Tobias Pulls* and Rasmus Dahlberg

Website Fingerprinting with Website Oracles

Abstract: Website Fingerprinting (WF) attacks are a subset of traffic analysis attacks where a local passive attacker attempts to infer which websites a target victim is visiting over an encrypted tunnel, such as the anonymity network Tor. We introduce the security notion of a *Website Oracle* (WO) that gives a WF attacker the capability to determine whether a particular monitored website was among the websites visited by Tor clients at the time of a victim's trace. Our simulations show that combining a WO with a WF attack—which we refer to as a WF+WO attack—significantly reduces false positives for about half of all website visits and for the vast majority of websites visited over Tor. The measured false positive rate is on the order one false positive per million classified website trace for websites around Alexa rank 10,000. Less popular monitored websites show orders of magnitude lower false positive rates.

We argue that WOs are inherent to the setting of anonymity networks and should be an assumed capability of attackers when assessing WF attacks and defenses. Sources of WOs are abundant and available to a wide range of realistic attackers, e.g., due to the use of DNS, OCSP, and real-time bidding for online advertisement on the Internet, as well as the abundance of middleboxes and access logs. Access to a WO indicates that the evaluation of WF defenses in the open world should focus on the highest possible recall an attacker can achieve. Our simulations show that augmenting the Deep Fingerprinting WF attack by Sirinam *et al.* [60] with access to a WO significantly improves the attack against five state-of-the-art WF defenses, rendering some of them largely ineffective in this new WF+WO setting.

Keywords: website fingerprinting, website oracles, traffic analysis, security model, design

DOI 10.2478/popets-2020-0013

Received 2019-02-28; revised 2019-09-15; accepted 2019-09-16.

1 Introduction

A Website Fingerprinting (WF) attack is a type of traffic analysis attack where an attacker attempts to learn which websites are visited through encrypted network tunnels—such as the low-latency anonymity network Tor [20] or Virtual Private Networks (VPNs)—by analysing the encrypted network traffic [11, 26, 27, 35, 48, 62]. The analysis considers only the size and timing of encrypted packets sent over the network to and from a target client. This makes it possible for attackers that only have the limited *capability* of observing the encrypted network traffic (sometimes referred to as a *local eavesdropper*) to perform WF attacks. Sources of such capabilities include ISPs, routers, network interface cards, WiFi hotspots, and guard relays in the Tor network, among others. Access to encrypted network traffic is typically not well-protected over the Internet because it is already in a form that is considered safe to expose to attackers due to the use of encryption.

The last decade has seen significant work on improved WF attacks (e.g., [8, 25, 60, 70]) and defenses (e.g, [6, 7, 31, 36]) accompanied by an ongoing debate on the real-world impact of these attacks justifying the deployment of defenses or not, in particular surrounding Tor (e.g., [30, 49, 71]). There are significant real-world challenges for an attacker to successfully perform WF attacks, such as the sheer size of the web (about 200 million active websites)[1], detecting the beginning of website loads in encrypted network traces, background traffic, maintaining a realistic and fresh training data set, and dealing with false positives.

Compared to most VPN implementations, Tor has some basic but rather ineffective defenses in place against WF attacks, such as padding packets to a constant size and randomized HTTP request pipelining [8, 20, 70]. Furthermore, Tor recently started implementing a framework for circuit padding machines to make it easier to implement traffic analysis de-

***Corresponding Author: Tobias Pulls:** Karlstad University, E-mail: tobias.pulls@kau.se

Rasmus Dahlberg: Karlstad University, E-mail: rasmus.dahlberg@kau.se

1 Netcraft January 2019 Web Server Survey, https://web.archive.org/web/20190208081915/https://news.netcraft.com/archives/category/web-server-survey/.

fenses[2] based on adaptive padding [31, 59]. However, the unclear real-world impact of WF attacks makes deployment of proposed effective (and often prohibitively costly in terms of bandwidth and/or latency overheads) WF defenses a complicated topic for both researchers to reach consensus on and the Tor Project to decide upon.

1.1 Introducing Website Oracles

In this paper, we introduce the security notion of a *Website Oracle* (WO) that can be used by attackers to augment any WF attack. A WO answers "yes" or "no" to the question "was a particular website visited over Tor at this point in time?". We show through simulation that such a *capability*—access to a WO—greatly reduces the false positive rate for an attacker attempting to fingerprint the majority of websites and website visits through the Tor network. The reduction is to such a great extent that our simulations suggest that false positives are no longer a significant reason for why WF attacks lack real-world impact. This is in particular the case for onion services where the estimated number of websites is a fraction compared to the "regular" web [28].

Our simulations are based on the privacy-preserving network measurement results of the live Tor network in early 2018 by Mani *et al.* [38]. Besides simulating WOs we also identify a significant number of potential sources of WOs that are available to a wide range of attackers, such as nation state actors, advertisement networks (including their customers), and operators of relays in the Tor network. Some particularly practical sources—due to DNS and how onion services are accessed—can be used by anyone with modest computing resources.

We argue that sources of WOs are inherent in Tor due to its design goal of providing *anonymous* and not *unobservable* communication: observable anonymity sets are inherent for anonymity [32, 50, 53], and a WO can be viewed as simply being able to query for membership in the destination/recipient anonymity set (the potential websites visited by a Tor client). The solution to the effectiveness of WF+WO attacks is therefore not to eliminate all sources—that would be impossible without unobservable communication [32, 50, 53]—but to assume that an attacker has WO access when evaluating the effectiveness of WF attacks and defenses, even for weak attackers like local (passive) eavesdroppers.

The introduction of a WO in the setting of WF attacks is similar to how encryption schemes are constructed to be secure in the presence of an attacker with access to *encryption* and *decryption* oracles (chosen plaintext and ciphertext attacks, respectively) [23, 43, 52]. This is motivated by the real-world prevalence of such oracles, and the high impact on security when paired with other weaknesses of the encryption schemes: e.g., Bleichenbacher [4] padding oracle attacks remain an issue in modern cryptosystems today despite being discovered about twenty years ago [40, 56].

1.2 Contributions and Structure

Further background on anonymity, Tor, and WF are presented in Section 2. Section 3 defines a WO and describes two generic constructions for combining a WO with *any* WF attack. Our generic constructions are a type of Classify-Verify method by Stolerman *et al.* [61], first used in the context of WF attacks by Juarez *et al.* [30] and later by Greschbach *et al.* [24]. Section 4 presents a number of sources of WOs that can be used by a wide range of attackers. We focus on practical sources based on DNS and onion service directories in Tor, offering *probabilistic* WOs that anyone can use with modest resources. We describe how we simulate access to a WO throughout the rest of the paper in Section 5, based on Tor network measurement data from Mani *et al.* [38].

Section 6 experimentally evaluates the performance of augmenting the state-of-the-art WF attack Deep Fingerprinting (DF) by Sirinam *et al.* [60] with WO access using one of our generic constructions. We show significantly improved classification performance against unprotected Tor as well as against traces defended with the WF defenses WTF-PAD by Juarez *et al.* [31] and Walkie-Talkie by Wang and Goldberg [72], concluding that the defenses are ineffective in this new setting where an attacker has access to a WO. Further, we also evaluate DF with WO access against Wang *et al.*'s dataset [70] with simulated traces for the constant-rate WF defenses CS-BuFLO and Tamaraw by Cai et al. [6, 7]. Our results show that constant-rate defenses are overall effective defenses but not efficient due to the significant induced overheads. We then evaluate two configurations of the WF defense DynaFlow by Lu *et al.* [36], observing similar effectiveness as CS-BuFLO but at lower overheads approaching that of WTF-PAD and Walkie-Talkie.

2 Tor blog post by Nick Mathewson, https://web.archive.org/web/20190208085701/https://blog.torproject.org/new-release-tor-0401-alpha.

In Section 7 we discuss our results, focusing on the impact on false positives with WO access, how imperfect sources for WOs impact WF+WO attacks, limitations of our work, and possible mitigations. Our simulations indicate that WF defenses should be evaluated against WF attacks based on how they minimise *recall*. We present related work in Section 8, including how WF+WO attacks relate to traffic correlation and confirmation attacks. Section 9 briefly concludes this paper.

2 Background

Here we present background on terminology, the anonymity network Tor, and WF attacks and defenses.

2.1 Anonymity and Unobservability

Anonymity is the state of a subject not being identifiable from an attackers perspective within the *anonymity set* of possible subjects that performed an action such as sending or receiving a message [50]. For an anonymity network, an attacker may not be able to determine who sent a message into the network—providing a sender anonymity set of all possible senders—and conversely, not be able to determine the recipient of a message from the network out of all possible recipients in the recipient anonymity set. Inherent for anonymity is that the subjects in an anonymity set change based on what the attacker observes, e.g., when some subjects send or receive messages [32, 53]. In gist, anonymity is concerned with hiding the *relationship* between a sender and recipient, not its existence.

Unobservability is a strictly stronger notion than anonymity [32, 50, 53]. In addition to anonymity of the relationship between a sender and recipient, unobservability also requires that an attacker (not acting as either the sender or recipient) cannot sufficiently distinguish if there is a sender or recipient or not [50]. Perfect unobservability is therefore the state of an attacker being unable to determine if a sender/recipient should be part of the anonymity set or not.

2.2 Tor

Tor is a low-latency anonymity network for anonymising TCP streams with about eight million daily users, primarily used for anonymous browsing, censorship circumvention, and providing anonymous (onion) services [20, 38]. Because Tor is designed to be usable for low-latency tasks such as web browsing, its threat model and design does not consider powerful attackers, e.g., global passive adversaries that can observe all network traffic on the Internet [18, 20]. However, less powerful attackers such as ISPs and ASes that observe a fraction of network traffic on the Internet are in scope.

Users typically use Tor Browser—a customised version of Mozilla Firefox (bundled with a local relay)—as a client that sends traffic through three *relays* when browsing a website on the regular Internet: a guard, middle, and exit relay. Traffic from the client to the exit is encrypted in multiple layers as part of fixed-size cells such that only the guard relay knows the IP-address of the client and only the exit relay knows the destination website. There are about 7000 public relays at the time of writing, all available in the *consensus* generated periodically by the network. The consensus is public and therefore anyone can trivially determine if traffic is coming from the Tor network by checking if the IP-address is in the consensus. Note that the encrypted network traffic in Tor is exposed to network adversaries as well as relays as it traverses the Internet. Figure 1 depicts the setting just described, highlighting the anonymity sets of users of Tor Browser and the possible destination websites.

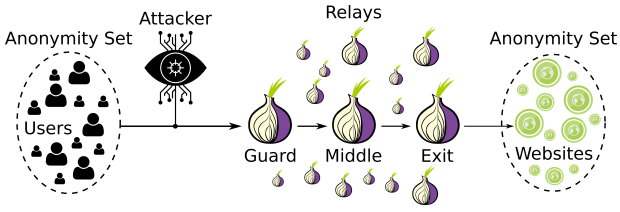


Fig. 1. Using Tor to browse to a website, where an attacker observes the encrypted traffic into the Tor network for a target user, attempting to determine the website the user is visiting.

2.3 Website Fingerprinting

As mentioned in the introduction, attacks that analyse the encrypted network traffic (a trace) between a Tor client and a guard relay with the goal to detect the website a client is visiting are referred to as *website fingerprinting* (WF) attacks. Figure 1 shows the typical location of the attacker, who can also be the guard itself. WF attacks are evaluated in either the *closed* or the *open* world. In the closed world, an attacker *monitors* a number of websites and it is the goal of the attacker

to determine which website out of all the possible monitored websites a target is visiting. The open world is like the closed world with one significant change: the target user may also visit *unmonitored* websites. This means that in the open world the attacker may also classify a trace as unmonitored in addition to monitored, posing a significantly greater challenge for the attacker in a more realistic setting than the closed world. The ratio between monitored and unmonitored traces in a dataset is further a significant challenge for WF attacks when assessing their real-world significance for Tor [30]. Typically, WF attacks are evaluated on the frontpages of web*sites*: web*page* fingerprinting is presumably much more challenging due to the orders of magnitude of more webpages than websites. Unless otherwise stated, we only consider the frontpages of websites in this paper.

2.3.1 Website Fingerprinting Attacks

Prior to WF attacks being considered for use on Tor, they were used against HTTPS [11], web proxies [27, 62], SSH tunnels [35], and VPNs [26]. For Tor, WF attacks are typically based on machine learning and can be categorized based on if they use deep learning or not.

Traditional WF attacks in the literature use manually engineered features extracted from both the size and timing of packets (and/or cells) sent by Tor. State of the art attacks with manually engineered features are Wang-kNN [70], CUMUL [46], and k-FP [25]. For reference, Wang-kNN has 1225 features, CUMUL 104 features, and k-FP 125 features. In terms of accuracy, k-FP appears to have a slight edge over the other two, but all three report over 90% accuracy against significantly sized datasets. As traditional WF attacks progressed, the features more than the type of machine learning method have shown to be vital for the success of attacks, with an emerging consensus on what are important features (e.g., coarse features like number of incoming and outgoing packets) [12, 25, 46].

Deep learning was first used for WF attacks by Abe and Goto in 2016 [2]. Relatively quickly, Rimmer *et al.* reached parity with traditional WF attacks, lending credence to the emerging consensus that the research community had found the most important features for WF [55]. However, recently Sirinam et al. [60] with Deep Fingerprinting (DF) significantly improved on other WF attacks, also on the WTF-PAD and Walkie-Talkie defenses, and is at the time of writing considered state-of-the-art. DF is based on a Convolutional Neural Network (CNN) with a customized architecture for WF. Each packet trace as input to DF is simply a constant size (5000) list of cells (or packets) and their direction (positive for outgoing, negative for incoming), ignoring size and timings. Based on the input, the CNN learns features on its own: we do not know what they are, other than preliminary work indicating that the CNN gives more weight to input early in the trace [39].

The last layer of the CNN-based architecture of DF is a *softmax* function: it assigns (relative) probabilities to each class as the output of classification. These probabilities allow a threshold to be defined for the final classification in the open world, requiring that the probability of the most likely class is above the threshold to classify as a monitored website.

2.3.2 Website Fingerprinting Defenses

WF defenses for Tor modify the timing and number of (fixed-size) cells sent over Tor when a website is visited. The modifications are done by injecting dummy traffic and introducing artificial delays. Defenses can typically be classified as either based on constant-rate traffic or not, where constant rate defenses force all traffic to fit a pre-determined structure, forming *collision sets* for websites where their traffic traces appear identical to an attacker. Non-constant rate defenses simply more-or-less randomly inject dummy traffic and/or artificial delays with the hope of obfuscating the resulting network traces. WF defenses are typically compared in terms of their induced *bandwidth* (BOH) and *time* (TOH) overheads compared to no defense. Further, different WF defenses make more or less realistic and/or practical assumptions, making comparing overheads necessary but not nearly sufficient for reaching conclusions.

We briefly describe WF defenses that we later use to evaluate the impact of attackers performing enhanced WF attacks with access to WOs:

Walkie-Talkie by Wang and Goldberg [72] puts Tor Browser into half duplex mode and pads traffic such that different websites result in the same cell sequences. This creates a collision set between a visited website and a target *decoy website* which results in the same cell sequence with the defense. Their evaluation shows 31% BOH and 34% TOH. Collision sets grow beyond size two at the cost of BOH.

WTF-PAD by Juarez *et al.* [31] is based on the idea of *adaptive padding* [59] where fake padding is injected only when there is no real traffic to send. The defense is simulated on collected packet traces and its design is the foundation of the circuit padding

framework recently implemented in Tor. The simulations report 50-60% BOH and 0% TOH.

CS-BuFLO by Cai *et al.* [6] is a *constant rate* defense where traffic is always sent at a constant rate between a sender and receiver, improving on prior work by Dyer et al. [21]. Their evaluation shows 220-270% BOH and 270-340% TOH.

Tamaraw by Cai *et al.* [7] is another constant rate defense that further improves on CS-BuFLO. In the evaluation by Wang and Goldberg, they report 103% BOH and 140% TOH for Tamaraw [72].

DynaFlow by Lu *et al.* [36] is a *dynamic* constant-rate defense that allows for the defense to adjust its parameters (notably the "inter-packet interval") based on configuration and on the observed traffic. The evaluation shows an overall improvement over Tamaraw when configured to use similar overheads.

The primary downside of defenses like Walkie-Talkie that depend on creating collision sets for websites is that they require up-to-date knowledge of the target website(s) to create collisions with (to know how to morph the traffic traces): this is a significant practical issue for deployment [45, 70, 72]. Constant rate defenses like CS-BuFLO and Tamaraw are easier to deploy but suffer from significant overheads [6, 7]. WTF-PAD is hard to implement both efficiently and effectively in practice due to only being simulated on packet traces as-is and also being vulnerable to attacks like Deep Fingerprinting [31, 60]. While DynaFlow shows great promise, but requires changes at the client (Tor Browser, local relay, or both) and at exit relays to *combine* packets with payloads smaller than Tor's cell size [36]. Without combined packets its advantage in terms of overhead compared to Tamaraw likely shrinks.

2.4 Challenges for WF Attacks in Practice

A number of practical challenges for an attacker performing WF attacks have been highlighted over the years, notably comprehensively so by Mike Perry of the Tor Project [49] and Juarez et al. [30]. Wang and Goldberg have showed that several of the highlighted challenges—such as maintaining a fresh data set and determining when websites are visited—are practical to overcome [71]. What remains are two notably significant challenges: distinguishing between different goals of the attacker and addressing false positives.

For attacker goals when performing WF attacks, an attacker may want to detect website visits with the goal of censoring access to it, to identify all users that visit particular websites, or to identify every single website visited by a target [49]. Clearly, these goals put different constraints on the attacker. For censorship, classification must happen before content is actually allowed to be completely transferred to the victim. For monitoring only a select number of websites the attacker has the most freedom, while detecting all website visits by a victim requires the attacker to have knowledge of all possible websites on the web.

For addressing false positives there are a number of aspects to take into account. First, the web has millions of websites that could be visited by a victim (not the case for onion services [28]), and each website has a significant number of webpages that are often dynamically generated and frequently changed [30, 49]. Secondly, how often victims potentially visit websites that are monitored by an attacker is unknown to the attacker, i.e., the *base rate* of victims are unknown. The base rate leads to even a small false positive rate of a WF attack overwhelming an attacker with orders of magnitude more false positives than true positives, leaving WF attacks impractical for most attacker goals in practice.

3 Website Oracles

We first define a WO and then present two generic constructions for use with WF attacks based on the kind of output the WF attack supports.

3.1 Defining Website Oracles

Definition 1. A website oracle answers true or false to the question "was a particular monitored website w visited over the Tor network at time t?".

A WO considers only web*sites* and not web*pages* for w, but note that even for webpage fingerprinting being able to narrow down the possible websites that webpages belong to through WO access is a significant advantage to an attacker. The time t refers to a *period of time* or *timeframe* during which a visit should have taken place. Notably, different sources of WOs may provide different *resolutions* for time, forcing an attacker to consider a timeframe in which a visit could have taken place. For example, timestamps in Apache or nginx access logs use regular Unix timestamps as default (i.e., seconds), while

CDNs like Cloudflare maintain logs with Unix nanosecond precision. Further, there are inherent limitations in approximating t for the query when the attacker in addition to WO access can only directly observe traffic from the victim into Tor. We explore this later in Section 5.3.

One important limitation we place on the use of a WO with WF is that the attacker can only query the WO for *monitored* websites. The open world setting is intended to capture a more realistic setting for evaluating attacks, and inherent in this is that the attacker cannot train (or even enumerate) all possible websites on the web. Given the ability to enumerate and query all possible websites gives the adversary a capability in line with a global passive adversary performing correlation attacks, which is clearly outside of the threat model of Tor [20]. We further relate correlation and confirmation attacks to WF+WO attacks in Section 8.

Definition 1 defines the ideal WO: it never fails to observe a *monitored* website visit, it has no false positives, and it can answer for an arbitrary t. This is similar to how encryption and decryption oracles always encrypt and decrypt when modelling security for encryption schemes [23, 43, 52]. In practice, sources of all of these oracles may be more or less ideal and challenging for an attacker to use. Nevertheless, the prevalence of sources of these imperfect oracles motivate the assumption of an attacker with access to an ideal oracle. Similarly, for WOs, we motivate this assumption in Sections 4 and 5, in particular wrt. a timeframe on the order of (milli)seconds. Section 7 further considers non-ideal sources of WOs and the effect on WF+WO attacks, both when the WO can produce false positives and when the source only observes a fraction of visits to monitored websites.

3.2 Generic Website Fingerprinting Attacks with Website Oracles

As mentioned in Section 2.3, a WF attack is a classifier that is given as input a packet trace and provides as output a classification. The classification is either a monitored site or a class representing unmonitored (in the open world). Figure 2 shows the setting where an attacker capable of performing WF attacks also has access to a WO. We define a generic construction for WF+WO attacks that works with *any* WF attack in the open world in Definition 2:

Definition 2 (Binary verifier)**.** Given a website oracle o and WF classification c of a trace collected at time t, if c is a monitored class, query the oracle $o(c, t)$. Return c if the oracle returns true, otherwise return the unmonitored class.

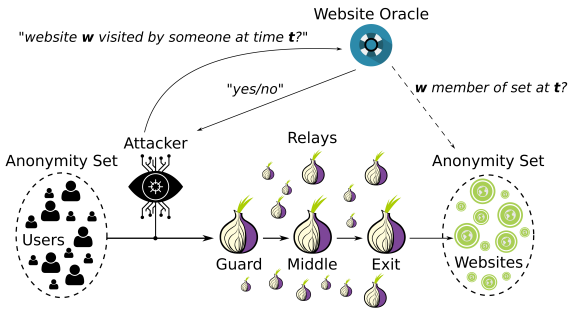


Fig. 2. WF+WO attacks, where the WO infers membership of a particular website w in the website anonymity set of all possible websites visited over Tor during a particular timeframe t.

Note that the WO is only queried when the WF *classification* is for a monitored website and that Definition 2 is a generalisation of the "high precision" DefecTor attack by Greschbach *et al.* [24]. In terms of *precision* and *false positives*, the above generic WF+WO construction is strictly superior to a WF attack without a WO. Assume that the WF classification incorrectly classified an unmonitored trace as monitored, then there is *only a probability* that a WO also returns true, depending on the probability that someone else visited the website in the same timeframe over Tor. If it does not, then a false positive is prevented. That is, a WF attack without WO access is identical to a WF attack with access to a useless WO that always returns true; any improvements beyond that will only help the attacker in ruling out false positives. We consider the impact on *recall* later.

We can further refine the use of WOs for the subset of WF attacks that support providing as output an ordered list of predictions in decreasing likelihood, optionally with probabilities, as shown in Definition 3:

Definition 3 (List verifier)**.** Given an ordered list of predictions in the open world and a website oracle:

for top prediction p **in** list **do**
  if p is unmonitored or oracle says p visited **then**
    return list
  move p **to** last **in** list and optionally update probabilities

First, we observe that if the WF attack thinks that it is most likely an unmonitored website, then we accept that because a WO can only teach us something new

about monitored websites. Secondly, if the most likely prediction has been visited according to the WO then we also accept that classification result. Finally, all that is left to do is to consider this while repeatedly iterating over the top predictions: if the top classification is a monitored website that has not been visited according to the WO, then move it from the top of the list and optionally update probabilities (if applicable, then also set $p = 0.0$ before updating) and try again. Per definition, we will either hit the case of a monitored website that has been visited according to the WO or an unmonitored prediction. As mentioned in Section 2.3, WF output that has some sort of probability or threshold associated with classifications are useful for attackers with different requirements wrt. false positives and negatives.

One could consider a third approach based on repeatedly querying a WO to first determine if any monitored websites have been visited and then train an optimised classifier (discarding monitored websites that we know have not been visited). While this may give a minor improvement, our results later in this paper as well as earlier work show that confusing monitored websites is a minor issue compared to confusing an unmonitored website as monitored [24, 30, 70].

4 Sources of Website Oracles

There are a wide range of potential sources of WOs. Table 1 summarizes a selection of sources that are more thoroughly detailed in Appendix C. The table shows the availability of the source, i.e., if the attacker needs to query the source in near real-time as a website visit occurs or if it can be accessed retroactively, e.g., through a legal request. We also estimate qualitatively the false positive rate of the source, its coverage of websites it can monitor (or fraction of Tor network traffic, depending on source), as well as the estimated effort to access the source. Finally, the table gives an example of an actor with access to the source.

Next we focus on a number of sources of WOs that we find particularly relevant: several due to DNS in Section 4.1, the DHT of Tor onion directory services in Section 4.2, and real-time bidding platforms in Section 4.3.

4.1 DNS

Before a website visit the corresponding domain name must be resolved to an IP address. For a user that uses Tor browser, the exit relay of the current circuit resolves the domain name. If the DNS record of the domain name is already cached in the DNS cache of the exit relay, then the exit relay uses that record. Otherwise the domain name is resolved and subsequently cached using whichever DNS resolution mechanism that the exit relay has configured. Based on this process we present three sources of WOs that work for unpopular websites.

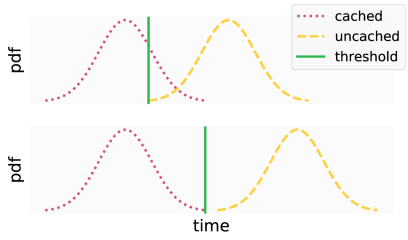


Fig. 3. The two cases when deciding on a classifier's threshold.

4.1.1 Shared Pending DNS Resolutions

If an exit relay is asked to resolve a domain name that is uncached it will create a list of pending connections waiting for the domain resolution to finish. If another connection asks that the same domain name be resolved, it is added to the list of pending connections. When a result is available all pending connections are informed. This is the basis of a WO: if a request to resolve a domain name returns a record *more quickly than previously measured by the attacker for uncached entries*, the entry was either pending resolution at the time of the request or already cached. Notably this works regardless of if exit relays have DNS caches or not. However, the timing constraints of shared pending connections are significant and thus a practical hurdle to overcome.

4.1.2 Tor's DNS Cache at Exit Relays

If an unpopular website is visited by a user, the resolved domain name will likely be cached by a *single* exit relay. We performed 411 `exitmap` [73] measurements between April 1–10 (2019), collecting on average 3544 (un)cached data points for each exit relay using a domain name under our control that is not in use by anyone else.

Given a labelled data set of (un)cached times for each exit relay, we can construct distinct *per-relay* classifiers that predict whether a measured time corresponds to an (un)cached domain name. While there are

Table 1. Comparison of a number of WO sources based on their *estimated* time of availability (when attacker likely has to collect data, i.e., retroactively or real-time), False Positive Rate (FPR), coverage of website/network visits, and primary entities with access.

Source	Availability	FPR	Coverage	Effort	Access
Dragnet surveillance programmes	retroactive	negl.	high	high	intelligence agencies
Content Delivery Networks	retroactive	negl.	high	high	operators
Real-time bidding	real-time (retroactive)	negl.	high	modest	customers (operator)
Webserver access logs	retroactive	negl.	high	medium	operators
Middleboxes	retroactive [1]	negl.	medium	medium	operators
OCSP	retroactive	low	high	medium	few CAs, plaintext
8.8.8.8 operator	retroactive	low [24]	16.8% of visits	high	Google, plaintext
1.1.1.1 operator	retroactive	low [24]	7.4% of visits	high	Cloudflare, plaintext
Exit relays	real-time	negl.	low	low	operators
Exit relays DNS cache	real-time	medium	high	medium	anyone
Query DNS resolvers	real-time	high	low	low	anyone
Onion v2 (v3)	real-time	negl.	high (low)	low (high)	anyone

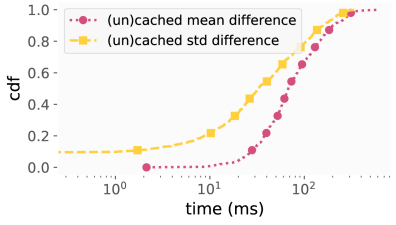


Fig. 4. The difference between (un)cached standard deviation and mean times without any absolute values, i.e., a negative value implies that the uncached time is smaller than the cached time.

many different approaches that could be used to build such a classifier, we decided to use a simple heuristic that should result in little or no false positives: output 'cached' iff no uncached query has *ever* been this fast before. Figure 3 shows the idea of this classifier in greater detail, namely create a *threshold* that is the minimum of the largest cached time and the smallest uncached time and then say cached iff the measured time is smaller than the threshold. Regardless of how well this heuristic performs (see below), it should be possible to construct other classifiers that exploit the trend of smaller resolve times and less standard deviation for cached queries (Figure 4). For example, 69.1% of all exit relays take at least 50 ms more time to resolve an uncached domain on average.

To estimate an *upper bound* on how effective the composite classifier of all per-relay classifiers could be *without any false positives* using our heuristic, we applied ten-fold cross-validation to simply exclude every exit relay that had false positives during any fold and then weighted the observed bandwidth for the remaining classifiers by the individual true positive rates. This gives us an estimate of how much bandwidth we could predict true positives for without having any false positives. By comparing it to the total exit bandwidth of the Tor network, we obtain an estimated upper bound true positive rate for the composite classifier of 17.3%.

When an attacker measures if a domain is cached or not the domain will, after the measurement, be cached for up to an hour (current highest caching duration in Tor, independent of TTL) at every exit. However, if a an attacker can cause an exit to run low on memory, the entire DNS cache will be removed (instead of only parts of it) due to a bug in the out-of-memory manager of Tor. We have reported this to the Tor Project[3]. We further discuss in Section 7 how frequently an attacker on average can be expected to query a WO.

4.1.3 Caching at Recursive DNS Resolvers

For a website that is unpopular enough, there is a high chance that nobody on the web visited the website within a given timeframe. This is the basis of our next idea for a WO which is *not mutually exclusive* to the Tor network: wait a couple of seconds after observing a connection, then probe all recursive DNS resolvers of Tor exits that can be accessed to determine whether any monitored website was cached approximately at the time of observing the connection by inspecting TTLs.

In 2016 Greschbach *et al.* [24] showed that remote DNS resolvers like Google's `8.8.8.8` receive a large portion of all DNS traffic that exit relays generate. To better understand how the DNS resolver landscape looks today, we repeated their experiment setup for 35 hours

3 https://trac.torproject.org/projects/tor/ticket/29617

in February 17–18 (2019), measuring every 30 minutes. Our results show that Google (16.8%) and Cloudflare (7.4%) are both popular. Many exits use a same-AS resolver which is presumably the ISP (42.3%), while other exits resolve themselves (15.2%) or use a remote DNS resolver that we did not identify (18.2%). Further, we note that there are at least one RIPE Atlas[4] network measurement probe in the same AS as 53.3% of all exits, providing access to many of the same DNS resolvers as used by exits from a similar network vantage point.

Instead of using RIPE Atlas nodes we opted for a different approach which is *strictly worse*: query Google's and Cloudflare's DNS resolvers from VMs in 16 Amazon EC2 regions. With a simple experiment of first visiting a unique domain (once again under our control and only used by us) using `torify curl` and then querying the DNS resolvers from each Amazon VM to observe TTLs, we got true positive rates of 2.9% and 0.9% for Google and Cloudflare with 1000 repetitions. While this may seem low, the cost for an attacker is at the time of writing about 2 USD per day using on-demand pricing. Using an identical setup we were also able to find a subset of monitored websites that yield alarmingly high true positive rates: 61.4% (Google) and 8.0% (Cloudflare). Presumably this was due to the cached entries being shared over a wider geographical area for some reason (however, not globally). Regardless, coupled with the fact that anyone can *globally purge* the DNS caches of Google[5] and Cloudflare[6] for arbitrary domain names, this is a noteworthy WO source.

4.2 Onion Service Directories in Tor

To access an onion service a user first obtains the service's *descriptor* from a Distributed Hash Table (DHT) maintained by *onion service directories*. From the descriptor the user learns of *introduction points* selected by the host of the onion service in the Tor network that are used to establish a connection to the onion service in a couple of more steps [64, 65] that are irrelevant here. Observing a request for the descriptor of a monitored onion service is a source for a WO. To observe visits for a target (known) onion service in the DHT, a relay first has to be selected as one out of six or eight (depending on version) relays to host the descriptor in the DHT, and then the victim has to select that relay to retrieve the descriptor. For v2 of onion services, the location in the DHT is deterministic [64] and an attacker can position its relays in such a way to always be selected for hosting target descriptors. Version 3 of onion services addresses this issue by randomising the process every 24 hours [65], forcing an attacker to host a significant number of relays to get a WO for onion services with high coverage. At the time of writing, there are about 3,500 relays operating as onion service directories.

4.3 Real-Time Bidding

Real-Time Bidding (RTB) is an approach towards online advertisement that allows a publisher to auction ad space to advertisers on a *per-visit* basis in real time [68]. Google's Display Network includes more than two million websites that reach 90% of all Internet users[7], and an advertiser that uses RTB must respond to submitted bid requests[8] containing information such as the three first network bytes of an IPv4 address, the second-level domain name of the visited website, and the user agent string within ≈100 ms. While the exact information available to the bidder depends on the ad platform and the publisher's advertisement settings, anonymous modes provide less revenue[9]. Combined with many flavours of pre-targeting such as IP and location filtering [67], it is likely that the bidder knows whether a user used Tor while accessing a monitored website. Vines et al. [67] further note that "35% of the DSPs also allow arbitrary IP white-and blacklisting (Admedo, AdWords, Bing, BluAgile, Criteo, Centro, Choozle, Go2Mobi, Simpli.fi)". Finally, observe that an attacker need not win a bid to use RTB as a WO.

4 https://web.archive.org/web/20190228145306/https://atlas.ripe.net/

5 https://web.archive.org/web/20190228150306/https://developers.google.com/speed/public-dns/cache

6 https://web.archive.org/web/20190228150344/https://1.1.1.1/purge-cache/

7 https://web.archive.org/web/20190228122431/https://support.google.com/google-ads/answer/2404191?hl=en&ref_topic=3121944%5C

8 https://web.archive.org/web/20190228122615/https://developers.google.com/authorized-buyers/rtb/downloads/realtime-bidding-proto

9 https://web.archive.org/web/20190228123602/https://support.google.com/admanager/answer/2913411?hl=en&ref_topic=2912022

5 Simulating Website Oracles

To be able to *simulate* access to a WO for *arbitrary monitored websites* we need to simulate the entire website anonymity set of Tor, because the anonymity set is what a WO queries for membership. We opt for simulation for ethical reasons. The simulation has three key parts: how those visits are distributed, the number of visits to websites over Tor, and the timeframe (resolution) of the oracle source. Note that the first two parts are easy for an attacker to estimate by simply observing traffic from live Tor exit relays, something we cannot trivially do as researchers adhering to Tor's research safety guidelines[10]. Another option available to an attacker is to repeatedly query a WO to learn about the popularity of its monitored websites and based on those figures infer the utility of the WO. We opted to not perform such measurements ourselves, despite access to several WOs, due to fears of inadvertently harming Tor users. Instead we base our simulations on results from the privacy-preserving measurements of the Tor network in early 2018 by Mani *et al.* [38].

5.1 How Website Visits are Distributed

Table 2 shows the average inferred website popularity from Mani *et al.* [38]. The average percentage does not add up to 100%, presumably due to the privacy-preserving measurement technique or rounding errors. Their results show that torproject.org is very popular (perhaps due to a bug in software using Tor), and beyond that focus on Alexa's[11] top one million most popular websites as bins. The "other" category is for websites identified not part of Alexa's top one million websites ranking. For the rest of the analysis (not simulation) in this paper we *exclude* torproject.org: for one, that Tor users visit that website is unlikely to be an interesting fact for an attacker to monitor, and its overrepresentation (perhaps due to a bug) will skew our analysis. Excluding torproject.org, about one third of all website visits go to Alexa (0,1k], one third to Alexa (1k,1m], and one third to other websites. The third column of Table 2 contains adjusted average percentages.

In our simulations for website visits we treat the entries in column two of Table 2 as bins of a histogram with the relative size indicated by the average website popularity. After randomly selecting a bin (weighted by popularity), in the case of an Alexa range we uniformly select a website within the range, and for the other category we uniformly select from one million other websites. This is a conservative choice given that there are hundreds of millions of active websites on the Internet. Uniformly selecting within a bin will make the more popular websites in the bin likely underrepresented while less popular websites in the bin get overrepresented. However, we typically simulate an attacker that monitors ≈100 websites and use the website popularity as the starting rank of the first monitored website. For the most popular websites, monitoring 100 websites covers the entire or significant portions of the bins (Alexa ≤1k), and for less popular websites (Alexa >1k), as our results later show, this does not matter.

Table 2. Inferred average website popularity for the entire Tor network early 2018, from Mani *et al.* [38, Figure 2].

Website	Average primary domain (%)	Without torproject.org
torproject.org	40.1	
Alexa (0,10]	8.4	13.9
Alexa (10,100]	5.1	8.4
Alexa (100,1k]	6.2	10.3
Alexa (1k,10k]	4.3	7.1
Alexa (10k,100k]	7.7	12.7
Alexa (100k,1m]	7.0	11.6
other	21.7	35.9

5.2 The Number of Website Visits

Mani *et al.* also inferred with a 95% confidence interval that $(104 \pm 36) * 10^6$ *initial* streams are created during a 24 hour period in the entire Tor network [38]. Based on this, in our simulation we assume 140 million website visits per day that are distributed as described above and occur uniformly throughout the day. While assuming uniformity is naive, we selected the upper limit of the confidence interval to somewhat negate any unreasonable advantage to the attacker.

5.3 A Reasonable Timeframe

Wang and Goldberg show that it is realistic to assume that an attacker can determine the start of a webpage load even in the presence of background noise and multiple concurrent website visits [71]. An attacker can fur-

10 https://web.archive.org/web/20190213130918/https://research.torproject.org/safetyboard.html

11 https://www.alexa.com/topsites

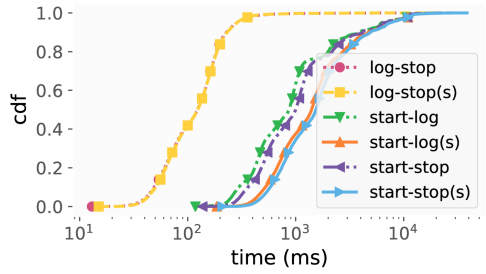


Fig. 5. Time differences between start, log, and stop events when visiting a website over HTTP(S) using Tor.

ther determine if a circuit is used for onion services or not [34, 47]. Now, consider an attacker that observes traffic between a Tor client and its guard. The initial stream contains the first HTTP GET request for a typical website visit. The request will be the first outgoing packet as part of a website visit once a connection has been established. When the request arrives at the destination is the point in time when an oracle, e.g., instantiated by access logs would record this time as the time of visit. Clearly, the exact time is between the request and the response packets and the attacker observes the timing of those packets. So what is a realistic timeframe for the attacker to use when it queries a WO?

Between January 22–30 (2019) we performed Round-Trip Time (RTT) measurements using four Amazon EC2 instances that ran *their own* nginx HTTP(S) servers to visit *themselves* over Tor (with `torify curl`) using a fresh circuit for each visit. This allowed us easy access to start and stop times for the RTT measurement, as well as the time a request appeared in the nginx access log (without any clock-drift). In total we collected 21,497 HTTP traces and 21,492 HTTPS traces, where each trace contains start, log, and stop timestamps. Our results are shown in Figure 5. It is clear that that log-to-stop times are independent of HTTP(S). More than half of all log-to-stop times (54.5%) are within a 100 ms window (see 40–140 ms), and nearly all log-to-stop times are less than 1000 ms.

Based on our experiment results we consider three timeframes relevant: 10 ms, 100 ms, and 1000 ms. First, 10 ms is relevant as close to optimal for any attacker. On average, there are only 17 website visits during a 10 ms window in the entire Tor network. 100 ms is our default for the WF experiments we perform: we consider it realistic for many sources of WOs (e.g., Cloudflare logs and real-time bidding). We also consider a 1000 ms timeframe relevant due to the prevalence of sources of WOs with a resolution in seconds (e.g., due to Unix timestamps or TTLs for DNS). Based on our simulations and the different timeframes, Appendix A contains an analysis of the utility of WOs using Bayes' law. Appendix B presents some key lessons from the simulation, in particular that while the resolution and resulting timeframe is an important metric in our simulation, it is minor in comparison to the overall website popularity in Tor of the monitored websites.

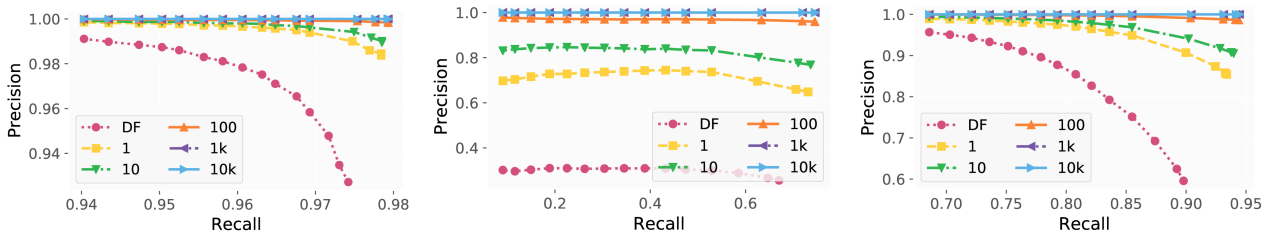
6 Deep Fingerprinting with Website Oracles

We first describe how we augment the Deep Fingerprinting (DF) attack by Sirinam *et al.* [60] with WO access. Next we evaluate the augmented classifier on three different datasets with five different WF defenses. Source code and datasets for simulating WF+WO attacks as well as steps to reproduce all of the following results using DF are available at github.com/pylls/wfwo.

6.1 The Augmented Classifier

As covered in the background (Section 2.3), DF is a CNN where the last layer is a softmax. The output is an array of probabilities for each possible class. Compared to the implementation of DF used by Sirinam *et al.*, we changed DF to not first use binary classification in the open world to determine if it is an unmonitored trace or not, but rather such that there is one class for each monitored website and one for unmonitored. Conceptually, this slightly lowers the performance of DF in our analysis, but our metrics show that mistaking one monitored website for another is insignificant for the datasets used in the analysis of this paper. The principal source of false positives is mistaking an unmonitored website for a monitored.

Given the probability of each possible class as output of DF, we used the second generic construction (Definition 3) from Section 3.2 to combine DF with a WO. To update the remaining probabilities after removing a (monitored) prediction with the help of the WO, we use a softmax again. However, due to how the softmax function is defined, it emphasizes differences in values above one and actually de-emphasizes values between



(a) No defense. **(b)** Walkie-Talkie [72]. **(c)** WTF-PAD [31].

Fig. 6. Attack simulation for Deep Fingerprinting (DF) with website oracles (100 ms timeframe) on Sirinam *et al.*'s dataset [60]. The lines in each sub-figure show DF with and without website oracle access for different starting Alexa ranks for monitored websites.

zero and one[12]. This is problematic for us because all values we send through the softmax are probabilities that per definition are between zero and one. To account for this, we first divide each probability with the maximum probability and multiply with a constant before performing the softmax. Through trial-and-error, a constant of five gave us a reasonable threshold in probabilities. Note that this does not in any way affect the order of likely classes from DF, it simply puts the probabilities in a span that makes it easier for us to retain a threshold value between zero and one after multiple calls to the softmax function.

6.2 WTF-PAD and Walkie-Talkie

We use the original dataset of Sirinam *et al.* [60] that consists of 95 monitored websites with 1,000 instances each as well as 20,000 unmonitored websites (95x1k+20k). The dataset is split 8:1:1 for training, validation, and testing, respectively. Given the dataset and our changes to DF to not do binary classification means that our testing dataset is unbalanced in terms of instances per class. Therefore we show precision-recall curves generated by alternating the threshold for DF with and without WO access.

Figure 6 shows the results of DF and DF+WO with a simulated WO on Sirinam *et al.*'s dataset with no defense (Figure 6a), Walkie-Talkie (Figure 6b), and WTF-PAD (Figure 6c). For the WO we use a 100 ms timeframe and plot the results for different starting Alexa ranks of the 95 monitored websites. Regardless of defense or not, we observe that for Alexa ranks 1k and less popular websites the precision is perfect (1.0) regardless of threshold. This indicates that—for an attacker monitoring frontpages of websites—a 100 ms WO significantly reduces false positives for two-thirds of all website visits made over Tor, for the vast majority of potentially monitored frontpages of websites. Recall is also slightly improved.

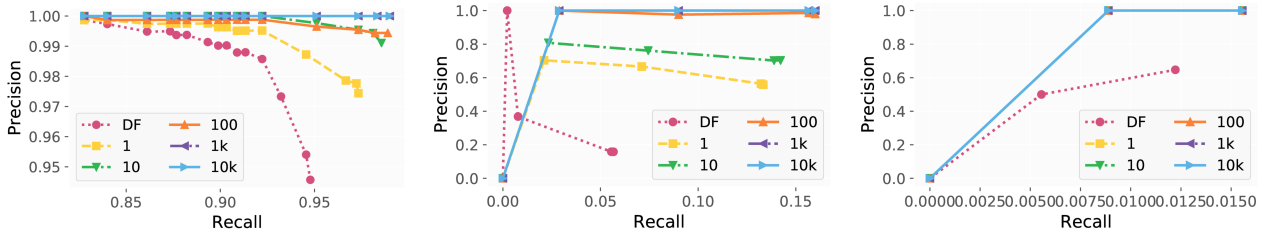
For Walkie-Talkie we observe a significant improvement in precision due to WO access. Wang and Goldberg note that the use of popular websites as decoy (non-sensitive) websites protects less-popular sensitive websites due to the base rate: an attacker claiming that the user visited the less-popular website is (per definition) likely wrong, given that the attacker is able to detect both potential website visits [72]. Access to a WO flips this observation on its head: if a WO detects the sensitive less-popular website, the base rate works in reverse. The probability of an unpopular website being both miss-classified and visited in the timeframe is small for all but the most popular websites. The key question becomes one of belief in the base rate of the network and that of the target user, as analysed in Appendix A.

Further, WO access improves both recall and precision for all monitored websites against WTF-PAD. WTF-PAD only provides a three percentage points decrease in recall compared to no defense for monitored websites with Alexa ranks 1k and above.

6.3 CS-BuFLO and Tamaraw

To evaluate the constant-rate defenses CS-BuFLO and Tamaraw by Cai et al. [6, 7] we use Wang *et al.*'s dataset in the open world [70]. The dataset consists of 100 monitored websites with 90 instances each and 9000 unmonitored sites (100x90+9k), that we randomly split (stratified) into 8:1:1 for training, validation, and testing. We had to increase the length of the input to DF for this dataset, from 5000 to 25000, to ensure that we capture most of the dataset. To get defended traces for

12 https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=883834589

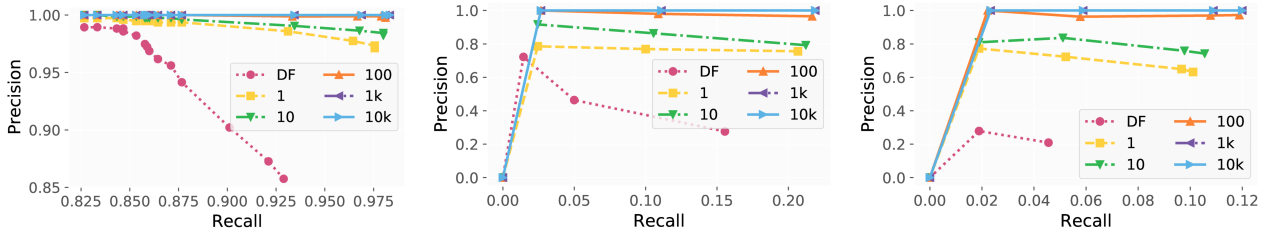


(a) No defense.

(b) CS-BuFLO [6], with reported 67.2% BOW and 575.6% TOH [12].

(c) Tamaraw [7], with reported 256.7% BOW and 341.4% TOH [12].

Fig. 7. Attack simulation for Deep Fingerprinting (DF) [60] with website oracles (100 ms timeframe) on Wang *et al.*'s dataset [70]. The lines in each sub-figure show DF with and without website oracle access for different starting Alexa ranks for monitored websites.



(a) No defense.

(b) DynaFlow [36] config 1, with measured 59% BOH and 24% TOH.

(c) DynaFlow [36] config 2, with measured 109% BOH and 30% TOH.

Fig. 8. Attack simulation for Deep Fingerprinting (DF) [60] with website oracles (100 ms timeframe) on Lu *et al.*'s dataset [36]. The lines in each sub-figure show DF with and without website oracle access for different starting Alexa ranks for monitored websites.

CS-BuFLO and Tamaraw we use the slightly modified implementations as part of Cherubin's framework [12].

Figure 7 shows the results of our simulations. DF alone is also highly effective against the original Wang dataset—as expected—and our attack simulation shows that we can further improve it with access to website oracles. Most importantly, both CS-BuFLO and Tamaraw offer protection against DF with and without oracle access by *significantly lowering recall*. Tamaraw offers an order of magnitude better defense in terms of recall. As implemented in the framework by Cherubin, CS-BuFLO and Tamaraw reportedly has BOH 67.2% and 256.7%, and TOH 575.6% and 341.4%, respectively. This kind of overhead is likely prohibitively large for real-world deployment in Tor [6, 7, 31, 60, 72].

6.4 DynaFlow

DynaFlow is a *dynamic* constant-rate defense by Lu *et al.* [36] with two configurations that result in different overheads and levels of protection. Lu *et al.* gathered their own dataset of 100 monitored websites with 90 instances each and 9000 unmonitored websites (100x90+9k, same as Wang *et al.*'s [70]) to be able to combine smaller packets, as discussed briefly in Section 2.3. As for CS-BuFLO and Tamaraw, we had to increase the length of the input to DF for this dataset to 25000 to ensure that we capture most of the dataset.

Figure 8 shows the results of our simulations for no defense as well as the two configurations of DynaFlow. As for Wang *et al.*'s dataset [70], we see as expected that DF is highly effective and WO access further improves the attack. Further, both configurations of DynaFlow are effective defenses, comparable to CS-BuFLO with significantly lower overheads at first glance. However, note that the comparison is problematic due to DynaFlow combining smaller packets. The extra overhead for config 2 over 1 is not wasted: recall is significantly reduced, more than halved for regular DF and slightly less than half with a WO.

7 Discussion

For defenses that are based on the idea of creating collision sets between packet traces generated by websites, oracle access is equivalent to being able to perform set intersection between the set of websites in a collision set and monitored websites visited at the time of fingerprinting. As the results show from Section 6, some

defenses can significantly reduce the recall of WF attacks with WOs, but not the precision for the majority of websites and website visits in Tor.

Next, in Section 7.1 we further cement that our simulations show that WOs significantly reduces false positives, highlighting that a WF+WO attacker surprisingly infrequently have to query a WO when classifying unmonitored traces. Section 7.2 discusses the impact of imperfect WO sources with limited observability and false positives on the joint WF+WO attack. Finally, Section 7.3 covers limitations of our work, and Section 7.4 discusses possible mitigations.

7.1 A Large Unmonitored Dataset

We look at the number of false positives for a large test dataset consisting of only unmonitored websites (representing a target user(s) with base rate 0, i.e., that never visits any monitored website). Using the dataset of Greschbach et al. [24], we trained DF on 100x100 monitored and 10k unmonitored websites (8:2 stratified split for validation), resulting in about 80% validation accuracy after 30 epochs (so DF is clearly successful also on this dataset). We then tested the trained classifier on *only* 100k *unmonitored* traces, with and without oracle access (100ms resolution) for different assumptions of the popularity of the *monitored* websites. With ten repetitions of the above experiment, we observed a false positive rate in the order of 10^{-6} for monitored websites with Alexa popularity 10k. Excluding torproject.org, this indicates that an attacker would have close to no false positives for about half of all website visits in Tor, according to the distribution of Mani *et al.* [38] (see Section 5.1). Without access to a WO, DF had a false positive rate in the order of 10^{-3} to 10^{-4}, depending on the threshold used by the attacker.

Recall how WOs are used as part of WF+WO attacks in Section 3.2: the WO is only used if the WF attack classifies a trace as monitored[13]. This means that, in the example above, the WO is used on average every 10^3 to 10^4 trace, to (potentially) rule out a false positive. Clearly, this means that WO sources that can only be used infrequently, e.g., due to caching as in DNS, are still valuable for an attacker.

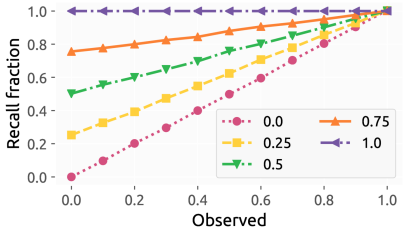


Fig. 9. How limited WO observability effects the final recall of a WF+WO attack for five different WO false positive rates.

7.2 Imperfect Website Oracle Sources

Our analysis considered an ideal source of a WO that observes all visits to targeted *monitored* websites of the attacker and that produces no false positives. Next, using the same dataset and setup as for Figure 6a with an Alexa starting rank of 10^4, we simulate the impact on recall and the False-Negative-to-Positive-rate[14] (FNP) of the *joint* WF+WO attack for five false positive rates *of the WO* and a fraction of observed website visits.

Figure 9 shows the impact on the joint recall in the above setting. We see that recall is directly proportional to the fraction of observed visits, as per the results of Greschbach *et al.* [24]. Further, false positives for the WO have a positive impact on the fraction of recall, counteracting visits missed due to limited observability. For the same reason, a larger timeframe or monitoring more popular websites would also improve recall.

Figure 10 shows the impact on the joint FNP. Note that lower FNP is better for an attacker. We see that limited observability has no impact on FNP. This makes sense, because a WO cannot confirm anything it does not observe. The FNP fraction for the joint FNP is roughly proportional to the FP of the WO. We also see that the FNP fraction consistently is above the FP of the WO: this is because—beyond the simulated FP—there is a slight probability that someone else (in our simulation of the Tor network) visited the website for each classified trace. A larger timeframe or monitoring more popular websites would also increase FNP.

From above results, our simulations indicate that even with a deeply imperfect WO source an attacker can get significant advantage in terms of reduced false

13 For WF attacks like DF that produces a list of probabilities (Definition 3), just assume that the attacker picks the threshold and checks if the probability is above as part of the if-statement before using the WO.

14 For a classifier with multiple monitored classes and an unmonitored class (as for our modified DF, see Section 6.1), FNP captures the case when the classifier classifies an unmonitored testing trace as any monitored class. In addition to FNP, such a classifier can also confuse one monitored website for another. Both these cases are false positives [69].

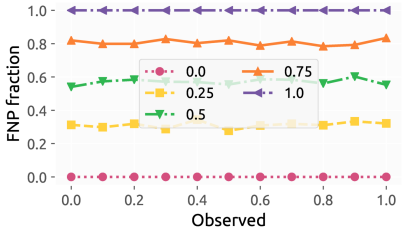


Fig. 10. How limited WO observability effects the final False-Negative-to-Positive-rate (FNP) of a WF+WO attack for five different WO false positive rates. Lower is better.

positives at a comparatively small cost of recall. For example, given a WO with 50% observability and false positives, the resulting WF+WO attack has about 75% of the recall of the WF attack and slightly more than half the false positives.

7.3 Limitations

As discussed in Section 2.3, there are a number of practical limitations in general for WF attacks. Regarding attacker goals, WOs are likely less useful for the purpose of censorship than for other goals. Many sources of WOs cannot be accessed in real-time, giving little utility for an attacker that needs to make a near real-time censorship decision. An attacker that only wants to detect visits to a few selected monitored websites gains significant utility from WOs, as long as the detection does not have to be in real-time. It is also noteworthy that an attacker that wants to detect all possible website visits by a victim can use the WO to in essence "close the world" from all possible websites to only those visited over Tor while the victim is actively browsing. Granted, this requires a source for the WO that is slightly different from our definition, but some do offer this: e.g., an attacker that gains comprehensive control over the DNS resolvers used by Tor exits [24].

When it comes to false positives a significant limitation of our simulations is that we consider fingerprinting the frontpages of websites and not specific webpages. Several sources or WOs are not able to detect webpage visits. This is also true for subsequent webpage visits on the same website after first visiting the frontpage of a website (e.g., DNS and OCSP will be cached). An attacker with the goal of detecting each such page visit will thus suffer more false positives or fail at detecting them for some sources of WOs.

7.4 Mitigations

The best defense against WOs is WF defenses that significantly reduce the recall of WF attacks. In particular, if an attacker can significantly reduce the website anonymity set *after* accounting for information from the WO, then attacks are likelier to succeed. This implies that most websites need to (at least have the potential to) result in the same network traces, as we see with DynaFlow, Tamaraw, and CS-BuFLO.

For onion websites we note that the DHT source of a WO from Section 4 is inherent to the design of onion services in Tor. Defenses that try to make it harder to distinguish between regular website visits and visits to onion websites should also consider this WO source as part of their analysis, in particular for v2 onion services.

Finally, some sources of WOs could be minimized. If you run a potentially sensitive website: do not use RTB ads, staple OCSP, have as few DNS entries as possible[15] with a high TTL, do not use CDNs, do not retain any access logs, and consider if your website, web server, or operating system have any information leaks that can be used as an oracle. If you run a Tor exit, consider not using Google or Cloudflare for your DNS but instead use your ISP's resolver if possible [24].

8 Related Work

The combination of a WF attack with a WO is a type of Classify-Verify method as proposed by Stolerman et al. [61], which in turn is a type of rejection function as described by Chow [13]. Such a method was first used in the context of WF by Juarez *et al.* [30] and later by Greschbach *et al.* [24] to augment WF attacks with inferences from observed DNS traffic. Note that the attack by Greschbach et al. can be seen as a probabilistic WO due to the attacker under their threat model only observing a fraction of DNS traffic from the Tor network. Our work builds upon and generalises their work where DNS traffic is just one of many possible sources to infer website visits from. Further, our DNS-based sources are usable by anyone instead of relatively strong network attackers (or Google or Cloudflare).

All anonymity networks produce anonymity sets (per definition) that change with observations by an at-

15 As noted by Greschbach *et al.* [24], websites may have several unique domain names. Each of those could be used independently to query several sources (e.g., DNS) of WOs.

tacker over time [53]. Modelling the behaviour of an anonymity system (as a mix), what the attacker observes, and how the anonymity sets change over time allows us to reason about how the attacker can perform traffic analysis and break the anonymity provided by the system [19, 32, 58]. Attacks along these lines are many with more-or-less consistent terminology, including intersection attacks, (statistical) disclosure attacks, and traffic confirmation attacks [3, 15–17, 33, 53, 54, 66].

WOs are nothing more than applying the notion of anonymity sets to the potential destination websites visited over an anonymity network like Tor and giving an attacker the ability to query this anonymity set for membership for a limited number of monitored websites. The way we use WOs in our generic attacks is *not to learn long-term statistically unlikely relationships* between senders and recipients in a network. Rather, the WO is only used to learn *part of the anonymity set at the time of the attack*. That an attacker can observe anonymity sets is not novel, what is novel in our work is how we apply it to the WF domain and argue for its inclusion as a core attacker capability when modelling WF attacks and defenses.

Murdoch and Danezis showed how to use observed latency in Tor as an oracle to perform traffic analysis attacks [42]. Chakravarty *et al.* detailed similar attacks but based on bandwidth estimation [10] and Mittal *et al.* using throughput estimation [41]. Attackers in these cases do not need to be directly in control of significant fractions Tor, but rather use network measurements to infer the state of the network and create an oracle that an attacker can utilize, similar to WOs.

Correlation of input and output flows is at the core of many attacks on anonymity networks like Tor [5, 29, 63]. Flow correlation attacks correlate traffic on the network layer, considering packet sizes and timing of sent traffic. The RAPTOR attack by Sun et al. [63] needs about 100MB of data sent over five minutes to correlate flows with high accuracy. The recent state-of-the-art attack DeepCorr by Nasr *et al.* [44]—based on deep learning like Deep Fingerprinting by Sirinam *et al.* [60]—needs only about 900KB of data (900 packets) for comparable accuracy to RAPTOR. While flow correlation attacks like RAPTOR and DeepCorr operate on the network layer, WF+WO attacks can be viewed as *application layer* correlation attacks. WF attacks extract the application-layer data (the website) while WOs reconstruct parts of the anonymity set of possible monitored websites visited. WF attacks need to observe most of the traffic generated when visiting a website that goes into the anonymity network. While a WO does not have to directly view any of the output flows of the network, it needs to be able to infer if a particular website was visited during a period of time, as shown in Section 4.

9 Conclusions

WF+WO attacks use the base rate of all users of the network against victims, significantly reducing false positives in the case of all but the most popular websites visited over Tor. This is troubling in many ways, in part because presumably many sensitive website visits are to unpopular websites used only by local communities in regions of the world where the potential consequences of identification are the worst.

The threat model of Tor explicitly states that Tor does not consider attackers that can observe both incoming and outgoing traffic [20]. Clearly, a WO gives the capability to infer what the outgoing traffic of the network encodes on the application layer (the website visits). This is in a sense a violation of Tor's threat model when combined with a WF attacker that also observes incoming traffic. However, we argue that because of the plethora of possible ways for an attacker to infer membership in the anonymity sets of Tor, WOs should be considered within scope simply because Tor asserts that it is an anonymity network.

While the real-world impact of WF attacks on Tor users remains an open question, our simulations show that false positives can be signficantly reduced by many attackers with little extra effort for some WO sources. Depending on WO source, this comes at a trade-off of less recall. For many attackers and attacker goals, however, this is a worthwhile trade. To us, the threat of WF attacks appears more real than ever, especially when also considering recent advances by deep learning based attacks like DF [60] and DeepCorr [44].

Acknowledgements

We would like to thank Jari Appelgren, Roger Dingledine, Nicholas Hopper, Marc Juarez, George Kadianakis, Linus Nordberg, Mike Perry, Erik Wästlund, and the PETS reviewers for their valuable feedback. Simulations were performed using the Swedish National Infrastructure for Computing (SNIC) at High Performance Computing Center North (HPC2N). This research was funded by the Swedish Internet Foundation and the Knowledge Foundation of Sweden.

References

[1] C. Abdelberi, T. Chen, M. Cunche, E. D. Cristofaro, A. Friedman, and M. A. Kâafar. Censorship in the wild: Analyzing internet filtering in syria. In *IMC*, 2014.

[2] K. Abe and S. Goto. Fingerprinting attack on Tor anonymity using deep learning. *Proceedings of the Asia-Pacific Advanced Network*, 42:15–20, 2016.

[3] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free MIX routes and how to overcome them. In *International Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[4] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *CRYPTO*, 1998.

[5] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? In *CCS*, 2007.

[6] X. Cai, R. Nithyanand, and R. Johnson. CS-BuFLO: A congestion sensitive website fingerprinting defense. In *WPES*, 2014.

[7] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *CCS*, 2014.

[8] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: website fingerprinting attacks and defenses. In *CCS*, 2012.

[9] Y. Cao, Z. Qian, Z. Wang, T. Dao, S. V. Krishnamurthy, and L. M. Marvel. Off-path TCP exploits of the challenge ACK global rate limit. *IEEE/ACM Trans. Netw.*, 26(2):765–778, 2018.

[10] S. Chakravarty, A. Stavrou, and A. D. Keromytis. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *ESORICS*, 2010.

[11] H. Cheng and R. Avnur. Traffic analysis of SSL encrypted web browsing. *Project paper, University of Berkeley*, 1998.

[12] G. Cherubin. Bayes, not naïve: Security bounds on website fingerprinting defenses. *PoPETs*, 2017.

[13] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1), 1970.

[14] T. Chung, J. Lok, B. Chandrasekaran, D. R. Choffnes, D. Levin, B. M. Maggs, A. Mislove, J. P. Rula, N. Sullivan, and C. Wilson. Is the web ready for OCSP must-staple? In *IMC*, 2018.

[15] G. Danezis. Statistical disclosure attacks. In *Security and Privacy in the Age of Uncertainty, IFIP SEC*, 2003.

[16] G. Danezis. The traffic analysis of continuous-time mixes. In *PET*, 2004.

[17] G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Information Hiding, 6th International Workshop, IH*, 2004.

[18] D. Das, S. Meiser, E. Mohammadi, and A. Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *IEEE S&P*, 2018.

[19] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *PET*, 2002.

[20] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

[21] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *IEEE S&P*, 2012.

[22] R. Ensafi, J. C. Park, D. Kapur, and J. R. Crandall. Idle port scanning and non-interference analysis of network protocol stacks using model checking. In *USENIX Security*, 2010.

[23] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[24] B. Greschbach, T. Pulls, L. M. Roberts, P. Winter, and N. Feamster. The effect of DNS on Tor's anonymity. In *NDSS*, 2017.

[25] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security*, 2016.

[26] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *CCSW*, 2009.

[27] A. Hintz. Fingerprinting websites using traffic analysis. In *PET*, 2002.

[28] R. Jansen, M. Juárez, R. Galvez, T. Elahi, and C. Díaz. Inside job: Applying traffic analysis to measure Tor from within. In *NDSS*, 2018.

[29] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson. Users get routed: traffic correlation on Tor by realistic adversaries. In *CCS*, 2013.

[30] M. Juárez, S. Afroz, G. Acar, C. Díaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *CCS*, 2014.

[31] M. Juárez, M. Imani, M. Perry, C. Díaz, and M. Wright. Toward an efficient website fingerprinting defense. In *ESORICS*, 2016.

[32] D. Kesdogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In *Information Hiding, 5th International Workshop, IH*, 2002.

[33] D. Kesdogan and L. Pimenidis. The hitting set attack on anonymity protocols. In *Information Hiding, 6th International Workshop, IH*, 2004.

[34] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas. Circuit fingerprinting attacks: Passive deanonymization of Tor hidden services. In *USENIX Security*, 2015.

[35] M. Liberatore and B. N. Levine. Inferring the source of encrypted HTTP connections. In *CCS*, 2006.

[36] D. Lu, S. Bhat, A. Kwon, and S. Devadas. Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows. In *WPES*, 2018.

[37] D. Lyon. Surveillance, snowden, and big data: Capacities, consequences, critique. *Big Data & Society*, 1(2), 2014.

[38] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr. Understanding tor usage with privacy-preserving measurement. In *IMC*, 2018.

[39] N. Mathews, P. Sirinam, and M. Wright. Understanding feature discovery in website fingerprinting attacks. In *2018 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–5. IEEE, 2018.

[40] R. Merget, J. Somorovsky, N. Aviram, C. Young, J. Fliegenschmidt, J. Schwenk, and Y. Shavitt. Scalable scanning and automatic classification of tls padding oracle vulnerabilities. In *USENIX Security*, 2019. to appear.

[41] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *CCS*, 2011.

[42] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *IEEE S&P*, 2005.

[43] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.

[44] M. Nasr, A. Bahramali, and A. Houmansadr. Deepcorr: Strong flow correlation attacks on Tor using deep learning. In *CCS*, 2018.

[45] R. Nithyanand, X. Cai, and R. Johnson. Glove: A bespoke website fingerprinting defense. In *WPES*, 2014.

[46] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle. Website fingerprinting at internet scale. In *NDSS*, 2016.

[47] A. Panchenko, A. Mitseva, M. Henze, F. Lanze, K. Wehrle, and T. Engel. Analysis of fingerprinting techniques for Tor hidden services. In *WPES*, 2017.

[48] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *WPES*, 2011.

[49] M. Perry. A critique of website traffic fingerprinting attacks, https://web.archive.org/web/20190208082403/https://blog.torproject.org/critique-website-traffic-fingerprinting-attacks.

[50] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. 34, 01 2010.

[51] Z. Qian, Z. M. Mao, and Y. Xie. Collaborative TCP sequence number inference attack: how to crack sequence number under a second. In *CCS*, 2012.

[52] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, 1991.

[53] J. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *International Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[54] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998.

[55] V. Rimmer, D. Preuveneers, M. Juárez, T. van Goethem, and W. Joosen. Automated website fingerprinting through deep learning. In *NDSS*, 2018.

[56] E. Ronen, R. Gillham, D. Genkin, A. Shamir, D. Wong, and Y. Yarom. The 9 lives of bleichenbacher's CAT: new cache attacks on TLS implementations. *IACR Cryptology ePrint Archive*, 2018:1173, 2018.

[57] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S. D. Strowes, and N. Vallina-Rodriguez. A long way to the top: Significance, structure, and stability of internet top lists. In *IMC*, 2018.

[58] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *PET*, 2002.

[59] V. Shmatikov and M. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *ESORICS*, 2006.

[60] P. Sirinam, M. Imani, M. Juárez, and M. Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *CCS*, 2018.

[61] A. Stolerman, R. Overdorf, S. Afroz, and R. Greenstadt. Classify, but verify: Breaking the closed-world assumption in stylometric authorship attribution. In *IFIP Working Group*, volume 11, page 64, 2013.

[62] Q. Sun, D. R. Simon, Y. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE S&P*, 2002.

[63] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: routing attacks on privacy in Tor. In *USENIX Security*, 2015.

[64] Tor Project. Tor rendezvous specification - version 2, https://gitweb.torproject.org/torspec.git/tree/rend-spec-v2.txt.

[65] Tor Project. Tor rendezvous specification - version 3, https://gitweb.torproject.org/torspec.git/tree/rend-spec-v3.txt.

[66] C. Troncoso, B. Gierlichs, B. Preneel, and I. Verbauwhede. Perfect matching disclosure attacks. In *PETS*, 2008.

[67] P. Vines, F. Roesner, and T. Kohno. Exploring ADINT: using ad targeting for surveillance on a budget - or - how alice can buy ads to track bob. In *WPES*, 2017.

[68] J. Wang, W. Zhang, and S. Yuan. Display advertising with real-time bidding (RTB) and behavioural targeting. *Foundations and Trends in Information Retrieval*, 2017.

[69] T. Wang. *Website Fingerprinting: Attacks and Defenses*. PhD thesis, University of Waterloo, 2015.

[70] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security*, 2014.

[71] T. Wang and I. Goldberg. On realistically attacking Tor with website fingerprinting. *PoPETs*, 2016(4):21–36, 2016.

[72] T. Wang and I. Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *USENIX Security*, 2017.

[73] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. R. Weippl. Spoiled onions: Exposing malicious tor exit relays. In *PETS*, 2014.

A Bayes' Law for Estimating Utility of Website Oracles

To reason about the advantage to an attacker of having access to a WO, we estimate the conditional probability of a target user visiting a monitored website. For conditional probability we know that:

$$P(C_0 \cap C_1) = P(C_0|C_1)P(C_1) \tag{1}$$

For a hypothesis H given conditional evidence E, Bayes' theorem states that:

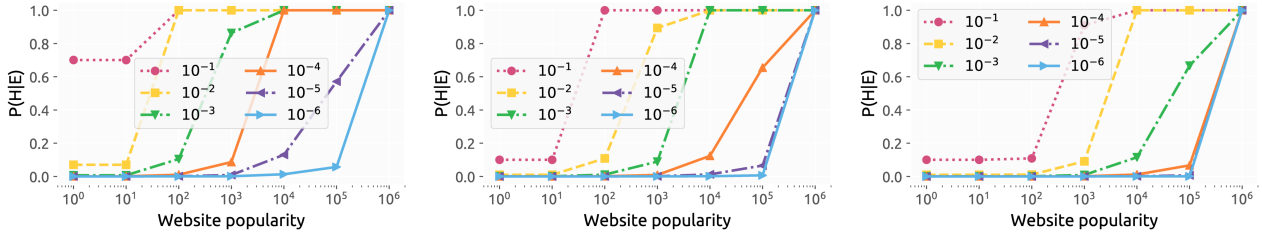
$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{2}$$

Assume that $\mathrm{E} = E_0 \cap E_1$, then:

$$P(H|E_0 \cap E_1) = \frac{P(E_0 \cap E_1|H)P(H)}{P(E_0 \cap E_1)} \tag{3}$$

Substituting $P(E_0 \cap E_1)$ with (1) we get:

$$P(H|E_0 \cap E_1) = \frac{P(E_0 \cap E_1|H)P(H)}{P(E_0|E_1)P(E_1)} \tag{4}$$



(a) 10 ms **(b)** 100 ms **(c)** 1000 ms

Fig. 11. The conditional probability as a function of user base rate and website popularity (Alexa) for three different timeframes.

For a timeframe t, we define

H the probability that target user(s) visited website w over Tor in t

E_0 the probability that target user(s) visited a website over Tor in t

E_1 the probability that someone visited website w over Tor in t

We see that $P(E_0 \cap E_1|H) = 1$ by definition and get:

$$P(H|E_0 \cap E_1) = \frac{P(H)}{P(E_0|E_1)P(E_1)} \tag{5}$$

Consider $P(E_0|E_1)$: while the conditional E_1 may have some minor affect on user behaviour (in particular for overall popular websites), we assume that the popularity of using Tor to visit a particular website (by any of the users of Tor) has negligible impact on E_0 and treat E_0 and E_1 as independent:

$$P(H|E_0 \cap E_1) = \frac{P(H)}{P(E_0)P(E_1)} \tag{6}$$

We can further refine $P(H)$ as being composed of at least:

$$P(H) = P(E_0) \cap P(B_w) = P(E_0|B_w)P(B_w) \tag{7}$$

Where $P(B_w)$ is the base rate (prior) of the user(s) visiting website w out of all possible websites they visit ($P(E_0)$). We again assume (perhaps naively) that E_0 is also independent of B_w, which gives us:

$$P(H|E_0 \cap E_1) = \frac{P(E_0)P(B_w)}{P(E_0)P(E_1)} = \frac{P(B_w)}{P(E_1)} \tag{8}$$

In other words, if an attacker learns that target user(s) visited a website (E_0) over Tor and that website w was also visited over Tor by some user (E_1), then we can estimate the probability that it was target user(s) that visited website w (H) as the ratio between the base rate (prior) for visiting w of target user(s) (B_w) and the probability that someone visited the website over Tor (E_1), all within a timeframe t.

Figure 11 shows the results for simulating the probability $P(H|E_0 \cap E_1)$ for different website popularities of w, base rates, and timeframes. We see that with a realistic timeframe of 100 ms, for all base-rates but 10^{-6} there is non-zero conditional probability (and therefore utility of WO access) for Alexa top 100k or less popular websites, which covers about half of all website visits over Tor (excluding torproject.org).

B Lessons from Simulation

With the ability to simulate access to WOs we can now simulate the entire website anonymity set for Tor. To get a better understanding of why WOs are so useful for an attacker performing WF attacks, we look at two results from the simulation below.

B.1 Time Until Website Visited over Tor

Figure 12 shows the time until there is a 50% probability that a website has been visited over Tor depending on website popularity (Alexa, as discussed in Section 5.1). Within ten seconds, we expect that most of Alexa top 1k has been visited. Recall that this represents about one third of all website visits over Tor. The less popular websites on Alexa top one-million represent another third of all visits, quickly approaching hundreds of seconds between visits. For the remaining third of all website visits we expect them to be even less frequent.

B.2 Visits Until First False Positive

Assume that target user(s) have a base rate of 0, i.e., they never visit the attacker's monitored websites. With

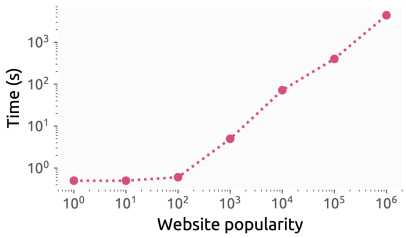


Fig. 12. The simulated time until there is a 50% probability that a website for different Alexa ranks has been visited over Tor.

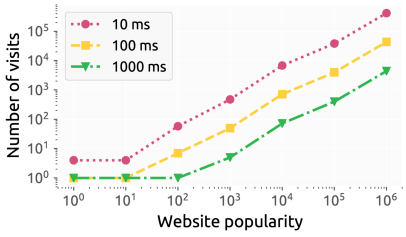


Fig. 13. The number of website visits until there is a 50% probability that a website oracle would contribute to a false positive.

WO access, we can determine how many (naively assumed independent) website visits it *at least* takes until there is a 50% chance that the attacker's classifier gets a false positive. This is because if the attacker's website classifier without oracle access always returns a false positive, then the false positive rate by the WF+WO attack will be determined by when the WO says that the—incorrectly classified as monitored—website has been visited. Figure 13 shows the expected number of visits *by the victim(s)* for different timeframes based on the popularity of the monitored websites. Note that the attacker per definition chooses which websites are monitored and can therefore take the probability of false positives into account.

C Sources of Website Oracles

There are a wide number of possible sources to instantiate WOs. Here we present some details on a selection of sources, far from exhaustive.

C.1 Surveillance Programmes

Intelligence agencies operate surveillance programmes that perform bulk collection and retention of communications metadata, including web-browsing [37]. For example, the Snowden revelations included *Marina*:

> Of the more distinguishing features, Marina has the ability to look back on the last 365 days' worth of DNI (Digital Network Intelligence) metadata seen by the Sigint collection system, *regardless* whether or not it was tasked for collection.[16]

Another example is the prevalence of nation states to monitor Internet traffic that crosses geographic borders. For example, China operates the Great Firewall of China that is also used for censorship purposes. Due to the nature of Tor and how exits are selected, visits to websites that are not operated by world-wide reaching hosting providers are highly likely to cross multiple nation borders as traffic goes from an exit to the website. It is also worth to highlight that any sensitive website hosted from within a country where a state actor is interested in identifying visitors are likely to capture traffic to that website due to the Tor traffic crossing its borders more often than not.

C.2 Content Delivery Networks

Content Delivery Networks (CDNs), such as Akamai, Google, and Amazon host different types of content for a significant fraction of all websites on the Internet [57]. Inherently, all requests for these resources are easily identified as coming from Tor exits, and depending on content, things like unique identifiers and HTTP referrer headers enable the CDN provider to infer the website the content is hosted on.

C.3 Internet Giants

Internet giants like Google, Apple, Facebook, Amazon, Microsoft, and Cloudflare make up a large fraction the web as we know it. For example the use of Google Analytics is wide-spread, so is hosting in clouds provided by several of these giants, and Cloudflare with its "cloud network platform" hosts over 13 million domains[17]. While some of them may do what is in their power to protect the valuable data they process and retain, they are still subject to many legal frameworks across the world that might not offer the best of protections for, say, access logs pertaining to "anonymous"

16 https://web.archive.org/web/20190227151029/https://www.theguardian.com/world/2013/sep/30/nsa-americans-metadata-year-documents

17 https://web.archive.org/web/20190227165133/https://www.cloudflare.com/

users of Tor when requested by authorities of nation states. As another example, Cloudflare offers a nice API for their customers to get their access logs with Unix nanosecond precision. The logs are retained for up to seven days[18], giving ample time for legal requests.

C.4 Access Logs of Web Servers

The vast majority of web servers retain access logs by default. Typically, they provide unix timestamps with seconds as the resolution (the case for Apache and nginx). Further, the access logs may be shipped to centralised security information and event management (SIEM) systems for analysis, with varying retention times and rigour in storage. For example, it is common to "anonymize" logs by removing parts of the IP-addresses and then retaining them indefinitely, as is the case for Google who removes part of IP addresses in logs after nine months[19].

C.5 Middleboxes

Network middleboxes that observe, analyse, and potentially retain network traffic abound. Especially in more oppressive countries, middleboxes are often used for censorship or dragnet surveillance, e.g., as seen with Blue Coat in Syria [1].

C.6 OCSP Responders

Chung *et al.* [14] found in a recent study that 95.4% of all certificates support the Online Certificate Status Protocol (OCSP), which allows a client to query the responsible CA in real-time for a certificate's revocation status via HTTP. As such, the browsed website will be exposed to the CA in question. From a privacy-standpoint this could be solved if the server *stapled* a recently fetched OCSP response with the served certificate. Unfortunately, only 35% of Alexa's top-one-million uses OCSP stapling [14].

Unless an OCSP response is stapled while visiting a website in a default configuration of the Tor browser, the status of a certificate is checked in real-time using OCSP. As such, any CA that issued a certificate for a website without OCSP stapling could instantiate a WO with an RTT-based resolution. Similarly, any actor that observes most OCSP traffic (which is in plaintext due to HTTP) gets the same capability. To better understand who could instantiate a WO based on OCSP we performed preliminary traceroute measurements[20] on the RIPE Atlas network towards four OCSP responders that are hosted by particularly large CAs: Let's Encrypt, Sectigo, DigiCert, and GoDaddy. Let's Encrypt and Sectigo are fronted by a variety of actors (mainly due to CDN caching), while DigiCert is fronted by a single CDN. Requests towards GoDaddy's OCSP responder always end-up in an AS hosted by GoDaddy.

C.7 Tor Exit Relays

Anyone can run a Tor exit relay and have it be used by all Tor users. Obviously, the operator of the exit relay can observe when its relay is used and the destination websites. At the time of writing, the consumed exit bandwidth of the entire Tor network is around 50 Gbit/s. This makes the necessary investment for an attacker that wishes to get a decent chunk of exit bandwidth more a question of stealthily deploying new exit relays than prohibitively large monetary costs.

C.8 Information Leaks

More sophisticated attackers can look for information leaks at the application, network, and operating system levels that allow them to infer that websites have been visited. Application level information leaks are particularly of concern for onion services: any observable state that can be tied to a new visitor is a WO for an onion visit (this is not the case for "regular" websites). Such state can include online status or the number of online users of a service, any observable activity with timestamps, a predictable caching structure, and so on. Similar information leaks can also occur on the network and operating system level [9, 22, 51].

18 https://web.archive.org/web/20190227165850/https://developers.cloudflare.com/logs/faq/

19 https://web.archive.org/web/20190227170903/https://policies.google.com/technologies/retention

20 Every RIPE Atlas probe used its configured DNS resolver(s). In total we requested 2048 WW-probes for one-off measurements.