Analysis of Fingerprinting Techniques for Tor Hidden Services

Andriy Panchenko
University of Luxembourg
andriy.panchenko@uni.lu

Asya Mitseva
University of Luxembourg
asya.mitseva@uni.lu

Martin Henze
RWTH Aachen University
henze@comsys.rwth-aachen.de

Fabian Lanze
Huf Secure Mobile GmbH
fabian@lanze.net

Klaus Wehrle
RWTH Aachen University
wehrle@comsys.rwth-aachen.de

Thomas Engel
University of Luxembourg
thomas.engel@uni.lu

ABSTRACT

The website fingerprinting attack aims to infer the content of encrypted and anonymized connections by analyzing traffic patterns such as packet sizes, their order, and direction. Although it has been shown that no existing fingerprinting method scales in Tor when applied in realistic settings, the case of Tor hidden (onion) services has not yet been considered in such scenarios. Recent works claim the feasibility of the attack in the context of hidden services using limited datasets.

In this work, we propose a novel two-phase approach for fingerprinting hidden services that does not rely on malicious Tor nodes. In our attack, the adversary merely needs to be on the link between the client and the first anonymization node. In the first phase, we detect a connection to a hidden service. Once a hidden service communication is detected, we determine the visited hidden service (phase two) within the hidden service universe. To estimate the scalability of our and other existing methods, we constructed the most extensive and realistic dataset of existing hidden services. Using this dataset, we show the feasibility of phase one of the attack and establish that phase two does not scale using existing classifiers. We present a comprehensive comparison of the performance and limits of the state-of-the-art website fingerprinting attacks with respect to Tor hidden services.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**; **Privacy-preserving protocols**; • **Networks → Network privacy and anonymity**; *Network security*;

KEYWORDS

Traffic Analysis; Website Fingerprinting; Privacy; Anonymous Communication; Onion Routing; Tor Hidden Services; Onion Services; Web Privacy

WPES'17, October 30, 2017, Dallas, TX, USA.

ACM ISBN 978-1-4503-5175-1/17/10…$15.00
https://doi.org/10.1145/3139550.3139564

1 INTRODUCTION

In the age of mass surveillance, privacy on the Internet has become a concern. One of the fundamental building blocks for achieving privacy is *anonymity*. Several approaches have been proposed for anonymous communication, but only a few of these have reached widespread deployment. Tor is the most popular low-latency anonymization network that aims to hide the user's IP address while communicating on the Internet [13]. The traffic exchange between the sender and the receiver is tunneled over multiple nodes, known as *onion relays* (ORs). The client, as initiator of a connection runs an onion proxy (OP) and creates a virtual tunnel, called a *circuit*, to the destination over typically three nodes: *entry*, *middle*, and *exit* [12]. By applying Diffie-Hellman key exchange, the sender negotiates a separate symmetric key with each OR in the chain. The symmetric keys are used to encrypt the actual user data with multiple layers of encryption [12]. While forwarding the user data, each of the ORs removes (or adds, depending on the direction) a layer of encryption. This ensures that each node in the chain knows only its direct predecessor and successor. Beside protecting the client's privacy, Tor also allows servers to operate anonymously by offering (location-)*hidden services*[1] (HSs). HSs allow users, in particular those living in oppressive regimes, e.g., human right activists and whistle-blowers, to bypass censorship and to exercise freedom of speech by publishing and offering access to content without being pursued, arrested, or forced to shut down their services. Hence, it is vital to know the level of protection offered by this popular anonymization technique.

The website fingerprinting (WFP) attack is a special case of *traffic analysis*. It attempts to infer information about the content (i.e., the website visited) of encrypted and anonymized connections by observing patterns of data flows. Here, the attacker is merely a passive local observer and utilizes meta information, such as packet size and direction, without breaking the encryption. Though Tor claims to protect against local observers, the attack has been shown to be feasible [29] and study of its limits has become an active field of research. Later, it was shown that none of the existing WFP attacks scales when applied in realistic settings, due a large set of all possible websites [28]. However, the universe of hidden services is small (a few thousands) and the impact of the WFP attack can be alarming. Recent studies confirm this fact, but only using datasets limited in size [16, 20]. In this paper, we analyze the impact of this attack on Tor hidden services using realistic datasets and applying state-of-the-art classifiers. Our contribution is as follows:

[1] Also known as *onion services*.

(1) We propose a novel two-phase fingerprinting attack which does not rely on malicious entry nodes. In our case, the attacker observes the link between the client and the entry node. In phase one, we detect whether the considered connection is to a HS or not. In phase two, we detect which particular HS content is being accessed.
(2) We collect the most comprehensive dataset of hidden services representing the real world. Instead of restricting the dataset only to publicly available hidden service addresses, we gather all HSs available in the Tor network to study the impact of the attack in real-world settings.
(3) Using our dataset, we show that detection of a connection establishment to a hidden service is feasible, whereas particular content recognition does not scale using any existing methods when applied in realistic settings.

2 BACKGROUND

In this section, we explain the functionality of the Tor hidden services. Figure 1 illustrates the steps to set up a hidden service in Tor and to establish a connection to it. The server operator runs an onion proxy in order to offer the server via Tor. Afterwards, the server's OP selects three ORs, called *introduction points* (InPs), and builds a separate circuit to each of them (step 1) [6]. The server informs the InPs about a service key[2] currently associated with its hidden service. The use of service keys prevents an InP from tracking the HS's activities, since the InPs are not able to recognize which service they are serving. Next, the server generates a piece of information called *hidden service descriptor*. This descriptor contains the public key currently associated with its HS and a list of its InPs with the corresponding service keys [6]. It publishes the descriptor anonymously to a subset of ORs, called *hidden service directories* (HSDirs), in order to advertise its service in Tor (step 2).

An OR is a HSDir if it provides an open port to store and serve hidden service descriptors and has an uptime of at least 96 hours [6]. Based on the descriptor identifier (ID), the HS determines whether a certain HSDir is responsible for keeping its descriptor. In Tor, the HSDirs are represented in the form of a distributed hash table (DHT) where each HSDir is identified by its fingerprint. The first three consecutive HSDirs, whose fingerprints are greater than the descriptor ID, are selected to store the HS descriptor [6].

To access a HS, the client needs to obtain its address. The HS address, also called *onion address*, is in the form of *x.onion* where x denotes the first 80 bits of the hash of the HS public key [6]. The client computes the ID of the corresponding HS descriptor to retrieve a list of responsible HSDirs. It fetches the HS descriptor from one of these HSDirs (step 3) to learn the InPs' addresses of the target HS and the corresponding service keys associated with these InPs. Before connecting to an InP, the client creates a circuit to a randomly selected OR, called *rendezvous point* (RP), (step 4) and sends an arbitrary value, i.e., *rendezvous cookie* (RC), to that OR. Next, the client builds a new circuit to one of the InPs of the HS to inform the HS about the ID of the selected RP and the RC (step 5). The InP forwards this information to the HS (step 6). If the HS accepts the client's request, it builds a new circuit to the RP (step 7).

[2]Instead of sharing its public key, the server generates separate short-term single-use keys, called *service keys*, used for communication with the InPs.

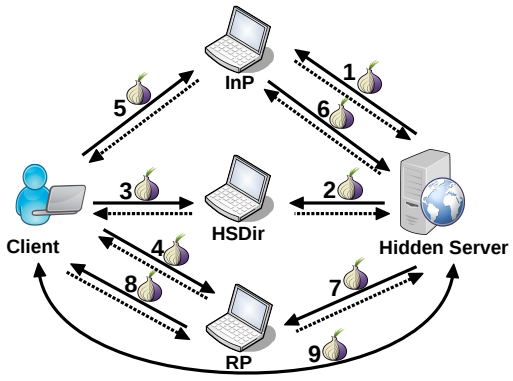


Figure 1: A hidden service protocol flow [6].

If the RP recognizes the RC sent by the HS, it informs the client that the connection with the HS is established (step 8). Subsequently, the RP starts transparently relaying encrypted packets between the client and the hidden server (step 9).

Beside publicly accessible HSs, the server can be configured to permit only authorized clients to access its service. The current Tor implementation specifies two protocols to set up client authorization; interested readers are referred to [6] for further information.

3 RELATED WORK

In the following, we survey known attacks on HSs and briefly overview the state-of-the-art WFP techniques targeting Tor users.

Attacks on Tor hidden services: The first documented attack on HSs was presented by Øverlier et al. [27]. They considered the first version of the HS protocol [13] where no guard nodes (persistently used entry relays) were used. In that scenario, the attacker controls (at least) one Tor node and repeatedly attempts to connect to a HS until his node is chosen as a first hop of the rendezvous circuit established by the HS. This leads to a situation where the attacker controls both ends of the communication. Then, the attacker sends specific traffic patterns along the circuits to perform traffic correlation and, thus, reveals the HS's location. To prevent such attacks, Øverlier et al. introduced the use of guard nodes, initially proposed by Wright et al. [36].

Murdoch et al. [24, 37] exploited the fact that the frequency of computers' system clocks is affected by the temperature of the CPU. The authors put loads on HSs by initiating connections and fetching content while remotely measuring the changes in the frequency of the system clocks by observing TCP/ICMP timestamps. This attack assumes the list of candidate HSs is known, and has limited scalability as each candidate from the list has to be probed.

Biryukov et al. [8] proposed a method for enumerating HS addresses in a short time interval. Given the list of gathered onion addresses, the authors further showed how to measure the popularity of any HS and how to perform a selective Denial of Service (DoS) attack on them; how to reveal guard nodes of HSs; and how to deanonymize a significant portion of HSs. To collect almost all HS addresses, they ran a large number of ORs with predefined fingerprints using a limited number of IP addresses. The fingerprints fall into every second interval between two consecutive HSDirs,

and, thus, cover all intervals of the DHT. Although only two ORs per IP address appear in the live network according to the Tor specification, the authors exploited the fact that directory authorities still collect statistics and allow bootstrapping for all ORs controlled by the adversary. When one of the active relays becomes unreachable, the authorities replace it with another OR from the set of attacker's relays. The new OR has all the flags, including HSDir flag, according to its real run time and not to the time for which it was in the consensus. In their follow-up work, Biryukov et al. [7] exploited the collected onion addresses to analyze the landscape of HSs and present statistics about open ports used by HSs, content distribution, and a geographical distribution of clients of HSs. Please note that the Tor project is currently designing and implementing countermeasures to hamper harvesting of onion addresses [21]. The same applies for predictable positioning of HSDirs in the DHT: the use of distributed randomness commitments affecting DHT positioning over time hinders this.

Jansen et al. [18] presented a memory-based DoS attack where the attacker identifies and disables the entry nodes of a target HS and, thus, forces the server to select new guards. Matic et al. [22] exploited information leakages in the configuration and content of HSs to reveal their location. They looked for URLs or email addresses pointing to regular websites and hosted on the same server, investigated HTTPS certificates to extract candidate IP addresses, searched for HS-specific strings, and used search engines to identify candidates hosting similar content.

In summary, the majority of attacks on HSs presented so far either assume an active attacker model or a malicious client that permanently tries to connect to a target HS. Furthermore, most of them are heavily dependent on specific protocol flaws already fixed by the Tor project [1, 3] or misconfigurations made by the administrators. In contrast to this, our adversary merely needs to be on the link between a HS client and its entry node to be successful.

Website Fingerprinting: The first work applying website fingerprinting to deanonymize Tor users was presented by Herrmann et al. [17]. On a dataset consisting of 775 websites, the authors achieved a recognition rate of only 2.95%. Thus, Tor was considered to provide a high level of security against WFP. The first successful WFP attack against Tor was presented in 2011 [29]. The authors proposed several types of features to represent a page loading, e.g., the total number of transmitted bytes, the total size of incoming and outgoing packets, and applied a Support Vector Machines (SVM) classification technique. In a closed-world scenario, they recognized 775 index pages with a classification accuracy of almost 55%. In addition, the authors evaluated the first WFP attack in an open-world setting, i.e., they aimed to identify a small number of websites within a set of thousands of random pages that had not previously been seen by the classifier, and achieved a true positive rate (TPR) of up to 73% and a false positive rate (FPR) of 0.05%. The results prompted a considerable amount of further investigation in the area of WFP.

Dyer et al. [14] compared existing classifiers and evaluated additional features on datasets consisting of two, 128, and 775 websites. However, they did not achieve better classification accuracy than [29]. In 2012, Cai et al. [9] improved the recognition rate to over 80% for a set of 100 pages and to over 70% for a set of 800 pages by using features based on the optimal string alignment distance (OSAD), and applying a SVM. Wang et al. [33] proposed modifications to the optimal string alignment distance of Cai et al. and introduced a new layer for feature extraction, namely Tor cells. The authors achieved more than 90% of classification accuracy for both the closed-world (100 pages) and open-world (1,000 pages) scenarios. Further research by Wang et al. [32] suggested a novel k-Nearest Neighbor (k-NN) classifier, which reduced the computation time significantly. The authors achieved a recognition rate of 91% in closed-world (100 pages) and 85% in open-world (> 5,000 pages) scenarios.

Juarez et al. [19] assessed several assumptions made by previous works in the field of WFP. The authors showed that the recognition rate of pages reduces dramatically over time due to content change. Furthermore, they observed a significant reduction in the classification accuracy if a user performs multitab browsing or if different Tor Browser Bundle (TBB)[3,4] versions are applied for training and testing. Juarez et al. were the first who considered the base-rate fallacy in the scope of WFP. Due to the enormous variety of pages in the world wide web, in most cases the adversary would wrongly conclude that a user has visited a monitored page. Taking into account the conclusions derived from [19], other works [15, 34] continued investigating the feasibility of WFP in general.

In 2016, Panchenko et al. [28] proposed a new classifier, called CUMUL, that outperforms all existing methods both in terms of classification accuracy and computational complexity. The authors were the first to collect a comprehensive and representative dataset and to evaluate the attack against it. They showed that fingerprinting websites, i.e., a set of webpages served under the same domain, scales significantly better than webpage fingerprinting. Moreover, they also showed that no existing fingerprinting method scales when applied in realistic settings, i.e., for every webpage in the dataset there are at least several others that look similar to the classifier and, thus, cause confusion.

Contrary to the cases above, the number of HSs is significantly smaller than the huge universe size of the world wide web. Hence, if the adversary is able to reliably distinguish a HS connection from a regular one, HS fingerprinting attack becomes similar to a closed-world setting, where the WFP attack is known to be successful.

Kwon et al. [20] introduced the first passive fingerprinting attack against HSs and their clients. Assuming that the adversary controls an entry node, the authors detected the presence of a HS activity by observing circuit-level information. Compared to our approach, this attack works if and only if all circuits to a HS go through a single entry node (according to our measurements, this strongly depends on the Tor version used). Moreover, due to the use of guard nodes, it is a challenge for an attacker to get into this position to target individual clients. In contrast, our attack does not require control of one or more entry nodes or to see circuit-level information at all, but only to passively observe the link between the client and the guards. By using a dataset consisting of 1,000 HSs and 1,000 Alexa websites, Kwon et al. successfully distinguished a HS connection from a non-HS one with a TPR of more than 98% and a FPR of less than 0.1%. However, they had only moderate success in differentiating between HSs. In a closed-world scenario, they

[3] Also known as *Tor Browser*.
[4] https://www.torproject.org/projects/torbrowser.html.en

monitored only 50 HS pages with 50 instances each and were able to deanonymize 97% of the client-side connections and 94.7% of the server-side connections. In an open-world setting, the authors again considered 50 HS pages monitored by the attacker with 50 instances each and 950 uncensored HS pages with one instance each and achieved TPR of 88% and FPR of nearly 3%.

To the best of our knowledge, Hayes et al. [16] presented the most recent WFP attack, called k-FP. Their technique is based on random decision forests. In the scope of HSs, they achieved a TPR of 85% with a FPR of 0.02% when trying to recognize 30 HSs from a set of world wide web pages. Moreover, the authors showed that their classifier is more robust against existing countermeasures compared to the other state-of-the-art classifiers.

To sum up, both HS fingerprinting approaches presented so far either made strong assumptions regarding the considered attacker model or lacked a large-scale evaluation by applying unrealistically small datasets. In contrast, our work particularly addresses these issues.

4 DATASETS

This section introduces the datasets utilized in the rest of the paper. To explore the feasibility of HS fingerprinting in practice, we need to obtain a comprehensive and realistic sample of accessible onion addresses. However, the collection of HS addresses is a challenging task. In order to limit potential information leakages, Tor does not provide any way of retrieving a complete list of currently available HSs. To overcome this, we used the following two-step approach. First, we implemented a tool that automatically crawled all publicly known HS search engines once per day. We collected 13,243 unique addresses for the period of four months (April to July 2015).

Second, we exploited the role of the HSDirs in the HS protocol to gather onion addresses directly from the live Tor network. We launched several ORs in Tor that fulfilled certain requirements to obtain a *HSDir flag*, i.e., to become HSDirs. Since the HSDirs see the content of unencrypted descriptors served by them [6], we modified the Tor implementation to harvest HS descriptors published on our HSDirs. Moreover, we exploited the fact that, in addition to the current version of the consensus, two predecessor documents also retain their validity [8, 11]. This means that ORs and OPs may use different consensuses containing different sets of ORs acting as HSDirs. As a consequence, a HS may attempt to publish a descriptor to a HSDir which is no longer responsible for it, or clients may request a HS descriptor from a HSDir that is not responsible for this descriptor. Therefore, we further modified the Tor source code to allow for storing of descriptors not in the range of our HSDirs and keep IDs not present in the memory of our HSDirs but requested by clients. On a receipt of an unknown ID, our malicious HSDirs immediately initiated a descriptor fetch from other responsible HSDirs.

We also tried to ensure that our HSDirs were located far away from each other in the DHT. Thus, we increased the probability of capturing as many different descriptors as possible. Like [8], we injected our ORs with predefined fingerprints. In other words, we started randomly generating private/public key pairs until we found a public key whose hash was in a predefined area of the DHT. We utilized *PlanetLab* nodes in different subnetworks to run our relays.

Table 1: Statistics for collected web onion addresses.

Description	Number
Addresses with valid traces	10,476
Addresses without any traces	754
Addresses pointing to *HTTP error code 4xx*	1,172
Addresses pointing to *HTTP error code 5xx*	418
Invalid header received from the client	325
Total number of addresses	13,145

We further used different Tor versions with various combinations of configuration settings to avoid revealing the relationship among our nodes (otherwise these would be detected as malicious and banned from the network)[5]. On completing the experiment, we collected 41,936 unique onion addresses by capturing HS descriptors.

Finally, we parsed the collected descriptors and extracted the public keys of the HSs. To obtain the corresponding URLs, we computed the hash of each HS's public key and stored the first 80 bits of this hash concatenated with the virtual domain *.onion* in our database. After merging both sets of collected HS addresses and removing duplicates, we obtained 48,418 unique onion addresses in total. According to the Tor statistics[6], for the corresponding time period there were between 23,000 and 35,000 unique HS addresses. In addition, the work of Biryukov et al. [8] – the only one previous research showing a global picture of all existing HSs – reported on 39,824 unique onion addresses collected within approximately two days. The difference can be explained by the fact that our network view is not a snapshot, but a collection of HS addresses over a longer period of time. Hence, our dataset contains more HSs than were actually present at any one time during the study period.

The goal of our fingerprinting attack is to detect HSs that provide HTTP(S) access. Therefore, we limited our dataset to onion addresses that serve HTTP or HTTPS content. We obtained 13,145 unique HTTP(S) hidden services. In comparison, Biryukov et al. [7] reported on 6,579 HSs providing HTTP and HTTPS access in 2012. Even though the number of addresses collected from search engines in general is less than 30% of all HSs gathered, almost 60% from them have been filtered as HTTP and HTTPS hidden services. On the other hand, only 26.61% of the set of HSs collected from HS descriptors were web services. A possible reason may be the fact that HS operators who maintain a web server, e.g., web shops, are usually interested in advertising their services publicly.

After completing the collection and processing of the data, we gathered at least one network trace (i.e., sequence of packet sizes) representing a correct page load for 10,476 HS addresses. Table 1 shows the obtained results and reasons for failing to gather traces for many HTTP(S) services. 1,590 addresses returned to a HTTP status code indicating a load error. Additionally, we observed a group of addresses indicating the following client error code: *Invalid header received from the client.* This error message may be caused by some HSs requiring either a special authentication cookie to

[5]The Tor community is continuously improving algorithms for identifying sybils (entities controlling many identities) in the network.
[6]https://metrics.torproject.org/hidserv-dir-onions-seen.html

access them, or client support of a specific protocol running over HTTP. For 754 addresses, we could not collect any trace. This may be due to the extremely short life time of some HSs.

To construct representative and realistic set of non-HS webpages, we used the largest available dataset in the community [28]. This dataset reflects webpages that are actually accessed through Tor. In total, we utilized 211,148 webpages containing 65,409 unique web domains. Additionally, for some evaluations we created a sample of index pages representing the Alexa[7] Top 1,000 most popular sites.

Ethical considerations. The collection of onion addresses for our dataset does not strictly follow the guidelines for ethical Tor research, which were published after we finished our experiments. Nonetheless, we believe that it is important to know the degree of protection offered by the real Tor network. There can be no better evaluation than using real onion addresses. While running our directories, we made every effort to minimize any potential harm to the users and tried to examine all the risks that may exist, as well as getting advice from the Tor Research Safety Board[8].

Among other things, we removed all absolute timestamps from our traces, which makes identifying exactly when a service was active difficult. We will not publish fingerprints of non-publicly-discoverable (i.e., private) hidden services. We replaced identifiers of these HSs (i.e., onion addresses) with pseudonyms and removed their content after conducting the experiments. We performed evaluations to assess how representative the WFP results are when using only public onion services. Table 4 confirms our hypothesis that, in general, using only publicly available HSs slightly underestimates the complexity of the underlying classification problem. Still, we conclude that, at least at the time we performed our experiments, evaluation results obtained when using public hidden services were representative for the whole universe of hidden services.

5 EXPERIMENTAL SETUP

Without loss of generality, we assume that the adversary collects a sufficient number of network traces for each of the HSs of interest and some non-HSs, i.e., regular webpages, as *training data* for fingerprinting. The transferred packets are recorded with a traffic analyzing tool that provides information about the length and order of the packets that were sent and received. The collected dumps are then analyzed to create a profile of each HS page and public webpage, a *fingerprint* (FP). Wiretapping the victim's traffic, the adversary attempts to match the collected test data to a previously-known fingerprint. Due to indeterministic packet fragmentation, updates in the pages, etc., differences between patterns in training and test data are inevitable. Therefore, the attacker needs to apply statistical methods to compare the recorded information to the fingerprints and to probabilistically match it to the known fingerprints.

5.1 Data Collection

Our experimental setup was similar to that applied in [28]. We utilized TBB 3.6.2 to access HSs and regular webpages via Tor. Note that we used the same TBB version used to fetch the largest available set of regular webpages for WFP as we employ this set for our evaluation. We also provide comparison with the more recent TBB 6.0.4. TBB consists of a pre-configured Tor client and a stand-alone web browser based on Mozilla Firefox with privacy-friendly settings. It also provides different patches to combat WFP, including randomized pipelining. We ensured automatic filtering of web HSs and autonomous page loads by using the plug-ins Chickenfoot[9], iMacros, and Scriptish[10]. The functionality of Tor was controlled through Stem[11], a Python implementation of the Tor Control protocol. Together with the page sources recorded by iMacros, we were able to select HTTP(S) HSs only. Finally, we recorded traces for these pages using *tcpdump*. To train our classifier, we extracted and analyzed multiple instances for each page (a general requirement of every WFP method). We ensured that we never used the same circuit to download more than one instance of a single page.

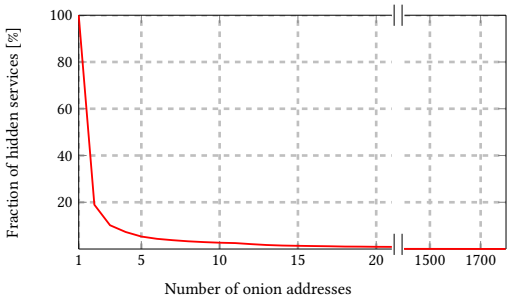


Figure 2: Complementary cumulative distribution function (CCDF) for HS content distribution.

5.2 Data Extraction and Processing

During data collection, we observed hundreds of HS addresses showing identical content. On the one hand, this phenomenon may be due to the existence of phishing addresses [25]. On the other hand, due to the lack of a load balancing for Tor hidden services at the time of executing these experiments, several HS operators might provide backup onion addresses that point to the same page, and, so increase the availability and capacity of their services[12]. Similar observations are also described by Biryukov et al. [8]. As in [20], we also found a large number of HS pages indicating that the corresponding hidden service had been seized[13].

In contrast to Kwon et al. [20], where the authors consider only a small HS sample and manually exclude repeated content, we used an automated approach to group all collected traces according to their content. To this end, we applied a cosine similarity metric to the collected HS page source dumps. This is a standard vector-based measure used to determine how similar two text documents are in terms of their subject matter [31]. We also introduced rules to define what kind of HS contents can be classified in a group. Beside pages that are obviously similar or even identical, we observed

[7] http://www.alexa.com/
[8] https://research.torproject.org/safetyboard.html
[9] http://groups.csail.mit.edu/uid/chickenfoot/
[10] http://imacros.net/ and http://scriptish.org/
[11] https://stem.torproject.org/
[12] During the study period, HS load balancing approaches [4, 5, 26] were still in progress.
[13] In November 2014, a large amount of HSs were seized by Europol and other government entities. For more details, we refer the reader to [2].

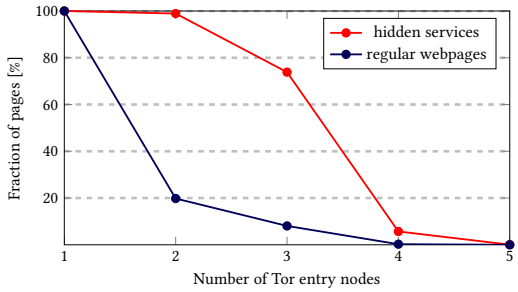


Figure 3: CCDF of the number of entry nodes used to load a HS vs. a regular webpage.

many different onion addresses pointing to subpages of a certain website. Although these subpages may contain different text and resources, e.g., images, they share the same page template and logo. Therefore, we considered them as one class. In addition, as in [8], we found hundreds of blank pages or pages containing only a few words that are inappropriate for classification. To avoid introducing noise into our attack, we grouped such HS pages together. In total, we created 3,145 groups representing different page content, on which we focus in the rest of this work. Figure 2 illustrates the distribution of onion addresses among the HS contents. Whereas almost 20% of the generated groups consist of at least two unique onion addresses, less than 0.05% include more than 1,000 addresses. Furthermore, we observed that most of the onion addresses that look alphabetically similar, also point to similar or even identical HS contents.

6 OUR FINGERPRINTING TECHNIQUE

In contrast to previous works on WFP [28, 32], where the authors assumed that a webpage is typically loaded over one circuit, i.e., the clients communicate with one entry node only, we noticed that our clients typically connect to several distinct entry nodes (depending on the TBB version) to fetch a HS page. Figure 3 shows the statistics obtained from our datasets. While more than 90% of the HSs were fetched over at least two different entry nodes, for less than 20% of the regular webpages the client used more than one entry node. To take advantage of the observed information leakage caused by the hidden service protocol, we define the following features. We first count the total number of distinct entry nodes N_G used for a single page load. According to our observations, this number varies from one to five. Next, we compute the sum of packet sizes transmitted between the client and each of the entry nodes, denoted by S_{G_i}, where $S_{G_1} \geq S_{G_2} \geq S_{G_3} \geq S_{G_4} \geq S_{G_5}$. If a given page is loaded over smaller number of entry nodes, then the remaining values of the features S_{G_i}, where $2 \leq i \leq 5$, are zero. In the remainder of this work, we refer to these new features as *hidden service features*.

We also include information regarding the complete page load in our fingerprints. To do this, we rely on the approach proposed in [28] and known as *CUMUL*. We extract information about the total number of incoming packets N_{in} and the total number of outgoing packets N_{out} transfered while fetching a page without taking into account the existence of different entry nodes. Afterwards, we calculate the total sum, S_{out}, of the data that the client has sent during the page load, and the sum of the received data, S_{in}. Our fingerprints also contain information that characterizes the load pattern. As explained in [28], given a trace of packet sizes $T = (p_1, ..., p_n)$, where $p_i > 0$ indicates an incoming packet and $p_i < 0$ an outgoing packet, the cumulative representation of this trace is computed as $C(T) = ((0,0),(a_1,c_1),...,(a_n,c_n))$ where $c_1 = p_1$, $a_1 = |p_1|$, and $c_i = c_{i-1} + p_i$, $a_i = a_{i-1} + |p_i|$ for $i = 2, ..., N$. As recommended in [28], we derive 100 features by sampling the piecewise linear interpolant of C at equidistant points. Thus, we are able to represent information about the chronological sequence of incoming and outgoing packets transfered during a page load.

To evaluate our attack, we apply LibSVM [10] with a radial basis function (RBF) kernel with parameters c and γ. LibSVM includes a tool to optimize these parameters using cross-validation. It applies a grid search, i.e., various combinations are tested and the one with the best cross-validation accuracy is selected. Before applying the SVM, we scale each feature linearly to the range $[-1, 1]$. This prevents features in greater numeric ranges dominating those in smaller numeric ranges [35]. For all following evaluations in this paper where we do not explicitly mention a different methodology, we always apply 10-fold cross-validation. This means, the data is split into 10 evenly large parts, i.e., *folds*. Then, the entire process of training and testing is repeated 10 times, using one of the 10 folds as test data and the remaining 9 folds as training data in turn.

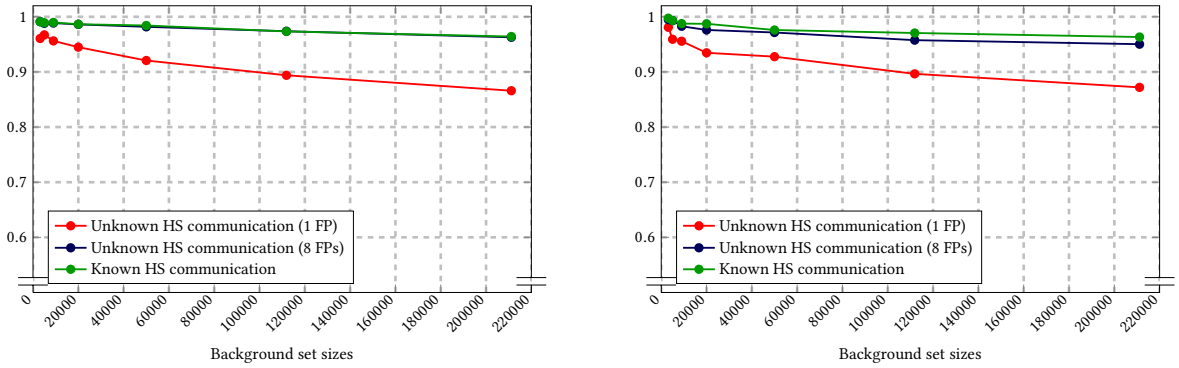
7 HIDDEN SERVICE RECOGNITION

Our fingerprinting approach consists of two classification phases. In *phase one*, we try to detect whether a client has connected to a HS or visited a regular webpage. Moreover, we aim to answer the question whether it is possible to identify a communication to a new, i.e., not seen before, HS by using FPs of already known HSs. Once a HS connection is detected, in *phase two* we try to recognize exactly which HS content has been visited. In the following, we describe and evaluate both classification phases in detail.

7.1 Phase 1: Detection of Communication to HS

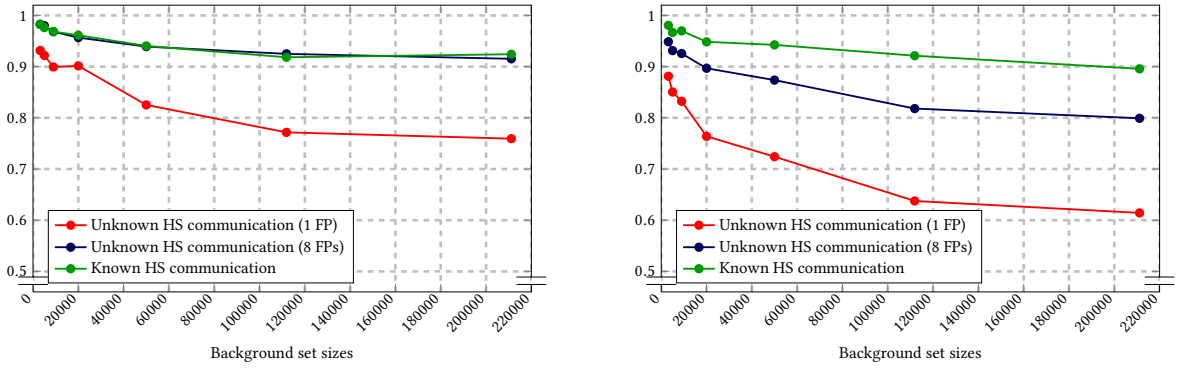
In phase one, we differentiate between two scenarios, depending on whether the adversary is attempting to detect a connection establishment to an already-known or a unknown (i.e., not seen before) HS. In both cases, we apply a binary classifier. The set of HS pages forms a single class and is denoted as the *foreground* class. The set of regular webpages is called the *background* class.

Detection of unknown HS communication. For evaluation, our foreground set consists of 3,145 unique HS contents with one fingerprint per content. Our background set contains increasing sizes b of regular webpages with one fingerprint per page and $b \in \{3145, 5000, 9000, 20000, 50000, 111884, 211148\}$, where 211,148 is the maximum available background set size. Here, the accuracy, i.e., the probability of a correct identification (either true positive or true negative), is not a representative indicator, since classes are not balanced. Therefore, we utilize metrics that are commonly applied in such situations, namely *recall* and *precision*. Recall represents the probability that a connection to a hidden service is successfully detected. On the other hand, precision indicates the probability that the classifier is actually correct when claiming to have detected a connection to a HS. From the adversary's perspective, precision



(a) Precision (b) Recall

Figure 4: Results for classification phase one when using TBB 3.6.2.



(a) Precision (b) Recall

Figure 5: Results for classification phase one when using TBB 6.0.4.

and recall should both be ideally equal or close to one. In this case, it is highly likely that all users connected to a hidden service are detected and the detection is practically always correct.

We computed precision and recall for all values of b. As shown in Figure 4(a) and 4(b) (red line), we observe a slow decrease of both metrics for increasing background set sizes. Still, for the maximum considered universe size, the precision is as high as 0.87 and recall is about 0.88. To overcome the observed degradation in the results, our first intuition was to increase the quantity of fingerprints per hidden service used for training by the attacker. Instead of one, we utilized eight fingerprints per HS while keeping the background set as described above. We also implemented an additional enclosing 10-fold cross-validation to ensure that all fingerprints belonging to a certain hidden service are used only for training or for testing, but not for both. Figure 4(a) and 4(b) (blue line) illustrate the obtained results. We can see that the precision remains high, i.e., close to 1.0, and almost constant. While the foreground pages are detectable with a precision almost 1.0 if the universe is small, this value decreases by only less than 0.03 for the largest background set. Similar results are observed with respect to recall. For $b = 3,145$, the adversary is able to detect every hidden service, while increasing b to 211,148 stills allows more than 95% of the foreground pages to be detected. In summary, we conclude that the adversary needs only a few more fingerprints (but still a moderate quantity) per hidden service to be successful.

Detection of known HS communication. So far, we have considered the scenario where the adversary tries to detect a connection establishment to a new HS that has not previously been seen by the classifier. Due to the limited hidden service universe size, it is also feasible to assume that the attacker knows all available HS pages visited by the clients (or, alternatively, is interested in detecting connection establishment only to some hidden services). Therefore, we also explore the severity of the attack in the scenario where the adversary tries to identify known HS communication. To evaluate this, we extended our foreground set by one fingerprint per hidden service (i.e., nine fingerprints per HS) and kept the same background set sizes. Contrary to our previous experiment, here we utilized those fingerprints for testing that are not included in the training set, but belong to the same HSs that the classifier has been trained on. While the size of the training set remains the same as the one used above, we further restricted the testing set size to that applied for unknown HS communication in order to keep both

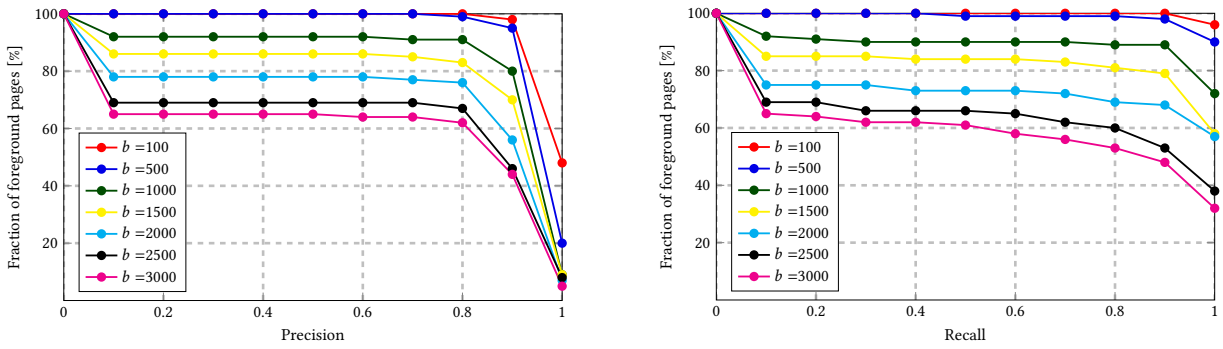


Figure 6: CCDF for an open-world scenario within the hidden service universe.

experiments comparable to each other. The background set was divided into training and testing sets as above. We then calculated precision and recall for all values of b. As shown in Figure 4(a) and 4(b) (green line), the foreground pages are detectable with a precision grater that 0.96 for all values of b. For the recall, we make similar observations. The adversary is able to detect almost every hidden service page, i.e., the recall equals 0.9975, if $b = 3,145$, and this metric remains greater than 0.96 for $b = 211,148$. Hence, as expected, the detection of connection establishment to known HSs is even easier classification task.

Discussion. First, we explore the significance of our hidden service features introduced in Section 6. To this end, we compare the classification results of our classifier with those obtained by CUMUL [28]. As shown in Table 2, the impact of the proposed HS features becomes significant for large background set sizes. For instance, the recall dramatically drops by 0.4 on average when excluding the HS features compared to the corresponding outcomes obtained by using fingerprints that contain these features. These significant improvements occur particularly when the attacker relies on prior knowledge in the area of WFP and uses grid parameters known to be successful for regular webpages [28]. Although the effect of the HS features is not always so distinctive if more fine-grained parameter search is performed, they are especially useful when the adversary is trying to save time and computational resources by using limited, standardly-suggested intervals of grid parameters.

Table 2: Impact of our novel fingerprinting features.

	our features		CUMUL features	
Background	Precision	Recall	Precision	Recall
3,145	0.9916	0.9935	0.9706	0.9918
5,000	0.9882	0.9907	0.9389	0.9883
9,000	0.9794	0.9868	0.9063	0.9819
20,000	0.9628	0.9825	0.9388	0.9651
50,000	0.9335	0.9733	0.9203	0.9313
111,884	0.8612	0.9595	0.8934	0.7460
211,148	0.8391	0.9357	0.8701	0.5282

Second, another interesting aspect is to investigate the impact of a newer TBB version on the performance of our attack. The use of several entry nodes for a communication with a HS has also attracted attention of the Tor developers, who have introduced improvements in newer TBB versions [30]. To evaluate this, we repeated the process of data collection for both the foreground and the background set by using the experimental setup described above, but applying TBB 6.0.4. While we kept the same background set sizes, we were able to gather 1,309 accessible hidden services. We repeated the experiments described above and obtained the results shown in Figure 5. Both precision and recall still remain high, especially for known hidden services. Nevertheless, in some scenarios (e.g., unknown HS communication with one fingerprint), correct classification clearly becomes more challenging. Also, the difference between unknown and known HS communication detection becomes more marked here. A reason for this might be the fact that most of clients already use only one entry node to connect to a HS [23]. Compared to the old TBB version where the length of the smallest packet sizes transfered for the both types of pages (HSs vs. regular webpages) is different, we no longer observe such variations. Last but not least, one has to take into account the fact that the foreground set used for this evaluation is significantly smaller, i.e., the training set size used by the attacker is restricted. However, a connection establishment to a HS can still be correctly identified with a precision more than 0.9 and a recall at least 0.8. In terms of training time, for the largest universe size of 211,148 pages, it takes one hour and 54 minutes on Intel Xeon 2.4GHz; for 111,884 about one hour and 4 minutes; for 50,000, less than 5 minutes and for 3,145 pages, 12 seconds. To sum up, though the new TBB version slightly increases the protection against an eavesdropper on the link between the client and its entry node, it is more vulnerable with regard to a malicious entry node as the latter sees all HS traffic and can reliably detect HS communication [20].

7.2 Phase 2: Detection of Particular HS Content

Once a communication to a HS is detected, in *phase two* we try to identify the particular HS content visited by the client. In accordance with phase one, here we also differentiate between two scenarios depending on whether the adversary's classifier has been trained on the whole hidden service universe. Regardless of the

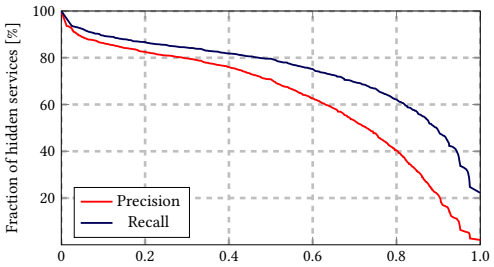


Figure 7: CCDF for a closed-world scenario within the hidden service universe.

scenario, we assume that the attacker has detected a HS communication by applying classification techniques from phase one. Here we do not apply HS specific features anymore (as unpredictability in the number of used circuits might disturb the recognition) but use the standard *CUMUL* classifier. First, we evaluate our attack in an open-world setting within the HS world only. In this case, the adversary wants to detect a particular set of HSs and is not interested in (and/or was not able to collect fingerprints for) the rest of the HS universe. Second, we also consider the scenario where the complete list of all hidden services is known to the attacker, who desires to know which one of the HSs is being accessed by the client. This use case is comparable to a closed-world setting in common WFP approaches. In the following, we present the classification outcomes obtained in both scenarios.

Open-world within hidden services. To evaluate this scenario, we divided our HS dataset described above into a foreground set consisting of 100 randomly selected HSs with 40 fingerprints each and a background set representing increasing background sizes $b \in \{100, 500, 1000, 1500, 2000, 2500, 3000\}$ with one fingerprint per page. Here, we again focus on recall and precision due to the existence of unbalanced datasets. Figure 6 shows the results. As illustrated in the figure, both recall and precision clearly decline for increasing background set sizes. For instance, less than 50% of the hidden services are identified with a recall at least 0.9 for the largest background size, though almost all had recall of one with the smallest HS universe size. For the precision, we observe an even more remarkable decrease. For precision of 0.9, the drop-off is from almost 100% to about 40%. To sum up, contrary to the adversary's success in phase one, the fingerprinting attack does not scale when applied in open-world setting within the hidden service universe only. Possibly, the attack would become feasible if the adversary is able to fingerprint the whole HS universe. The results of this scenario are shown and discussed in the next subsection.

Table 3: Accuracy of different classification problems.

Dataset	Accuracy [%]
1000 random HSs	69.86
1000 random webpages	69.51
1000 Alexa Top pages	76.21

Table 4: Comparison of state-of-the-art classifiers in closed-world setting.

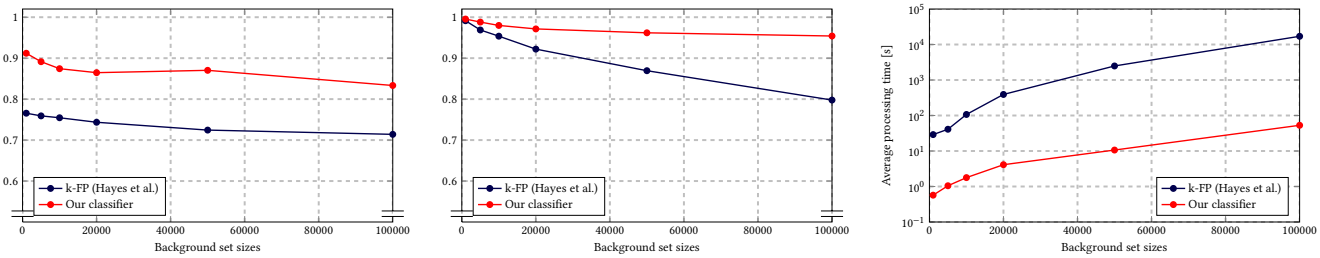
Classifier	Accuracy [%]			Avg. runtime per fold [s]
	random HSs		public HSs	
	1000	500	500	500
Our classifier (SVM)	69.86	80.65	81.7	62
Kwon (*k*-NN)	39.31	47.37	46.75	19
Wang (*k*-NN)	29.55	34.63	34.95	1916
Hayes (k-FP)	–	47.60	49.61	991

Closed-world within hidden services. For this case study, we consider a *multi-class* scenario where each HS content is treated as a different class and each class is evaluated against all others. For this evaluation, we were able to create a hidden service set consisting of 2,744 unique HS contents with 40 fingerprints per content. Besides precision and recall, we also computed the accuracy, since in contrast to the experiments above, we consider a balanced dataset. We used 10-fold cross-validation and achieved a recognition rate of 60.97%. In addition, Figure 7 shows the CCDF of precision and recall for this experiment. As we can see, there are fewer than 20% of all HSs that have both precision and recall at least as good as 0.9 and about 5% of all HSs that are not detectable at all. Trying to better understand the obtained classification outcomes, we found that 161 hidden services cannot be identified at all and 99 HSs are always correctly recognized. We further analyze the problem of detectability of Tor hidden services in Section 7.3.

Discussion. Contrary to our expectation based on existing results for closed-world settings in common WFP attack scenarios [9, 32], the recognition rate obtained in the latter experiment is significantly lower. This could be because our dataset is significantly larger than the datasets used to evaluate state-of-the-art classifiers (2,744 vs. 100 pages). Another hypothesis is that the detection of a particular HS content is a more difficult classification problem than the detection of regular webpages. Moreover, typically most evaluations are restricted to detection of the index pages of most popular websites. Therefore, we performed an additional closed-world experiment to investigate these hypotheses. To this end, we selected the first 1,000 index pages from the Alexa Top list, 1,000 random HSs, and 1,000 webpages representing an unbiased random sample of the world wide web [28]. Table 3 shows the obtained classification results. We see that HSs are identified with an accuracy similar to that of random webpages. Only Alexa Top index pages have a slightly better recognition rate. Possibly, this is due to the greater variability of the most popular websites. Therefore, this experiment confirms the fact that most popular websites are not necessarily a good sample of pages for a performance evaluation of classifiers with respect to WFP attacks [28].

7.3 Comparison with Related Work

In this section, we compare our classifier to other state-of-the-art WFP classifiers. First, we focus on phase two of our attack, namely closed-world within hidden services. To do this, we used the same



(a) Precision (b) Recall (c) Runtime (The y-axis scales logarithmically.)

Figure 8: Performance evaluation of our classifier vs. k-FP of Hayes et al. [16].

set of 1,000 random hidden services described above and generated the corresponding feature set not only for our classifier, but also for the known state-of-the-art classifiers k-NN of Wang et al. [32], k-NN of Kwon et al. [20], and k-FP of Hayes et al. [16]. Furthermore, all classifiers were (if necessary) adapted to perform 10-fold cross-validation to make results comparable to each other. Table 4 shows the obtained classification outcomes and the corresponding runtime per fold. As we can see, our classifier outperforms all other state-of-the-art approaches in terms of the classification accuracy. Please note that we were not able to successfully run the attack of Hayes et al. for 1,000 pages due to its huge memory requirements[14]. Therefore, we decreased out dataset to 500 hidden services and repeated the experiments. While other state-of-the-art approaches cannot reach even 50% accuracy on the reduced dataset, our classifier achieves more than 80% recognition rate. The worst performance is provided by Wang et al.'s k-NN classifier. The approach of Hayes et al. achieves similar results to the k-NN classifier of Kwon. In terms of runtime, our classifier is faster by a factor of 15 than the k-FP of Hayes et al., and a factor of 30 than the k-NN of Wang et al. The k-NN of Kwon et al. is by a factor of two faster than our method, however, at the price of a significantly lower classification accuracy. Please note that we could reproduce a high accuracy of Kwon's classifier on his dataset only by selecting the checkbox "use training set for testing" in the Weka toolbox[15]. Performing correct classification (i.e., using disjoint data for training and testing) yields a significantly lower classification accuracy.

As the results from the closed-world experiments are not necessarily transferable into open world [29], we further compare classifiers in the open-world scenario. Please note that the classifier of Hayes et al. [16] is the only WFP classifier for HSs that does not require circuit level information (i.e., does not rely on a malicious entry node). Hence, only comparison with this classifier is possible in the open world. We selected 30 HSs with 40 fingerprints each for the foreground set and increasing background set sizes b consisting of regular webpages with one fingerprint per page and $b \in \{1000, 5000, 10000, 20000, 50000, 100000\}$. Moreover, we assume that the adversary tries to detect already-known HSs. Figure 8 shows the obtained results[16]. As we can see, in the open-world setting our classifier also significantly outperforms state-of-the-art with respect to both precision and recall. In particular, while the recall obtained by using our classifier remains high and constant for increasing background sizes, the degradation of the metric is severe when applying the approach of Hayes et al. In terms of precision, both classifiers exhibit similar behavior for growing number of pages, with our classifier consistently outperforming that of Hayes et al. by 10%. The promising results of classification in this experiment should not be overestimated, as this is a simplified scenario, where in phase two we distinguish between 30 HSs only, for the purpose of comparing classifiers to each other. Also in terms of runtime, our classifier is by several orders of magnitude faster than the approach of Hayes et al. We conclude that our classifier provides the best classification performance compared to state-of-the-art in both closed and open-world settings.

[14] Around 250 GB of memory was not sufficient for the successful execution of the approach of Hayes et al.

[15] The use of this option is confirmed by the author in a private email exchange.

[16] Please note that incorrect predictions with respect to the foreground set are counted as false positives.

8 CONCLUSION

In this paper we have analyzed the susceptibility of Tor hidden services to the website fingerprinting attack. To this end, we proposed a novel two-phase fingerprinting approach for hidden services that does not rely on malicious Tor nodes. We show that our approach significantly outperforms state-of-the-art methods with respect to classification accuracy. Though recent works claim the feasibility of attack in the context of hidden services, we show that in general neither our attack nor other existing approaches scale when applied in realistic settings. Moreover, as the size of HS universe is constantly growing, it is likely that the feasibility of the attack in the future will decrease even further. We present a comprehensive comparison of the performance and limits of state-of-the-art website fingerprinting attacks with respect to Tor hidden services and reveal the fact that the attack still works well on some of them despite considering full universe size at the time of our evaluation.

9 ACKNOWLEDGMENTS

The authors would like to thank Markus Peuhkuri from Aalto University in Finnland for helping them with accessing the PlanetLab testbed. Parts of this work have been funded by the Luxembourg National Research Fund (FNR) within the CORE Junior Track project PETIT, the EU H2020 projects Privacy Flag and SAINT.

REFERENCES

[1] 2014. Better, fairer circuit OOM handling. https://trac.torproject.org/projects/tor/ticket/9093. (2014).

[2] 2014. Thoughts and Concerns about Operation Onymous. https://blog.torproject.org/blog/thoughts-and-concerns-about-operation-onymous. (2014).

[3] 2015. Getting the HSDir flag should require the Stable flag. https://github.com/DonnchaC/torspec/blob/master/proposals/243-hsdir-flag-need-stable.txt. (2015).

[4] 2015. Load Balancing/High Availability Hidden Services. http://archives.seul.org/or/talk/Mar-2015/msg00218.html. (2015).

[5] 2015. Possible Solutions for Increasing the Capacity of a Hidden Service. https://lists.torproject.org/pipermail/tor-talk/2015-March/037173.html. (2015).

[6] 2017. Tor Rendezvous Specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=rend-spec.txt. (2017).

[7] Alex Biryukov, Ivan Pustogarov, Fabrice Thill, and Ralf-Philipp Weinmann. 2014. Content and Popularity Analysis of Tor Hidden Services. In *34th International Conference on Distributed Computing Systems Workshops*. IEEE, Madrid, Spain, 188–193.

[8] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. 2013. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Symposium on Security and Privacy (S&P)*. IEEE, Berkeley, CA, USA, 80–94.

[9] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: website fingerprinting attacks and defenses. In *ACM conference on Computer and communications security (CCS)*. ACM, Raleigh, NC, USA, 605–616.

[10] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (April 2011). Issue 3. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] Roger Dingledine and Nick Mathewson. 2017. Tor directory protocol, Version 3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. (2017).

[12] Roger Dingledine and Nick Mathewson. 2017. Tor Protocol Specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=tor-spec.txt. (2017).

[13] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-generation Onion Router. In *13th conference on USENIX Security Symposium*. USENIX Association.

[14] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *Symposium on Security and Privacy (S&P)*. IEEE, San Francisco, CA, USA, 332–346.

[15] Rafael Gálvez, Marc Juarez, and Claudia Diaz. 2016. Profiling Tor Users with Unsupervised Learning Techniques. In *International Workshop on Inference and Privacy in a Hyperconnected World (INFER)*. DE GRUYTER, Darmstadt, Germany.

[16] Jamie Hayes and George Danezis. 2016. *k*-fingerprinting: a Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium*. USENIX Association, Austin, TX, 1187–1204.

[17] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *ACM workshop on Cloud computing security*. ACM, Chicago, IL, USA, 31–42.

[18] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Bjorn Scheuermann. 2014. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *21st Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA, USA.

[19] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *21st ACM Conference on Computer and Communications Security (CCS)*. ACM, Scottsdale, Arizona, USA, 263–274.

[20] 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 Association, Washington, D.C., 287–302.

[21] Nick Mathewson. 2015. Next-Generation Hidden Services in Tor. https://gitweb.torproject.org/torspec.git/tree/proposals/224-rend-spec-ng.txt. (2015).

[22] Srdjan Matic, Platon Kotzias, and Juan Caballero. 2015. Caronte: Detecting Location Leaks for Deanonymizing Tor Hidden Services. In *22nd ACM SIGSAC conference on Computer and communications security (CCS)*. ACM, Denver, Colorado, USA, 1455–1466.

[23] Asya Mitseva, Andriy Panchenko, Fabian Lanze, Martin Henze, Klaus Wehrle, and Thomas Engel. 2016. POSTER: Fingerprinting Tor Hidden Services. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, Vienna, Austria, 1766–1768.

[24] Steven Murdoch. 2006. Hot or not: Revealing hidden services by their clock skew. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, Alexandria, VA, USA, 27–36.

[25] Juha Nurmi. 2015. Warning: 255 fake and booby trapped onion sites. (2015). https://lists.torproject.org/pipermail/tor-talk/2015-July/038318.html

[26] Donncha O'Cearbhaill. 2017. OnionBalance. https://onionbalance.readthedocs.org/en/latest/. (2017).

[27] Lasse Øverlier and Paul Syverson. 2006. Locating Hidden Servers. In *Symposium on Security and Privacy (S&P)*. IEEE, Oakland, CA, USA, 99–114.

[28] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. 2016. Website Fingerprinting at Internet Scale. In *the 23rd Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA, USA.

[29] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *10th ACM Computer and Communications Security Workshop on Privacy in the Electronic Society*. ACM, Chicago, Illinois, USA, 103–114.

[30] Mike Perry. 2015. Notes and Action Items from Hidden Service Fingerprinting Session. https://lists.torproject.org/pipermail/tor-dev/2015-October/009632.html. (2015).

[31] Sandeep Tata and Jignesh M. Patel. 2007. Estimating the Selectivity of tf-idf Based Cosine Similarity Predicates. *Newsletter ACM SIGMOD Record* 36 (June 2007), 7–12. Issue 2.

[32] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *23rd USENIX conference on Security Symposium*. USENIX Association, 1–15.

[33] Tao Wang and Ian Goldberg. 2013. Improved website fingerprinting on Tor. In *12th ACM Computer and Communications Security Workshop on Privacy in the Electronic Society*. ACM, Berlin, Germany, 201–212.

[34] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. In *Privacy Enhancing Technologies (PETS)*. DE GRUYTER, Darmstadt, Germany, 21–36.

[35] Chih wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2010. A Practical Guide to Support Vector Classification. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf. (2010).

[36] Matthew Wright, Micah Adler, Brian Levine, and Clay Shields. 2003. Defending Anonymous Communication Against Passive Logging Attacks. In *Symposium on Security and Privacy (S&P)*. IEEE, Oakland, CA, USA, 28–43.

[37] Sebastian Zander and Steven Murdoch. 2008. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *17th conference on USENIX Security symposium*. USENIX Association, Berkeley, CA, USA, 211–225.