# The Effect of DNS on Tor's Anonymity

Benjamin Greschbach* *KTH Royal Institute of Technology* bgre@kth.se

Tobias Pulls* *Karlstad University* tobias.pulls@kau.se

Laura M. Roberts* *Princeton University* laurar@cs.princeton.edu

Philipp Winter* *Princeton University* pwinter@cs.princeton.edu

Nick Feamster *Princeton University* feamster@cs.princeton.edu

***Abstract*—Previous attacks that link the sender and receiver of traffic in the Tor network ("correlation attacks") have generally relied on analyzing traffic from TCP connections. The TCP connections of a typical client application, however, are often accompanied by DNS requests and responses. This additional traffic presents more opportunities for correlation attacks. This paper quantifies how DNS traffic can make Tor users more vulnerable to correlation attacks. We investigate how incorporating DNS traffic can make existing correlation attacks more powerful and how DNS lookups can leak information to third parties about anonymous communication. We (*i*) develop a method to identify the DNS resolvers of Tor exit relays; (*ii*) develop a new set of correlation attacks (DefecTor attacks) that incorporate DNS traffic to improve precision; (*iii*) analyze the Internet-scale effects of these new attacks on Tor users; and (*iv*) develop improved methods to evaluate correlation attacks. First, we find that there exist adversaries that can mount DefecTor attacks: for example, Google's DNS resolver observes almost 40% of all DNS requests exiting the Tor network. We also find that DNS requests often traverse ASes that the corresponding TCP connections do not transit, enabling additional ASes to gain information about Tor users' traffic. We then show that an adversary that can mount a DefecTor attack can often determine the website that a Tor user is visiting with perfect precision, particularly for less popular websites where the set of DNS names associated with that website may be unique to the site. We also use the Tor Path Simulator (TorPS) in combination with traceroute data from vantage points co-located with Tor exit relays to estimate the power of AS-level adversaries that might mount DefecTor attacks in practice.**

## I. Introduction

We have yet to learn how to build anonymity networks that resist global adversaries, provide low latency, and scale well. Remailer systems such as Mixmaster [32] and Mixminion [12] eschew low latency in favor of strong anonymity. In contrast, Tor [14] trades off strong anonymity to achieve low latency; Tor therefore enables latency-sensitive applications such as web browsing but is vulnerable to adversaries that can observe traffic both entering and exiting its network, thus enabling deanonymization. Although Tor does not consider global adversaries in its threat model, adversaries that can observe traffic for extended periods of time in multiple network locations (*i.e.*, "semi-global" adversaries) are a real concern [15, 24]; we need to better understand the nature to which these adversaries exist in operational networks and their ability to deanonymize users.

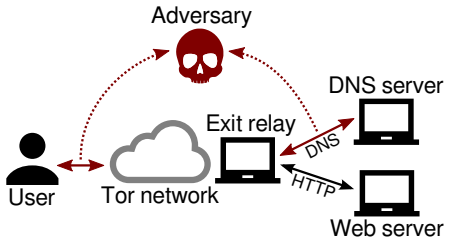



Fig. 1. Past traffic correlation studies have focused on linking the TCP stream entering the Tor network to the one(s) exiting the network. We show that an adversary can also link the associated DNS traffic, which can be exposed to many more ASes than the TCP stream.

Past work has quantified the extent to which an adversary that observes TCP flows between clients and servers (*e.g.*, HTTP requests, BitTorrent connections, and IRC sessions) can correlate traffic flows between the client and the entry to the anonymity network and between the exit of the anonymity network and its ultimate destination [24, 33]. The ability to correlate these two flows—a so-called *correlation attack*—can link the sender and receiver of a traffic flow, thus compromising the anonymity of both endpoints. Although TCP connections are an important part of communications, the Domain Name System (DNS) traffic is also quite revealing: for example, even loading a single webpage can generate hundreds of DNS requests to many different domains. No previous analysis of correlation attacks has studied how DNS traffic can exacerbate these attacks.

DNS traffic is highly relevant for correlation attacks because it often traverses completely different paths and autonomous systems (ASes) than the subsequent corresponding TCP connections. An attacker that can observe occasional DNS requests may still be able to link both ends of the communication, even if the attacker cannot observe TCP traffic between the exit of the anonymity network and the server. Figure 1 illustrates how an adversary may monitor the connection between a user and the guard relay, and between the exit relay and its DNS resolvers or servers. This territory—to-date, completely unexplored—is the focus of this work.

We first explore how Tor exit relays resolve DNS names. By developing a new method to identify all exit relays' DNS resolvers, we learn that Google currently sees almost 40% of all DNS requests exiting the Tor network. Second, we investigate which organizations can observe DNS requests that originate from Tor exit relays. To answer this question, we

*All four authors contributed substantially, and share first authorship. The names are ordered alphabetically.


NDSS '17, 26 February - 1 March 2017, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2017.23311

emulate DNS resolution for the Alexa top 1,000 domains from several ASes. We find that DNS resolution for half of these domains traverses numerous ASes that are not traversed for the subsequent HTTP connection to the web site. Next, we show how the ability to observe DNS traffic from Tor exit relays can augment existing website fingerprinting attacks, yielding perfectly precise DefecTor[1] attacks for unpopular websites. We further introduce a new method to perform traceroutes from the networks where exit relays are located, making our results significantly more accurate and comprehensive than previous work. Finally, we use the Tor Path Simulator (TorPS) [23] to investigate the effects of Internet-scale DefecTor attacks.

We demonstrate that DNS requests significantly increase the opportunity for adversaries to perform correlation attacks. This finding should encourage future work on correlation attacks to consider both TCP traffic and the corresponding DNS traffic; future design decisions should also be cognizant of this threat. Our work (*i*) serves as guidance to Tor exit relay operators and Tor network developers, (*ii*) improves state-of-the-art measurement techniques for analysis of correlation attacks, and (*iii*) provides even stronger justification for introducing website fingerprinting defenses in Tor. To foster future work and facilitate the replication of our results, we publish both our code and datasets.[2] In summary, we make the following contributions:

- We show how existing website fingerprinting attacks can be augmented with observed DNS requests by an AS-level adversary to yield perfectly precise DefecTor attacks for unpopular websites.
- We develop a method to identify the DNS resolver of exit relays. We find that Tor exit relays comprising 40% of Tor's exit bandwidth rely on Google's public DNS servers to resolve DNS queries.
- We quantify the extent to which DNS resolution exposes Tor users to additional AS-level adversaries who are not on the path between the sender and receiver. We find that for the Alexa top 1,000 most popular websites, many ASes that are on the paths between the exit relay and the DNS servers required to resolve the sites' domain names are not on the path between the exit relay and the website.
- We develop a new measurement method to evaluate the extent to which ASes are on-path between exit relays and DNS resolvers. We use the RIPE Atlas [39] platform to achieve previously unprecedented path coverage and accuracy for evaluating the capabilities of AS-level adversaries.

The rest of this paper is organized as follows. Section II presents background, and Section III relates our study to previous work. In Section IV, we shed light on the landscape of DNS in Tor. Section V discusses our DefecTor attacks, which we evaluate in Section VI. We then model the Internet-scale effect of our attacks in Section VII. Finally, we discuss our work in Section VIII and conclude the paper in Section IX.

[1]The acronym is short for DNS-enhanced fingerprinting and egress correlation on Tor.

[2]Our project page is available at https://nymity.ch/tor-dns/.

## II. Background

We now provide an introduction to the Tor network, website fingerprinting attacks, as well as how the Tor network implements DNS resolution.

*a) The Tor network:* The Tor network is an overlay network that anonymizes TCP streams such as web traffic. As of August 2016, it comprises approximately 7,000 relays and about two million users. The hourly published network consensus summarizes all relays that are currently online. Clients send data over the Tor network by randomly selecting three relays—typically called the guard, middle, and exit relay—that then form a virtual tunnel called a *circuit*. The guard relay learns the client's IP address, but not its web activity, while the exit relay gets to learn the client's web activity, but not its IP address. Relays and clients talk to each other using the Tor protocol, which uses 512-byte *cells*. Finally, each relay is uniquely identified by its fingerprint—a hash over its public key.

*b) Website fingerprinting attacks:* The Tor network encrypts relayed traffic as it travels from the client to the exit relay. Therefore, intermediate parties such as the user's Internet service provider (ISP) cannot read the contents of any packet. Tor does not, however, protect other statistics about the network traffic, such as packet inter-arrival timing, directions, and frequency. The ISP can analyze these properties to infer the destinations that a user is visiting. The literature calls this attack *website fingerprinting*.

Past work evaluated website fingerprinting attacks in two settings: a *closed-world* setting consists of a set of $n$ monitored websites, and the attacker tries to learn which among all $n$ sites the user is visiting with the notable restriction that the user can only browse to one of the $n$ websites. The *open-world* setting is more realistic: the user can browse to unmonitored sites in addition to the monitored sites. Unmonitored sites are, per definition, not known to the attacker; thus, the attacker's traffic classifier cannot train on unmonitored sites the user visits. The attacker's classifier can train on whatever unmonitored sites it chooses, as long as the classifier has not trained on an unmonitored site used for testing. Two relevant metrics in the open-world setting are *recall* and *precision*. Recall measures the probability that a visit to a monitored site will be detected, while precision measures the probability that a classification by the classifier of a visit to a monitored site (positive test outcome) is the correct one. Consider a classifier with 0.25 recall and 0.5 precision: on average, every fourth visit by the user to a monitored site will be detected, and half of the classifications by the classifier will be wrong. Errors can either classify a monitored site as unmonitored (lowering recall) or vice versa (lowering precision). Mistaking one monitored *website* for another is less likely [44].

Wa-kNN is a website fingerprinting attack by Wang *et al.* [45] that uses a k-nearest neighbor classifier with a custom weight-learning algorithm, WLLCC [44, § 3.2.5]. From packet traces between a Tor client and its guard, Wa-kNN extracts a number of features to classify each website. Useful features include the number of outgoing packets and bursts of packets in the same direction. In the training phase, WLLCC adjusts weights of features extracted from sites of known classes such that the distance between instances of the same site

(class) are minimized (collapsed). In the testing phase, Wa-kNN determines the distance of a testing traffic trace to all known training traces. The distance calculation results in the $k$ nearest classes: if all classes are the same, then the testing trace is classified as that class, otherwise it is classified as unmonitored. In the open-world setting, one class represents all unmonitored sites both during training and testing. By increasing $k$, Wa-kNN can trade decreased recall for increased precision. We set $k = 2$ when using Wa-kNN for higher recall since DefecTor is a highly precise attack.

Tor could eliminate website fingerprinting attacks with encrypted, constant-bitrate channels between a Tor client and its guard; other anonymity networks could use a similar technique. Unfortunately, the Tor network's limited spare capacity does not allow for such a throughput-intensive defense, but some research has worked on making this type of defense more efficient [8, 26, 38, 44].

*c) How Tor resolves DNS requests:* Tor clients must send DNS requests over Tor to prevent DNS leakage (*e.g.*, having a DNS request travel over an unencrypted channel as opposed to over Tor itself). Tor does not transport UDP traffic, but it implements a workaround to wrap DNS requests into Tor cells. Using the SOCKS protocol, applications can instruct the Tor client to establish a circuit to a given domain and port.[3] After the user types in a domain, say example.com, the Tor browser establishes a connection to the SOCKS proxy exposed by the local Tor client. The Tor client then selects an exit relay whose exit policy supports example.com and port 443. Next, the client sends a `RELAY_BEGIN` Tor cell to the exit relay, instructing it to first resolve example.com, and then establish a TCP connection to the resolved address at port 443 [13, § 6.2]. After successfully establishing a connection, the exit relay responds with a `RELAY_CONNECTED` cell. The client can then exchange data with its intended destination. Another type of cell, `RELAY_RESOLVE`, supports pure name resolution, without establishing a subsequent TCP connection [13, § 6.4]. The exit relay responds with a `RELAY_RESOLVED` cell.

Exit relays send their DNS requests to the system resolver, which Linux systems read from `/etc/resolv.conf`. Tor does not modify the system resolver and uses whatever the exit relay operator configured, such as the ISP's resolver, or public resolvers such as Google's public DNS resolver 8.8.8.8. As of August 2016, exit relays cache DNS responses to speed up repeated lookups. The caching layer for Tor clients, however, is off by default to prevent tracking attacks due to modified DNS responses [31].

## III. Related Work

This paper combines traffic analysis methods for correlation attacks with website fingerprinting attacks; we discuss related work in each of these two areas.

### A. Traffic analysis and correlation attacks

Tor's threat model excludes global adversaries [14], but the practical threat of such adversaries is an important question that the research community has spent considerable effort on answering. In 2004, when the Tor network comprised only 33 relays, Feamster and Dingledine investigated the practical threat that AS-level adversaries pose to anonymity networks [16]. The authors considered an attacker that controls an AS that is traversed both for ingress and egress traffic, allowing the attacker to correlate both streams. Using AS path prediction [19], Feamster and Dingledine found that powerful tier-1 ISPs reduce location diversity of anonymity networks. In 2007, Murdoch and Zieliński drew attention to IXP-level adversaries, a class of adversaries that was missing in Feamster and Dingledine's work [33]. Murdoch and Zieliński showed that IXP adversaries are able to correlate traffic streams, even in the presence of packet sampling rates as low as one in 2,000.

In 2013, Johnson *et al.* [24] presented the first large-scale study on the risk of Tor users facing relay-level and AS-level adversaries. The authors developed TorPS [23] that simulates Tor circuits for a number of user models. By combining TorPS with AS path prediction, Johnson *et al.* could answer questions such as the average time until a Tor user's circuit is deanonymized by an AS or IXP. Most recently in 2016, Nithyanand *et al.* [35] used AS path prediction to evaluate the practical threat faced by users in the top ten countries using Tor. In 2015, Juen *et al.* [27] examined the accuracy of path prediction algorithms that prior work [16, 24] used to estimate the threat of correlation attacks. The authors compared AS path predictions to millions of traceroutes, initiated from 25% of Tor relays by bandwidth at the AS level, and found that only 20% of predicted paths matched the paths observed in traceroutes. Juen *et al.* could not consider the reverse path in traceroutes. In 2015, Sun *et al.* [40] addressed this shortcoming; although past work treated routing as static, Sun *et al.* showed that the dynamic nature of Internet routing makes AS-level adversaries stronger than previous work had considered.

We improve on previous work in two significant ways: (*i*) we are the first to evaluate how DNS traffic exacerbates traffic correlation attacks, both in concept and in practice; and (*ii*) we develop and deploy a scalable, sustainable version of the measurement method proposed by Juen *et al.* [27]. Our method uses the volunteer-run RIPE Atlas measurement platform [39], as opposed to relying on relay operators to run third-party scripts. This approach allows us to fully automate our method and achieve previously unprecedented scale.

### B. Website fingerprinting

In 2009, Herrmann *et al.* [21] demonstrated the first website fingerprinting attack against anonymity systems—including Tor—in a closed-world setting. In 2011, Panchenko *et al.* [37] greatly improved on Herrmann *et al.*'s detection rate and provided insight into an open-world setting. In 2012, Cai *et al.* [10] improved on previous work by proposing an attack that used Hidden Markov Models to determine whether a sequence of page requests all come from the same site. The authors used an open-world setting for their evaluation. Wang and Goldberg [46] proposed an improved attack that employed a new method for data gathering. In 2014, Wang *et al.* [45] further improved on their results with a k-nearest neighbor classifier Wa-kNN and a custom weight-learning algorithm (WLLCC [44, § 3.2.5]) that in several rounds determine the optimal weights for features extracted from traffic traces. Cai *et al.* [9] determined which traffic features provide the

[3]The SOCKS versions 4a and 5 support connection initiation using domain names in addition to IP addresses.

most predictive power to detect websites, proved a lower bound of any defense that achieves a certain level of security, and provided a framework to investigate the performance of website fingerprinting attacks. Juarez *et al.* [25] showed that all previous attacks made several simplifying assumptions; the work suggested that attacks are still difficult to run outside a lab setting as an attacker will have to consider operating system differences, page changes, and background traffic. Recently, in 2016, Wang and Goldberg addressed many practical roadblocks to website fingerprinting, such as noisy data and maintaining a training set, further highlighting the need for website fingerprinting defenses in Tor [47].

Panchenko *et al.* [36] showed that web*page* fingerprinting (*i.e.*, fingerprinting of any page on a site) is significantly harder than web*site* fingerprinting (*i.e.*, fingerprinting of only the start page of a site). Hayes and Danezis proposed k-fingerprinting, an attack with notably better performance than Wa-kNN even in the face of defenses [20]. Their attack retains 30% accuracy in a closed-world setting against the WTF-PAD defense by Juarez *et al.* [26]—a prime candidate for deployment in Tor [38]—at the cost of 50% bandwidth overhead. Juarez *et al.* used Wa-kNN to evaluate WTF-PAD and set $k = 5$, as recommended by Wang *et al.* for an optimal trade-off between recall and the false positive rate.

In our work, we show how to correlate and use observed DNS requests in concert with website fingerprinting attacks, which significantly improves precision for web*site* fingerprinting. In scenarios where precision is paramount, DefecTor attacks pose an even bigger threat than website fingerprinting attacks from attackers that can observe even a modest fraction of DNS traffic from the Tor network. Mitigating the two DefecTor attacks that we present has implications for the design of website fingerprinting defenses: open-world evaluations of the website fingerprinting defense should minimize recall even when the website fingerprinting attack is tuned to sacrifice precision for recall. In the case of Wa-kNN, this means a low $k$: our results are based on $k = 2$.

## IV. Understanding the Landscape

Before explaining our attack, we need to better understand how Tor performs DNS resolution. We begin by investigating how common it is for adversaries to be able to observe DNS requests but *not* subsequent TCP connections of Tor users (Section IV-A). We then seek to understand how these results connect to the Tor network by determining the DNS resolvers used by exit relays (Section IV-B).

### A. Quantifying the additional AS exposure of DNS queries

Adversaries that can observe both DNS and subsequent TCP traffic (*e.g.*, the ISP of an exit relay) gain no benefit from seeing the client's DNS traffic, since TCP traffic is sufficient to mount correlation attacks [33]. In this work, we consider adversaries that can observe traffic entering the Tor network and *some* DNS requests exiting the network—such as requests addressed to DNS root servers—but *not* subsequent TCP traffic from exit relays. We first determine the prevalence of these adversaries by measuring the number of ASes that DNS queries traverse versus the number of ASes subsequent web traffic traverses.

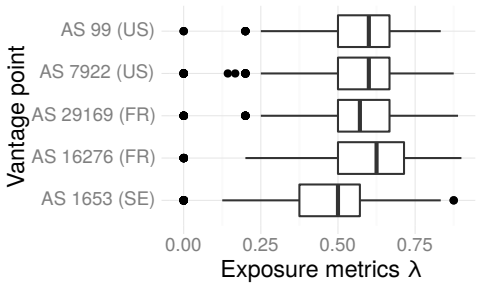



Fig. 2. Five box plots capturing the AS exposure metric $\lambda$ for Alexa's top 1,000 web sites. The box plots represent five autonomous systems in three countries.

We quantify the *exposure* of DNS traffic versus TCP traffic as follows. We begin with Alexa's top 1,000 [4], a list of the 1,000 most popular web sites as estimated by Alexa. For each site, we conducted two experiments. First, we ran a TCP traceroute to the site, targeting port 80 to mimic web traffic. Second, we determined the DNS delegation path for the website's DNS name using the `dig` command's `+trace` feature. The delegation path of a domain name, say www.example.com, is a hierarchy of authoritative DNS servers, such as the authoritative server for .com pointing to the authoritative server for example.com, which in turn points to the authoritative server responsible for www.example.com. We also ran UDP traceroutes to each server in the delegation path, targeting port 53 to mimic DNS resolution.[4] For both experiments, we then mapped all IP addresses in the traceroutes to AS numbers [41], generating both a set of traversed ASes for DNS traceroutes ($\mathscr{D}$) and a set of traversed ASes for web traceroutes ($\mathscr{W}$). Given these two sets for each of Alexa's top 1,000, we compute the fraction of ASes that are *only* traversed for DNS traffic, but *not* for web traffic ($\lambda$):

$$\lambda \in [0,1] = \frac{|\mathscr{D} \setminus \mathscr{W}|}{|\mathscr{D} \cup \mathscr{W}|}. \quad (1)$$

The metric approaches 1 as the number of ASes that are only traversed for DNS increases. For example, if $\mathscr{D} = \{1,2,3\}$ and $\mathscr{W} = \{2,3,4\}$, then $\lambda = \frac{|\{1,2,3\} \setminus \{2,3,4\}|}{|\{1,2,3\} \cup \{2,3,4\}|} = \frac{|\{1\}|}{|\{1,2,3,4\}|} = \frac{1}{4} = 0.25$. We determined $\lambda$ for each site in the Alexa top 1,000 from five autonomous systems in three countries.[5] One of our vantage points, the French OVH, is the most popular AS by exit bandwidth as of August 2016. It sees 10.98% of exit traffic, closely followed by AS 12876 (owned by the French Online) that sees 9.33% of exit traffic. Our experiment consisted of 5,000 traceroute runs, 4,773 (95.5%) of which succeeded, and 227 (4.5%) failed.

The result is illustrated in Figure 2, which shows five box plots capturing $\lambda$ values for Alexa's top 1,000 sites. The median of all 4,773 $\lambda$ values is 0.571, so for half of all runs, DNS-only ASes account for 57% or more of all traversed ASes. This result only applies to exit relays that do their own DNS resolution; for relays that use a third-party resolver, the ASes that are traversed between the exit relay and its

[4]The tool we developed for this purpose is available online at https://github.com/NullHypothesis/ddptr.

[5]The ASes are: OVH (France), Gandi (France), Karlstad University (Sweden), Princeton University (U.S.), and Comcast (U.S.).

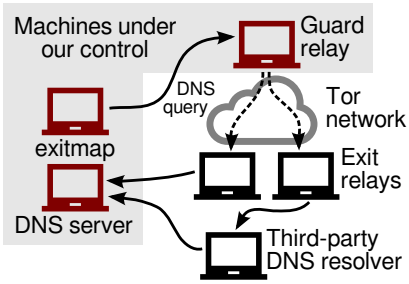



Fig. 3. Our method to identify the DNS resolvers of exit relays. Over each exit relay, we resolve relay-specific domain names that are under our control. By inspecting our DNS server logs, we can then identify the IP address of all exit relay resolvers.

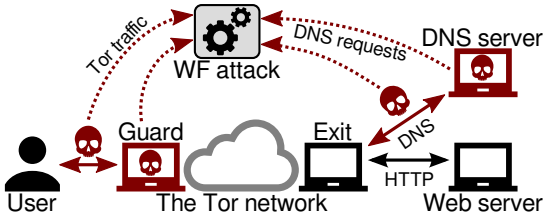



Fig. 5. An overview of the DefecTor attack. An adversary must monitor both ingress (encrypted Tor traffic) and egress (DNS request) traffic. A AS-level adversary between the client and its guard monitors ingress traffic. The same adversary monitors egress traffic between the exit and a DNS server, or the DNS server itself. Both ingress and egress traffic then serve as input to the DefecTor attack.

DNS resolver is the metric of interest. We further believe that relays in regions other than Western Europe or North America are likely to witness significantly different exposure of DNS queries because many websites outsource their DNS setup to providers such as Cloudflare whose points of presence are centered around Western Europe and North America. We conclude that adversaries that are unable to observe a Tor user's TCP connection still have many opportunities to see a TCP connection's corresponding DNS request. Such adversaries include (*i*) popular open DNS resolvers such as Google and OpenDNS, (*ii*) DNS root servers, and (*iii*) network adversaries located on the path to the previous two entities.

### B. Determining how Tor exit relays resolve DNS queries

Having shown that the Internet provides ample opportunity for AS-level adversaries to snoop on DNS traffic from exit relays, we now investigate how the exit relays in the Tor network resolve DNS queries in practice. Before this study, we only had anecdotal evidence (*e.g.*, from OpenDNS-powered error messages [49, § 4.1]) that some exit relays would occasionally show.

We identify the DNS resolver of all exit relays by using `exitmap` [48], a scanner for Tor exit relays. The tool automates running a task such as fetching a webpage over all one thousand exit relays, making it possible to see the Internet through the "eyes" of every single exit relay. Using `exitmap`, we resolve unique, relay-specific domains over each exit relay, to a DNS server under our control. Figure 3 illustrates this experiment. To improve reliability, we configured `exitmap` to use two-hop circuits instead of the standard three-hop circuits. The first hop was a guard relay under our control. Over each exit relay, we resolved a unique domain PREFIX.tor.nymity.ch. The prefix consisted of the relay's unique 160-bit fingerprint, concatenated to a random 40-bit string whose purpose is to prevent caching, so exit relays indeed resolve each query instead of responding with a cached element. We controlled the authoritative DNS server of tor.nymity.ch, so we could capture both the IP address and packet content of every single query for tor.nymity.ch.

An exit relay can either run its own resolver, as shown in the left exit relay in Figure 3; or rely on a third-party resolver, such as the one provided by its ISP, as shown in the right exit relay in Figure 3. If an exit relay runs its own resolver, we expect to receive a DNS request from the exit relay's IP address, but if an exit relay uses a third-party resolver, we expect to receive a request from an unrelated IP address. Having encoded relay-specific fingerprints in the query names, we are able to map queries to exit relays in such cases. We ran this experiment from September 2015 to May 2016, at least once a day.

On Linux relays, DNS resolution is controlled by the file `/etc/resolv.conf`, which contains up to three DNS resolvers that are queried in order. If the primary resolver does not respond in time, the system falls back to the second, and finally the third resolver. Our data suggests that several exit relays used different resolvers in subsequent `exitmap` scans—one relay, for example, used both Google's DNS resolver and one provided by its ISP. For our visualization, we only consider the first resolver we observed for an exit relay, which is likely but not guaranteed to be the primary resolver.

Figure 4 illustrates the fraction of DNS requests that four of the most popular organizations could observe. Google averages at 33%, but at times saw more than 40% of all DNS requests exiting the Tor network—an alarming number for a single organization. Second to Google is "Local"—exit relays that run their own resolver, averaging at 12%. Next is OVH, which used to be as popular as local resolvers, but slowly lost its share over time. Note that in contrast to Google, OVH does not run a public DNS server; the company's resolvers are only accessible to its customers. Finally, there is OpenDNS, which also runs public DNS resolvers. OpenDNS saw occasional spikes in popularity but always remained in the single digits. Apart from the illustrated top resolver setups, the distribution has a long tail, presumably consisting of many ISP resolvers.

## V. DefecTor Attacks

As with conventional correlation attacks, an attacker must observe traffic that is both entering and exiting the Tor network; in contrast to threat models from previous work, we incorporate DNS instead of only TCP traffic. Figure 5 illustrates our correlation attack; it requires the following building blocks:

- *Ingress sniffing:* An attacker must observe traffic that is entering the Tor network. The attacker can operate on the network level, as a malicious ISP or an intelligence agency. In addition, the attacker can operate on the relay level by running a malicious Tor guard relay. In both cases, the attacker can only observe encrypted data, so packet lengths and directions are the main inputs for website fingerprinting [36].

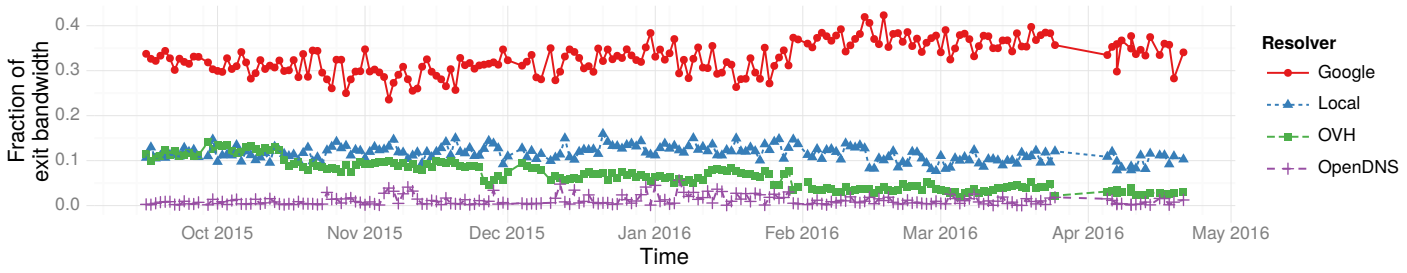



Fig. 4. The popularity of some of the most popular DNS resolvers of exit relays over time. The *y* axis depicts the fraction of exit bandwidth that the respective resolver is responsible for. Google's DNS resolver is by far the most popular, at times serving more than 40% of all DNS requests coming out of the Tor network. Google is followed by local resolvers, which average at around 12%. Once serving a fair amount of traffic, OVH dropped in popularity, and is now close to OpenDNS, an organization that runs an open resolver.

- *Egress sniffing:* To observe both ends of the communication, an attacker must also observe egress DNS traffic. We expect the adversary either to be on the path between exit relay and a DNS server or to run a malicious DNS resolver or server. We do not expect an attacker to run an exit relay because in this case conventional end-to-end correlation attacks are at least as effective as those we describe here [33].

We combine a conventional website fingerprinting attack operating on traffic from ingress sniffing with DNS traffic observed by egress sniffing, creating DefecTor attacks. Our attacks correlate the web*sites* observed by the website fingerprinting attack in ingress traffic with the web*sites* identified from DNS traffic.[6] Next, we describe how we simulate the DNS traffic from Tor exits, how we map DNS requests to websites, and finally present our two DefecTor attacks.

### A. Approximating DNS traffic from Tor exits

We first investigate the type and volume of DNS traffic that Tor's exit relays send. There are no logs of outgoing traffic from Tor exit relays available to us, and ethical considerations kept us from trying to collect them (*e.g.*, by operating exit relays and recording all the outgoing traffic). We therefore opt to approximate the DNS traffic emerging from Tor exit relays by (*i*) building a model of typical Tor users' website browsing patterns, (*ii*) collecting a minimally invasive dataset of DNS traffic, and (*iii*) accounting for the effects of DNS caching.

*1) Modeling which sites Tor users visit:* We first build a model to approximate which websites Tor users visit. As of July 2016, there are about 173 million active websites [34]; the Alexa ranking [4] gives insights into their popularity based on the browsing behavior of a sample of all Internet users. The distribution of the popularity of these websites has previously been fit to a power-law distribution based on the rank of the website [2, 11, 30]. For the pageview numbers of the Alexa top 10,000 websites, we found a power-law distribution to be a good fit as neither a log-normal nor a power-law distribution with exponential cutoff (*i.e.*, a truncated power-law distribution) offered significantly better fits. We used the Python `powerlaw` package [3] for fitting and picked a power-law distribution with an $\alpha$ parameter of 1.13. When varying the fitting parameter $x_{min}$ that determines beyond which minimum value the power-law behavior should hold in the provided data, we can get different $\alpha$ values. We made a conservative choice of picking this smaller $\alpha$ value as it underestimates the popularity of popular websites and therefore performs worse for the attacker.[7] Thus, we use a power-law distribution to model what websites Tor users visit. On the one hand, this might overestimate the popularity of higher-ranked websites such as Facebook and YouTube because we believe that Tor users—who tend to be privacy-conscious—are more likely to seek out alternatives than the typical Internet user. On the other hand, highly sensitive sites tend to be offered as onion services. We will discuss the implications of our model for browsing behavior later.

*2) Modeling how often Tor users visit each site:* Next, we determine how many websites Tor users visit in a certain time span. We approximated this number by setting up an exit relay whose exit policy included only the ports 80 and 443, so our relay would only forward web traffic. We then used the tool `tshark` to capture the timestamps of DNS requests—but no DNS responses. We made sure that our `tshark` filter did not capture packet payloads or headers, so we were unable to learn what websites Tor users were visiting. In addition, we patched `tshark` to log timestamps at a five-minute granularity. The coarse timing granularity allows us to publish this dataset with minimal privacy implications; Section VIII-A discusses the ethical implications of this experiment in more detail. We ran the experiment for two weeks, from May 15, 2016 to May 31, 2016, which allowed us to determine the number of DNS requests for 4,832 five-minute intervals. Figure 6 shows this time series, but for clarity we only plot May 25, 2016. The distribution's median is 105. The time series features several spikes; the most significant one counts 1,410 DNS requests. We repeated the same experiment with the so-called *reduced exit policy*[8] because it contains several dozen more ports and it is more popular among Tor relay operators; as of August 2016, it is used by 7.8% of exit relays by capacity. In comparison, the exit policy containing only port 80 and 443 only accounts for 1.5%. The reduced exit policy resulted in a median of 102 DNS requests per five minutes, so the difference between both policies is only three DNS requests.

We then interpolate these numbers to all Tor exit relays

[6] Our work can be understood as DNS-enhanced traffic correlation attack, *or* as DNS-enhanced website fingerprinting attack.

[7] Alexa's page-view numbers ignore multiple visits by the same user on the same day (see https://support.alexa.com/hc/en-us/articles/200449744), so the ranking might be slightly off when modeling website visit patterns.

[8] The reduced exit policy is available online at https://trac.torproject.org/projects/tor/wiki/doc/ReducedExitPolicy.

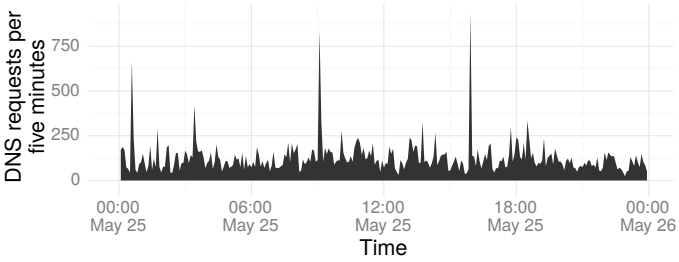



Fig. 6. The number of DNS requests per five-minute interval on our exit relay for May 25, 2016. Using a privacy-preserving measurement method, we only determined approximate timestamps and no content.

based on their published bandwidth statistics. While we measured a median of 105, the mean of the distribution was 119.3 per five minutes during a two-week period. From DNS statistics of the Alexa top one million websites (see Section V-B) we know that one website visit causes outgoing DNS requests for 10.3 domains on average (assuming a power-law distribution of site popularity as described above, and taking into account Tor's caching of pending DNS requests, ensuring that multiple requests sent by clients for the same domain name only result in one outgoing request by the exit). This means that our exit relay saw an average of 23.2 website visits per ten minutes. Assuming that the two main factors influencing the volume of DNS requests are a relay's *bandwidth* and its *exit policy*, and having shown that the exit policy does not significantly impact the number of DNS requests, we can scale this number up to the whole Tor network using the self-reported bandwidth statistics of exit relays. In particular, we use the bandwidth information reported in the extra-info descriptors that are available on CollecTor [42] and estimate the number of website visits on each of the about 1,200 exit relays active at that time. The resulting average number of websites visited through the Tor network is 288,000 per ten minutes. However, this number is merely an estimate because the interpolation is based on a single exit relay, and the bandwidth data of exit relays is self-reported and might therefore be incorrect.

Recently, Jansen and Johnson measured that the average number of active web (port 80 and 443) circuits in Tor amounts to about 700,000 per ten minutes [22, § 5.3]. Tor Browser, The Tor Project's fork of Firefox, builds one circuit per website entered in the URL bar. How long the circuit remains active depends on Tor Browser settings (primarily `MaxCircuitDirtiness` currently set to ten minutes) and how long TCP streams in the circuit are active: as long as at least one stream is active, the circuit remains active. Each time a new stream is attached to a circuit, the circuit's dirtiness timeout is reset. The number of active circuits serves as an upper bound for the number of websites visited over Tor: visiting different pages of a website will use the same circuit, and visiting a new website will construct a new circuit. Users visiting several pages of a website and websites with long-lived reoccuring connections, like Twitter and Facebook with continuously updating feeds, all lower the number of websites visited in Tor relative to the number of active circuits. For our model we consider the upper bound of 700,000 to be the number of websites visited through the Tor network per ten minutes. This is a conservative choice as more website visits increase the anonymity set of websites possibly visited by a Tor user—and therefore reduces the information an attacker can gain from observed DNS traffic. Later, we revisit the implications of our choice by both scaling the Tor network *up* to ten times its estimated size, and scaling it *down* to the size of 288,000 website visits per ten minutes that we got from our own interpolation described above.

*3) Modeling the effects of DNS caching at Tor exits:* To learn what DNS requests the adversary can see, we need to take into account caching of DNS responses. We ignore client-side DNS caching since it is disabled by default, as described in Section II. Exit relays, however, do cache DNS requests and we take it into account because all Tor clients using the same exit relay share its cache. In addition to their resolver's cache, exit relays maintain their own DNS cache[9] and enforce a minimum TTL of 60 seconds and a maximum TTL of 30 minutes.[10] We refer to this as Tor's *TTL clipping*. However, due to a bug that we identified,[11] the TTL of all DNS responses is set to 60 seconds.

If a Tor client attempts to resolve a domain that an exit relay has cached, the adversary will be unable to observe this request. However, the adversary can record all observed DNS requests over the past $x$ seconds, where $x$ is the maximum TTL value (*i.e.*, maintain a sliding window of length $x$). If a Tor client is attempting to resolve a domain name, the request is either cached or not. If it is not cached, the adversary will see it as a new, outgoing DNS request from the exit relay. If it is cached, it must have been resolved by the exit relay in the last $x$ seconds, and will therefore be in the sliding window. The sliding window technique allows the attacker to capture all relevant DNS requests, regardless of if they are cached or not. We assume that an adversary applies this sliding window technique and models the observable DNS traffic accordingly. The attacker observes a fraction of Tor's exit bandwidth for a specific window length, and together with our website visit frequency estimation, this triggers a number of website visits in our simulation. For each visit event, we randomly draw a website using the power-law website popularity distribution described above and put its DNS requests into the window. As we will see next, we do not need to simulate or consider the fact that the observed fraction of Tor exit bandwidth corresponds to many different exits with individual caches.

### B. Inferring website visits from DNS requests

Given a sliding window full of DNS requests, we investigate how this information can help determine whether a user has visited a website of interest. In April 2016, we visited the Alexa top one million websites five times, and collected all DNS requests that each visit of a website's frontpage generated. We refer to the data collected for one visit as a *sample*. We performed these measurements in five rounds from Karlstad University. Each round browsed all one million websites in random order before visiting the same website again. We used Tor Browser 5.5.4 and configured it *not to browse over Tor*: Tor Browser ensures that the browser behavior is identical to a Tor Browser user over Tor. By

[9]The code is available online at https://gitweb.torproject.org/tor.git/tree/src/or/dns.c?id=tor-0.2.9.1-alpha.

[10]The code is available online at https://gitweb.torproject.org/tor.git/tree/src/or/dns.c?id=tor-0.2.9.1-alpha#n209.

[11]The bug report is available online at https://bugs.torproject.org/19025.

Tab. 1. The percentage of websites in Alexa's top 1 million that use providers that restrict access from Tor [28].

| Description | Percentage |
|---|---|
| Website behind Cloudflare IP address | 6.44 |
| Domain on website uses Cloudflare | 25.81 |
| Domain on website uses Akamai | 33.86 |
| Domain on website uses Google | 77.43 |

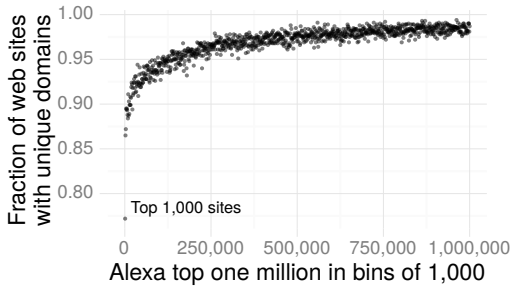



Fig. 7. The fraction of websites in Alexa's top one million that have at least one unique domain. We grouped all domains into 1,000 consecutive, non-overlapping bins of size 1,000. The vast majority of sites (96.8%) have unique domains.

not using Tor, we can bypass IP blacklists and CAPTCHAs that Tor users are frequently struggling with. Table 1 shows the percentage of websites in our dataset that are hosted by Cloudflare or Akamai. We might not be able to access these websites programatically over Tor because they block or filter exit relays, as identified by Khattak *et al.* [28]. We also include Google, which is prevalent in our dataset and restricts access to Tor users for Google's search.

We collected 2,540,941 unique domain names from a total of 60,828,453 DNS requests. The dataset contains 2,260,534 domains that are unique to a particular website, *i.e.*, are not embedded on any other top million site; we call these domains *unique domains*. Unique domains are particularly interesting because they reveal to the adversary what sites among the top million the user has visited. This is not possible for domains such as youtube.com, simply because many websites embed YouTube videos. Figure 7 shows the fraction of sites with unique domains for websites up to Alexa's top one million. We grouped all domains into 1,000 consecutive, non-overlapping bins of size 1,000. For 96.8% of all sites on the Alexa top one million there exists at least one unique domain. Interestingly, more popular websites are less likely to have a unique domain associated with them: only 77% of the first bin—the most popular 1,000 domains—contain at least one unique domain.

Table 2 shows summary statistics for the number of domains per website. At least half of the sites have ten domains per website, two of them are unique, suggesting that an adversary can identify many website visits by observing a single unique DNS request.

To evaluate the feasibility of mapping DNS requests to websites, we construct a naïve website classifier that maps the unique domains in a set of DNS requests to the corresponding website that contains a matching set of domains. With five-fold cross-validation on our Alexa top one million dataset (with five samples per site), we consider a closed world and an open world. In the closed world, the attacker can use samples from all sites in training; in the open world, some

Tab. 2. Summary statistics for the number of domains per website in the Alexa top 1 million. More than half of the sites embed two domains that are unique to that site.

| Domains | Median | Mean ± Stddev | Min. | Max. |
|---|---|---|---|---|
| Per site | 10 | 12.2 ± 11.2 | 1 | 397 |
| Unique per site | 2 | 2.3 ± 1.8 | 0 | 363 |

sites are unmonitored and therefore unknown (as per the fold). The closed-world evaluation yields 0.955 recall. In the open-world evaluation, we monitor the Alexa top 500,000 with five samples each and consider 433,000 unmonitored sites. The number of unmonitored sites is determined by our power-law distribution to represent a realistic base rate (for the entire Tor network) for evaluating our classifier: on average, for sites in the Alexa top 500,000 to be visited 2.5 million times, there will be about 433,000 visits to sites outside of Alexa's top 500,000. Our classifier does not take into account the popularity of websites. The open-world evaluation yields a recall of 0.947 for a precision of 0.984. By accounting for request order, per-exit partitioning of DNS requests, TTLs, and website popularity, we expect that classifying website visits from DNS requests can be made even more accurate. Further, a closed world is realistic in our setting: determining the DNS requests made by all 173 million active websites on the Internet is practical, even with modest resources. We use the conservative open world results when simulating the Tor network and the attacker's success in mapping DNS requests to websites. We conclude that for the purpose of identifying websites, observing DNS requests coming out of Tor is almost as effective as observing the web traffic itself.

### C. Classifiers for DefecTor attacks

We extend Wa-kNN from Wang *et al.* [45] (described in Section II) by having it take as input a list of sites derived from observing DNS requests. In particular, we implement two DefecTor attacks:

- `ctw` We "close the world" on a Wa-kNN classifier that we modified to consider only the distance to observed sites when calculating the $k$-nearest neighbors. The classifier still considers the distance to all unmonitored sites.
- `hp` When Wa-kNN classifies a trace as a monitored site, confirm that we observed the same site in the DNS traffic (ensuring *high precision*). If not, make the final classification unmonitored.

These approaches apply to any website fingerprinting attack. The `ctw` attack increases the effectiveness of conventional website fingerprinting attacks by making them more akin to a closed-world setting, where websites have known fingerprints and the world is often of limited size. Conceptually, the attack could also include a custom weight-learning run—training only on observed sites—but our initial results showed little to no gain, despite significant increases in testing time. We assume that this is due to the fact that some features of traffic traces are more useful than others, regardless of the training data [20]. The `hp` attack only produces a positive classification if both ingress and egress traffic are consistent, resulting in a simple but effective classifier.

## VI. Evaluating DefecTor Attacks

### A. Attack precision and recall

To evaluate our DefecTor attacks, we collected traffic traces in May 2016 using Tor Browser 5.5.4. We modified Tor Browser to not generate network traffic on launch (*i.e.*, check for updates, extensions, *etc.*), and we modified Tor (bundled with Tor Browser) to log incoming and outgoing cells. We then performed 100 downloads for each site in the Alexa top 1,000 and one download for each site in the Alexa top (1k,101k]. We randomly distributed these measurement tasks to a Docker fleet; each download used a fresh circuit without guard relay, and a fresh copy of Tor Browser for up to 60 seconds, in line with the recommendations by Wang and Goldberg [46, § 4]. We cached Tor's network consensus to minimize load on the network. We labeled a measurement as successful if we managed to resolve the domain of the site; we did not prune our dataset further, neglecting issues like Cloudflare CAPTCHAs, outliers, control cells, and localized domains [25]. Presumably, this means that we will underestimate the effectiveness of our attack, but we are primarily interested in the difference between website fingerprinting attacks and DefecTor attacks [46].

We perform ten-fold cross-validation for all of our experiments in the open world setting, monitoring 1,000 sites with 100 instances each, and 100,000 unmonitored sites. The 1:1 ratio between monitored traces and unmonitored traces is to ensure that for the classifier there is equal probability in the testing phase that a trace is a monitored or unmonitored site. In other words, the *base rate* is 0.5 in our experiments. Furthermore, for all experiments we specify the starting Alexa rank of the monitored sites *when simulating sites visited over the Tor network*. We always use the same sample data for website fingerprinting. The popularity of monitored sites is a key factor in the effectiveness of our attacks.

Figure 8 shows the recall and precision of our DefecTor attacks as a function of the percentage of observed Tor exit bandwidth by the attacker monitoring Alexa sites for sites whose ranks is 10,000 or less. For recall, both `ctw` and `hp` are bound by the percentage of exit bandwidth observed by the attacker (the percentage is an upper bound). It is simply not possible to identify a monitored site in the DNS traffic that the attacker does not see. At 100% of exit bandwidth, `ctw` sees better recall than `wf`. For `hp` the results suggest that:

$$\text{recall}_{\texttt{hp}} = \text{recall}_{\texttt{wf}} * \text{pct}. \tag{2}$$

This relationship only holds when observing DNS requests gives a clear advantage to `hp` in terms of precision over `wf` (see the following paragraph). For precision, the `hp` attack has an immediate gain over `wf` as soon as the attacker can observe *any exit bandwidth*. Although the `hp` attack has near-perfect precision, the `ctw` attack benefits from observing increasingly more exit traffic, nearly reaching the same levels as `hp` at 100% of the exit bandwidth.

Figure 9 shows recall and precision at 100% of observed Tor exit bandwidth as a function of the starting Alexa rank of monitored sites (we still monitor 1,000 sites). For popular websites (*i.e.*, websites with a high Alexa ranking), there is no difference between our attacks and the `wf` attack. This is because even with a window of only 60 seconds, it is almost certain that at least one user visited any of the most

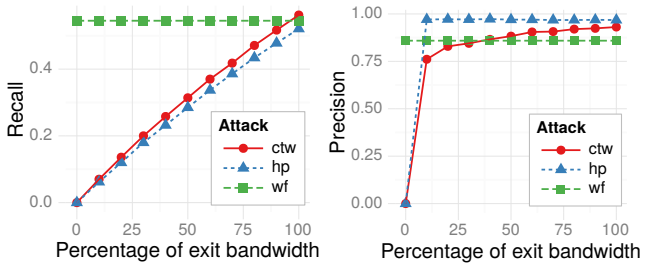



Fig. 8. Recall and precision for an open-world dataset with monitored sites at Alexa rank 10k and lower. We compare our DefecTor attacks (`ctw and hp`) to a conventional website fingerprinting attack (`wf`) for different percentages of observed exit bandwidth.

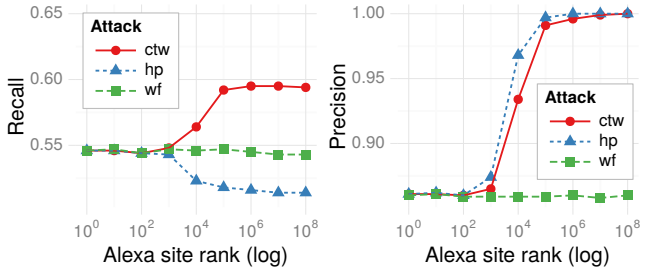


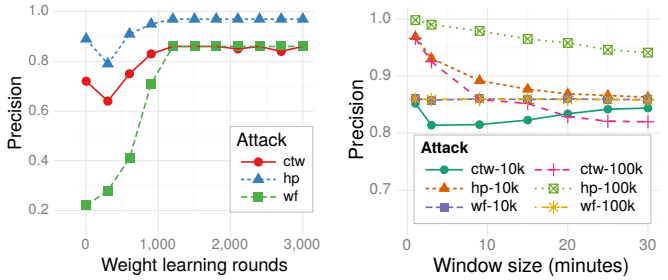

Fig. 9. The recall and precision when varying the starting Alexa rank of monitored sites for 100 percentage of exit bandwidth.

popular sites over Tor. For sites that rank 1,000 or lower (*i.e.*, less popular sites), both DefecTor attacks show a clear improvement in precision while `ctw` also shows improved recall—but only at 100% observed exit bandwidth, as shown in Figure 8. These results paint a bleak picture: an attacker that observes the vast majority of exit bandwidth can use the `ctw` attack as a perfectly precise attack with increased recall over a traditional `wf` attack. On the other hand, an attacker that can observe a small fraction of exit bandwidth can use the `hp` attack as a perfectly precise attack on relatively unpopular sites such as wikileaks.org, which had Alexa rank 10,808 on April 15, 2016. However, Equation 2 suggests that recall will be low.

### B. Sensitivity analysis

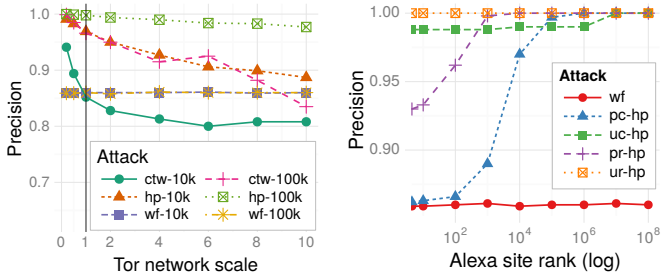
To better understand the extent and limitations of our attacks, we now study the sensitivity of our DefecTor attacks to website fingerprinting defenses, TTL clipping, the growth of the Tor network, and website popularity distribution. In this section, we assume that an adversary can observe Tor exit relays representing 33% of exit bandwidth (as observed on average by Google) and consider only precision (where we see clear gain from both our attacks). Note that the following results largely also apply to weaker attackers that observe a smaller fraction of exit bandwidth for the `hp` attack, but that the `ctw` attack is more sensitive in terms of precision to different bandwidth fractions, as shown above. Unless stated otherwise, we (*i*) perform our evaluation on websites starting from Alexa rank 10,000 upwards, (*ii*) use 2,500 weight-learning rounds, (*iii*) have a 60-second window size, (*iv*) a Tor network scale of 1.0, and (*v*) use the conservative power-law distribution from Section V-A1.

*1) Effect of website fingerprinting defenses:* The Tor Project is working on a website fingerprinting defense [38].




(a) Estimating the effect of website fingerprinting defenses.

(b) Effect of increasing the analysis time window due to TTL clipping.




(c) Effect of Tor network scale for Alexa ranks 10k and 100k.

(d) Effect of different website popularity distributions.

Fig. 10. The effect on attack precision. The defaults are: Alexa from top 10,000, 2,500 weight-learning rounds, 60-second window size, Tor network scale 1.0, and the conservative power-law distribution (pc) with $\alpha = 1.13$.

Most defenses produce bandwidth and/or latency overhead, with a significant increase in overhead as the defense becomes stronger. For example, Juarez *et al.* observe an exponential increase in bandwidth overhead as the protection of the WTF-PAD defense increases [26, § 4.3]. The goal is to find an optimum that provides strong protection while keeping the overhead tolerable for Tor users. To approximate the effect of fingerprinting defenses on DefecTor attacks, we use Wa-kNN with random weights and no weight-learning, which significantly reduces the effectiveness of the attack since some features (like indices of outgoing packets) are several orders of magnitude more useful than others [26].

Figure 10(a) shows the effect of weight-learning between 0 and 3,000 rounds. At few to no rounds, the precision for the `wf` attack is below 50%—a positive classification is more likely to be wrong than right—while there is a relatively small impact on the `hp` and `ctw` attacks. For recall, which is not shown in the figure, the bound and relationship is as in Equation 2: for `wf`, at zero rounds, recall is 0.055; for `hp` at zero rounds, recall is 0.019. These results suggest that for website fingerprinting defenses to be effective against DefecTor attacks, the defense must be tuned to cause low recall even if the parameters of website fingerprinting attacks are optimized for high recall.

*2) Effect of Tor's TTL clipping:* As discussed in Section V-A, due to a bug in Tor, all exit relays cache DNS responses for 60 seconds, regardless of the DNS response's TTL. Therefore, a sliding window covering the last 60 seconds of observed DNS requests suffices to capture all monitored sites through Tor (subject to the fraction of observed Tor exit bandwidth, and mapping DNS requests to sites).

Table 3 shows the TTL of DNS records in our Alexa top one million dataset from Section V-B both for the TTL as-is (raw) and when clipped (Tor). We calculate the intended values for TTL clipping, assuming that The Tor Project will fix the aforementioned bug. For each of these cases, we also consider TTLs for all unique domains, and for only the unique domain for each website with the lowest TTL. About half of all sites on Alexa's top one million have a unique domain with a TTL of 60 seconds or less; 48% of the raw unique TTLs are below 60 seconds and only 26% above 30 minutes. Fixing the Tor clipping bug is therefore not sufficient; to mitigate DefecTor attacks, the minimum TTL should be significantly increased. In this case, we find that Tor's TTL clipping has no effect on the median TTL, but significantly reduces the mean TTL.

Tab. 3. Median and mean DNS TTL values across Alexa top one million sites. Raw TTLs are unprocessed, as they appear in DNS lookup traces. Tor TTLs adhere to Tor's TTL clipping. Unique refers to the TTLs for unique domains; min unique only considers the unique domains with the minimum TTL for each website.

| TTLs | Median TTL (sec) | Mean TTL (sec) ± Stddev |
|---|---|---|
| Raw | 255 | 9,780.0 ± 42,930.5 |
| Tor | | 701.5 ± 755.3 |
| Unique raw | 900 | 13,022.2 ± 35,054.4 |
| Unique Tor | | 1,005.3 ± 789.6 |
| Min unique raw | 60 | 3,833.9 ± 11,073.6 |
| Min unique Tor | | 644.2 ± 763.8 |

Suppose that Tor eventually fixes the DNS TTL bug, requiring the attacker to monitor DNS lookups for a time interval equal to the maximum TTL of all unique domains for any monitored site. Figure 10(b) shows the effect on precision for different time intervals from 60 seconds to 30 minutes (Tor's `MAX_DNS_ENTRY_AGE` for keeping entries in an exit's DNS resolver cache), and for Alexa starting rank 10,000 and 100,000. For `ctw`, the time interval has a significant effect on both Alexa starting ranks, while `hp` is only affected for sites ranked 10,000 or higher; for less popular sites, the DNS lookup data still significantly improves fingerprinting precision, even with the larger window size.

*3) Effect of Tor network growth:* Figure 10(c) scales the size of the Tor network with respect to site visits from the estimated status quo to ten times its size, for Alexa starting rank 10,000 and 100,000. At twice its current size, the impact on DefecTor attacks is smaller than increasing the minimum TTL for DNS caching to three minutes, as shown in Figure 10(b). These results indicate that DefecTor attacks will remain practical for many sites in the Alexa top one million, even as the Tor network grows. If we overestimated the current Tor network size in the analysis in Section V-A2, our DefecTor attacks would have even higher precision, as shown by the data points to the left of the gray line in Figure 10(c).

*4) Sensitivity to website popularity distribution:* To explore the sensitivity of our results to different distributions in how users visit websites, we now evaluate the effectiveness of DefecTor attacks with four different website distributions:

- pc — A conservative power-law distribution (with $\alpha = 1.13$) that we manually fitted to the Alexa top 10,000 data, slightly underrepresenting the popularity of top Alexa sites. We described this distribution in Section V-A1.
- pr — A realistic power-law distribution (with $\alpha = 1.98$)

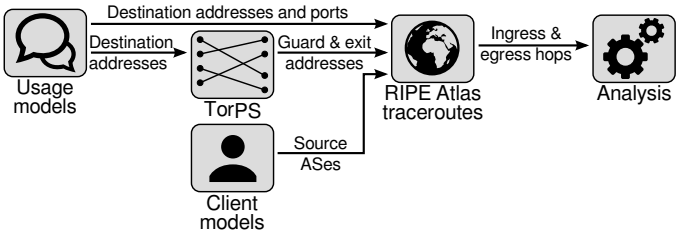



Fig. 11. The relation among our simulation components. Our goal is to determine the ASes a Tor user's traffic traverses into and out of the Tor network. Duplicate ASes on both sides can deanonymize streams.

that is the best fit according to the Python `powerlaw` library by Alstott *et al.* [3] for the Alexa top 10,000 data.

uc A conservative uniformly random distribution that only considers one million active websites browsed over Tor.

ur A realistic uniformly random distribution that considers 173 million active websites, as reported by Netcraft in July 2016 for the Internet [34].

Figure 10(d) shows the effect on the precision of the `hp` attack for the different distributions as we vary the starting Alexa rank. The uniform distributions always have nearly perfect precision. The difference between the two power-law distributions is about one order of magnitude in terms of starting Alexa rank: the realistic distribution gets near perfect at 1,000 and the conservative at 10,000. We conclude that DefecTor attacks are perfectly precise for unpopular sites because it is unlikely that more than one person is browsing a monitored site within the timeframe determined by the window length.

## VII. Internet-scale analysis

In the preceding sections we have presented our DefecTor attacks and evaluated their effectiveness, but we have yet to understand what entities can mount them. In this section, we aim to quantify the likelihood that any AS is in a position to mount DefecTor attacks.

### A. Approach

Figure 11 summarizes our simulation approach, which we detail in the next section. In short, we model the activity of Tor users and simulate their path selection using TorPS [23]. TorPS returns guard and exit relays, which we then feed as input—together with source ASes and destination addresses—into our framework that runs traceroutes from RIPE Atlas nodes. The rest of this section describes our approach in detail.

*1) Attack model:* We assume that an AS can mount DefecTor attacks if it can see both traffic *entering* the Tor network and DNS traffic *exiting* the Tor network. Recall that an exit relay can perform DNS resolution in two ways; by running a local resolver, or by relying on a third-party resolver, such as its ISP's or Google's public resolver. In the case of exit relays that perform local resolution, an effective position for an attacker is both (*i*) anywhere on the AS path between a Tor client and its guard relay; and (*ii*) anywhere on the path between an exit relay and any of the name servers the exit has to communicate with to resolve a domain. These name servers include the full DNS delegation path, *i.e.*, a root name server plus subsequent name servers in the DNS hierarchy. All ASes along the path from the exit relay to the name servers will be able to see the domain names that the exit relay is querying. For exit relays that rely on third-party resolvers, the adversary instead has to be on the path between the exit relay and its DNS resolver.

*2) Simulating Tor user activity with TorPS:* To measure the likelihood that an AS can be in a position to perform a DefecTor attack, we use TorPS [23]—short for Tor Path Simulator—which mimics how a Tor client constructs circuits (see "TorPS" in Figure 11). TorPS takes as input archived Tor network data [42] and usage models, which are sets of IP addresses that Tor clients talk to—*e.g.* web servers. Given this input, TorPS then simulates for a configurable number of "virtual" Tor clients the way they would select guard and exit relays. TorPS is based on the Tor stable release in version 0.2.4.23. For each simulated client, TorPS uses one guard; this guard selection expires after 270 days. We use TorPS to simulate the behavior of 100,000 Tor clients for the entire month of March 2016.

We need to place our simulated Tor clients into an AS (see "Client models" in Figure 11). We selected clients in major ISPs in the top-five most popular countries of Tor usage according to Tor Metrics [43]. As of August 2016, the top five countries are the U.S., Russia, Germany, France, and the U.K. For the U.S., we chose Comcast (AS 7922); for Russia, Rostelecom (AS 42610); for Germany, Deutsche Telekom (AS 3320); for France, Orange (AS 3215); and for the U.K., British Telecom (AS 2856).

Having placed simulated Tor clients into ASes, we now model their activity over Tor (see "Usage models" in Figure 11). We model each client to have visited several websites every day in March 2016.[12] At 9 a.m. EST, the client visits mail.google.com and www.twitter.com. At 12 p.m. EST, the client visits calendar.google.com and docs.google.com. At 3 p.m. EST, the client visits www.facebook.com and www.instagram.com. Finally, at 6 p.m. EST, the client visits www.google.com, www.startpage.com, and www.ixquick.com, and at 6:20 p.m. EST, the client visits www.google.com, www.startpage.com, and www.ixquick.com again. Each of the 100,000 simulated Tor clients thus had $12 \cdot 31 = 372$ opportunities to be compromised given 31 days and 12 site visits per day. TorPS provided a new circuit every ten minutes, regardless of how many distinct connections the client made to different sites; it did not provide a new circuit for different websites if the client visited the group of sites within the same ten-minute window.

For simplicity, we assume that only one DNS request occurs every time a client visits a site. For example, in our model, at 9 a.m. one DNS request will occur for mail.google.com and one DNS request will occur for www.twitter.com. At 6 p.m. three DNS requests will occur, and at 6:20 p.m. those same three DNS requests will occur again. For now, we do not take embedded requests (*i.e.* for embedded website content such as YouTube videos) or caching into account.

[12] We modeled our client behavior off of the "Typical" model that Johnson *et al.* [24, § 5.1.2] used.

Tab. 4. The coverage of RIPE Atlas nodes that are co-located with Tor guard and exit relays as of May 2016.

| Atlas probe coverage | Tor guard ASes (%) | Tor exit ASes (%) |
|---|---|---|
| By bandwidth | 73.59 | 57.53 |
| By number | 50.69 | 52.25 |

*3) Inferring AS-level paths using traceroutes and* `pyasn`*:* Our Internet-scale analysis also requires learning the AS-level paths from each client to its guard, and from its exit to the destination (see "RIPE Atlas traceroutes" in Figure 11). We decided against the commonly applied AS path inference because Juen *et al.* showed that it can be quite inaccurate [27]. Traceroutes, in contrast, yield significantly more accurate paths, but are difficult to run from Tor relays: Past work involved asking relay operators to run traceroutes on behalf of the researchers [27, § 4]. This approach yielded traceroutes from relays representing 26% of exit bandwidth, but does not scale well. Instead of running traceroutes from Tor relays, we leverage the RIPE Atlas [39] platform, a volunteer-run network measurement platform consisting of thousands of lightweight and geographically spread *probes* that can be used as vantage points for traceroutes. Our key observation is that RIPE Atlas has probes in many ASes that also have Tor relays. We leverage this observation by designing measurements to run traceroutes from Atlas probes that are located in the same AS as exit relays, to each of the destinations in question.

Table 4 shows that for a day in May 2016, we found that RIPE Atlas had probes in 52% of ASes that contain exit relays, and in 51% of ASes that contain Tor guard relays. More importantly, we found that Atlas ASes cover 58% of exit *bandwidth* and 74% of guard *bandwidth*. This statistic is important given that Tor clients select relays weighted by their bandwidth, and the bandwidth of Tor relays is not uniformly distributed. Given the growth of both Tor and Atlas, we expect these numbers to increase in the future. In addition to Atlas, we also considered using PlanetLab [1] to initiate traceroutes, but unfortunately most PlanetLab nodes are located in research and education networks [6], and are thus not suited for performing our measurements.

We performed traceroutes from the five Tor client ASes outlined above to all their respective guard relay IP addresses that TorPS determined. To measure the paths from exit relays to their DNS resolvers, we performed the following traceroutes, simulating four different DNS configurations:

- *ISP DNS:* To investigate the scenario in which an exit relay uses its ISP's resolver, we chose to represent this as the resolver being in the same AS as the exit relay. Thus, no traceroutes were necessary for this experiment. We acknowledge that this is not necessarily the case, but assume that it holds for the majority of exit relays.
- *Google DNS:* This scenario represents an exit relay using Google's public resolver. To measure the AS path, we perform traceroutes from a RIPE Atlas node in the AS of the exit relay to Google's public DNS resolver, *i.e.*, 8.8.8.8.
- *Local DNS:* To measure the paths that would be traversed if an exit relay were running its own, local resolver (*e.g.*, the popular service `unbound`), we used the command line tool `dig` with the `+trace` option to determine the iterative resolution process. We tracked all name server IP addresses from referrals at each level of the delegation path, and performed traceroutes to those IP addresses.
- *Status quo:* This scenario represents the state of the Tor network as of March 2016, a combination of the above three configurations. Recall that in Section IV-B, we determined the IP addresses of the resolvers that exit relays use. We ran traceroutes to these very IP addresses. For the exit relays that used several resolvers during March, we randomly assigned one to the relay. We ended up having data for 73% of the exit relays that TorPS ended up picking.[13]

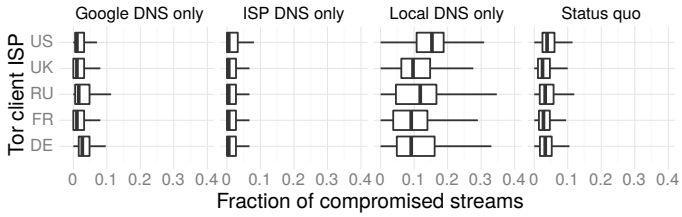
We then mapped each IP address in every traceroute to its corresponding AS (see "Analysis" in Figure 11). The Python module `pyasn` [5] relies on BGP routing tables to perform these mappings; by using a routing table that coincides with the time when we performed our traceroutes, we can obtain accurate AS-level mappings. This method is subject to inaccuracies due to BGP route hijacks or leaks, but we expect those events to be relatively unlikely for the time period and IP prefixes that we are concerned with.

*4) Putting it all together:* We consider the same two security metrics that Johnson *et al.* [24, § 4.2] originally proposed; we aim to estimate (*i*) the fraction of compromised streams per simulated Tor user, and (*ii*) the amount of time it would take for the first compromise to occur. For both metrics, we consider the four DNS configurations outlined above. Our simulation can reveal the respective average threat that a given DNS configuration poses for Tor users.

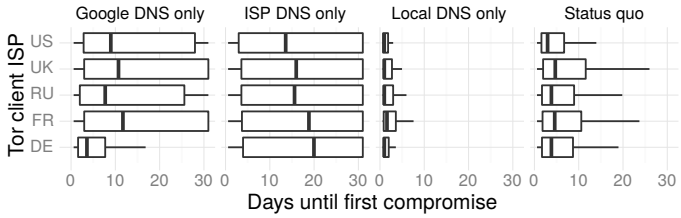
Each traceroutes run yielded two sets of ASes, one from the Tor clients' ASes to their guard relays, and one from approximately half of the exit relays' ASes to the different destinations, which depend on the exit relays' DNS configurations. We intersect both AS sets (the "ingress" and "egress" hops of Figure 11) and classify a website visit as compromised if the intersection is non-empty. As stated earlier, for some exit relays we did not have associated AS-level paths to a particular destination, either due to a lack of co-located RIPE Atlas probes, or because of missing traceroute information. In these cases, we checked if the exit AS had the potential to launch an attack by itself, and if not, we labelled the stream as uncompromised to err on the conservative side.

To compute the fraction of compromised streams, we counted the streams that were compromised for every simulated user out of a possible maximum of 372. To compute the time until first compromise, we determined the first stream in which the user was compromised, took its timestamp, and calculated the offset from the beginning of March 1, 2016. For users that were not compromised during the month of March, we assigned the maximum value of 31 days as the time until first compromise, which is reflected in the plots in our next section. Users who were compromised immediately would have a value of 0, signifying that they were compromised at the very beginning of March 1.

[13]The missing 27% are due to the churn in exit relays. Since we did not run our `exitmap` experiment each hour, we were bound to miss some exit relays.




(a) The fraction of compromised streams of simulated Tor clients.




(b) The time until simulated Tor clients got first compromised.

Fig. 12. The fraction of compromised streams and the time until first compromise for our simulated Tor clients. We placed these clients in five popular client ASes in the U.S., the U.K., Russia, France, and Germany. For exit relays, we consider the status quo (on the very right) plus three hypothetical DNS configurations for all exit relays. We do not plot outliers beyond the box plots' whiskers. In both experiments, the safest configuration is "ISP DNS only," *i.e.* have all exit relays use their ISP's DNS resolver.

### B. Results

Figures 12(a) and 12(b) illustrate our results as box plots. Each figure contains four subfigures, one for each DNS configuration. Each box plot contains five rows, one for each Tor client AS. For clarity, we did not plot any outliers beyond the box whiskers. For the fraction of compromised streams, an ideal setup has its median at 0. For the time until first compromise, an ideal median is 31. Both figures show that the "ISP DNS only" setup is the safest for Tor users, *i.e.*, it exhibits on average the least number of compromised streams while also on average counting the most days until compromise. This setup is closely followed by "Google DNS only," the status quo, and finally "Local DNS only," which fares worse than all other setups. We expected "ISP DNS only" to do best because if all exit relays use their ISP's resolvers, there is only one AS to contend with on the egress side—the exit relay's. The Google setup fares similarly well; most likely because of Google's heavily anycast infrastructure which minimizes the number of AS hops. The status quo does significantly better than the "Local DNS" results, presumably because only around 12% of Tor exit relays actually do their own resolution. The large variance observed in Figure 12(b) for "ISP DNS" and "Google DNS" is due to using 31 days as a placeholder for simulated clients who were never compromised. However, a safe configuration against AS-level adversaries, which our figures capture, is not necessarily the best setup for Tor users. For example, ISP-provided DNS resolvers can be misconfigured, subject to censorship, or simply be a forwarder to Google's resolver, which already serves numerous exit relays and whose centralization poses a threat to the anonymity of Tor users. We will explore this trade-off in greater detail in Section VIII.

Interestingly, we find differences in our five client ASes. These differences are particularly striking in Figure 12(b). For "Google DNS only," the median time until compromise differs by around seven days between DE and UK, and around eight days between DE and FR. For "ISP DNS only," the median time until compromise differs by around six days between US and DE, and around five days between US and FR. Also, we notice that DE fares worse than the others in the "Google DNS only" scenario and better than the others in the "ISP DNS only" scenario. We conclude that the location of Tor clients matters and should be considered in future traffic correlation studies.

## VIII. Discussion

In this section, we briefly discuss the ethics of our research and ways to defend against DefecTor attacks.

### A. Ethical considerations

In Section V-A2, we discussed setting up an exit relay to determine the number of DNS requests per five minute interval. Since our exit relay was forwarding traffic of Tor users, we contacted Princeton University's institutional review board (IRB) before running the experiment. Our IRB deemed that this research did not fall within the realm of human subjects research. In addition to contacting our IRB, we adhered to The Tor Project's ethics guidelines [29]. Specifically, (*i*) we ensured that we only collected data that is safe to publish, (*ii*) we only collected data we needed, and (*iii*) we limited the granularity of the data to minimize the likelihood of reidentification. The risk to Tor users of this experiment is negligible. As for the benefits, by conducting this experiment, we can improve our understanding of the risks that DNS poses to the anonymity of Tor users and use this understanding to improve protection for Tor users in the future. Thus, we believe that the benefits of our experiment outweigh the risks.

### B. Defending against DefecTor attacks

We now discuss ways to defend against DefecTor attacks. We distinguish between short-term solutions that can be implemented quickly (Section VIII-B1), and long-term solutions that need significantly more work (Section VIII-B2). Our discussion of countermeasures is not comprehensive, and we defer a more detailed analysis to future work.

*1) Short-term solutions:* Exit relay operators face a dilemma: they must either operate their own resolver, which exposes DNS queries to network adversaries; or, they must use a third-party DNS resolver, which exposes DNS queries to a third party. Clearly, the goal is to minimize exposure of DNS requests, but there are several dimensions to this. In lieu of substantial DNS protocol improvements, we envision three extreme design points, in which *all* exit relays use (*i*) Google's DNS resolver; (*ii*) their own, local resolver; or (*iii*) the resolver provided by their ISP.

If all exit relays were to use Google's public resolver, the company would obtain metadata about the activity of all Tor users, which runs counter to Tor's design goal of distributing trust. We clearly should avoid this scenario. Fifield *et al.*'s [18] censorship circumvention system meek used to use Google's cloud infrastructure to tunnel the traffic of censored users up until May 2016 [17]. While the system was operational, thousands of meek clients selected exit relays that use Google's public resolver, which means that Google saw both traffic entering and, partially, exiting the Tor network, allowing the company to mount DefecTor attacks. Next, consider a Tor

network that only uses local resolvers. In this case, Tor is fully independent of third-party resolvers, at the cost of each iterative DNS query being exposed to a diverse set of ASes in the network, allowing several parties to learn the DNS queries of Tor users. Finally, all exit relays could simply use their ISP-provided resolver. This would minimize the network exposure of DNS requests as resolvers are frequently in the same AS as exit relays, and AS-level adversaries would be unable to distinguish between DNS requests from exit relays and unrelated ISP customers. However, this setup introduces the possibility of misconfigured and censored DNS resolvers [49, § 4.1]. Besides, just a few ASes—OVH, for example—host a disproportionate amount of exit relays, turning them into the very centralized data sinks that Tor aims to avoid.

Considering the above, we believe that exit relay operators should avoid public resolvers such as Google and OpenDNS. Instead, they should either use the resolvers provided by their ISP, or run their own, particularly if the operator's ISP already hosts many other exit relays. Local resolvers can further be configured to minimize information leakage, by enabling QNAME minimization [7]. There likely is a measurable performance difference between a local resolver and Google's resolver, but we believe that this difference pales in comparison to other performance issues in Tor such as head-of-line blocking.

Finally, Tor can fix the Tor clipping bug we discovered and consider significantly increasing the minimum TTL for the DNS cache at exit relays to make DefecTor attacks less precise. This adjustment requires finding the longest acceptable TTL that does not have a notable negative detriment to user experience. Further, as soon as the clipping bug is fixed, website operators of sensitive websites can opt to increase the TTL of their DNS records.

*2) Long-term solutions:* Additional practical defenses are on the horizon. Zhu *et al.* [50] proposed T-DNS, which employs several TCP optimizations to transport the DNS protocol over TLS and TCP. The TLS layer provides confidentiality between exit relays and their resolvers. Finally, site operators whose users are particularly concerned about safety should offer an onion service as an alternative. Facebook, for example, set up facebookcorewwwi.onion. When connecting to the onion service, Tor users never leave the Tor network, and hence do not need DNS—as long as the onion service does not embed non-onion service content.

Deploying defenses against website fingerprinting attacks in Tor should be an important long-term goal, as well. Although growing the Tor network will help defend against DefecTor attacks to some degree, the most important change is to deploy defenses against these attacks. Since DefecTor attacks significantly increase precision of website fingerprinting attacks, defenses should be designed to significantly reduce the recall of website fingerprinting attacks, even when the website fingerprinting attack is configured to sacrifice precision for recall.

## IX. Conclusion

In this paper, we have demonstrated how AS-level adversaries can use DNS traffic from Tor exit relays to launch more effective website fingerprinting attacks, to learn what websites Tor users are visiting. Mapping DNS traffic to websites is highly accurate even with simple techniques, and improves the precision when monitoring relatively unpopular websites. We further developed a method to identify the DNS resolver for each Tor exit relay, and found that a set of exit relays comprising 40% of all Tor exit relay bandwidth use the Google public DNS servers. Although this concentration of DNS query traffic reduces the expanse of ASes that can see DNS query traffic emanating from exit nodes, this configuration nonetheless gives a single administrative entity considerable visibility into the traffic that is exiting the Tor network. Tor relay operators should take steps to ensure that the network maintains more diversity into how exit relays resolve DNS domains. To mitigate the risk of website fingerprinting attacks in light of our work, we suggest that local DNS resolvers on Tor exit relays implement privacy-preserving techniques such as DNS QNAME minimization, which minimizes the amount of information about the domain name that each iterative query contains. We publish all our code, data, and replication instructions on our project page, which is available online at https://nymity.ch/tor-dns/.

## Acknowledgments

We want to thank Jedidiah R. Crandall for providing infrastructure for our measurements, Aaron Johnson for help with TorPS, Robayet Nasim for running DefecTor experiments, and Tom Ritter and the anonymous reviewers for providing helpful feedback. This research was supported in part by the Swedish Foundation for Strategic Research grant SSF FFL09-0086; the Swedish Research Council grant VR 2009-3793; the Swedish Internet Fund grant "Hoppet till Tor"; the National Science Foundation Awards CNS-1540055 and CNS-1602399; and the Center for Information Technology Policy at Princeton University.

## References

[1] "PlanetLab – an open platform for developing, deploying, and accessing planetary-scale services." URL: https://www.planet-lab.org (Cited on p. 12)

[2] K. Ali and M. Scarr, "Robust methodologies for modeling web click distributions," in *WWW*. ACM, 2007. URL: https://nymity.ch/tor-dns/pdf/Ali2007a.pdf (Cited on p. 6)

[3] J. Alstott, E. Bullmore, and D. Plenz, "powerlaw: A Python package for analysis of heavy-tailed distributions," *PLoS ONE*, vol. 9, no. 1, 2014. URL: https://nymity.ch/tor-dns/pdf/Alstott2014a.pdf (Cited on pp. 6 and 11)

[4] Amazon Web Services, "Alexa top sites." URL: https://aws.amazon.com/alexa-top-sites/ (Cited on pp. 4 and 6)

[5] H. Asghari, "pyasn – Python IP address to autonomous system number lookup module." URL: https://github.com/hadiasghari/pyasn (Cited on p. 12)

[6] S. Banerjee, T. G. Griffin, and M. Pias, "The interdomain connectivity of PlanetLab nodes," in *PAM*. Springer, 2004. URL: https://nymity.ch/tor-dns/pdf/Banerjee2004a.pdf (Cited on p. 12)

[7] S. Bortzmeyer, "RFC 7816 – DNS query name minimisation to improve privacy," Mar. 2016. URL: https://tools.ietf.org/html/rfc7816 (Cited on p. 14)

[8] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A congestion sensitive website fingerprinting defense," in *WPES*. ACM, 2014. URL: https://nymity.ch/tor-dns/pdf/Cai2014a.pdf (Cited on p. 3)

[9] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *CCS*. ACM, 2014. URL: https://nymity.ch/tor-dns/pdf/Cai2014b.pdf (Cited on p. 3)

[10] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *CCS*. ACM, 2012. URL: https://nymity.ch/tor-dns/pdf/Cai2012a.pdf (Cited on p. 3)

[11] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, 2009. URL: https://arxiv.org/pdf/0706.1062 (Cited on p. 6)

[12] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Security & Privacy*. IEEE, 2003. URL: https://nymity.ch/tor-dns/pdf/Danezis2003a.pdf (Cited on p. 1)

[13] R. Dingledine and N. Mathewson, "Tor protocol specification." URL: https://spec.torproject.org/tor-spec (Cited on p. 3)

[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security*. USENIX, 2004. URL: https://nymity.ch/tor-dns/pdf/Dingledine2004a.pdf (Cited on pp. 1 and 3)

[15] S. Farrell and H. Tschofenig, "RFC 7258 – pervasive monitoring is an attack," May 2014. URL: https://tools.ietf.org/html/rfc7258 (Cited on p. 1)

[16] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *WPES*. ACM, 2004. URL: https://nymity.ch/tor-dns/pdf/Feamster2004a.pdf (Cited on p. 3)

[17] D. Fifield, "meek-google suspended for terms of service violations (how to set up your own)," Jun. 2016. URL: https://lists.torproject.org/pipermail/tor-talk/2016-June/041699.html (Cited on p. 13)

[18] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," *PoPETS*, vol. 2015, no. 2, 2015. URL: https://nymity.ch/tor-dns/pdf/Fifield2015a.pdf (Cited on p. 13)

[19] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, 2001. URL: https://nymity.ch/tor-dns/pdf/Gao2001a.pdf (Cited on p. 3)

[20] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *USENIX Security*. USENIX, 2016. URL: https://nymity.ch/tor-dns/pdf/Hayes2016a.pdf (Cited on pp. 4 and 8)

[21] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *CCSW*. ACM, 2009. URL: https://nymity.ch/tor-dns/pdf/Herrmann2009a.pdf (Cited on p. 3)

[22] R. Jansen and A. Johnson, "Safely measuring Tor," in *CCS*. ACM, 2016. URL: https://nymity.ch/tor-dns/pdf/Jansen2016a.pdf (Cited on p. 7)

[23] A. Johnson, "The Tor path simulator." URL: https://github.com/torps/torps (Cited on pp. 2, 3, and 11)

[24] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on Tor by realistic adversaries," in *CCS*. ACM, 2013. URL: https://nymity.ch/tor-dns/pdf/Johnson2013a.pdf (Cited on pp. 1, 3, 11, and 12)

[25] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *CCS*. ACM, 2014. URL: https://nymity.ch/tor-dns/pdf/Juarez2014a.pdf (Cited on pp. 4 and 9)

[26] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *ESORICS*. Springer, 2016. URL: https://nymity.ch/tor-dns/pdf/Juarez2016a.pdf (Cited on pp. 3, 4, and 10)

[27] J. Juen, A. Johnson, A. Das, N. Borisov, and M. Caesar, "Defending Tor from network adversaries: A case study of network path prediction," *PoPETS*, vol. 2015, no. 2, 2015. URL: https://nymity.ch/tor-dns/pdf/Juen2015a.pdf (Cited on pp. 3 and 12)

[28] S. Khattak, D. Fifield, S. Afroz, M. Javed, S. Sundaresan, D. McCoy, V. Paxson, and S. J. Murdoch, "Do you see what I see? differential treatment of anonymous users," in *NDSS*. The Internet Society, 2016. URL: https://nymity.ch/tor-dns/pdf/Khattak2016a.pdf (Cited on p. 8)

[29] K. Krauss, "Ethical Tor research: Guidelines," Nov. 2015. URL: https://blog.torproject.org/blog/ethical-tor-research-guidelines (Cited on p. 13)

[30] A. Mahanti, N. Carlsson, A. Mahanti, M. Arlitt, and C. Williamson, "A tale of the tails: Power-laws in Internet measurements," *IEEE Network*, vol. 27, no. 1, 2013. URL: https://nymity.ch/tor-dns/pdf/Mahanti2013a.pdf (Cited on p. 6)

[31] N. Mathewson, "Remove global client-side DNS caching," Jul. 2012. URL: https://gitweb.torproject.org/torspec.git/tree/proposals/205-local-dnscache.txt (Cited on p. 3)

[32] "Mixmaster." URL: http://mixmaster.sourceforge.net (Cited on p. 1)

[33] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by Internet-exchange-level adversaries," in *PET*. Springer, 2007. URL: https://nymity.ch/tor-dns/pdf/Murdoch2007a.pdf (Cited on pp. 1, 3, 4, and 6)

[34] Netcraft, "July 2016 web server survey," Jul. 2016. URL: https://news.netcraft.com/archives/2016/07/19/july-2016-web-server-survey.html (Cited on pp. 6 and 11)

[35] R. Nithyanand, O. Starov, A. Zair, P. Gill, and M. Schapira, "Measuring and mitigating AS-level adversaries against Tor," in *NDSS*. The Internet Society, 2016. URL: https://nymity.ch/tor-dns/pdf/Nithyanand2016a.pdf (Cited on p. 3)

[36] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at Internet scale," in *NDSS*. The Internet Society, 2016. URL: https://nymity.ch/tor-dns/pdf/Panchenko2016a.pdf (Cited on pp. 4 and 5)

[37] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networls," in *WPES*. ACM, 2011. URL: https://nymity.ch/tor-dns/pdf/Panchenko2011a.pdf (Cited on p. 3)

[38] M. Perry, "Padding negotiation," Sep. 2015. URL: https://gitweb.torproject.org/torspec.git/tree/proposals/254-padding-negotiation.txt (Cited on pp. 3, 4, and 9)

[39] RIPE Network Coordination Centre, "RIPE Atlas." URL: https://atlas.ripe.net/ (Cited on pp. 2, 3, and 12)

[40] 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*. USENIX, 2015. URL: https://nymity.ch/tor-dns/pdf/Sun2015a.pdf (Cited on p. 3)

[41] Team Cymru, "IP to ASN mapping." URL: https://www.team-cymru.org/IP-ASN-mapping.html (Cited on p. 4)

[42] The Tor Project, "CollecTor." URL: https://collector.torproject.org (Cited on pp. 7 and 11)

[43] The Tor Project, "Tor Metrics–Top-10 countries by directly connecting users." URL: https://metrics.torproject.org/userstats-relay-table.html (Cited on p. 11)

[44] T. Wang, "Website fingerprinting: Attacks and defenses," Ph.D. dissertation, University of Waterloo, 2015. URL: https://nymity.ch/tor-dns/pdf/Wang2015a.pdf (Cited on pp. 2 and 3)

[45] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *USENIX Security*. USENIX, 2014. URL: https://nymity.ch/tor-dns/pdf/Wang2014a.pdf (Cited on pp. 2, 3, and 8)

[46] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in *WPES*. ACM, 2013. URL: https://nymity.ch/tor-dns/pdf/Wang2013a.pdf (Cited on pp. 3 and 9)

[47] T. Wang and I. Goldberg, "On realistically attacking Tor with website fingerprinting," *PoPETs*, vol. 2016, no. 4, 2016. URL: https://nymity.ch/tor-dns/pdf/Wang2016a.pdf (Cited on p. 4)

[48] P. Winter, "exitmap: A fast and modular scanner for Tor exit relays." URL: https://github.com/NullHypothesis/exitmap (Cited on p. 5)

[49] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. Weippl, "Spoiled onions: Exposing malicious Tor exit relays," in *PETS*. Springer, 2014. URL: https://nymity.ch/tor-dns/pdf/Winter2014b.pdf (Cited on pp. 5 and 14)

[50] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, "Connection-oriented DNS to improve privacy and security," in *Security & Privacy*. IEEE, 2015. URL: https://nymity.ch/tor-dns/pdf/Zhu2015a.pdf (Cited on p. 14)