencoders for WF

attacks. (Autoencoders reduce feature input dimensions when stacked.) Kota et al. form input vectors from Tor cell directions (+1 or -1). They use no other features. Using a (small) data set from Wang [wang2016website], the classifier achieves a 86% true positive rate and 2% false positive rate in the open world model. Rimmer et al. [rimmer2018automated] applied DL for automated feature generation and classifier construction. Trained with 2,500 traces per website, their system achieves 96.3% accuracy in the open world model. Recently, Bhat et al. [bhat2018var], Oh et al. [oh2017pfp], and Sirinam et al. [sirinam2018deep] used Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs) for WF attacks. Results from Sirinam et al. show the best results - 98% on Tor without recent defenses (in Section {{defenses}}) - while performing favorably when select defenses are used for both open and closed world models.

Yan et al. [yan2018feature] studied manual high-information feature extraction from packet traces. They "exhaustively" examined different levels of features, including packet, burst, TCP, port, and IP address, summing to 35,683 in total, and distilled them into a diverse set of uncorrelated features for eight different communication scenarios. Rahman [rahman2018using] studied the utility of features derived from packet interarrival times, including: median interarrival time (per burst), burst packet arrival time variance, cross-burst interarrival median differences, and others. Using a CNN, results show that these features yield a non-negligible increase in WF attack accuracy.

5. **Base Rate Fallacy**

For all WF attacks, one limitation worth highlighting is the base rate fallacy. This can be summarized as follows: highly accurate classifiers with a reliable false positive rate (FPR) decrease in efficacy as the world size increases. Juarez et al. [juarez2014critical] studied its impact by measuring the Bayesian detection rate (BDR) in comparison to the FPR as a function of world size. As the world size increases, the BDR approaches 0 while the FPR remains stable, meaning that the probability of incorrect classifier results increase as well. Juarez et al. partially address the base rate fallacy problem by adding a confirmation step to their classifier. Another problem is that web content is (increasingly) dynamic. Most WF attacks, especially those in closed world models, assume that traces are static. However, Juarez et al. [juarez2014critical] show this is not the case even for "simple" pages such as google.com. Thus, due to the base fallacy rate and dynamic nature of content, classifiers require continual retraining in order to ensure accuracy.

6. Defenses

There are at least two types of WF defenses: traffic shaping or
morphing algorithms, and traffic splitting algorithms. This section
describes and illustrates examples of both.

6.1. Traffic Morphing

WF defenses are deterministic or randomized algorithms that take as
input application data or packet sequences and return modified
application data or packet sequences. Viable defenses seek to
minimize the transformation cost and maximum (theoretical and
perfect) attacker accuracy. Naive defenses such as sending a
constant stream of (possibly random) bytes between client and server
may be effective though clearly not viable from a cost perspective.
Relevant cost metrics include bandwidth overhead, added time or
latency (and its impact on related metrics such as page load time),
and even CPU cost, though the latter is often ignored in favor of
the former two. Wang [wang2016website] describe defenses as either
limited or general. A limited defense is one which only helps
mitigate specific WF attacks by transforming packets in a way to
obviate a particular (set of) feature(s) used by said attacks. In
contrast, general defenses help mitigate a variety of attacks.

Several general defenses have been proposed, including BuFLO
[dyer2012peek], which pads packets to a fixed length of 1500B (the
normal MTU) and schedules packets for transmission at fixed period
intervals (and sends fake data if nothing is yet available). Tamaraw
[wang2014comparing] is an improvement over BuFLO that uses two
different fixed lengths for packet transmission, rather than one, to
save on bandwidth overhead. Tamaraw also uses two different
scheduling rates for ingress and egress packets. The authors chose
to make the ingress packet period smaller than the egress packet
period since HTTP responses are often larger in size and count - if
HTTP Push is used - than requests. While provably correct, both
BuFLO and Tamaraw limit the rate at which clients send traffic, and
requires all clients to send at a uniform rate. Both requirements
therefore make it difficult to apply as a generic defense in IETF
protocols.

Wang et al. also developed Supersequence [wang2016website], which
attempts to approximate a bandwidth-optimal deterministic defense.
This is done by casting the padding and flow control problem as the
shortest common subsequence (SCS) of the transformed packet trace.
Supersequence approximates the solution by learning the optimal
packet scheduling rate; it uses the same padding scheme as Tamaraw.

Walkie-Talkie [wang2015walkie] is a collection of mechanisms for WF
defense. It includes running the client (browser) in half-duplex

mode to batch requests and responses together, as well as randomly padding traffic so as to mimic traffic of benign websites. It assumes knowledge of traffic patterns for benign websites, which can be information learned over time or provided by a cooperating peer. Goldberg and Wang also propose a "randomized" variant that pads real bursts of requests and generates random request bursts according to a uniform distribution. The half-duplex mode could be implemented as an extension to a protocol such as HTTP/2, QUIC, or TLS.

Many limited defenses have also been proposed. We list prominent works below.

  *Shmatikov and Wang [shmatikov2006timing] developed adaptive padding which adds packets to mask inter-packet times. (This mechanism does not ever delay application data being sent, in contrast to other padding mechanisms such as BuFLO; see below.) Juarez et al. [juarez2015wtf]}[juarez2016toward] also created a WF defense based on adaptive padding called WTF-PAD. This variant uses application data and "gap" distribution to generate padding for delays. Specifically, when not sending application data, senders use the gap distribution to drive fake packet transmission. WTF-PAD can be run by a single endpoint, though it is assumed that both client and server participate. As mentioned above, protocols such as HTTP/2, QUIC, and TLS 1.3 offer a mechanism by which applications can send padding. WTF-PAD could therefore be implemented as an extension to any of these protocols, either by applications supplying padding distributions or the system learning them over time.

  *In the context of HTTP, Danezis [danezis2009traffic] proposed padding: URLs, content, and even HTML document structures to mask application data lengths.

  *Wright et al. [wright2009traffic] developed traffic morphing, which pads packets in such a way so as to make the sequence from one page have characteristics of another (non-monitored or benign) page. This technique requires application-specific knowledge about benign pages and is therefore best implemented outside of the transport layer.

  *Nithyanand et al. [nithyanand2014glove] developed a mechanism called Glove, wherein traces were first clustered and then morphed (via dummy insertion, packet merging, splitting, and delaying) to look indistinguishable within clusters. When used to protect the Alexa top 500 domains, Glove performs well with respect to bandwidth overhead when compared to BuFLO and CS-BuFLO. Varying the cluster size can tune Glove's bandwidth overhead.

*Pironti et al. [pironti2012identifying] developed a TLS-based
 fragmentation and padding scheme designed to hide the length of
 application data within a certain range with record padding. The
 mechanism works by iteratively splitting application data into
 variable sized segments. Applications can guide the range of
 viable lengths provided such information is available.

*Luo et al. [luo2011httpos] created HTTPS with Obfuscation
 (HTTPOS), which is a client-side mechanism for obfuscating HTTP
 traffic. It uses the HTTP Range method to receive resources in
 chunks, TCP MSS to limit the size of individual chunks, and
 advertised window size to control the flow of chunks in
 transmission.

*Panchenko et al. [panchenko2011website] developed Decoy, which is
 a simple mechanism that loads a benign page alongside a real
 page. This seeks to mask the real page load by properties of the
 "decoy" page. As with morphing, this defense requires
 application-specific knowledge about benign pages and is best
 implemented outside of the transport layer.

*The Tor project implemented HTTP pipelining
 [perry2011experimental], which bundles egress HTTP/1.1 requests
 into batches of varying sizes with random orders. Batching
 requests to mask request and response sizes could be made easier
 with HTTP/2 [RFC7540], HTTP/3, and QUIC, since these protocol
 naturally support multiplexing. However, pipelining and batching
 may necessarily introduce latency delays that negatively impact
 the user experience.

*Cherubin et al. [cherubin2017website] design two application-
 layer defenses called Application Layer Padding Concerns
 Adversaries (ALPaCA) and Lightweight application-Layer
 Masquerading Add-on (LLaMA). ALPaCA is a server-side defense that
 pads first-party content (deterministically or probabilistically)
 according to a known distribution. (Deterministic padding similar
 to Tamaraw performs worse than probabilistic padding.) LLaMA is
 similar to randomized pipelining, yet differs in that requests
 are also delayed (if necessary) and spurious requests are
 generated according to some probability distribution.
 Comparatively, ALPaCA yields a greater reduction in WF attack
 accuracy than LLaMA.

*Lu et al. [lu2018dynaflow] designed DynaFlow, which is a defense
 that dynamically adjusts traffic flows using a combination of
 burst pattern morphing, constant traffic flow with flexible
 intervals, and burst padding. DynaFlow overhead is 40% less than
 that of Tamaraw and was shown to have similar benefits.

*Rahman [rahman18gan] uses generative adversarial networks (GANs) to modify candidate burst properties of packet traces, i.e., by inserting dummy packets, such that they appear indistinguishable from other traces. Normally, the generator component in a GAN uses random noise to produce information that matches a target data distribution as classified by the discriminator component. Rahman uses a modified GAN architecture wherein the generator produces padding (dummy packets) for input data such that the discriminator cannot distinguish it from noise, or a desired burst packet sequence. Preliminary results with the GAN trained and tested on defended traffic, i.e., traffic already subject to some form of WF defense, show a 9% increase in bandwidth and 15% decrease in attack accuracy (from 98% to 85% in a closed world setting).

*Imani et al. [imanimockingbird] developed Mockingbird, a defense built on using generated adversarial examples, i.e., dummy traffic designed to disrupt classifier behavior, aimed towards model misclassification. When run on classifiers trained without adversarial examples, Mockingbird reduced state-of-the-art DF attacks and CUMUL attacks from [panchenko2016website] from 98% to 3% and 92% to 31%, respectively. Conversely, classifiers trained and hardened with adversarial examples only reduce attack accuracy from 95% to between 25-57%, respectively. Classification results for half-duplex traces, i.e., those in which traffic flows in half-duplex mode, are lower. Mockingbird's bandwidth overhead is tunable based on parameters that control the internal traffic shaping algorithm.

*Gong et al. developed [gong2020zero] is a lightweight defense that does not delay any packets, minimizing its effect on user experience. Instead of adding packets during a packet trace in order to obfuscate which page it came from, GLUE adds packets between packet traces (during user think time/downtime) to merge them together, creating a seamless sequence of packets covering multiple page loads. Attackers are unable to train classifiers for multiple contiguous traces and also unable to identify individual page traces from the sequence. This is in part because the GLUE used is itself a real packet trace, thwarting attacker classification. GLUE also adds extra noise packets ("FRONT") in the first trace as it is vulnerable otherwise.

6.2. Traffic Splitting

Traffic splitting is a type of defense wherein application data is sent over multiple, disjoint network paths. Multipath TCP (MPTCP) is one type of "traffic splitting" protocol, wherein an endpoint may send TCP segments for a single connection over multiple interfaces. This is commonly done for multi-homed devices, such as mobile

devices with cellular and WiFi or wired network connections. Traffic splitting assumes that guided traffic distribution reduces information available to an adversary, and thereby decreases the success probability of WF attacks. Traffic splitting defenses differ in the algorithm used for traffic distribution.

Henri et al. [henri2020protecting] studied several traffic splitting algorithms, including: weighted and non-weighted round-robin path splitting, incoming and outgoing path split, fixed-probability splitting, and variants of per-connection uniform probability splitting. The best results came from a per-connection path splitting variant where the maximum number of packets sent on any given path was limited by a random variable chosen from a geometric distribution. (Once this limit was reached, a new path was selected uniformly at random.) De la Cadena et al. [de2019poster] also study path splitting algorithmss. They conclude that a weighted random path selection algorithm works best. (The authors do not give specifics of path weight probability derivation.)

7. Open Problems and Directions

To date, WF attacks target clients running over Tor or some other anonymizing service, meaning that WF attacks are likely more accurate on normal TLS-protected connections. Moreover, attacks normally assume clients use HTTP/1.1 with parallel connections for parallel resource fetches. In recent years, however, protocols such as SPDY, HTTP/2, and QUIC with built-in padding support and multiplexed stream-based connections should make existing attacks more difficult to carry out. That said, it is unclear how exactly these protocol design trends will impact WF attacks. A non-exhaustive list of questions that warrant further research are below:

1. How does connection coalescing and consolidation affect WF attacks? Technologies such as DNS-over-HTTPS and ESNI favor architectures wherein a single network or connection can serve multiple origins or resources. With connection coalescing, traffic for multiple resources is sent on the same connection, thereby adding effects similar to that of the Decoy defense mechanism described in Section 6

2. To what extent does protocol multiplexing increase WF attack difficulty? Using a single connection with multiple streams to avoid HoL blocking saves on connection startup and bandwidth costs while simultaneously mixing information from multiple requests and resources on the same connection.

3. How can protocol features such as HTTP Push be used to improve WF defense efficacy? Defenses without cooperative peer support

often induce suboptimal bandwidth or latency costs. If both endpoints of a connection participate in the defense, even proactively with Push, perhaps this could be improved.

4. Can connection bootstrapping techniques such as those used by ESNI be used to distribute WF defense information? One possible approach is to distribute client padding profiles derived from CDN knowledge of serviced resources.

5. How can clients build, use, and possibly share WF defense information to benefit others?

6. How can applications package websites and subresources in such a way that limits unique information? For example, websites link to third party resources in an ad-hoc fashion, causing the subsequent trace of browser fetches to possibly uniquely identify the website.

Research into the above questions will help the IETF community better understand the extent to which WF attacks are a problem for Internet users in general.

It is worth mentioning that traffic-based WF attacks may not be required to achieve the desired goal of learning a connection's destination. Network connections by nature reveal information about endpoint behavior. The relationship between network address and domains, especially when stable and unique, are a strong signal for website fingerprinting. Trevisan et al. [trevisan2016towards] explored use of this signal as a reliable mechanism for website fingerprinting. They find that most major services (domains) have clearly associated IP address(es), though these addresses may change over time. Jiang et al. [jiang2007lightweight] and Tammaro et al. [tammaro2012exploiting] also previously came to the same conclusion. Address-based website fingerprinting was also explored by Patil and Borisov [patil2019can], wherein they showed that addresses, especially when grouped together as part of a single web page load, leak a substantial amount of information about the corresponding domain. Thus, in general, classifiers that rely solely on network addresses may be used to aid website fingerprinting attacks.

8. **Protocol Design Considerations**

New protocols such as TLS 1.3 and QUIC are designed with privacy-protections in mind. TLS 1.3, for example, supports record-layer padding [RFC8446], although it is not used widely in practice. Despite this, TLS connections still leak metadata, including negotiatied ciphersuites. (See [fordTLSMetadata] for a discussion of this issue.) QUIC is more aggressive in its use of encryption as both a mitigation for middlebox ossificatiion and privacy

enhancement. IPsec Traffic Flow Confidentiality [RFC4303] and Traffic Flow Security [I-D.ietf-ipsecme-iptfs] are two mechanisms by which endpoints can ESP datagrams to mask size metadata.

Future protocols should follow these trends when possible to remove unnecessary metadata from the network.

9. Security Considerations

This document surveys security and privacy attacks and defenses on encrypted TLS connections. It does not introduce, specify, or recommend any particular mitigation to the aforementioned attacks.

10. IANA Considerations

This document makes no IANA requests.

11. Informative References

[abe2016fingerprinting] "Fingerprinting attack on tor anonymity using deep learning", Asia-Pacific Advanced Network, 2016 , n.d..

[backes2013preventing] "Preventing Side-Channel Leaks in Web Traffic -- A Formal Approach", NDSS, 2013 , n.d..

[bhat2018var] "Var-CNN and DynaFlow -- Improved Attacks and Defenses for Website Fingerprinting", arXiv preprint arXiv: 1802.10215 , n.d..

[bissias2005privacy] "Privacy vulnerabilities in encrypted HTTP streams", International Workshop on Privacy Enhancing Technologies, 2005 , n.d..

[cai2012touching] "Touching from a distance -- Website fingerprinting attacks and defenses", ACM conference on Computer and communications security, 2012 , n.d..

[cheng1998traffic] "Traffic analysis of SSL encrypted web browsing", n.d..

[cherubin2017website] "Website fingerprinting defenses at the application layer", Privacy Enhancing Technologies, 2017 , n.d..

[coull2007web] "On Web Browsing Privacy in Anonymized NetFlows", USENIX Security Symposium , n.d..

[CRIME]   "The CRIME Attack", n.d., <https://www.ekoparty.org/ archive/2012/CRIME_ekoparty2012.pdf>.

[danezis2009traffic]
              "Traffic Analysis of the HTTP Protocol over
              TLS", 2009 , n.d..

[de2019poster] "Traffic Splitting to Counter Website
              Fingerprinting", 2019, <https://dl.acm.org/doi/
              10.1145/3319535.3363249>.

[dyer2012peek] "Peek-a-boo, i still see you -- Why efficient traffic
              analysis countermeasures fail", IEEE Symposium on
              Security and Privacy, 2012 , n.d..

[fordTLSMetadata] "Metadata Protection Considerations for TLS
              Present and Future", n.d., <http://bford.info/pub/net/
              tlsmeta.pdf>.

[foremski2014dns] "DNS-Class -- immediate classification of IP flows
              using DNS", International Journal of Network Management ,
              n.d..

[gong2010fingerprinting] "Fingerprinting websites using remote
              traffic analysis", Proceedings of the 17th ACM conference
              on Computer and communications security , n.d..

[gong2020zero] "Zero-delay Lightweight Defenses against Website
              Fingerprinting", n.d., <https://www.usenix.org/system/
              files/sec20summer_gong_prepub.pdf>.

[hayes2016k] "k-fingerprinting -- A Robust Scalable Website
              Fingerprinting Technique", USENIX Security Symposium,
              2016 , n.d..

[henri2020protecting] "Protecting against Website Fingerprinting
              with Multihoming", 2020, <https://petsymposium.org/2020/
              files/papers/issue2/popets-2020-0019.pdf>.

[herrmann2009website] "Website fingerprinting -- attacking popular
              privacy enhancing technologies with the multinomial
              naive-bayes classifier", ACM workshop on Cloud computing
              security, 2009 , n.d..

[hintz2002fingerprinting] "Fingerprinting websites using traffic
              analysis", International Workshop on Privacy Enhancing
              Technologies, 2002 , n.d..

[I-D.ietf-ipsecme-iptfs] Hopps, C., "IP Traffic Flow Security", Work
              in Progress, Internet-Draft, draft-ietf-ipsecme-iptfs-01,
              March 2, 2020, <http://www.ietf.org/internet-drafts/
              draft-ietf-ipsecme-iptfs-01.txt>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-29, June 9, 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-29.txt>.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-07, June 1, 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-esni-07.txt>.

[imanimockingbird] "Mockingbird -- Defending Against Deep-Learning-Based Website Fingerprinting Attacks with Adversarial Traces", n.d., <https://arxiv.org/pdf/1902.06626.pdf>.

[jiang2007lightweight] "Lightweight application classification for network management", SIGCOMM workshop on Internet network management, 2007 , n.d..

[juarez2014critical] "A critical evaluation of website fingerprinting attacks", ACM SIGSAC Conference on Computer and Communications Security, 2014 , n.d..

[juarez2015wtf] "WTF-PAD -- toward an efficient website fingerprinting defense for tor", CoRR, abs/1512.00524 , n.d., <https://pdfs.semanticscholar.org/0f54/4d0845cb9f317722759dc49e1493ef30d83d.pdf>.

[juarez2016toward] "Toward an efficient website fingerprinting defense", European Symposium on Research in Computer Security, 2016 , n.d..

[li2018can] "Can We Learn What People Are Doing from Raw DNS Queries?", IEEE INFOCOM 2018-IEEE Conference on Computer Communications , n.d..

[liberatore2006inferring] "Inferring the source of encrypted HTTP connections", ACM Conference on Computer and Communications Security, 2006 , n.d..

[lu2010website] "Website fingerprinting and identification using ordered feature sequences", European Symposium on Research in Computer Security, 2010 , n.d..

[lu2018dynaflow] "DynaFlow -- An Efficient Website Fingerprinting Defense Based on Dynamically-Adjusting Flows", Workshop on Privacy in the Electronic Society, 2018 , n.d..

[luo2011httpos]
          "HTTPOS -- Sealing Information Leaks with Browser-
          side Obfuscation of Encrypted Flows", NDSS, 2011 , n.d..

[miller2014know] "I know why you went to the clinic -- Risks and
          realization of https traffic analysis", International
          Symposium on Privacy Enhancing Technologies Symposium,
          2014 , n.d..

[nithyanand2014glove] "Glove -- A bespoke website fingerprinting
          defense", Proceedings of the 13th Workshop on Privacy in
          the Electronic Society , n.d..

[oh2017pfp] "p-FP -- Extraction, Classification, and Prediction of
          Website Fingerprints with Deep Learning", n.d..

[panchenko2011website] "Website fingerprinting in onion routing
          based anonymization networks", ACM workshop on Privacy in
          the electronic society, 2011 , n.d..

[panchenko2016website] "Website Fingerprinting at Internet Scale",
          n.d., <https://www.freehaven.net/anonbib/cache/
          fingerprinting-ndss2016.pdf>.

[patil2019can] "What can you learn from an IP?", n.d., <https://
          irtf.org/anrw/2019/anrw2019-final44-acmpaginated.pdf>.

[perry2011experimental] "Experimental defense for website traffic
          fingerprinting", n.d., <https://blog.torproject.org/
          experimental-defense-website-traffic-fingerprinting>.

[pironti2012identifying] "Identifying website users by TLS traffic
          analysis -- New attacks and effective countermeasures",
          n.d..

[rahman18gan] "Generating Adversarial Packets for Website
          Fingerprinting Defense", n.d., <https://
          www.rahmanmsaidur.com/projects/
          Fall_18_Generating_Adversarial_Packets.pdf>.

[rahman2018using] "Using Packet Timing Information in Website
          Fingerprinting", n.d..

[rahman2019tik] "Tik-Tok -- The Utility of Packet Timing in Website
          Fingerprinting Attacks", n.d., <https://arxiv.org/pdf/
          1902.06421.pdf>.

[reed2017identifying] "Identifying https-protected netflix videos in
          real-time", ACM on Conference on Data and Application
          Security and Privacy, 2017 , n.d..

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC
           4303, DOI 10.17487/RFC4303, December 2005, <https://
           www.rfc-editor.org/info/rfc4303>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext
           Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI
           10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/
           info/rfc7540>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS)
           Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446,
           August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS
           (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018,
           <https://www.rfc-editor.org/info/rfc8484>.

[rimmer2018automated] "Automated website fingerprinting through deep
           learning", Network & Distributed System Security
           Symposium (NDSS), 2018 , n.d..

[schuster2017beauty] "Beauty and the burst -- Remote identification
           of encrypted video streams", USENIX Security, 2017 ,
           n.d..

[shmatikov2006timing] "Timing analysis in low-latency mix networks
           -- Attacks and defenses", European Symposium on Research
           in Computer Security, 2006 , n.d..

[shulman2014pretty] "Pretty bad privacy -- Pitfalls of DNS
           encryption", Workshop on Privacy in the Electronic
           Society, 2014 , n.d..

[siby2018dns] "DNS Privacy not so private -- the traffic analysis
           perspective", n.d..

[sirinam2018deep] "Deep fingerprinting -- Undermining website
           fingerprinting defenses with deep learning", arXiv
           preprint arXiv:1801.02265 , n.d..

[sun2002statistical] "Statistical identification of encrypted web
           browsing traffic", IEEE, 2002 , n.d..

[tammaro2012exploiting] "Exploiting packet-sampling measurements for
           traffic characterization and classification",
           International Journal of Network Management, 2012 , n.d..

[TIME]   "A Perfect CRIME? Only TIME Will Tell", Black Hat Europe
           2013 , n.d..

[trevisan2016towards]
                "Towards web service classification using
          addresses and DNS", Wireless Communications and Mobile
          Computing Conference (IWCMC), 2016 International. IEEE,
          2016 , n.d..

[wagner1996analysis] "Analysis of the SSL 3.0 protocol", USENIX
          Workshop on Electronic Commerce Proceedings, 1996 , n.d..

[wang2013improved] "Improved website fingerprinting on tor",
          Workshop on privacy in the electronic society, 2013 ,
          n.d..

[wang2014comparing] "Comparing website fingerprinting attacks and
          defenses", Technical Report 2013-30, CACR, 2013. , n.d..

[wang2014effective] "Effective Attacks and Provable Defenses for
          Website Fingerprinting", USENIX Security Symposium, 2014
          , n.d..

[wang2015walkie] "Walkie-talkie -- An effective and efficient
          defense against website fingerprinting", n.d..

[wang2016website] "Website fingerprinting -- Attacks and defenses",
          University of Waterloo , n.d..

[wright2009traffic] "Traffic Morphing -- An Efficient Defense
          Against Statistical Traffic Analysis", NDSS, 2009 , n.d..

[yan2018feature] "Feature selection for website fingerprinting",
          Proceedings on Privacy Enhancing Technologies, 2018 ,
          n.d..

Appendix A. Acknowledgements

Authors' Addresses

Ian Goldberg
University of Waterloo

Email: iang@uwaterloo.ca

Tao Wang
HK University of Science and Technology

Email: taow@cse.ust.hk

Christopher A. Wood
Apple, Inc.

Email: cawood@apple.com