Onion-Location Measurements and Fingerprinting

Paul Syverson
paul.f.syverson.civ@us.navy.mil
U.S. Naval Research Laboratory
USA

Rasmus Dahlberg
rasmus@rgdd.se
Independent
Sweden

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

Rob Jansen
robert.g.jansen7.civ@us.navy.mil
U.S. Naval Research Laboratory
USA

Abstract

Onion-Location makes it easy for websites offering onion service access to support automatic discovery in Tor Browser of the random-looking onion address associated with their domain. We provide the first measurement study of how many websites are currently using Onion-Location. We also describe the open-source tools we created to conduct the study. Onion-Location has been criticized elsewhere for its lack of transparency and vulnerability to blocking. Perhaps even more troubling, we show that Onion-Location is vulnerable to very accurate fingerprinting. We present recommended changes to and alternatives to Onion-Location as well as steps towards even more secure onion discovery and association.

Keywords

onion services, Onion-Location, website fingerprinting, circuit fingerprinting, network measurement

1 Introduction

Websites available at onion addresses via onion service protocols have numerous authentication, reachability, security, and privacy advantages [33, 49]. For this reason thousands of sites reachable by traditional protocols at ordinary registered domains such as torproject.org are also available at a .onion domain. One advantage of an onion address is that it is self-authenticating: it encodes its own public key. A corresponding disadvantage is that onion addresses are random-looking strings, hard to either remember or recognize: the onion address for torproject.org is 2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion.

For several years, users of Tor Browser who know a domain name or follow a link to its URL, have been able to easily reach the associated onion address if the site offers Onion-Location [53] (O-L). For example, connecting to https://www.torproject.org will yield an ".onion available" button displayed in the URL bar (see Figure 1). Clicking on the button will redirect the connection to the onion address. Alternatively, Tor Browser could be configured so that the redirect happens automatically—until this option was disabled as a result of the research herein [55].

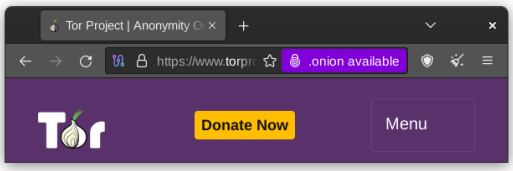

Figure 1: Tor Browser's Onion-Location button.

We will more fully describe features of onion services and various previously published aspects of and issues with O-L in Section 2. This paper focuses on the measurement and fingerprintability of sites accessed using O-L and variants of it. *Website fingerprinting* (WF) is a traffic analysis attack in which an adversary matches the pattern of (encrypted) traffic to and from a client against fingerprint patterns of known websites to try to determine the site connected to without needing to observe the other end of the connection [19]. *Circuit fingerprinting* (CF) identifies the type of circuit in Tor. For example, it has been very effective at distinguishing the introduction and rendezvous circuits used exclusively for connecting to an onion service from general-use circuits [21, 25, 29]. One reason CF is especially effective is that it is fingerprinting based on a small closed world: there are only a handful of possible circuit types in Tor. "Closed world" is sometimes used in WF to mean that any observed client is assumed to only visit destinations within a selected subset of possibilities. To disambiguate, we will herein use *complete world* to signify that the world that observed clients are selecting from comprises all possibilities.

O-L sites would appear to be particularly susceptible to fingerprinting given an access pattern of ordinary web connection followed immediately by an onion service lookup and connection to the same website (often in fact the same services on the same server, albeit over onion service circuits). Our analysis confirms this, showing with over 99% accuracy that a standard fingerprinting technique identifies a connection as to an O-L site. Note that O-L fingerprinting is more akin to CF than WF but is actually neither: it uses the combined pattern and timing of circuits (identified via

Rasmus Dahlberg produced part of this work while at Karstad University.
This paper was submitted and accepted, subject to revision, prior to Rob Jansen becoming Editor-in-Chief of PoPETs. The final adjudication of the revision was overseen by the other Editor-in-Chief, Zubair Shafiq, without any input from, or visibility to, Rob Jansen. Reviewers were not made aware of authorship prior to the final decision.

Proceedings on Privacy Enhancing Technologies 2025(2), 512–526

https://doi.org/10.56553/popets-2025-0074

CF) to fingerprint that an O-L session is occurring. We will focus herein specifically on *Onion-Location fingerprinting* (OLF) and closely related types of activity fingerprinting.

Although our focus is on OLF, it is possible to use methods other than O-L for *onion association* (OA), i.e., for discovering and associating an onion address for a given registered domain. For example, if lookup of the associated onion address can be done entirely at the client, then the only connection will be to the onion service. In this case *onion association fingerprinting* (OAF) would seem to merely reveal an onion service access. Consequently, WF of an accessed OA site would be rendered equivalent to WF in the much larger world of general onion-service access.

To investigate the current state of O-L fingerprintability, we first determine the O-L sites currently available. Since an O-L redirect header is only accepted by Tor Browser from a URL with a certified TLS connection, and since it is now standard to record the issuance of a TLS certificate in Certificate Transparency (CT) logs [30], we first download the CT logs to learn about *nearly all available certified TLS sites*. We then visit these sites to determine those configured for O-L, thus producing a list of available O-L sites. We access them to create a dataset of access patterns for use in evaluating fingerprintability. We show that an attacker can use CF as a building block to create nearly perfect OLF attacks. This in turn allows an attacker to create highly tailored WF attacks on all the O-L sites on the list. (We describe how OLF facilitates WF attacks and the likely expected impacts, but we do not analyze WF attacks in this paper.)

Contributions: In this paper:

- We conduct the first Internet-wide measurement of O-L. We introduce novel tools and use them to measure and then analyze how many clearnet websites use O-L to also offer onionsites, finding 1,505 stable (available after six months) O-L onions. Our measurement tools and data are available so that this can be repeated and tracked over time [7, 8].
- We show that both O-L and circuit types are extremely fingerprintable (99% accuracy). We show that such fingerprinting shrinks the anonymity set of an onion service associated with a registered domain by nearly an order of magnitude (versus non-O-L onion services [38]) and makes the onionsite susceptible to Website Oracle [41] attacks through DNS [9].
- We recommend changes and alternatives to O-L for onion discovery and association that are less susceptible to fingerprinting. We discuss the steps needed to produce OA that is usable and resistant to fingerprinting and other attacks. We recommend immediate actions to reduce the threat of OLF as well as longer-term research actions that should make onion discovery and association even less fingerprintable.

The next section presents background on Tor, onion services, O-L, and WF, as well as our fingerprinting threat model. In Section 3 we describe OLF, as well as the attacks we evaluate. In Section 4 we describe the open-source tools we created to learn the list of available O-L sites, the browsing of onion-associated pairs, as well as the data gathered from running these on the Internet. In Section 5 we describe the results of conducting OLF attacks on the sites on which we gathered data. In Section 6 we discuss our recommendations for O-L, trade-offs of recommended choices, and suggestions to further reduce onion service fingerprintability. In Section 7 we describe related work, and we conclude in Section 8.

2 Background

In this section, we provide a general background on Tor and onion services. Next we will describe O-L as currently deployed. Then we will describe alternative versions of O-L as well as alternative means of OA and their known pros and cons. Finally we will describe the basics of WF on Tor and our threat model.

2.1 Tor and onion services

Tor is best known as an anonymous networking tool. By separating identification from routing it offers security and privacy protections to its millions of client users, typically accessing the network via Tor Browser. Onion services offer somewhat similar protections to servers. Since we are concerned with WF, we will need to understand a bit of how these work to be able to explain our threat model. But our primary focus for the offered protections is on a different aspect of Tor that is less generally appreciated, authentication that is both stronger and more in control of the authenticated service.

The vast majority of Tor traffic is not to onion services but on connections that exit the Tor network for ordinary destinations on the Internet, such as the webpage at www.torproject.org. The Tor network is comprised of thousands of volunteer-run *relays*. The Tor client learns about the relays in the network from a directory system. For our purposes we do not need to understand how. (Description of unexplained aspects of Tor and further details on explained aspects can be found at [52].) The client builds a three-hop cryptographic circuit through the relays of the Tor network, exiting the last of these for the intended Internet destination. In building the circuit the client establishes a session key with each relay known only to those two parties and kept only for the duration of the circuit. Tor relays in a circuit are chosen at random, though not uniformly. Such a three-hop circuit is comprised of a *guard*, a *middle*, and an *exit*. A guard is selected by a client and used for several months as the entrance to all Tor circuits. If the guard is not controlled by an adversary, a relay-based adversary will not find itself selected to be on a Tor circuit adjacent to (thus identifying) the client.

An onion service essentially acts as another client from the perspective of the Tor network. Instead of a circuit to an exit, the onion service connects to an *introduction point*, a relay that listens for client requests to connect to that onion service and passes them along. Upon receiving such a request, the onion service connects to the end of a circuit the client has constructed to a *rendezvous point* which mates the circuits from the client and from the onion service. For clients to know how to find introduction points, there is an onion service directory system on a distributed hash table (DHT) comprised of thousands of Tor relays. Each such relay (called an *HSDir*) stores a list of introduction points and other information about an onion service in a *descriptor* associated with the onion address. The DHT is constructed so that an HSDir that does not already know an onion address will not know it is storing the descriptor for that onion address. Descriptors are regularly reassigned to new HSDirs based on a distributed source of randomness so that relays cannot predictably serve as directories for particular onion addresses to facilitate mining of lookup activity for given addresses.

The onion address is a 56-character string encoding an ed25519 public key that authenticates the onion service. The top level domain .onion is reserved by RFC 7686 [3] and must not be resolved by DNS. Because the onion address encodes its own public key, it is self-authenticating. But, it thus generally appears to be a meaningless, random-looking 56-character string, though some sites spend the resources to make a small portion of that meaningful.

Publicly available sites typically do not need to take advantage of the network location protection that onion service protocols offer. But, onion services offer other advantages. Exit relays in the Tor network are subject to resource contention, so offering onion service access can improve user experience. Also, DNS lookup occurs at exits. Though Tor detects and removes exits that abuse DNS, such abuse is not unheard of: those steps are necessary. Abuse also occurs once beyond the control of Tor network elements. Several years ago, Cloudflare began offering its DNS resolver 1.1.1.1 at an onion service in part to cope with such concerns [44].

Another advantage of offering onion service access to your website is that the address is self-authenticating. It offers authority independence from DNS hijack or TLS certificate hijack. Even if those occur, lookup, routing, and site authentication cannot be usurped by anyone not in possession of the private key corresponding to the address itself. But this is true only if the client knows the address, which is a random-looking string by itself not memorable, recognizable, or easily discovered as associated with a commonly known entity. This is a motivation for O-L.

2.2 Onion-Location and onion association

O-L is one way to discover and access an onion service associated with a registered domain. There are others. For example, Cloudflare began offering onion alternative (alt) services for its supported domains in 2018 [43]. Tor users could request connection to an HTTPS URL hosted by Cloudflare but be routed and connected to that website through its onion address. The URL address bar shows the requested HTTPS URL, even though the connection is via the onion address. No changes to Tor Browser were needed for these to function as intended, although the lack of transparency to users or browsers was shown to be the basis of multiple attacks [49, 50].

We will use "onion association" (OA) to mean an association between a registered domain and an onion address such that the onion address is visible to the user, as is whether an onion service protocol is being used for the connection. Thus onion alt services do not count as OA. We will reserve "Onion-Location" (O-L) for OA that requires for each access a connection to a registered domain from which an onion service connection is induced.

We wish to also require that the domain responsible for the redirection is also visible to the user, but we must be careful what we are claiming if we intend to included O-L under "onion association", which we do. It is unfortunately ambiguous to claim that current O-L offers such visibility. Once ".onion available" is selected and Tor Browser redirects to the onion address, the original domain is no longer visible. If the Tor Browser setting "Prioritize .onion sites when known." is set to "Always", then the redirect is automatic when clicking on a link to a domain offering O-L.[1] It is thus easy for a momentarily distracted user to not even notice the original domain in the URL bar, which can be leveraged to create attacks [49]. Nonetheless, it was previously possible to set onionsite prioritization so that the domain associating with the onionsite (and its TLS certificate) could be inspected before selecting ".onion available". And that setting is now the only option. We will count as OA O-L that offers to cautious users this level of visibility of both registered domain and onion address.

Our focus is not hijacks but OAF by a client's guard relay. We thus now describe the connection pattern occurring in O-L and other variants.

2.2.1 Current Onion-Location: A client making an O-L connection, e.g., to torproject.org first creates an exit circuit, and the exit's DNS resolver will resolve torproject.org. (We assume that, like the overwhelming majority, the client is not configured to resolve domains on its own, e.g., via an onion service such as Cloudflare's mentioned above.) The same circuit used to resolve the address is then used to connect to the IP address of the torproject.org server, and the index page is loaded. This includes either an O-L HTTP header or an HTML `<meta> http equiv` attribute that states the onion address to be associated with the domain [53]. For ease of exposition, we use "header-based" to refer to O-L using either HTTP headers or HTML tags. This example describes O-L redirection from an index page. O-L works the same way from another page at a domain, however. Thus, following O-L from torproject.org/about/history/ will redirect to 2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion/about/history/.

Tor Browser only allows O-L on certified HTTPS connections. For such a connection, the client can use O-L to select connecting to the onion address. The next actions are the same as if the onion address were entered directly into the URL bar (or a link to it clicked on), as described above in Section 2.1. If the client's request to connect is accepted by the onion service, the onion service will then send over the rendezvous circuit the index page of the website it is serving. Typically this would be the same index page as was sent over the exit service, as we show later in Section 4.

For privacy reasons, Tor Browser will not cache OA information. The pattern of first connecting to the clearnet website and downloading its index page before connecting to the onion service happens each time O-L is invoked (whether automatically or manually). This makes O-L trivially censorable if the destination is censored on the open Internet. For this and other reasons, an alternative has been designed and implemented.

2.2.2 Onion-Location via Sauteed Onions using TLS handshake. Current O-L is not transparently consistent. O-L headers (or tags) can direct connections to different onion addresses, and there is no easy way to detect this without collaboration amongst the clients directly affected. However, if the OA were based on public append-only logs such as those which Certificate Transparency (CT) provides, then there would be a consistent public record of the OAs available. This is a feature of sauteed-onion based O-L [10], so called because actual onions became transparent when sauteed.

[1] This describes the state of O-L prior to the release of Tor Browser 13.0.12, in which automatic O-L was removed as a result of our findings reported herein [55]. Similar comments apply to Brave Browser. See Section 6 below.

To offer sauteed O-L using TLS handshake, a domain must first obtain a sauteed-onion TLS certificate. This adds to the certificate, a Subject Alternative Name (SAN) that encodes the associated onion address as a subdomain. For example, mullvad.net has a TLS certificate with the SAN o54hon2e2vj6c7m3aqqu6uyece65by3vgoxx hlqlsvkmacw6a7m7kiadonion.mullvad.net. Such a certificate can easily be obtained for free from Let's Encrypt. Instructions for this are available at https://www.sauteed-onions.org, which also has a sauteed-onion certificate.

Client configuration for sauteed O-L is currently implemented using a WebExtension for easy implementation and adoption, though ultimately it is best built into the browser. Connection patterns are the same as in current O-L, except the redirect happens not after loading the index page but immediately upon the TLS handshake. As a side benefit, this reduces the traffic load and cuts down on the overall time to load the index page from the onion service.

2.2.3 Onion Association via Sauteed Onions using CT Log Monitors. Because modern certificate issuance requires proof of submission of a certificate to CT logs, certificate-based OA need not require O-L, that is, OA requiring a client to connect to the registered domain each time. One can do a lookup of, e.g., mullvad.net at a CT-log-based certificate search site such as https://crt.sh/. The aforementioned subdomain onion SAN appears in the returned values under "Matching Identities". An advantage of this approach is that if mullvad.net is blocked by an adversary, it will not preclude discovery of the associated onion address. Looking only at the pattern of connections, rather than any pattern differences of traffic served and time for human processing of returned information, this will be roughly the same as O-L. First, there is an exit circuit to an Internet site (following DNS lookup). This is followed by onion service connections to the discovered associated onion address.

We assume that connecting to an onion address follows immediately after the association is retrieved. If the OA lookup is done enough in advance that there is no apparent association between it and visiting the onion service, then the relevant pattern will essentially be the same as just visiting an onion service. We do not consider this case further since it assumes significant planning and/or usability inconvenience on the part of users.

A sauteed-onion search API based on multiple CT logs has been implemented and made available at api.sauteed-onions.org [10]. This is also available at zpadxxmoi42k45iifrzuktwqktihf5didbae c3xo4dhvlw2hj54doiqd.onion, thus improving privacy, security, and blocking resistance versus O-L. OA based on this onion service is the first pattern of connections not superficially similar to O-L, instead characterized by two successive patterns of onion service connections. As above, if the discovery of OA is done far enough in advance of the connection to the onion service, then the pattern will essentially be just that of connecting to one onion service. But, also as above, that will require planning and/or inconvenience. To mirror the convenience offered by O-L it will be necessary to have the association be immediately available or automatic.

2.2.4 Local Onion Association via Sauteed Onions. All of the above means of OA are implemented and available now. A future implementation of sauteed OA could periodically use the above search API to download all currently known onion associations and then do each particular OA locally as needed, similar to HTTPS-Everywhere rulesets [16]. (HTTPS-Everywhere loads into participating browsers rulesets that locally rewrite requested HTTP URLs to associated HTTPS URLs. As HTTPS sites have become all but ubiquitous and HTTPS-only mode has been adopted by more browsers, HTTPS-Everywhere has increasingly become moot [17].) As we will see in Section 4, the number of available OA sites could grow by more than an order of magnitude over what is currently provided by O-L before this will begin to be comparable to what is already downloaded regularly by Tor clients. Such periodic download is thus not a significant overhead at existing levels. If the retrieval of OAs from the API is done automatically and periodically, then the pattern of connections to associated onions will be essentially indistinguishable from any onion service access. Unlike manual use of a certificate search or of an OA API, this will also not imply a usability hit. If anything, it is likely to improve usability because no additional Tor circuits or external retrieval is needed before accessing a locally associated onion address.

2.3 Adversary model

Our attack is based on CF, not WF, but our adversary model also follows that of prior work on WF. Also, prior work we cite shows that WF of onion services in general is quite effective—even before (1) applying our OLF or OAF methods which effectively reduces us from an open world of tens of thousands of onionsites to a world an order of magnitude smaller, and (2) using website oracles as described below to further shrink the size of the world.

Website Fingerprinting allows an adversary observing client traffic patterns to infer the destination to which a client is connecting even if the adversary observes neither the destination end of communication nor the specific content exchanged. WF on onion routing and other anonymous communication systems predates Tor itself [19]. For recent surveys of WF attacks and defenses specifically on Tor, see [6, 32, 47]. Importantly, these cover Tor but do not discuss WF on onion services. Explorations of WF that specifically analyze onion service fingerprintability include [21, 29, 36, 37, 41].

A WF adversary may also be modelled to have access to a Website Oracle (WO) [41]. A WO is an abstraction of a source of information for the adversary that simply answers the question of whether a given website was visited over Tor by someone (or something) at a given time, for example, by observing whether a DNS lookup of a relevant domain occurred. One can observe the timing difference of lookups at recursive resolvers to know whether a given domain is cached, and thus requested recently. Tor exit relays cache recently requested domains for up to an hour, and can be probed [9]. Note that there are many passive sources of WOs [41]. Since O-L requires an exit connection to a domain, a WO based on DNS would greatly reduce the False Positive Rate (FPR) of WF attacks on all but the most popular websites visited over the Tor network [41].

Our adversary will be assumed to be conducting attacks from the entry guard of the client. The adversary is not assumed to control the guard of the onion service or otherwise observe the server end of an onion service connection. The adversary might be controlling the guard of a particular client by pure luck. This might be a "hoovering" adversary [20] that is fingerprinting the behavior of any clients unlucky enough to have selected a guard under the adversary's control. But it may also be that the adversary has targeted the guard

of particular clients for takeover, possibly expending significant resources to do so. The clients might, for example, reside in an area in which the adversary controls the ISP and will attempt to compromise the guards of observed clients that are of interest for whatever reason. In the case of such a targeting adversary, it may be that the adversary merely controls the ISP and some middle relays as analyzed in [20] or even just middle relays [21]. Wang and Goldberg have shown that even an ISP adversary, seeing only encrypted TCP traffic (thus without direct access to cells, as described in the next paragraph), can successfully split packet sequences to identify separate page loads [58]. Our attacks should thus also work at an ISP adversary that uses such techniques to address splitting. We will assume a guard adversary to keep things simple.

As an entry guard, the adversary is able to observe the traffic patterns of packets going back and forth on circuits. Tor packages all traffic into cells, which are generally of uniform length. As a relay, the guard can also see which circuit a cell is associated with as well as the cell commands. Cell commands identify the purpose of the cell within the Tor protocol. Most will be RELAY cells, sending data up and down the circuit. Others include CREATE cells for establishing a circuit and DESTROY cells for tearing a circuit down. RELAY cells carrying data from the client will arrive triple-encrypted at the guard (once for each hop in the circuit). The guard will strip off one layer of encryption before forwarding to the middle. RELAY cells carrying data from the destination arrive at the guard from the middle and are double-encrypted. The guard adds its layer of encryption before sending to the client.

We also assume a purely passive adversary. This is not so much because actively altering, e.g., the timing patterns of packets passing through its control or dropping circuits, etc. is expected to be difficult for the adversary to perform. Rather, it is because a purely passive adversary is all that is needed to be extremely effective in performing the attacks we wish to evaluate.

3 Onion-Location Fingerprinting

A fingerprinting attack can attempt to directly identify via fingerprint the specific website being accessed (i.e., WF). But fingerprinting can also be used to attempt to identify *the type of access* rather than the specific site. For example, Kwon et al. [29] used a CF attack to identify that a connection was being made to an onion service, *any* onion service. While Jansen et al. [21] show how to distinguish onion service circuits based on circuit type (e.g., Rendezvous) and relay position (e.g., client-side middle relay). Recall that CF is inherently a complete world attack. And there are only four[2] types in the world of circuits originating from an ordinary client (general-use, HSDir, Introduction, Rendezvous).

This may be of interest in its own right but may be useful even if the adversary ultimately wants to identify the specific destination. According to metrics.torproject.org there have been roughly 800K onionsites available for the last few years, but the number of active reachable sites is significantly smaller. Estimates vary greatly, but the number in studies is typically around ten thousand or even less [38]. Thus, identifying the site being accessed as an onionsite greatly shrinks the world of possible destinations.

[2]Conflux [2, 14] was relatively recently added as a fifth type, see discussion in Section 5.

Our analysis to be described in Section 4 shows that the number of reachable O-L sites is almost an order of magnitude smaller than reachable onionspace, around 1,500 when we performed our study. The primary goal of our OLF adversary will thus be to accurately determine if a client is connecting to an O-L site.

If one adds to the adversary's arsenal a Website Oracle, then FPR becomes negligible [41]. As Pulls and Dahlberg observed, when onion services moved several years ago from version 2 to version 3, the resources needed for a WO (based on the DHT) dramatically increased and the percentage of covered onionsites by a given WO adversary dramatically decreased. But for O-L, which involves a DNS lookup and a DHT lookup, a WO remains readily available even though v3 onion services have been the default for years. In general our adversary's goal is to identify whether a connection pattern constitutes O-L, or another form of OA discussed earlier.

Further, we describe in Section 4 how to identify all the O-L sites available at a given time. Shrinking the world of onionsites in this way by roughly an order of magnitude is extremely impactful even without the WO that automatic O-L facilitates.

(1) Create a list of all registered domains with TLS certificates offering O-L.
  (a) Download CT logs to create a list of all SANs in TLS certificates issued by CAs.
  (b) Visit domains on the resulting list of SANs to find the ones offering O-L.
(2) Construct datasets to be used in fingerprinting O-L.
  (a) Set up a Tor client and a guard relay it will use.
  (b) Create traces from the guard of visits by the client to the list of O-L domains and onion addresses, both separately and with O-L turned on.
(3) Train classifiers to identify the types of circuits used in O-L: general-use, HSDir, Introduction, Rendezvous.
(4) Use the pattern of circuit types associated with O-L to decide if observed traffic through an adversary guard matches an O-L fingerprint.

Figure 2: Steps for an OLF attack.

Any moderately resourced attacker should be able to follow the specific steps we took to conduct OLF. The steps are shown in Figure 2, and more details will be set out in the next two sections. The steps in Figure 2 describe current header-based O-L. Because Tor Browser does not currently support TLS-handshake-based O-L natively, our traces for it are based on simulation using `curl` running through our Tor client. When TLS-handshake-based O-L is deployed, an adversary will be able to follow the steps in Figure 2 for it as well, without the need for any simulation.

OA via the onion service API described in Section 2.2.3 can either occur at the time of connection to the associated onion address or can utilize a list of associations periodically downloaded, as described in Section 2.2.4. As noted, in the local case there will be no traffic additional to what would occur in any onion service access. So, there should be no fingerprint distinguishing local OA from any other onion service access. A fingerprint in the case of (automated)

lookup at the time of connection will follow a slightly different pattern than O-L, and importantly will take an exit-based WO off the table. (Note that unless OA lookup at a server retrieves the entire list of OAs there, we make a large server-trust assumption: an adversary-run server would effectively be an ideal WO. Private Information Retrieval (PIR) or some other technique would be required to address this, such as an adaptation of the proposal to use PIR to reduce trust in the HSDir storing the onion service descriptor [13]. Discussion of possible ways to remove or reduce this trust assumption are beyond the scope of this paper.) The guard adversary and goal will be the same: look for a fingerprint indicating that the client is making an OA from the available list.

An unknown is how prevalent patterns indicating successive onion service connections will be, especially as onion service offerings continue to grow. Existing websites make extensive use of content loaded from other domains or from third parties when servicing connections to their index page. As the volume of providers offering onion service access continues to grow, it is thus conceivable that an onion-service-then-onion-service fingerprint will by itself not be indicative of OA by this means. For now, it is reasonable to assume that there is a very small increase in false positives if identifying OA by such a fingerprint versus an O-L fingerprint. In Section 6 we propose cover activity to add to general-purpose circuit creation and use, so as to make it harder to distinguish the traffic patterns of onion service access versus others. This will also increase the FPR of such OA and could even be extended to occasionally make general purpose activity harder to distinguish from two successive onion service accesses.

4 Measurement and datasets

To carry out our OLF attack, an adversary must distinguish patterns resulting from loading websites with and without O-L. We now describe several Internet measurement tasks to demonstrate how an adversary may (1) identify the set of websites that configure O-L, and (2) construct a dataset of encrypted traffic patterns for use in subsequent fingerprinting attacks. Figure 3 provides an overview.

4.1 Identifying sites with Onion-Location

4.1.1 Domain names in CT logs. CT is a system of logs that publish certificates issued by trusted certificate authorities. Because several major web browsers require that a website's certificate is promised to be in the logs before establishing a secure HTTPS connection, the logged certificates form a representative dataset of sites configuring HTTPS. In our pursuit of finding sites with O-L, we created a free and open-source tool named `ct-sans` that assembles a reproducible[3] dataset of Subject Alternative Names (SANs) found via CT. Data gathering takes place in three phases:

Snapshot Record the current state of all logs. The state of a log includes information such as the number of certificates, a Merkle tree hash, and a log's digital signature on its state.

Collect Download the logs up until the recorded snapshot. The SANs found in each log are persisted to disk, but not the complete certificates to reduce storage requirements.

Assemble Combine the SANs found in each log. The resulting dataset is one unique SAN per line in ascending order and the snapshot that fixates these SANs for reproducibility.

The list of logs is managed automatically by `ct-sans`: Google's signed log list is consulted to use the same logs as Google Chrome.[4] All cryptographic properties of the logs are also verified, such as checking that the next snapshot is consistent with the current one (given repeated snapshot phases) and that the downloaded certificates are really included with regard to the current snapshot.

We conducted our `ct-sans` data gathering between 2023-03-18 and 2023-04-03. The machine used for the data collection was an Ubuntu 22.04.2 LTS VM configured with 62.9GiB RAM, 32 (virtual) CPU cores, and a 2TiB SSD. We used 2-16 workers for logs nearby us (or if they contained few certificates), and 40 workers for Google's Xenon log which was further away from our measurement setup (to get comparable download speeds at around 1000 certificates/s). Each of these workers backed-off exponentially on rate-limit errors using a library created by Google.[5] The utilized bandwidth (all logs combined) ranged from 45-388 Mbps throughout the measurement. In total, 3.7B certificates were downloaded from 17 logs. 7,418 certificates were ignored because they or their SANs could not be parsed.[6] This resulted in 0.91B unique SANs (25.2 GiB).

The `ct-sans` source code (roughly 1,000 lines of Go), the exact measurement configuration that was used, and the assembled data set can be located at https://git.rgdd.se/ct-sans.

4.1.2 Sites with Onion-Location. We created a free and open-source tool named `onion-grab` to determine which HTTPS sites configure O-L. For overview, the input is a list of domain names. The output is a list of domain names and discovered onion addresses, categorized on if HTTP headers or HTML meta tags were used for configuration. We opted to create our own tool rather than integrating with existing general-purpose scanners like `zgrab`[7] to get fine-grained control over the number of requests/s and error handling. For example, `onion-grab` can be configured with an upper limit for requests/s and categorizes why a given request failed (e.g., TCP error, DNS error, TLS error, too many redirects, and timeout). The level of output also makes it easy to resume a previously crashed session without starting over again from scratch.

We used a measurement setup consisting of three VMs. One VM was configured with 62.9GiB RAM, 32 (virtual) CPU cores, and a 2TiB SSD. The other two VMs were configured with 62.9GiB RAM, 16 (virtual) CPU cores, and 60GiB SSDs. All VMs used Ubuntu 22.04.2 LTS and shared a 1x10Gbps link with each other and other network VMs we have no control over. To ensure there can be many concurrent connections, the maximum number of file descriptors were set to 1M.[8] The range of ports that can be used for our outgoing TCP connections were also increased.[9] For geographical diversity, the three VMs were configured to proxy their respective traffic

[3] Requires that the logs remain available for download, and that the same libraries and versions of these libraries are used for certificate and domain name parsing.

[4] https://github.com/google/certificate-transparency-community-site/blob/master/docs/google/known-logs.md

[5] https://github.com/google/certificate-transparency-go/blob/master/scanner/fetcher.go

[6] SANs that contain newlines are considered malformed because our dataset is line-terminated. No other value constraints were applied to minimize dataset assumptions.

[7] https://github.com/zmap/zgrab2

[8] `ulimit -Sn 1000000`

[9] `net.ipv4.ip_local_port_range="1024 65535"`

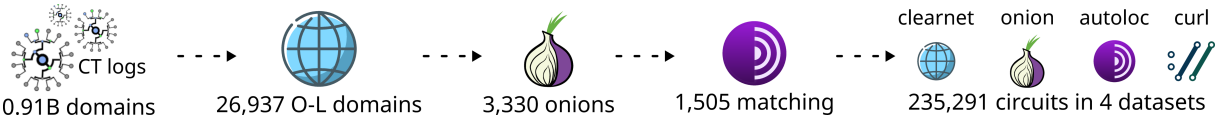

Figure 3: Overview of our measurement and data collection process using Certificate Transparency (CT) logs to identify sites using O-L with matching index pages and constructing four datasets for evaluating OLF.

through Frankfurt, New York, and Melbourne using Mullvad VPN. Google's `8.8.8.8` and `8.8.4.4` were set as primary and secondary DNS resolvers. Google's DNS policy permits 1,500 requests/s (per distinct IP) without any special negotiation.[10] We configured our scans to be just below this limit at 1,450 requests/s. We could expect each VM to spend 200-250 Mbps with such a configuration.

While measuring, we used the same list of domain names on each VM but in randomized order to reduce the impact of transient network errors and possible errors that only occur as a result of the used order. Any wildcard domain name in the input dataset was treated as a literal domain name. For example, `example.org` would be measured if the input dataset contains `*.example.org`. No web crawling or subdomain guessing was done to possibly identify additional O-L sites that get HTTPS through a wildcard. Because we do no web crawling and because our measurement is a one-off, `robots.txt` or similar are not taken into account.

After informal validation of our measurement setup using the (comparably small) Tranco top-1M list [40], we conducted a full measurement using the 0.91B domain names discovered by `ct-sans`. The full measurement took place between 2023-04-03 and 2023-04-13. Table 1 shows our results, and also includes what we learned about Tranco top-1M. 3,330 unique `.onion` addresses were configured by 26,937 unique domains. Post data-collection we realized the large number of unique domains was caused by `onion-grab` following HTTP redirects without properly taking note of this in the results. For example, a personal website that configures an HTTP redirect to X would in our dataset be shown as if the unique domain configured X's O-L using an HTML meta tag when it is in fact an HTTP redirect (≈14k instances of this). For Tranco top-1M, 207 unique `.onion` addresses were found from 285 unique domains.

The `onion-grab` source code (≈700 lines of Go), the exact measurement configuration that was used, and the collected datasets are at https://gitlab.torproject.org/tpo/onion-services/onion-grab.

4.1.3 Sites with mirrored Onion sites. About six months after identifying sites with O-L, on 2023-10-31, we attempted to visit each onion and the associated clearnet URLs using `torify curl`. We tried five times throughout the day for each onion and associated clearnet URL (after onion success, re-trying up to five times if we hit a Cloudflare CAPTCHA). We temporarily stored (only) the index HTML of the onion and the clearnet URLs, filtered out index pages below 100 bytes and with a Levenshtein distance between onion and clearnet pages below 0.9, saving the result. Since O-L works the same from any page, OLF attacks do as well. We used index pages simply to quantify the number TLS-certified domains offering O-L.

We originally had 3,330 onions with 26,984 clearnet URLs from 2023-04-03. Of the original 3,330 onions, we could still reach 2,101 onions (63%) on 2023-10-31. Of those 2,101 onions, 1,505 onions (72%) had reachable clearnet URLs with matching index pages. The remaining 596 "lonely" onions (28%) had no reachable clearnet URLs with matching index pages. Of those 596 onions, 208 (35%) had no reachable clearnet URL (7 of those returned less than 100 bytes though) and the remaining 388 (65%) had one or more reachable clearnet URLs but no matching index pages. As perhaps expected, the percentage of persistently available O-L sites is a bit higher than persistent onion addresses in general, e.g., in a recent study of onion services, Boshmaf et al. found "only 52.2% of the crawled domains were available and mapped to active services on any given day" [5]. A study of the types of sites offering O-L is a natural follow-on to our results that might shed light on the above numbers.

Table 1: The number of unique domains that when visited with Tor Browser would land at (or possibly redirect to) a site configuring O-L, as well as the method used for configuration and how many of the advertised onion addresses are unique. When looking at the method of discovering each unique onion address for the combined scan, 3,077 and 281 onion associations are available via HTTP headers and HTML tags.

	Domains	HTTP	HTML	Onions
Frankfurt	23,869	8,097	18,828	3,105
Melbourne	25,993	8,812	20,894	3,138
New York	25,827	8,850	20,655	3,283
Combined	26,937	-	-	3,330
Tranco 1M	285	-	-	207

4.1.4 Ethical considerations. Our `ct-sans` measurement used the public-good CT infrastructure. To avoid harm on the volunteer-operated logs that need to be available for other users than us at the same time, we did not try to speed up download times more than what was possible using a single IP address and one HTTPS connection per log. Within these HTTPS connections, multiple concurrent workers were used that each backed-off exponentially. The number of workers were tuned manually to find an equilibrium during which additional workers did not result in higher throughput. As a precaution, we ensured that it was possible for log operators to get in touch with us by using a university IP address and setting the HTTP agent to a value that identified us. The latter is a common but mostly undocumented best practise when downloading the logs.

To put our `onion-grab` measurement into perspective, the index page of each public HTTPS site was fetched three times over the course of 10 days. Such a one-off measurement has negligible impact on each individual site and does not cause harm. We ensured that our campus network was able to handle the load of each VM, and got permission to run our measurement by relevant system

[10] https://developers.google.com/speed/public-dns/docs/isp

administrators. As already described, no permission is needed for the (controlled) load we put on Google's DNS servers. Similarly, no permission is needed to use the (paid) Mullvad VPN service which operates at overcapacity without limits.[11] To not exacerbate congested VPN relays though, we monitored our effective bandwidth and timeout errors carefully and ended up switching to a less congested location (Frankfurt) one day into our measurements. The timeline for this and supplementary notes are available online.

Conclusion: We are the first to identify and measure the prevalence of websites that configure O-L. Such associations are critical for constructing informative datasets for OLF.

4.2 Constructing datasets for fingerprinting

Once the adversary obtains a set of onion URLs that are confirmed to be mirrors of clearnet sites, it may use the associations to construct fingerprinting datasets. In this section, we demonstrate how to construct such datasets, which we will use in Section 5 to evaluate the effectiveness of our OLF attack.

4.2.1 Measurement goals. An adversary using machine learning to conduct fingerprinting attacks generally needs informative datasets to train classification models to differentiate various types of patterns. As described in Section 3, the adversary we consider first uses CF to help it label the types of circuits it observes from the guard position, and then uses the sequence of circuit types and their patterns in OAF. To be successful, it should train on the circuit types and sequence of circuits it expects to observe in practice so that it can accurately predict when an O-L access occurs.

4.2.2 Measurement process. We suppose that the adversary uses website-to-onionsite associations to gather examples of the patterns that would be observed by a guard performing OLF. Thus, we first construct a list of website-to-onionsite pairs using the results of our measurement from Section 4.1; recall that we have a dataset of 1,505 websites that are confirmed to be mirrored at an onion URL. The dataset is a mapping of one onion URL to many domain names that point to the same website. We construct a list of (onion URL, domain name) pairs by choosing a single domain name at random among those associated with each onion URL.

We carry out measurement of the patterns resulting from loading the onion URLs and domain names through the Tor network. Our measurement utilizes a Tor client that pins a guard relay under our control in all circuits it creates. The client and guard run a version of Tor v0.4.7.10 that we modified to support in-protocol circuit identification and labeling (similar to the "opt-in" method of Cherubin et al. [6]). Our Tor modifications enable us to tag each client session (start Tor, load URL, stop Tor) with a string label. Whenever our client constructs a circuit through our guard, the client immediately sends to the guard a special `SIGNAL` cell type that embeds the session label, URL, and type of circuit created by the client. Upon receiving the `SIGNAL` cell from our client, our guard marks the circuit for measurement and records the label, circuit type, and the type and directionality of each of the first 5,000 cells sent in the circuit for subsequent analysis.

[11] See https://mullvad.net/en/help/improve-slow-speeds-throttling and https://mullvad.net/en/blog/unlimited-traffic-volume.

Table 2: Counts of circuit types composing our OLF datasets.

Dataset	General	HSDir.	Intro.	Rend.	Total
`clearnet`	29,697	10,627	10,195	10,083	60,602
`onion`	19,251	24,491	18,592	18,400	80,734
`autoloc`	31,916	21,888	18,159	17,998	89,961
`curl`	3,994	0	0	0	3,994
Total	84,858	57,006	46,946	46,481	**235,291**

With the Tor client and relay measurement process in place, we create an automated Python script using the open-source `stem` and `tbselenium` packages [1]. For each URL in a configured list, the script creates an ephemeral Tor client configured with a unique session id, loads the URL using Tor Browser, and then stops the ephemeral client. The script also optionally sets the Tor Browser preference `privacy.prioritizeonions.enabled` to true in order to enable O-L automatic redirect. Note that Tor Browser is configured *in its Safest level*, which includes disabling JavaScript. From a fingerprinting perspective, this is the most conservative both because it means that any differentiation discernible from the disabled features are not available to the attacker and because it reflects fingerprinting of clients even if they have opted for the most conservative settings.

4.2.3 Datasets. We apply the measurement process described above to our set of 1,505 web and onion URL pairs as follows:

`clearnet` Load each of the 1,505 web URLs through Tor. Repeat 6 times.

`onion` Load each of the 1,505 onion URLs through Tor. Repeat 6 times.

`autoloc` Load each of the 1,505 web URLs through Tor after enabling O-L automatic redirect. Repeat 6 times.

`curl` Load a standard TLS page using `curl` (through our Tor client) that results in a 302 redirect, to simulate a Sauteed Onion TLS handshake. Repeat 2,000 times.

We collect the resulting circuit traces and labels as observed by our guard relay into an independent dataset for each measurement type. Each URL load resulted in the creation of numerous circuits that are all tied to the same client session. We show in Table 2 the counts of the number of circuit traces of each type of circuit that we recorded in each dataset. As shown in the table, we collected a total of 235,291 circuits, 84,858 (36%) of which are general-purpose circuits, 57,006 (24%) of which are HSDir circuits, 46,946 (20%) of which are introduction circuits, and 46,481 (20%) of which are rendezvous circuits.

Note that circuits of every type are generally observed in all but the `curl` dataset because for each client session we use Tor Browser and follow its standard bootstrapping procedures (including onion service update checks). The `curl` dataset was collected without Tor Browser and thus update checks do not occur. Although we report the counts of all observed circuits here, some are filtered out during our fingerprinting evaluation as described in Section 5.

4.2.4 Ethical considerations. We conducted data measurement using the public Tor network. We limited the impact of our measurement in several ways. First, all circuits that are measured by our guard relay are created by a client under our control. This is

guaranteed through the use of the special SIGNAL cell type that we created and that is only understood by our measurement client and relay. Second, our client pins our guard relay in the first hop of all created circuits, which reduces our load on other public relays. Third, we limit our measurement to just six instances of each URL for each dataset (1,000 instances of one URL for the curl dataset), since this was enough to evaluate our hypothesis.

Conclusion: We demonstrate how an adversary can use existing methods and open-source tools to construct encrypted-traffic datasets for OLF.

5 Onion-Location Fingerprinting attack results

Recall that our adversary is a guard that can link all circuits belonging to a client, reliably determine the start and end of each circuit, and observe individual cells. We break the problem down into a number of *binary classifiers*, each of which identifies a particular type of circuit (or later, activity). The binary classifiers are trained on two classes: the target circuit (or activity) type and all other circuits. The classes come from filtering the datasets collected in Section 4. Using one or more of these binary classifiers, we show that a guard attacker can reliably do CF and then use that for OLF, thus reducing the possible-destination anonymity set of a client to a small world (relative to all possible destinations, and even relative to just all onionsite destinations).

We split our datasets 8:1:1, stratified by label, i.e., proportionally 8:1:1 for each label, but the labels might not be balanced. (We note the baseline accuracy in each experiment, i.e., the accuracy if the classifier just always picks the most prevalent label.) Results are presented as the mean and standard deviation accuracy and false positive rate (FPR) from 10-fold cross validation. We use the WF attack Deep Fingerprinting (DF) [48], which only considers cell sequences[12] and completely ignores time. For CF, we use the first 512 cells of traces (because we are fingerprinting handshakes) and for OLF we use up to 5000 cells (the default DF input size). We update DF to use 200 epochs with an early stop of 10 epochs on the training loss. We use the last model for testing (ignoring the validation results). It is likely that a combination of hyperparameter tuning, a more sophisticated model using time and approach to training, data collection artifacts and/or randomized learning algorithms will explain the last fractions of a percentage point of accuracy missing from making the following fingerprinting results perfect.

As a pre-processing step, we filtered out our data collection SIGNAL cell (see Section 4) and all general circuits with an observable (*to the guard*) BEGIN_DIR relay command. Note that all datasets but the curl dataset contain onion traffic, probably due to some Tor Browser extension checking for updates and the prevalence of Cloudflare with alt-svc headers.

5.1 Circuit Fingerprinting

To perform OLF we first show that it is still possible to reliably perform CF, as first shown by Kwon et al. [29], despite deployed partial defenses in Tor [25] after Kwon et al.'s work. A limitation of our work is that we intentionally excluded the recently added Conflux [2, 14] circuits. We suspect that the Conflux handshake is fingerprintable as well (as noted by Goulet and Perry [14]), especially when taking timing into account, but we leave this for future work. The types of circuits created by clients (that are not serving onion services, which is discernible by the guard) are general, introductory, rendezvous, and HSDir circuits. This constitutes a complete world. Any discrepancy between circuit counts below and in Table 2 is due to our pre-processing on general circuits.

5.1.1 General circuits. To fingerprint general exit circuits, we create two classes by filtering: include only general circuits from the clearnet dataset (21,795 samples) and exclude all general circuits from the clearnet dataset (30,905 samples). We get accuracy 99.94±0.03% (58.64% baseline) and FPR 0.02±0.02%.

5.1.2 HSDir circuits. To fingerprint HSDir circuits, we create two classes by filtering: include only HSDir circuits from the onion dataset (24,491 samples) and exclude all HSDir circuits from the onion dataset (48,323 samples). We get accuracy 99.91±0.02% (66.36% baseline) and FPR 0.10±0.05%.

5.1.3 Introductory circuits. To fingerprint introductory circuits, we create two classes by filtering: include only introductory circuits from the onion dataset (18,592 samples) and exclude all introductory circuits from the onion dataset (54,222 samples). We get accuracy 99.95±0.03% (74.47% baseline) and FPR 0.03±0.02%.

5.1.4 Rendezvous circuits. To fingerprint rendezvous circuits, we create two classes by filtering: include only rendezvous circuits from the onion dataset (18,400 samples) and exclude all rendezvous circuits from the onion dataset (54,414 samples). We get accuracy 99.94±0.02% (74.73% baseline) and FPR 0.03±0.02%.

Conclusion: Circuit fingerprinting of general, HSDir, introductory, and rendezvous circuits is practically perfect for *a guard attacker*, with at least 99.9% accuracy and a FPR below 0.1%.

5.2 Onion Association Fingerprinting

Armed with the ability to perform CF, we can now use that for OAF. We show that fingerprinting is reliable, further reducing the potential destination anonymity sets of clients. Figure 4 gives an overview complementing the following sections.

For OAF, we have to consider all circuits that are part of a website visit, as observed by the guard. During a time window in which the attacker considers a website visit to be ongoing, the attacker first uses CF to identify the kinds of circuits. The attacker looks for a general circuit followed by a rendezvous circuit with the HSDir and introductory circuits in between. The HSDir and introductory circuits are useful for CF, but since they will have the same fingerprint regardless of the visited onionsite they will not play a role in creating classifiers. We also note that an important characteristic of automatic and manual OLF is that the attacker can use the *overlap between the general and rendezvous circuits* to build a better classifier because both circuits will transport the same website, likely with the same page. Since the general and rendezvous circuits of an OLF session should overlap in the content they carry, classifiers can learn to detect this overlap, as our attacks demonstrate and as we experimentally evaluate. (See Section 5.2.2 below.)

[12] Representing a circuit trace as a sequence of numbers with -1 for cells received and 1 for cells sent, padded with zeroes if lacking cells to a fixed length.

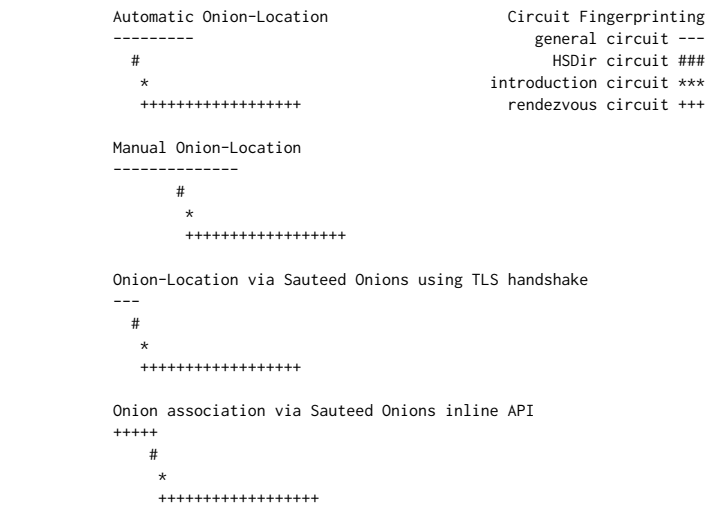

Figure 4: Overview of the different types of circuits and their use in O-L techniques. An attacker can use CF to identify circuits, and then use OAF to further reduce the potential destination anonymity sets of clients.

5.2.1 Classes, filtering, and input traces. Our goal is yet again to create binary classifiers. Here, we stress the difference between the *positive* and *negative* classes. The positive class is the target for what we are trying to fingerprint, while the negative class is everything else. An attacker, as part of the training data, is free to define the classes as they see fit. However, it is vital that the negative class is representative of the real-world traffic the attacker expects to see. Therefore, the only (indirect) filtering for the negative class is based on the existence of circuits: per definition, if a website visit does not contain a general or rendezvous circuit, it is not an O-L (or most kinds of OA) website visit (and trivially classified).

For the positive class—that defines the fingerprinting target—we perform three forms of filtering in our dataset. First, we filter out circuits to Cloudflare and their onion service[13], which uses alt-svc headers to transparently reroute to onionsites connection requests for content they host. Second, we filter out circuits caused by the Tor Browser extension HTTPS-Everywhere to SecureDrop, which hosts rulesets [49] that was checked on launch of Tor Browser in our data collection (for every website visit). Finally, for OLF, we set a minimum circuit length of 100 cells on both the general and rendezvous circuits. This is to ensure that the circuits are not just broken connections, and for utilizing the overlap between the general and rendezvous circuits. We emphasize again that the negative class is not filtered. Filtering of the positive class would not be needed if more care was taken during data collection to ensure that only fully complete website visits were included and only circuits that were part of the website visit were included.

With a more tailored classifier than DF, one could for example put each observed circuit during a website visit into a separate input to the classifier. Instead, we do a poor researcher's version by concatenating the first 2500 cells from the general circuit followed by the first 2500 cells from the rend circuit as the input trace. Note that this works because the rend circuit input always starts at position 2500 in the input trace. If cells are missing, we pad with zeroes. For the positive class, in the case where website visits have more than one general or rendezvous circuit after filtering, we use the largest circuit. For the negative class, in case of more than one circuit of any kind, we add every combination of general and rendezvous circuits to the negative class.

5.2.2 Automatic Onion-Location. We use the `autoloc` dataset for the positive class, using the largest general circuit and the largest rend circuit per visit (minimum 100 cells each), resulting in 4,392 samples. We consider two sources for the negative class: the `clearnet` dataset and the `onion` dataset.

With the `clearnet` dataset as the negative class, we have 14,143 samples. We get accuracy 99.87±0.09% (76.3% baseline) and FPR 0.16±0.10%. Note the skew towards the negative class.

With the `onion` dataset as the negative class, we have 4,685 samples. We get accuracy 98.81±0.74% (51.6% baseline) and FPR 1.23±0.74%. Note that in the `onion` dataset, the negative class is more balanced. However, 8,064 out of 9,566 website visits (84%) had no general circuit at all and were thus not considered for the negative class. Adjusting for this, the accuracy would be 99.8% ($(8064/9566) * 1.0 + (1 - 8064/9566) * 0.9881 = 0.9981$) with a FPR of 0.2% ($(1 - 8064/9566) * 0.0123 = 0.0019$).

To verify that fingerprinting is also robust in the presence of unknown O-L websites (e.g., due to new O-L websites being deployed after the attacker enumerates all O-L websites using the technique in Section 4), we also ran our experiment by splitting our dataset probabilistically based on website: for each website, with some probability, assign all positive and negative samples to either the training or testing set. This is similar to open-world WF, where unknown websites are present in the unmonitored set.

For the `clearnet` dataset, an 80/20 split (i.e., 80% used for training and 20% reserved for testing) resulted in accuracy 99.70±0.18% and FPR 0.30±0.23%, a 50/50 split resulted in accuracy 99.77±0.13% and FPR 0.24±0.13%, and a 30/70 split resulted in accuracy 99.68±0.09% and FPR 0.22±0.09%. Note that as the testing set size increases, the confidence interval shrinks and the accuracy remains within the confidence interval of the smaller testing sets. Similarly for FPR. This indicates that the attack generalizes beyond websites, because it is based on circuit kinds and overlap, not website content alone (unlike WF). The results for the `onion` dataset is similar (accounting for the 84% with no general circuit).

The 100-cell minimum for the general and rendezvous circuits in the positive class may seem arbitrary. We re-ran our experiments with decreasing minimum lengths, resulting in slowly degrading fingerprintability (albeit minor, comparing minimum length 100 and 0 the accuracy decreases from 99.87%±0.09% to 97.1%±0.39% and FPR increases from 0.16±0.10% to 1.24%±0.57%). However, we argue that the overlap is essential to consider in practice.

To establish how informative the overlap between the general and rendezvous circuits is in O-L, we create an artificial negative dataset. The dataset is constructed by randomly pairing general and rendezvous circuits from both the `clearnet` and `onion` datasets into samples, with the same 100-cell minimum and filtering as for the positive class. This ensures that the distinguishing factor is the overlap in content between circuits (with high probability), and not

[13] https://blog.cloudflare.com/cloudflare-onion-service

lengths or kinds of circuits. With 5,397 negative samples, we get accuracy 91.23±1.17% (55.1% baseline) and FPR 9.36±2.09%. That DF—a WF attack—can *correlate* general and rend circuits to such a high degree is a strong indication that the overlap is a distinguisher on its own. It is likely that flow correlation attacks could do even better [34]. Traces from Tor Browser with the Standard security level (where JavaScript is enabled) would likely also greatly improve WF and correlation attacks. We note that making a classifier that determines if a general circuit and a rendezvous circuit carries overlapping website traffic is closely related to both end-to-end correlation and WF. Specifically, the proposed one-page WF setting [57] by Wang showed this to be particularly advantageous to the attacker. We leave this as future work.

5.2.3 Manual Onion-Location. Identical to automatic O-L fingerprinting, except that the duration between the general and rendezvous circuits may be longer. This may lead to a larger number of circuits being observed in the time window, but the attacker can still use the—increasingly longer—overlap between the general and rendezvous circuits to filter out false positives. We did not experimentally evaluate manual O-L because it comes down to user behavior and background traffic assumptions where we lack real data from the Tor network (and means of ethically collecting it). Given what we know, it is plausible both that user behavior in the wild will significantly affect FPR and that it will not. For example, if users visit other clearnet or onion sites before eventually clicking the "onion available" button that could complicate OLF. But, users inclined to click as soon as the button appears would be just as fingerprintable as in automatic O-L.

5.2.4 Onion-Location via Sauteed Onions using TLS handshake. As a first step, an attacker would use CF to identify general circuits. A special case of a general circuit is when the client is redirected to an onion service by identifying its address as part of a TLS certificate, assumed to immediately stop any further requests on the general circuit. We create our two classes by using the tailored `curl` dataset (1,991 samples) and filter to include only general circuits from the `clearnet` dataset (21,795 samples). We get accuracy 100.00±0.01% (91.63% baseline) and FPR 0.00±0.00%.

5.2.5 Onion association via Sauteed Onions inline API. By necessity, this would involve at least two rendezvous circuits: one for communicating with the Sauteed Onions API (either newly created or reused, depending on implementation) and one for the onion service. If the API lookup does not block an additional general circuit will be observed, potentially with overlapping web traffic as for the O-L case. We did not experimentally evaluate this type of OA. As we discussed at the end of Section 3, it is possible that as onion service use continues to grow, such circuit patterns may become more common for other purposes than OA. For now, however, it is reasonable to assume at most a small increase in FPR over OLF.

Conclusion: OAF builds upon CF and is practical, allowing a guard attacker to greatly reduce the number of candidate websites, as long as the OA method results in creating and using additional active circuits.

6 Discussion

Fundamentally, our results in Section 5 show that the transition from clearnet to onion is extremely fingerprintable. This transition may happen due to O-L, users clicking an onion link from a clearnet website, users simply going to an onion after visiting an unrelated clearnet website, or something else. We do not have data from the live Tor network to say how prevalent these transitions are.

When the transition from clearnet to onion is to the same website—or even the same webpage as in Tor Browser today—then overlap of traffic in the general and rendezvous circuits will likely be detectable. Preventing detection is akin to defending against WF in the challenging one-page setting [57]. There is also a tradeoff between overlap and the attacker's timing signal if redirecting before a page has finished. The extreme here is redirects based on the certificate in the TLS handshake with no overlap but a clear timing signal.

Our results in Section 4 show that the work needed to uncover all recent O-L sites is well within the capabilities of even a relatively low-resource adversary. Thus, suppose an adversary Alice has done the CT and O-L dataset generation as well as the classifier training we have done; that Alice owns the guard of a client Bob that she intends to fingerprint; that example.com is a site among the roughly 1500 O-L sites on Alice's list; and that Bob visits example.com and follows an automated O-L redirect to the associated onionsite. (And assume for simplicity because it is left for future work that circuits are not using Conflux.) Recall that the attacks we analyze use data gathered by accessing real sites over the live Tor network via a guard we controlled. The site access fingerprints used in our analysis are thus real; analysis of detecting them is based on simulated rather than live connections in order to not place real Tor users at risk.

Our analyses show that Alice will succeed at an OLF of the connections to example.com and its onionsite with an accuracy of at least 99.8% and an FPR of no more than 0.2%. This is itself significant. But further, the limit to on the order of 1500 sites may prompt other reactions or other approaches to identify the destination, depending on the sites on the O-L list and adversary goals and capabilities.

Even if Alice uses only WF to identify the specific O-L site Bob is visiting and the particular site example.com is specifically targeted, previous results imply that accuracy of WF by Alice of just the exit connection can be roughly 95% [6]. If the target is the entire list, then the story is more complex and mostly less successful for an adversary conducting only a basic WF of a clearweb site. But, from the OLF attack Alice also knows the destination is an O-L site: any exit destination identified but not on the O-L list can be ruled out or trivially checked to see if it began offering O-L since the list was created. Since the base rate of onion service connections is only 5% of Tor traffic [21], OLF should dramatically improve accuracy and reduce FPR over an exit destination WF alone. Similarly, O-L facilitates Website Oracles (such as via DNS [9]), which have been shown to greatly improve accuracy and reduce FPR over WF of exit destinations alone [41]. Specific numbers for the results of a combined OLF, WO, and WF attack that O-L enables would require experiments and analysis beyond the scope of this paper, but the result should be a dramatic improvement in successful site identification over a WF attack by itself.

It is instructive to contrast O-L with onion discovery by simply putting on a website a link to an associated onion address. This may

be less convenient than O-L, but a circuit fingerprint of someone following the link in this case effectively identifies a connection to some onionsite or other. Since there is no systematic and practical way for the adversary to discover all onionsites after v3 [56], the adversary would need to have already been targeting the site or otherwise have discovered the onion address for a correct identification from the guard position to even be possible. By contrast, the adversary can systematically construct a list on which all recently active O-L sites will appear. Thus, correctly fingerprinting as O-L virtually guarantees that the onion address is one of the roughly 1500 on that list, whether or not the adversary had previously been aware of it. Note that even if we just assume that somehow the connection is to a known stable onionsite, that still makes for a possibility set at least 7-10 times larger than the list of O-L sites.

This work was done in coordination and collaboration with the Tor Project, and we will continue supporting them as they work to resolve problems we have uncovered. Our recommendations split into two groups: (1) things that are relatively straightforward to carry out and that we believe should be done immediately. (2) recommendations for research into making OA less fingerprintable.

6.1 Immediate recommendations

Simply offering an onionsite provides advantages versus just encouraging Tor access to a registered domain [33]. But the means of discovering the association between these matters with respect to resistance to multiple attacks. As noted in previous work, problems with O-L include facilitating censorship and hijack [10, 49], and as we have shown in this paper, facilitating fingerprinting.

For some sites, a significant fraction of their users would be at personal risk if fingerprinting revealed the user to be visiting that site. These include whistleblower sites and dissident or controversial information sites whose adversaries are well-resourced and repressive. For such sites, the convenience of O-L may not justify the risk. We do not know, however, what the fingerprint risk is (for example FPR) for *manual* O-L, nor can we easily decide which sites have user populations at significant risk. And manual O-L does provide convenient onion discovery for users. We thus have not recommended disabling manual O-L in general in the near future. But we do know that *automated* O-L is extremely fingerprintable due to its automated and deterministic nature.

Immediate recommendation: Tor Browser should immediately stop offering an automatic O-L option.

The work reported in this paper was done in coordination with other members and collaborators of the Tor Project. Automatic O-L was removed from Tor Browser as of version 13.0.12, released March 19, 2024 [55]. Our presentation herein has been based in Tor Browser. Brave Browser (https://brave.com) also supports Tor network connections and onion service access. Brave Browser offered essentially the same options of manual and automatic O-L up until March 2024 and was thus subject to the same OLF attacks. They similarly removed automatic O-L at that time.

Like its header-based counterparts, automated O-L based on sauteed-onion certificates would be highly fingerprintable. Sauteed-onion certificates do, however, enable right now manual use of CT log monitors for OA that is more secure in various ways, more resistant to blocking, and less fingerprintable (and less easy to use) than O-L, as well as not subject to easy Website Oracles. And since there is as yet no browser support for CT-log-monitor-based O-L, such manual OA is the only form of OA available using CT log monitors. Sauteed OA is thus a somewhat user-friendly form of OA that also encourages growth of associated-onion space. And a sauteed-onion certificate is generally free and easy to obtain [10]. A sauteed-onion certificate does not by itself reduce fingerprintability. But, especially for any domain that is likely to be subject to Internet blocks or disruption or is concerned about the transparent consistency of its OA, there is benefit and little cost to doing so.

6.2 Research recommendations

CF underpins the fingerprintability of most available approaches to OA and onion services in general. Improving CF defenses is thus a natural research direction. Ideally, rendezvous and general-purpose circuits should be indistinguishable, and fake HSDir and introductory circuits are needed as part of general-purpose circuits to not be a clear distinguisher. Kadianakis et al. [25] already did much of the ground work for such defenses, but work remains, also around the use of the Tor Circuit Padding Framework [39] for real deployment in the near to mid term. As noted, Tor already ships with padding machines, but they are not effective (see Section 5).

Within the Circuit Padding Framework, Tor could also deploy one or more practical WF defenses [32], perhaps after tailoring them to onionsites. While the real-world impact of WF attacks has been debated, the small world of onionsites has been recognized as more fingerprintable [21, 29, 36, 37, 41], and our work shows that the O-L world is even smaller. Note that deploying WF defenses only for rendezvous circuits may confound other efforts to reduce circuit fingerprintability, because web traffic over rendezvous circuits would potentially be even more peculiar than it is now [25].

Research recommendation: Continue the work of Kadianakis et al. [25] on CF defenses for the Tor Circuit Padding Framework [39]. Investigate practical WF defenses [32] and CF defenses that are tailored to onionsites.

Any automated O-L is highly susceptible to OLF, similarly for automated OA via a sauteed-onion-based discovery server at an onion service. For a subsequent WF attack, automated OA from an onion-service-hosted server will avoid easy exit Website Oracles, but until PIR or similar is developed for this setting we must make the potentially show-stopping assumption that the server itself is trusted to not act as an oracle for a WF adversary. Moving to local OA would raise the OA anonymity set by roughly an order of magnitude to that of general onionsite access. Local association lists could be based on periodic retrievals of all OAs from the sauteed-onion search API, as described in Section 2.2.4 or built by other criteria. Stability and dynamics of the OA site list is an important deployment question for how often such retrievals should occur.

Another open deployment question is simply how much the list of OAs could grow before it becomes too untenably large to download to clients regularly. It might take some time if growth is organic, but it might happen much faster if, intentionally or not, many OAs are set up without sustained intent to use the association.

For example, someone who wants to overwhelm the clients of this system could obtain sauteed-onion certificates for large numbers of subdomains of registered domains they own.

If it were possible to include both the domain name and the onion address in a single URL understandable to both Tor-aware and Tor-unaware browsers, it might be possible to include among the addresses comprising the nodes of the Web, addresses that have OA effectively built in. This could potentially avoid the CF issues of realtime lookup from a remote server and also the scaling issues of local lists of OAs. Self-Autheniticating Traditional Addresses (SATAs) [49, 50] were designed to address limitations of the existing Web authentication infrastructure, but they also employ just such addresses. They are thus another possible avenue to secure, scalable, fingerprint resistant OA.

Research recommendation: Investigate schemes that address or circumvent the scaling limitations of local onion association.

7 Related work

There are many different approaches to associating onion addresses with domain names. These include Onion-Location and sauteed onions, both of which have been described above.

The alt-svc approach, briefly described above, uses an HTTP header sent when connecting to the registered domain, as does O-L. Unlike O-L, the header has a max age parameter, so re-routing to the onion address does not require connecting to the original domain until the header expires. This might seem a good thing with respect to fingerprinting resistance: during that time there will only be onion service connections for that domain. But, there is no easy way for the user (or the browser) to know whether re-routing is taking place, thus no easy way to know how fingerprintable a connection to an alt-svc-supporting domain is (although a guard adversary should be able to tell easily). Also, onion alt services facilitate targeted tracking of users and have other problems [50]. Therefore, alt-svc support in Tor Browser remains extremely problematic despite potential gains in terms of fingerprinting resistance.

Another approach is to use DNS records, either by means of an HTTPS Resource Record, as proposed in RFC 9460 [46], or another existing record such as the TXT or SRV record. A potential fingerprintability advantage over O-L is that there would not need to be an exit connection to the domain for redirection or "clicked" selection to connect to the onion service. There would still need to be a DNS lookup, however. Thus, the circuit fingerprints would be comparable to those of O-L, and this would still support a WO fingerprinting of the onion association. This approach as well as the alt-svc approach and O-L and sauteed onions are described on the Tor Project page on "Traditional address translation" [54].

Many prior works measured CT logs in various ways, e.g., for the purpose of studying the log system itself [15, 26, 31] or the certificates that are being published [27, 28, 35, 42, 45]. Some develop their own tools to measure the logs in ad-hoc ways, whereas others depend on existing certificate datasets such as the one Censys puts together from CT and active scans [12]. Unlike prior work, our tool for measuring CT logs is tailored for easily creating a reproducible dataset of SANs that is tied to the logs' cryptographic states. Such a dataset can be used for a wide variety of active HTTPS/TLS measurements [51]. We conducted one such measurement ourselves, informing for the first time on the prevalence of sites configuring O-L as well as the characteristics of how such configuration is done.

As noted, we believe ours to be the first study of the number of O-L sites. The current number of onionsites in general is not easy to assess, and reported numbers vary widely. The Tor Metrics portal has been reporting roughly 800K addresses for over a year. But this is based on the number of addresses in the directory system, which is probably more than forty times the number of publicly reachable addresses. Pastor-Galindo et al. recently published a survey of onion address measurement studies [38] reflecting widely varying numbers. Most studies covering onion service availability have found that most addresses are unreachable or highly volatile. Wang et al. found, however, that of the roughly 50K addresses they deemed "public" based on being retrievable from popular search engines, roughly 80% were available roughly 80% of the time and 10% were available over 90% of the time [56]. Another factor affecting any measurement studies is that Tor moved from v2 to v3 onion services a few years ago (support for v2 in stable Tor clients was removed by the end of 2021). v3 onion services not only have incompatibly different address formats, but descriptors became encrypted, and where they are stored in the DHT became unpredictable. Thus, studies done more than a few years ago are even less likely to be reflective of the state of onionspace.

Our methodology of measuring cell traces at a relay using in-protocol client-to-relay signaling was first set out by Cherubin et al. [6] and recently used to create GTT23, the largest existing dataset of genuine Tor traces [23, 24]. Many other studies of website fingerprinting have constructed datasets of cell traces by programmatically accessing websites through the Tor network. The most recent examples are the Drift dataset [4], the Wikipedia browsing dataset [22], the Multi-tab dataset [11], and the BigEnough dataset [32]. However, relatively few existing datasets contain cell traces collected from onion service circuits [18], and we are not aware of any which contain associated traces of the same website loaded through both clearnet and onion services, as our dataset does. Additionally, we are the first to measure the cell traces resulting from automatic O-L redirection, which was required in order to explore the research questions in this paper.

8 Conclusion

We have provided the first Internet-wide measurement of Onion-Location, finding 1505 stable sites offering O-L. We have also created open-source tools to repeat such measurements, thus supporting the ability to track the state of O-L deployment over time. We have also performed fingerprintability analysis on O-L and other onion association schemes showing that O-L is easily fingerprintable by a guard adversary with high accuracy. Based on our analysis, we make several recommendations including some for easy immediate changes to counter O-L fingerprintability and others for research to create onion association solutions without the limitations of O-L.

Artifacts availability

https://github.com/pylls/ol-measurements-and-fp and [7, 8].

Acknowledgments

We would like to thank Björn Töpel for debugging help when creating `onion-grab`. We would also like to thank the Tor Project, especially Silvio Rhatto and Micah Anderson, for helpful comments and discussions while researching and writing this paper; as well as Pier Angelo Vendrame for reporting a bug in how `onion-grab` noted down results for sites with HTTP redirects to other O-L sites. Some authors used GitHub Copilot and Grammarly integrated into their LaTeX editor to revise the text in the paper (no explicit prompting, continuous feedback). Work by Tobias Pulls funded by the Knowledge Foundation of Sweden.

References

[1] Gunes Acar, Marc Juarez, and individual contributors. 2023. tor-browser-selenium - Tor Browser automation with Selenium. https://github.com/webfp/tor-browser-selenium.

[2] Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg. 2013. The path less travelled: Overcoming Tor's bottlenecks with traffic splitting. In *Privacy Enhancing Technologies: 13th International Workshop, PETS 2006*. Springer-Verlag, LNCS 7981. https://doi.org/10.1007/978-3-642-39077-7_8

[3] Jacob Appelbaum and Alec Muffett. 2015. The ".onion" Special-Use Domain Name. RFC 7686. https://doi.org/10.17487/RFC7686

[4] Alireza Bahramali, Ardavan Bozorgi, and Amir Houmansadr. 2023. Realistic Website Fingerprinting By Augmenting Network Traces. In *ACM CCS 2023: 30th Conference on Computer and Communications Security*. ACM Press. https://doi.org/10.1145/3576915.3616639

[5] Yazan Boshmaf, Isuranga Perera, Udesh Kumarasinghe, Sajitha Liyanage, and Husam Al Jawaheri. 2023. Dizzy: Large-Scale Crawling and Analysis of Onion Services. In *Proceedings of the 18th International Conference on Availability, Reliability and Security* (Benevento, Italy) *(ARES '23)*. Association for Computing Machinery, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/3600160.3600167

[6] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. 2022. Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world. In *31st USENIX Security Symposium (USENIX Security 22)*. 753–770.

[7] Rasmus Dahlberg. 2023. ct-sans. https://git.rgdd.se/ct-sans.

[8] Rasmus Dahlberg. 2023. onion-grab. https://gitlab.torproject.org/tpo/onion-services/onion-grab.

[9] Rasmus Dahlberg and Tobias Pulls. 2023. Timeless Timing Attacks and Preload Defenses in Tor's DNS Cache. In *32nd USENIX Security Symposium (USENIX Security 23)*.

[10] Rasmus Dahlberg, Paul Syverson, Linus Nordberg, and Matthew Finkel. 2022. Sauteed Onions: Transparent Associations from Domain Names to Onion Addresses. In *Proceedings of the 21st ACM Workshop on Workshop on Privacy in the Electronic Society (WPES '22)*. ACM, ACM Press, 35–40. https://doi.org/10.1145/3559613.3563208

[11] Xinhao Deng, Qilei Yin, Zhuotao Liu, Xiyuan Zhao, Qi Li, Mingwei Xu, Ke Xu, and Jianping Wu. 2023. Robust Multi-tab Website Fingerprinting Attacks in the Wild. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1005–1022. https://doi.org/10.1109/SP46215.2023.10179464

[12] Zakir Durumeric, David Adrian, Ariana Mirian, Michael D. Bailey, and J. Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM, 542–553. https://doi.org/10.1145/2810103.2813703

[13] Edward Eaton, Sajin Sasy, and Ian Goldberg. 2022. Improving the Privacy of Tor Onion Services. In *Applied Cryptography and Network Security (ACNS 2022)*, Giuseppe Ateniese and Daniele Venturi (Eds.). Springer-Verlag, LNCS 13269, 273–292. https://doi.org/10.1007/978-3-031-09234-3_14

[14] David Goulet and Mike Perry. 2020. Overcoming Tor's Bottlenecks with Traffic Splitting. https://gitlab.torproject.org/tpo/core/torspec/-/raw/main/proposals/329-traffic-splitting.txt.

[15] Josef Gustafsson, Gustaf Overier, Martin F. Arlitt, and Niklas Carlsson. 2017. A First Look at the CT Landscape: Certificate Transparency Logs in Practice. In *Passive and Active Measurement - 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10176)*, Mohamed Ali Kâafar, Steve Uhlig, and Johanna Amann (Eds.). Springer, 87–99. https://doi.org/10.1007/978-3-319-54328-4_7

[16] Alexis Hancock. 2020. 10 Years of HTTPS Everywhere. https://www.eff.org/deeplinks/2020/11/10-years-https-everywhere. Online; Retrieved Feb. 13 2024.

[17] Alexis Hancock. 2021. HTTPS is Actually Everywhere. https://www.eff.org/deeplinks/2021/09/https-actually-everywhere. Online; Retrieved Feb. 19 2024.

[18] Jamie Hayes and George Danezis. 2016. *k*-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *USENIX Security 2016: 25th USENIX Security Symposium*.

[19] Andrew Hintz. 2002. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies: Second International Workshop, PET 2002*, Roger Dingledine and Paul Syverson (Eds.). Springer-Verlag, LNCS 2482, San Francisco, CA, USA, 171–178.

[20] Aaron D. Jaggard and Paul Syverson. 2017. Onions in the Crosshairs: When The Man really is out to get you. In *Proceedings of the 16th ACM Workshop on Workshop on Privacy in the Electronic Society (WPES '17)*. ACM, ACM Press, Dallas, Texas, USA. https://doi.org/10.48550/arXiv.1706.10292

[21] Rob Jansen, Marc Juarez, Rafael Galvez, Tariq Elahi, and Claudia Diaz. 2018. Inside Job: Applying Traffic Analysis to Measure Tor from Within. In *25th Annual Network and Distributed System Security Symposium (NDSS 2018)*. The Internet Society. https://doi.org/10.14722/ndss.2018.23261

[22] Rob Jansen and Ryan Wails. 2023. Data-Explainable Website Fingerprinting with Network Simulation. *Proceedings on Privacy Enhancing Technologies* 2023, 4 (July 2023). https://doi.org/10.56553/popets-2023-0125

[23] Rob Jansen, Ryan Wails, and Aaron Johnson. 2024. *GTT23: A 2023 Dataset of Genuine Tor Traces*. https://doi.org/10.5281/zenodo.10620519

[24] Rob Jansen, Ryan Wails, and Aaron Johnson. 2024. A Measurement of Genuine Tor Traces for Realistic Website Fingerprinting. arXiv:2404.07892 [cs.CR]

[25] George Kadianakis, Theodoros Polyzos, Mike Perry, and Kostas Chatzikokolakis. 2022. Tor circuit fingerprinting defenses using adaptive padding. arXiv:2103.03831 [cs.CR]

[26] Nikita Korzhitskii and Niklas Carlsson. 2020. Characterizing the Root Landscape of Certificate Transparency Logs. In *2020 IFIP Networking Conference, Networking 2020, Paris, France, June 22-26, 2020*. IEEE, 190–198. https://ieeexplore.ieee.org/document/9142756

[27] Nikita Korzhitskii and Niklas Carlsson. 2021. Revocation Statuses on the Internet. In *Passive and Active Measurement - 22nd International Conference, PAM 2021, Virtual Event, March 29 - April 1, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12671)*, Oliver Hohlfeld, Andra Lutu, and Dave Levin (Eds.). Springer, 175–191. https://doi.org/10.1007/978-3-030-72582-2_11

[28] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J. Alex Halderman, and Michael D. Bailey. 2018. Tracking Certificate Misissuance in the Wild. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 785–798. https://doi.org/10.1109/SP.2018.00015

[29] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX, 287–302.

[30] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. *Certificate Transparency*. RFC 6962. IETF.

[31] Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Wei Wang, Qi Li, Guangshen Cheng, Jiwu Jing, and Congli Wang. 2022. The Invisible Side of Certificate Transparency: Exploring the Reliability of Monitors in the Wild. *IEEE/ACM Trans. Netw.* 30, 2 (2022), 749–765. https://doi.org/10.1109/TNET.2021.3123507

[32] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. 2022. SoK: A Critical Evaluation of Efficient Website Fingerprinting Defenses. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 344–361.

[33] Alec Muffett. 2022. Why offer an Onion Address rather than just encourage browsing-over-Tor? https://alecmuffett.com/article/16007. Online; Retrieved Feb. 16 2024.

[34] Se Eun Oh, Taiji Yang, Nate Mathews, James K. Holland, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. 2022. DeepCoFFEA: Improved Flow Correlation Attacks on Tor via Metric Learning and Amplification. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*.

[35] Olamide Omolola, Richard Roberts, Md. Ishtiaq Ashiq, Taejoong Chung, Dave Levin, and Alan Mislove. 2021. Measurement and Analysis of Automated Certificate Reissuance. In *Passive and Active Measurement - 22nd International Conference, PAM 2021, Virtual Event, March 29 - April 1, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12671)*, Oliver Hohlfeld, Andra Lutu, and Dave Levin (Eds.). Springer, 161–174. https://doi.org/10.1007/978-3-030-72582-2_10

[36] Rebekah Overdorf, Marc Juarez, Gunes Acar, Rachel Greenstadt, and Claudia Diaz. 2017. How Unique is Your .onion? An Analysis of the Fingerprintability of Tor Onion Services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, ACM Press, 2021–2036.

[37] Andriy Panchenko, Asya Mitseva, Martin Henze, Fabian Lanze, Klaus Wehrle, and Thomas Engel. 2017. Analysis of fingerprinting techniques for Tor hidden services. In *Proceedings of the 16th ACM Workshop on Workshop on Privacy in the Electronic Society (WPES '17)*. ACM, ACM Press, 165–175. https://doi.org/10.1145/3139550.3139564

[38] Javier Pastor-Galindo, Félix Gómez Mármol, and Gregorio Martínez Pérez. 2023. On the gathering of Tor onion addresses. *Future Generation Computer Systems* 145 (2023), 12–26. https://doi.org/10.1016/j.future.2023.02.024

[39] Mike Perry and George Kadianakis. 2022. Tor Padding Specification. https://spec.torproject.org/padding-spec/overview.html. Online; Retrieved Feb. 25 2024.

[40] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *26th Annual Network and Distributed System Security Symposium, (NDSS 2019*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/tranco-a-research-oriented-top-sites-ranking-hardened-against-manipulation/

[41] Tobias Pulls and Rasmus Dahlberg. 2020. Website Fingerprinting with Website Oracles. *Proceedings on Privacy Enhancing Technologies* 2020, 1 (2020), 235–255.

[42] Richard Roberts and Dave Levin. 2019. When Certificate Transparency Is Too Transparent: Analyzing Information Leakage in HTTPS Domain Names. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, (WPES '19), London, UK, November 11, 2019*, Lorenzo Cavallaro, Johannes Kinder, and Josep Domingo-Ferrer (Eds.). ACM, 87–92. https://doi.org/10.1145/3338498.3358655

[43] Mahrud Sayrafi. 02018. Introducing the Cloudflare Onion Service. https://blog.cloudflare.com/cloudflare-onion-service/. Online; Retrieved Feb. 12 2024.

[44] Mahrud Sayrafi. 2018. Introducing DNS Resolver for Tor. https://blog.cloudflare.com/welcome-hidden-resolver/. Online; Retrieved Feb. 12 2024.

[45] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. 2018. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*. ACM, 343–349. https://dl.acm.org/citation.cfm?id=3278562

[46] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren. 2023. Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records). RFC 9460. https://doi.org/10.17487/RFC9460

[47] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. 2023. Subverting Website Fingerprinting Defenses with Robust Traffic Representation. In *32nd USENIX Security Symposium (USENIX Security 23)*.

[48] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1928–1943. https://doi.org/10.1145/3243734.3243768

[49] Paul Syverson, Matthew Finkel, Saba Eskandarian, and Dan Boneh. 2021. Attacks on Onion Discovery and Remedies via Self-Authenticating Traditional Addresses. In *Proceedings of the 20th ACM Workshop on Privacy in the Electronic Society (WPES '21)*. ACM, ACM Press, 45–52. https://doi.org/10.1145/3463676.3485610

[50] Paul Syverson and Matt Traudt. 2019. Self-Authenticating Traditional Domain Names. In *2019 IEEE Secure Development (SecDev)*. IEEE, 147–160.

[51] Pouyan Fotouhi Tehrani, Eric Osterweil, Thomas C. Schmidt, and Matthias Wählisch. 2024. How to Measure TLS, X.509 Certificates, and Web PKI: A Tutorial and Brief Survey. *CoRR* abs/2401.18053 (2024). https://doi.org/10.48550/ARXIV.2401.18053 arXiv:2401.18053

[52] Tor Project 2019. *Getting up to speed on Tor's past, present, and future.* https://2019.www.torproject.org/docs/documentation.html.en Online; Retrieved Feb. 12 2024.

[53] Tor Project 2021. *Onion-Location.* https://community.torproject.org/onion-services/advanced/onion-location/ Online; Retrieved Feb. 8 2024.

[54] Tor Project. 2023. Traditional address translation. https://tpo.pages.torproject.net/onion-services/onionplan/proposals/usability/discovery/translation/. Online; Retrieved Feb. 25 2024.

[55] Tor Project 2024. *New Release: Tor Browser 13.0.12 | The Tor Project.* https://blog.torproject.org/new-release-tor-browser-13012/ Online; Retrieved Aug. 20 2024.

[56] Chunmian Wang, Junzhou Luo, Zhen Ling, Lan Luo, and Xinwen Fu. 2023. A Comprehensive and Long-term Evaluation of Tor V3 Onion Services. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM53939.2023.10229057

[57] Tao Wang. 2021. The One-Page Setting: A Higher Standard for Evaluating Website Fingerprinting Defenses. In *CCS '21: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. ACM, New York, NY, USA, 2794–2806. https://doi.org/10.1145/3460120.3484790

[58] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. *Proceedings on Privacy Enhancing Technologies* 2016, 4 (2016), 21–36. https://doi.org/10.1515/popets-2016-0027