Fingerprinting Search Keywords over HTTPS at Scale

Junhua Yan, Hasan Faik Alan and Jasleen Kaur
University of North Carolina at Chapel Hill
{junhuayan, alan, jasleen}@cs.unc.edu

Abstract—The possibility of fingerprinting the search keywords issued by a user on popular web search engines is a significant threat to user privacy. This threat has received surprisingly little attention in the network traffic analysis literature. In this work, we consider the problem of keyword fingerprinting of HTTPS traffic—we study the impact of several factors, including client platform diversity, choice of search engine, feature sets as well as classification frameworks. We conduct both closed-world and open-world evaluations using nearly 4 million search queries collected over a period of three months. Our analysis reveals several insights into the threat of keyword fingerprinting in modern HTTPS traffic.

I. INTRODUCTION

Search engines are among the most popular websites worldwide [5]. In fact, a significant fraction of referrals to even other popular websites (for instance, 55.3% of referrals to the fifth most popular website, wikipedia.org), come from search engines [56]. Indeed, search query keywords are invaluable for improving user search results, preventing click fraud, reducing irrelevant advertising, and even detecting the spread of epidemics like influenza [28]. However, the same keywords—if leaked—may also reveal quite sensitive information about the user, including health issues, marital problems, abuse, and controversial political stance [8]. Hence, the possibility of fingerprinting search queries, from the network traffic they generate, is a significant privacy concern.

Over the past decade, the traffic analysis literature has devoted significant attention to the topic of website and webpage fingerprinting—in which network traffic headers are used to fingerprint which website, or webpage within a website, a user visited [31, 32, 39, 45, 51, 30, 11, 57, 81, 3]. However, this large body of work has mostly ignored the more challenging problem of fingerprinting search queries issued by a user—in which instead of simply discovering that a user is visiting the search engine google.com, traffic analysis is used to fingerprint the actual keywords (such as pregnancy or depression) that she may be searching for.

The notable recent exception is [48], which studies keyword fingerprinting in the context of the Tor anonymization network—and shows that the presence of one among 300 targeted keywords can be flagged with 80% recall and 91% precision, and the specific keyword can be identified with up to 48% accuracy. To the best of our knowledge, however, there is no prior work that helps us understand how fingerprintable are search keywords in modern HTTPS traffic, which is the dominant transfer scenario used by the vast majority of Internet users [27]. It is important to note that keyword fingerprinting of HTTPS traffic is quite different from that of Tor traffic, due to differences in structural information revealed in the traffic.

In this paper, we focus on HTTPS traffic and explore the limits of fingerprinting search keywords at Internet scale, by considering the impact of several influencing factors. We ask:

1) How accurately can one of a targeted set of search queries be identified, when a user relies on prominent web search engines?
2) Does the use of some search engines make a user more vulnerable to privacy violations/attacks?
3) How does diversity in client browser platforms impact the accuracy of fingerprinting search queries? Do some browsers make a user more vulnerable to privacy attacks?
4) What type of traffic features aid in keyword fingerprinting? What type of noise is an impediment?
5) Given the volatility of search results, how often must a classifier be re-trained for keyword fingerprinting?
6) Finally, how fingerprintable are targeted keywords in large scale “open-world” scenarios, in which a large mix of non-targeted background keywords may also be encountered?

To address these questions, we collect a large scale dataset over a period of three months, consisting of nearly 4 million targeted and non-targeted web search queries, using four popular web search engines (DuckDuckGo, Google, Bing and Yahoo), and four prominent browsers (Chrome, Firefox, Edge, Safari). Our analysis reveals:

- **Vulnerability With Different Search Engines**: The ability to fingerprint search queries differs significantly across users of the four major search engines studied. Search traffic generated by DuckDuckGo yields the highest fingerprinting accuracy (up to 96%), followed by Google, Yahoo, and Bing (up to 45%). We hypothesize that these differences are partially due to the different levels of tracking/advertisement/news background traffic generated by different search engines, which is inherently dynamic in nature and may not reflect the search keyword. The presence of such noise hinders the ability to achieve high classification accuracy.

- **Impact of Browser**: Browsers differ in how vulnerable their traffic is to keyword fingerprinting. Furthermore, a classifier that is trained on traffic samples from a single-browser, can not be used successfully to test samples from different browsers—the classification accuracy achieved may be significantly lower. Attackers will either need to incorporate a diverse set of browsers/devices during training, or will need to first determine the client browser before testing.

- **Impact of Feature Selection**: We evaluate prominent feature sets used previously for website fingerprinting—the two best performing feature sets consistently outperform in all evaluations. Furthermore, our analysis reveals the presence of noisy traffic components that generate significant amount of dynamic traffic that is not indicative of the
search query keywords. When such noise is eliminated by considering only the primary domains that are directly related to a given search engine, classification accuracy improves significantly for the Yahoo search engine, even in cross-browser attacks.

- **Effectiveness of Countermeasures:** The presence of countermeasures that obfuscate packet sizes and packet ordering, can significantly lower the accuracy of keyword fingerprinting classifiers in HTTPS traffic.

- **Presence of Large Number of Non-targeted Keywords:** In an open-world scenario, a user may search for any keyword in the wild, including those not seen during training. Based on the goal of attackers, we consider three different classification tasks: binary classification, multi-level classification, and multi-class classification. For each task, we consider up to 250k non-targeted search queries for training and testing. We find even though the classification performance gradually degrades as the number of non-monitored testing samples increases, it is still quite promising. For instance, an attacker may be able to fingerprint among 1,440 targeted keywords, in the presence of around 190k non-targeted search samples, with nearly 90% average precision for DuckDuckGo/Chrome.

The rest of paper is organized as follows: Section II formulates the problem, Section III summarizes our data collection methodology. Section IV discusses the candidate feature sets and the classification algorithm used. Section V presents "closed-world" evaluations, and Section VI presents "open-world" evaluations. Section VIII presents our conclusions.

II. PROBLEM FORMULATION

**Keyword Fingerprinting Threat Model** In this paper, we consider the scenario in which a user uses a web search engine and attackers eavesdrops on the HTTPS traffic traversing the access link of the user and attempt to predict the search query keywords sent by the user. For every sensitive search keyword the attackers wants to identify, they train a machine-learning classifier to determine, given the network traffic generated by the web search, whether it is associated with the keyword. To collect labeled training data, attackers repeatedly conduct web searches for both the targeted keywords of interest to them, as well as other popular keywords (as negative examples), and capture the generated network traffic.

**State of the Art** Prior to 2012, traffic analysis had successfully demonstrated that the deterministic packet sizes generated by the auto-complete feature of the Google search engine can be used to successfully fingerprint the keywords being typed by a user. For every sensitive search keyword the attackers wants to identify, they train a machine-learning classifier to determine, given the network traffic generated by the web search, whether it is associated with the keyword. To collect labeled training data, attackers repeatedly conduct web searches for both the targeted keywords of interest to them, as well as other popular keywords (as negative examples), and capture the generated network traffic.

A broader discussion of related work is included in Appendix A. In fact, users in some countries do not even have access to advanced anonymization networks such as Tor. It is important to note that Oh et al. attempt to show that search query traffic can be distinguished from webpage traffic in Tor—however, the webpage dataset used in is provided by Panchenko et al. while the Tor search query dataset is collected by Oh et al. —the use of two different datasets (collected at different times and using different platforms) raises the question of whether the high accuracy is due to the classifier’s ability to differentiate between webpage/search query traces or differences in the the underlying data collection setups that may make the traffic differ. For instance, we run a simple experiment with two Tor datasets collected by Wang et al. —the classifier we train is able to determine which dataset a testing sample comes from with 100% accuracy!
Given the widespread usage of HTTPS, an attacker is likely to encounter a large volume of searches in practice. It is important for a study to use at-scale data sets that are representative of this volume (and are much larger than have been used in prior work).

**Our Approach** While it is heartening to witness the tremendous growth in traffic analysis studies over the past decade, it is also important for this line of research to avoid the pitfalls that may lead to significant exaggeration of fingerprinting accuracy in the real world [53], [34], [51], [3]. In this paper, we consider the impact of several factors, many of which have been shown in prior work to significantly influence the performance of traffic analysis techniques:

- **Client Platform Diversity:** Recent studies have shown that different client browsers may result in significant differences in the network traffic generated—so much so, that traffic classification accuracy is significantly impacted when different browsers are represented in the training and testing data sets [3]. It is, therefore, important to address the questions: how does the use of different browsers impact the accuracy of fingerprinting search keywords? And do some browsers make a user more vulnerable to keyword fingerprinting attacks?

- **Feature Set Design:** The choice of input features can significantly influence the performance of a learning-based classifier [51], [56]—distinguishing features that are stable even in the presence of noise are likely to yield better classification results in the real world. Hence, we ask the question: which features yield unique signatures across classification labels, but are also robust to the presence of real-world noise?

- **Choice of Search Engine:** For a given keyword search, the network traffic pattern generated may differ across different web search engines. Since HTTPS traffic lets us identify which search engine is being used, it can also help us answer: does the use of some search engines make a user more vulnerable to keyword fingerprinting attacks?

- **Large Scale Data Collection:** There are two well-accepted observations with respect to the use of machine learning for traffic classification: (i) training data sets that are large as well as representative of the noise and diversity likely to be encountered during testing, will lead to better classifier performance in the real world; and (ii) it is important to test a classifier in large-scale “open-world” settings that include a large number of unseen data points—classification results can often be exaggerations if based on small and narrow-focused testing data. Hence, we ask in this paper: how does keyword fingerprinting perform when larger number of unseen keywords are encountered?

In what follows, we describe our methodology to collect and analyze data in order to address the above questions.

**III. DataSet**

**A. Selecting Search Query Keywords**

**Targeted Keywords** For the purpose of this study, we consider the context in which the attacker is interested in tracking whether a given user is interested in blacklisted/sensitive content. For creating the corresponding list of targeted search keywords, we use 447 keywords blacklisted from Google Instant [43] (as reported by 2600.com [10]), as well as 1,000 keywords that are considered sensitive in China [16].

**Non-targeted Background Keywords** When collecting traffic from a given population of users, an attacker is likely to encounter web search traffic for non-targeted keywords as well—the additional task for the attacker, then, is to sift through the web searches to identify those that include targeted keywords. In order to understand the problem of keyword fingerprinting in the real world, therefore, it is important to study the “open-world” setting, in which regular background search traffic is also incorporated.

We assume that background web search traffic is modeled well by including popular search queries. For this, we consider the most popular queries reported by a commercial keyword tool [35], for two region/language settings: Global/English and Hong Kong/Chinese Simplified (China)—given an input sequence of characters, this tool returns a list of suggested search keywords that either start with that sequence or contain that sequence as a substring. The tool also provides the average number of search per month for each suggested search keyword. To harvest the most popular keywords for each region, we adopt a 3-step approach:

1) First, we extract the complete list of all suggested search queries that start with each of the 26 alphabet characters a-z.

2) From the above list, then, we select search queries that log more than 50,000 average search volume per month—this step yields 2,647 and 1,814 keywords for the Global/English and Hong Kong/Chinese Simplified settings, respectively.

3) For each search query obtained in the second step above (e.g., “apple”), we then input it back to the keyword tool [35]—which returns a list of suggested search terms that contain that search query in them (e.g., “apple sauce”). In this step, we select all query phrases that log more than 3,000 average search volume per month.

Our final list of background keywords consist of all search terms harvested in steps two and three above—in total, we harvest around 235,767 background keywords, with a diverse span of topics. The average search volume for these keywords during 2018 was around $2.89 \times 10^{11}$, which is about 1,052 hours of Google search volume according to [69].

**Semantic closeness of Non-targeted Keywords** Users may differ in how they search for a given topic—for instance, a user may prefer “tv” while another user may consider “television” instead. To reflect the variations, non-targeted keyword list is supposed to contain queries that are semantically close to each other. Thus we measure the semantic closeness of non-targeted keywords with a commonly-used technique in natural language processing. Specifically, we first embed the keywords and then calculate the cosine similarity between their embedding. By manually investigating groups of keywords grouped based on cosine similarity, we found 0.8 to be a reasonable threshold to

---

6Google Instant has been deprecated since 2017 [61] and not used in the paper.

7Examples of non-targeted search keywords can be found in Appendix B.
balance between the variation and semantic similarity. Fig. 1 shows the distribution of the number of non-targeted keywords with cosine similarity equal to or larger than 0.8. The results indicate that around 53% of keywords are unique while 17% of them have one semantic similar neighbor.

Fig. 1: Distribution of non-targeted keywords with cosine similarity equal or larger than 0.8.

B. Data Collection Methodology

Web Search Engines We focus on four web search engines for our analysis—Google (www.google.com), Yahoo (www.yahoo.com), Bing (www.bing.com) and DuckDuckGo (duckduckgo.com). The first three of these are the most popular web search engines worldwide [12], while DuckDuckGo is the default search engine used in the Tor browser—a prominent feature of DuckDuckGo is that it claims to enhance user privacy by not tracking users as well as by blocking hidden trackers from Google [78].

Client Browsers In order to study the impact of diverse client browser platforms on keyword fingerprinting, we consider four different browsers—Firefox, Chrome, Edge and Safari. A majority of the data is collected using Firefox and Chrome, which are run on 12 virtual machines and 3 desktop machines with Ubuntu 17.10. Edge instances are run on 8 Windows 10 virtual machines on Microsoft Azure [7], while Safari instances are running on a Mac Mini device with macOS High Sierra. We use Docker containers on the Linux virtual machines in order to scale data collection with Firefox and Chrome.

Our closed-world dataset is collected using all 4 browsers, while our larger open-world dataset is collected using Chrome.

Automated Traffic Capture We use Selenium for web browser automation [63]. We define a search session as the process of issuing a search using a given combination of keyword, browser, and web search engine. Within each search session, after successfully instantiating the browser, we capture the network traffic using either tcpdump on macOS/Ubuntu or WinDump on Windows 10.

Furthermore, a search session is divided into two consecutive phases. The first phase opens the homepage of the web search engine (e.g., duckduckgo.com), while the second phase simulates the typing of the search query by the user and pressing ENTER after the last character is typed. In order to capture all the information conveyed by auto-complete during typing, we impose a 1 second delay before typing the next character. After loading the search page, we wait for an additional 5 seconds before stopping tcpdump/WinDump and closing the browser. Consequently, along with the actual search results, traffic related to loading the search engine homepage as well as auto-complete is also captured.

Simulating User Searching Modes There are two prominent web search modes used by Internet users—homepage searching, in which a user first visits the homepage of the web search engine and then types in a query in the search box, and address bar searching, in which a user directly searches from the browser address bar (with the configured default search engine) without visiting the homepage of the search engine. In order to study keyword fingerprinting under both modes, we consider network traffic as follows. For homepage searching, we consider all network traffic collected for a given search session. For address bar searching, we filter out all packets captured before typing the first character in the search box.

C. Dataset Summary

TABLE I summarizes the total number of targeted and non-targeted search sessions captured with different web search engines and client browsers. This dataset is collected over a three-month period, in two phases. The first phase lasts seven weeks and focuses on studying the influence of several factors in a closed-world scenario. For a given browser platform, we conduct search sessions with the four search engines in a round-robin manner, iterating multiple times over the different targeted keywords. Within the first phase, we successfully searched 1,440 targeted keywords each at least 54 times using Chrome and Firefox, 40 times using Edge, and 4 times using Safari.

The second phase lasted six weeks and focused on open-world data collection, using only the Chrome browser and with the DuckDuckGo and Google search engines—this phase conducted search sessions with both targeted and non-targeted keywords. We randomly iterate over the non-targeted keywords, and randomly insert targeted keyword queries in between. Overall, each non-targeted keyword is queried at least 5 times each with DuckDuckGo and Google, and each targeted keyword is queried at least 106 times.

In total, our dataset contains nearly 4 million search query visits with four search engines and browsers, including both targeted and non-targeted search queries.

---

8An example list of keywords with a cosine similarity more than 0.8 are “maps maroon 5 lyrics”, “maps by maroon 5 lyrics”, “maps maroon 5” and “maps lyrics”.
9https://www.tcpdump.org
10https://www.winpcap.org/windump/
11In practice, the interval is likely to be smaller than 1s. In Appendix C, we study how typing speeds affect the traffic trace and the vulnerability of being fingerprinted—we do not find any obvious correlation.
12For TCP connections that are established before typing the first character in the search box, we filter out only the packets observed before typing—since these may be persistent connections that are used to later carry search traffic.
13Due to access to several Linux platforms, data collection with Firefox and Chrome on Ubuntu 17.10 took significantly less time.
14Due to limited availability of Apple devices, search sessions conducted using Safari are used for testing only (not for training the machine learning classifiers).
15The dataset can be shared to assist future studies per request.
The performance of learning-based classification may be significantly influenced by the choice of input features. Indeed, in the traffic analysis literature, several different types of features have been derived from the headers of TCP/IP network traffic for the purpose of website and webpage fingerprinting, especially in the context of Tor traffic [25], [24], [24], [30], [51], [81], [36]. In this paper, we consider and evaluate the following five feature sets—none of these have been considered in the context of keyword fingerprinting over HTTPS traffic, but most have been shown to yield good classification performance in their respective application domains:

- **k-FP (2016)**: K-fingerprinting is devised for fingerprinting web page visits over Tor using features based on packet number and ordering—including number of packets, ratio of incoming/outgoing packets, packet ordering, number of packets per second, concentration of outgoing packets, packet inter-arrival time, and the overall transmission time [30].

- **SvmResp/EtResp (2017)**: Targeted search keywords are identified in [48] by combining informative features for website fingerprinting, as well as additional novel features in Tor. The features considered include number of total/incoming/outgoing packets, number of incoming bursts [49] and the cumulative size of TLS records. We use the code provided by Oh et al. [49] for feature extraction.

- **Wfin (2018)**: Recent work on website fingerprinting has focused on extracting many more features than considered previously [81], [36]. Yan and Kaur [81] extract and analyze the importance of more than 36,000 fine-grained features (grouped into more than 100 feature categories) in different communication scenarios. Feature selection is conducted based on the importance of each feature category for classification—the selected feature categories are then used for website fingerprinting classification. Yan and Kaur [81] show that Wfin achieves comparable or better website fingerprinting accuracy in several communication scenarios, including HTTPS. We use the code provided by the authors of [81] to select features for each combination of search engine, browser, browsing mode.

- **Wfin++**: We also derive from Wfin a feature set in two steps. First, inspired by features introduced in [48] for fingerprinting search keywords in Tor traffic, in addition to the 109 feature categories identified in [81], we introduce features such as the sequence of reversed cumulative size of packets/bursts, total number of packets, maximum packet size, and the average packet size in the largest incoming burst.

- **Packet Size Count (PSC)**: Packet size count, for a given traffic direction, is the frequency of each packet size encountered—it has been shown to be one of the most informative features for identifying individual web pages under both HTTPS and encrypted tunnels [37], [31], [45], [81], [3]. Thus we also consider packet size count as the baseline feature set.

### Machine Learning Algorithms

Prior studies have used different types of machine learning algorithms for traffic analysis, including multinomial Bayes [31], Support Vector Machine (SVM) [59], and decision tree-based ensemble methods, including Random Forests [11] and Extra-Trees [24]. Among these, SVM and decision tree-based ensemble methods are able to achieve consistently high performance [30], [51], [81]—SvmResp uses SVM with 10-fold cross validation for parameters tuning, k-FP directly/indirectly uses Random Forests [59] while Wfin uses Extra-Trees. We observe in our evaluations that: (i) for large scale datasets, SVM has a prohibitively high training overhead during cross-validation; and (ii) with comparable performance, Extra-Trees is more computationally efficient than Random Forests, due to the randomization of cut-point choices and the use of whole learning samples for growing the trees [24]—hence, we choose Extra-Trees with the above five features sets for both closed-world and open-world evaluations [9].

### V. Closed World Evaluations

We start with the “closed-world” assumption (the attacker knows in advance that the search keyword to be fingerprinted belongs to a small known targeted set of keywords)—we use this constrained scenario to investigate the impact of five important factors on the performance of fingerprinting search keywords, including (1) the choice of search engines; (2) the traffic features used for classification; (3) the impact of client browser platforms; (4) the choice of searching modes; and (5) the impact of the increasing time gap between training and testing. In this section, we primarily use the closed-world dataset summarized in TABLE I.

#### A. Vulnerability of Different Search Engines

One of the key differences between HTTPS and Tor traffic is that the attacker may be able to learn which search engine is used by the user in the former case, based on the IP address of

---

**TABLE I: Total Number of Search Sessions Captured**

| Feature Set         | Google (k-FP) | Bing (k-FP) | Yahoo (k-FP) | DuckDuckGo (k-FP) | Time Span          |
|---------------------|---------------|-------------|--------------|-------------------|--------------------|
| Edge                | 81,020        | 80,832      | 84,662       | 88,246            | Jan-Feb, 2019      |
| Safari              | 5,784         | 5,784       | 5,784        | 5,784             | Jan-Feb, 2019      |
| Chrome-cw           | 98,296        | 98,316      | 96,610       | 96,327            | Feb, 2019          |
| Firefox             | 94,757        | 94,848      | 93,462       | 94,919            | Feb, 2019          |
| Chrome-ow           | 1,223,727     | 1,223,727   | 1,223,727    | 1,223,727         | Feb-Mar, 2019      |
| Chrome-targeted     | 150,011       | 150,011     | 150,011      | 150,011           | Feb-Mar, 2019      |

---

16A burst is defined as a sequence of back-to-back packets sent in one direction between two packets sent from the opposite direction [52].

17Examples of features considered for HTTPS traffic are packet size count, burst size count, total No. of TCP connections, transmitted bytes w.r.t. port 443, and hostname count (details in Section 11 of [81]).

18In closed-world settings, k-FP directly uses the classification output of the Random Forest; for open-world settings, the output of the forest is used as features to fit into k-NN.

19Performance of SVM and Extra-Trees is compared in Appendix E.
the server as well as the server name in the SNI extension of TLS \cite{47, 26}. That is, the attacker can be assumed to know that a user is visiting google.com—but the attacker wants to fingerprint the keywords that the user is using for their search query. Thus the first question we ask is: among Google, Bing, Yahoo and DuckDuckGo, which web search engine makes a user more vulnerable/robust to keyword fingerprinting?

To answer this question, we use our searches for the 1,440 targeted keywords on Yahoo, Bing, Google and DuckDuckGo, using Firefox, Chrome, and Edge. For Chrome and Firefox, the targeted dataset is split into three disjoint subsets for training, validation and testing, with the ratio of 4:1:1—for every 6 consecutive visits of a search query, the first 4 samples are used for training, the 5th for validation, and the 6th for testing. \footnote{The reason for adopting this sub-sampling scheme is to minimize the time gap between training and testing samples (the impact of time will be explicitly studied in Section V.E). In practice, it is indeed possible for the attacker to capture training and testing traces close in time, if it is an offline testing model.} Since each keyword is queried at least 54 times in the closed-world datasets for Chrome and Firefox, we consider 36 training samples, 9 validation samples, and 9 testing samples during evaluation of each targeted keyword. The validation dataset is used for tuning parameters in Extra-Trees and for deriving the feature sets for Wfin and Wfin++. To obtain test accuracy, we include both training and validation samples for building the machine learning model, and use test samples for the fingerprinting evaluation (45:9 samples). \footnote{Results with the Edge browser are included in Appendix F.}

![Network traffic generated using Chrome](image)

Fig. 2: Network traffic generated using Chrome (caption below each sub-figure indicates the time of search)

| Keyword | Bing | Yahoo | Google | DuckDuckGo |
|---------|------|-------|--------|------------|
| Wfin++  | 44.86 ± 0.07 | 64.63 ± 0.24 | 60.52 ± 0.07 | 96.15 ± 0.01 |
| PSC     | 44.98 ± 0.12 | 57.66 ± 0.09 | 57.72 ± 0.05 | 96.33 ± 0.04 |
| Wfin    | 41.63 ± 0.02 | 52.57 ± 0.05 | 58.25 ± 0.14 | 94.06 ± 0.04 |
| EtResp  | 15.54 ± 0.06 | 0.57 ± 0.03 | 27.39 ± 0.07 | 42.38 ± 0.08 |
| k-FP    | 8.95 ± 0.08 | 1.48 ± 0.06 | 20.70 ± 0.14 | 33.45 ± 0.08 |

TABLE II: Classification Accuracy Achieved (%)

| Keyword | Bing | Yahoo | Google | DuckDuckGo |
|---------|------|-------|--------|------------|
| Wfin++  | 44.73 ± 0.13 | 58.14 ± 0.23 | 75.56 ± 0.07 | 91.95 ± 0.10 |
| PSC     | 44.83 ± 0.08 | 49.90 ± 0.11 | 76.75 ± 0.02 | 92.23 ± 0.02 |
| Wfin    | 41.22 ± 0.03 | 43.06 ± 0.09 | 73.55 ± 0.21 | 89.70 ± 0.03 |
| EtResp  | 8.07 ± 0.10 | 0.12 ± 0.01 | 24.07 ± 0.02 | 28.08 ± 0.06 |
| k-FP    | 5.34 ± 0.06 | 0.81 ± 0.03 | 9.69 ± 0.06 | 15.82 ± 0.17 |

TABLE III summarizes the test accuracy (mean and standard deviation) for fingerprinting among 1,440 targeted keywords, obtained with different feature sets and search engines. We find that:

- **Feature Sets**: As reported for website fingerprinting \cite{81}, feature sets that work well for Tor may not work well for HTTPS—the performance of EtResp and k-FP are not comparable to Wfin, Wfin++, or PSC. For instance, the accuracy obtained for DuckDuckGo/Chrome is around 33-42% with EtResp and k-FP, but ranges around 94-96% for the others. \footnote{In Extra-Trees classifier, the split criteria is Gini Index with bootstrap enabled and the number of trees is set to 700. Details about parameter tuning are included in Appendix F.}
Second, the $Wfin++$ feature set outperforms $Wfin$ in all cases—the performance gains are as large as 15% for Yahoo/Firefox. In the rest of this paper, we only consider the derived $Wfin++$. Finally, $PSC$ and $Wfin++$ outperform each other in different cases. However, $Wfin++$ performs at least comparably well in all cases, while $PSC$ may sometimes yield a significantly lower accuracy (Yahoo).

- **Search Engines:** Among the four web search engines, traffic generated by targeted keywords with DuckDuckGo are most vulnerable to fingerprinting—the highest accuracy achieved (boxed in Table III) with DuckDuckGo is above 96%, while it ranges from 44% (Bing) to 76% (Google) for the rest. Below, we further investigate differences across search engines.

- **Client Browsers:** The vulnerability of fingerprinting searches on a given search engine also depends on the client browser used—this is most notable for Google, in which targeted keywords can be identified with 76% accuracy when using Firefox, but only 60% accuracy when using Chrome. We further investigate the impact of browsers in Section V-C.

- **HTTPS vs. Tor:** For Google, the highest keyword fingerprinting accuracy achieved with 1,400 keywords in the closed-world setting of Table II ranges from 60% to 76% with different client browsers. The closed-world fingerprinting accuracy achieved for Google with 100 keywords in the Tor browser in 48 was around 64%. However, it is important to resist the urge to compare these numbers—the dataset used in 48 is very different from the one used here (most notably, in the time period of data collection, the number of targeted keywords, and the number of training/testing samples). In Appendix E we provide a brief comparison of HTTPS and Tor by carefully controlling factors during data collection. In Appendix F we study the impact of smaller number of keywords on the classification accuracy (we find that the accuracy of fingerprinting Google/Chrome keywords decreases from 60% to 80% as the number of targeted keywords is reduced to 100).

To understand what makes the search engines differently vulnerable to keyword fingerprinting, we examine the network traffic. Fig. 2 illustrates the traffic generated using Chrome with four search engines, while searching for a targeted keyword at four different times within five hours on Feb 24-25, 2019. The $x$-axis represents the packet index within each TCP transfer, and the $y$-axis represents the index of the TCP connection in the network trace. Different colors are used to describe the packet length together with the direction (“-“ sign indicates packets sent from the client to the server). We observe that the network traffic differs significantly, when searching the exact same query at approximately the same time across different search engines (any given column in Fig. 2).

In Table III, we summarize statistics with respect to the number of packets generated and number of TCP connections initiated by the four search engines, for all search sessions collected using Chrome. We observe that:

**TABLE III: Network Traffic Statistics (Chrome)**

|            | Yahoo | Bing | Google | DuckDuckGo |
|------------|-------|------|--------|------------|
| Median no. of packets | 2989 | 1476 | 1195   | 723        |
| Std Dev of no. of packets | 343.3 | 192.3 | 128.1  | 66.1       |
| Median no. of TCP conn. | 27   | 4    | 8      | 6          |
| Std Dev of no. of TCP conn. | 6.4  | 0.7  | 3.2    | 1.1        |
| Median homepage load time (s) | 3.05 | 0.52 | 1.00   | 1.05       |

**TABLE IV: Second-level Server Names (Chrome)**

|            | 2019-02-23-23-58-37 | 2019-02-24-01-02-04 |
|------------|---------------------|---------------------|
| yimg.com   | 9                   | yimg.com            |
| yahoo.com  | 5                   | yimg.com            |
| atwola.com | 1                   | atwola.com          |
| scorecardresearch.com | 1           | scorecardresearch.com |
| google.com | 1                   | google.com          |
| bing.com   | 1                   | bing.com            |
| atwola.com | 1                   | atwola.com          |
| google.com | 1                   | google.com          |
| yimg.com   | 9                   | yimg.com            |
| yahoo.com  | 6                   | yahoo.com           |
| atwola.com | 5                   | atwola.com          |
| scorecardresearch.com | 1           | scorecardresearch.com |
| adtechus.com | 1              | adtechus.com         |
| krxd.net   | 1                   | krxd.net            |
| nexac.com  | 1                   | nexac.com           |
| addthis.com | 1              | addthis.com          |
| google.com | 1                   | google.com          |
| bing.com   | 1                   | bing.com            |

(a) DuckDuckGo

|            | 2019-02-24-02-03-53 | 2019-02-24-03-01-05 |
|------------|---------------------|---------------------|
| yimg.com   | 9                   | yimg.com            |
| yahoo.com  | 6                   | yahoo.com           |
| atwola.com | 5                   | atwola.com          |
| yahoodns.net | 5              | yahoodns.net        |
| scorecardresearch.com | 1           | scorecardresearch.com |
| adtechus.com | 1              | adtechus.com         |
| krxd.net   | 1                   | krxd.net            |
| nexac.com  | 1                   | nexac.com           |
| addthis.com | 1              | addthis.com          |
| google.com | 1                   | google.com          |
| bing.com   | 1                   | bing.com            |

(b) Yahoo

- **Amount of Traffic Generated:** The median number of packets (as well as standard deviation) is largest for Yahoo, followed by Bing, Google, and DuckDuckGo (Table III). A smaller number of packets may result from simpler content in the responses returned for a search query. For example, most of the responses returned by DuckDuckGo are text-based, in contrast to a significant presence of image-based responses from the other three search engines.

- **Third-party Connections:** Yahoo initiates a much larger number of TCP connections (and with largest deviation). To understand who a user communicates with when using different search engines, we study the second-level server names extracted from the SNI extension field of TLS in the network traces of Fig. 2. Table IV lists the number of TCP connections initiated to different servers when using DuckDuckGo and Yahoo. While DuckDuckGo searches contact only duckduckgo.com for serving query results, Yahoo searches involve communicating with several third-party servers. Among those servers, some are owned by other web search engines, such as google.com and bing.com24 while some are other marketing/advertisement service providers, such as adtechus.com25

---

23Only the first 100 packets in each TCP connection are depicted for ease of visualization. TCP connections are sorted according to the timestamp of the first packet observed in each TCP connection. The time gap between the start of different TCP connection is not shown in the figure.

24By further inspecting the traffic, we observe contents retrieved under google.com are mostly from ade.googlesyndication.com, www.googletagservices.com and googleleads.g.doubleclick.net. On the other hand, Yahoo retrieved some images on the result page from bing.com14.
connection from each of

We simulate two corresponding searching modes—

B. Homepage vs. Addressbar Searching

Users may enter keywords either in the search box on
the homepage of a search engine, or directly in the
address bar of their browser (using the default search engine).
We simulate two corresponding searching modes—

or not we include the network traffic generated when loading
the homepage of the search engine. TABLE V summarizes the
accuracy obtained in addressbar searching mode with 1,440
targeted keywords using the same setup as in Section V-A.
Compared with the results obtained in homepage searching
mode (TABLE II), we observe:

| Chrome  | Bing | Yahoo | Google | DuckDuckGo |
|---------|------|-------|--------|------------|
| Wfin++  | 45.18 ± 0.07 | 86.62 ± 0.08 | 94.05 ± 0.06 | 96.51 ± 0.02 |
| PSC     | 45.69 ± 0.14 | 84.22 ± 0.02 | 99.05 ± 0.05 | 96.61 ± 0.02 |
| EtResp  | 21.54 ± 0.04 | 0.80 ± 0.02  | 28.61 ± 0.05 | 55.59 ± 0.06 |
| k-FP    | 8.97 ± 0.12  | 2.83 ± 0.03  | 26.24 ± 0.13 | 41.12 ± 0.09 |

For Bing, Google, DuckDuckGo, the fingerprinting ac-


For Yahoo, the performance is increased significantly even
with Wfin++—from 60% to 86%. This may be attributed to
the elimination of dynamic traffic generated when loading
the homepage of Yahoo, which contains news, weather,
and ads. Indeed, when we repeat the analysis of Fig. 2 for
the addressbar searching mode with Yahoo, we observe a
significant decrease in the number of packets as well as
number of TCP connections (details in Appendix B).

In what follows, we present evaluations with only Wfin++ and
PSC, and homepage searching results are included only in
the appendices.

C. Impact of Disregarding Client Platforms?

Alan and Kaur [2] show that differences in client browser
platforms represented in training and test data can signifi-
cantly impair the accuracy of fingerprinting webpages. Our
evaluations presented so far have trained and tested with data
collected using the same client browser. Next, we investigate
the performance of keyword fingerprinting in cross-browser

25Most Bing/Chrome traffic traces consists of 4 TCP connections—one
connection from each of bing.com, bingparachute.com, live.com and microsoft-
online.com.
attacks (that train with data from one type of browser and test with another), using $W_{fn++}$ and $PSC$.

To minimize the variations introduced by different time of data collection on classification performance, we study the cross-browser impact with Firefox vs. Chrome samples (both collected in Feb 2019), and with Edge vs. Safari samples (both collected in Jan-Feb 2019). For comparison, we also evaluate the classification accuracy when training with samples from both Firefox and Chrome browsers.

**TABLE VI: Classification Accuracy: Cross-browser Attacks in Addressbar Searching**

| DDG      | Firefox | Chrome | Firefox | Chrome |
|----------|---------|--------|---------|--------|
|          | $W_{fn++}$ | $PSC$ | $W_{fn++}$ | $PSC$ |
| Firefox  | 92.87 ± 0.03 | 92.44 ± 0.04 | 0.69 ± 0.06 | 0.91 ± 0.03 |
| Chrome   | 0.80 ± 0.06 | 1.54 ± 0.07 | 96.51 ± 0.02 | 96.61 ± 0.02 |
| Fire/Chr | 92.26 ± 0.03 | 92.18 ± 0.03 | 96.49 ± 0.02 | 96.52 ± 0.04 |
| Google   | Firefox  | Chrome | Firefox | Chrome |
|          | $W_{fn++}$ | $PSC$ | $W_{fn++}$ | $PSC$ |
| Firefox  | 78.44 ± 0.13 | 79.27 ± 0.08 | 1.41 ± 0.08 | 1.61 ± 0.04 |
| Chrome   | 1.65 ± 0.08 | 0.77 ± 0.01 | 64.05 ± 0.06 | 59.05 ± 0.05 |
| Fire/Chr | 77.57 ± 0.22 | 77.62 ± 0.05 | 64.94 ± 0.09 | 58.48 ± 0.17 |
| Yahoo    | Firefox  | Chrome | Firefox | Chrome |
|          | $W_{fn++}$ | $PSC$ | $W_{fn++}$ | $PSC$ |
| Firefox  | 85.95 ± 0.09 | 83.63 ± 0.13 | 53.19 ± 0.52 | 48.68 ± 0.59 |
| Chrome   | 22.09 ± 0.38 | 25.39 ± 0.63 | 86.66 ± 0.11 | 84.17 ± 0.08 |
| Fire/Chr | 85.66 ± 0.17 | 87.71 ± 0.19 | 85.65 ± 0.07 | 83.88 ± 0.27 |

The classification results with Google, DuckDuckGo, and Yahoo are displayed in TABLE VI. In both addressbar searching and homepage searching modes, we find that when there is a mismatch between the training browser and testing browser, the classification accuracy is significantly lower. Furthermore, the performance achieved by including samples from both Firefox and Chrome during training is comparable to (or even slightly better than) what is achieved when trained with a single matching browser. For instance, when testing with DuckDuckGo/Chrome, the accuracy obtained by $PSC$ and $W_{fn++}$ is around 96% in both cases. This observation stresses the importance of incorporating diverse browser platforms during training, in order to achieve good fingerprinting accuracy in practice.

**Browser Specific Communication** We next examine browser-specific communication—TABLE VII lists the second-level server names observed uniquely in Firefox (not observed in Chrome), with DuckDuckGo and Yahoo. We find that Firefox generates a significant fraction of connections to its own servers—mozilla.com, mozilla.net and mozilla.org (this was true across all four search engines—DuckDuckGo: 40.06%, Google: 33.53%, Bing: 39.93%, and Yahoo: 9.73%). Furthermore, almost all unique connections came from mozilla servers when Firefox used DuckDuckGo, Google, and Bing; however, many other unique server names were observed with Yahoo, such as alephd.com and smartadserver.com—this indicates the impact of different browsers may also differ across search engines. This also suggests that the appearance of unique server names in a network trace may be used by an attacker to infer the browser (and then select the corresponding trained model for fingerprinting). An attacker may also train a multi-class classifier using features based on server names to help identify the browser.

**TABLE VII: Firefox-specific Server Names**

| Rank | Server Name          | Popularity (Continued) |
|------|----------------------|------------------------|
| 1    | smartadserver.com    | 0.12%                  |
| 2    | yieldmo.com          | 0.14%                  |
| 3    | smartadserver.com    | 0.12%                  |
| 4    | gstatic.com          | 0.14%                  |
| 5    | solocpm.com          | 0.12%                  |
| 6    | webmd.com            | 0.11%                  |
| 7    | jivox.com            | 0.11%                  |
| 8    | akamaihd.net         | 0.09%                  |

**D. Eliminate Noise From “Other” Domains?**

Based on our observations so far, there are at least two sources of noisy background traffic—tracking/advertisement connections, and browser specific connections—that (even in addressbar searching mode) may hinder achieving high fingerprinting accuracy. To eliminate such noise, we next study keyword fingerprinting when only TCP connections that serve the actual search results are considered. Specifically, based on manual analysis, we include connections from google.com and gstatic.com for Google, duckduckgo.com for DuckDuckGo, bing.com for Bing, and yahoo.com for Yahoo.

**TABLE VIII** summarizes the classification accuracy achieved with Yahoo. We observe a significant increase in classification accuracy, from around 86% (TABLE VI) to 95% (increase was less significant for the other search engines—see Appendix X). More importantly, the accuracy is high for Yahoo even in cross-browser attacks—around 91% when training (testing) with Chrome (Firefox), and 82% when training (testing) with Firefox (Chrome). Thus, we find that when significant amount of noise from other domains is eliminated, keyword fingerprinting may become vulnerable to even cross-browser attacks.

**E. How Often to Re-train Classifier?**

The auto-suggestion list and search results returned by a search engine depend on the current social trends and can be quite dynamic over time—thus, an attacker may need to...
TABLE VIII: Classification Accuracy (only yahoo.com connections considered with Yahoo, addressbar searching mode)

|        | Firefox       | Chrome       |
|--------|---------------|--------------|
|        | Wfin++        | PSC          | Wfin++        | PSC          |
| Firefox| 94.60 ± 0.03  | 94.37 ± 0.05 | 91.39 ± 0.09  | 91.65 ± 0.12 |
| Chrome | 76.91 ± 0.15  | 82.09 ± 0.18 | 96.27 ± 0.03  | 96.16 ± 0.06 |

frequently re-train the machine learning model to keep up its performance. In order to understand how frequently the model may need to be updated in a closed-world scenario, we next study how the classification accuracy changes as the time gap between training and test samples is increased. For this, we focus on Chrome and use the first 36 samples (out of 54) for each of the 1,440 targeted keywords for training, next 9 samples (37-45) for validation, and test with five datasets (test-3/4/8/10/14) collected in different time periods. Each test dataset is composed of 9 samples for each of the 1,440 targeted keywords—test-3 contains the last 9 samples (out of 54) from the Chrome-cw dataset (TABLE I); test-4/8/10/14 are composed of samples from the Chrome-targeted dataset collected during the second phase.

TABLE IX: Classification Accuracy: Impact of Time (Chrome, addressbar searching mode)

| Gap (Hours) | (20-30) | (38-86) | (164-188) | (219-254) | (331-352) |
|------------|---------|---------|-----------|-----------|-----------|
| DDDG       |         |         |           |           |           |
| test-3     | 92.20   | 90.86   | 81.84     | 72.90     | 69.15     |
| test-4     | 92.85   | 91.09   | 80.34     | 69.65     | 69.15     |
| PSC        | 95.14   | 94.34   | 92.33     | 99.99     | 92.16     |
| Wfin++     | 47.74   | 34.67   | 9.75      | 10.14     | 14.72     |

TABLE IX summarizes the classification accuracy (addressbar searching mode) against the time gap between the last training sample and the testing samples (specified as a range in number of hours). As expected, the accuracy decreases as the time gap between training and testing samples increases. However, the rate of decline differs across search engines. For instance, the accuracy achieved with Google decreases from 55% to 22%, while with DuckDuckGo, it decreases from 92% to 72%.

Limiting the Time Effect Our closed-world evaluations revealed that search traffic signatures can change with time, requiring frequent classifier re-training to achieve high performance. In order to reduce the impact of time in our open-world evaluations: (i) each keyword should be searched at approximately the same time with different search engines; and (ii) targeted and non-targeted keywords should be visited in similar time spans. Given our scale of 200K+ keywords, visiting each non-targeted keyword once, with even one search engine, takes more than 3 days. Thus, for open-world evaluations, we focus only on the DuckDuckGo and Google search engines, with Chrome browser in addressbar searching mode—DuckDuckGo is the most vulnerable to keyword fingerprinting (Section V), while Google is the most popular search engine worldwide.

For classifiers, we consider two best-performing feature sets in the closed-world scenario, Wfin++ and PSC, with Extra-Trees. We use the open-world dataset (TABLE I)—45 samples are used for training and 9 for testing for each targeted keyword; and 1 sample is considered for either training or testing for each selected non-targeted search keyword (from a list of 235,767 keywords). The maximum number of testing samples in our experiments is around 203k (including both targeted and non-targeted ones).

A. Binary Classification

We first consider the attacker task of determining whether or not a given user query is searching for a targeted keyword—this can be formulated as a binary classification problem with two labels: targeted or non-targeted. To evaluate classification performance, we consider:

- Precision-Recall-Curve (PRCbc): \( PRC_{bc} \) is based on two evaluation measures—precision and recall\(^{29}\). Precision, a measure of exactness, is the fraction of samples that are correctly classified, when being classified as a targeted one (\( \frac{TP}{TP+FP} \)). Recall, a measure of completeness, is the fraction of targeted samples that are classified correctly\(^{31}\).

\( k\)-FP and EtResp do not perform as well even in closed-world evaluations.

We do not expect them to perform well in open-world scenarios, which are more challenging due to large volumes of unseen non-targeted keywords.

\(^{29}\)Similar trend is observed in homepage searching (Appx. TABLE IX).

\(^{31}\)PRC is generated by sweeping over classifier score thresholds, and calculating the precision and recall at each threshold.
(b) AP: # Non-targeted Testing Samples

Fig. 4: Binary Classification: Impact of Number of Non-targeted Training/Testing Samples (DuckDuckGo and Google)

(c) Precision recall curve (DDG, Wfin++)

Fig. 5: Multi-level Classification: Impact of Number of Non-targeted Training Samples (DuckDuckGo and Google)

Fig. 6: Multi-level Classification: False negative rate with varying number of non-targeted testing samples (DuckDuckGo and Google).

Impact of Number of Non-targeted Training/Testing Samples: We plot the APbc achieved by Wfin++ and PSC for different number of non-targeted training and testing samples in Fig. 4a and 4b, respectively, and the PRCbc (Wfin++, DuckDuckGo) for different number of non-targeted testing samples in Fig. 4c. We observe:

- Search queries using DuckDuckGo, with both higher precision and higher recall, are consistently more vulnerable to be fingerprinted than those using Google (similar to observations in closed-world evaluations).
- Attackers can benefit from training on larger number of non-targeted training samples (Fig. 4a)—APbc increases from 90-94% to around 94-98% as the number of non-targeted training samples increases from 10k to 190k (the number of non-targeted testing samples is 50k).
- As the number of non-targeted test samples increases (testing dataset becomes increasingly imbalanced), APbc keeps decreasing (Fig. 4c). For example, APbc drops from 100% to around 82% as the number of non-targeted testing samples increases from 0 to around 190k with Google (the number of non-targeted training samples used is 50k). However, the attacker is still able to distinguish 12,960 targeted samples from 190k non-targeted ones with 80% APbc for Google and 90% APbc for DuckDuckGo.
- Both precision and recall decrease when the number of non-targeted test samples increases (Fig. 4c). For instance, as number of non-targeted test samples increases from 70k to 190k, a precision of 88% can be maintained only at the expense of recall decreasing from 90% to 70%.

We conclude that, in practice, it seems feasible for attackers to determine whether a query is targeted or not, even in the presence of large scale unseen samples.

---

**Notes:**

- True positives—TP (true negatives—TN) are number of the targeted (non-targeted) samples that are correctly classified. False positives—FP (false negatives—FN) are the number of non-targeted (targeted) samples that are incorrectly classified.

\[ P_{bc} = \frac{TP}{\sum R_n - R_{n-1}} \] * R_n, where P_n and R_n are precision and recall at the n'th decision threshold [79].

- A similar trend is observed with Google using Wfin++ and PSC—results are omitted due to space constraints.
B. Multi-level Classification

After determining that a given search query belongs to the targeted list (binary classification), an attacker can apply an additional classifier trained with only targeted samples (under the closed-world assumption), to further identify the specific targeted keyword. We evaluate this 2-level classification by feeding the samples that are classified as targeted by the binary classifier, to a second model that is trained with targeted samples (with 1,440 labels corresponding to the exact keyword). The label predicted by each classifier is the one with the highest mean probability estimate across the trees in the Extra-Trees classifier. We measure:

- **False Positive Rate (FPR\(_{ml}\))**: The fraction of non-targeted samples that are incorrectly classified, \(\frac{FP}{FP+TN}\) (binary classification).
- **False Negative Rate (FNR\(_{ml}\))**: The fraction of targeted samples that are incorrectly classified, \(\frac{FN}{TP+FN}\) (binary classification).
- **Accuracy\(_{ml}\)**: The fraction of true positives (identified by the first classifier) that are correctly classified at the second level.

The first two metrics reflect the performance of the binary classifier in terms of sifting targeted samples from non-targeted ones, while the last metric reflects the ability of the second classifier in terms of sifting targeted samples from non-targeted ones, while the last metric reflects the ability of the second classifier to differentiate among targeted search queries. Fig. 5 shows the performance achieved in multi-level classification scenario with different number of non-targeted training samples using Wfin++ and PSC.\(^{35}\) We observe:

- Training the binary classifier on a larger number of non-targeted training samples reduces the FPR\(_{ml}\), but increases the FNR\(_{ml}\) (Fig. 5a and 5b)—with 190k non-targeted training samples, the FNR\(_{ml}\) of Wfin++ is around 4% (out of 12,960) with DuckDuckGo and 11% with Google, which shows classifiers’ ability to identify targeted testing samples even in the presence of large number of non-targeted testing samples. Furthermore, Fig. 6 shows how FNR\(_{ml}\) changes as the number of non-targeted testing samples increases—we find that FNR\(_{ml}\) is not significantly affected and remains to around 1% for DuckDuckGo and 4% for Google with 50k non-targeted training samples.\(^{36}\)

- The Accuracy\(_{ml}\) of the second level classifier is not significantly impacted by the number of non-targeted training samples—it is around 50% for Google and 90% for DuckDuckGo (consistent with closed-world evaluations). However, as a larger number of non-targeted training samples are used, the binary classifier labels more targeted samples as false negatives—hence, the overall performance of determining which targeted keyword a user is searching for, decreases.

- In all performance measures, Wfin++ consistently outperforms PSC.

C. Multi-class Classification

The attacker may choose to simply train a single classifier to determine exactly which targeted query is being searched for. In this case, samples associated with different targeted keywords have different labels (as in closed-world scenario), while all non-targeted samples share the same label (e.g., -1).

To evaluate performance, we define below an additional metric—the *False Monitored Rate (FMR\(_{mc}\))*. Each targeted sample is categorized as either a true positive: \(TP_{mc}\) (the keyword is correctly identified), a false negative: \(FN_{mc}\) (the sample is labeled as non-targeted), or a false monitored: \(FM_{mc}\) (the sample is labeled as an incorrect targeted keyword). \(FMR_{mc}\) is computed as the ratio of false monitored to the total number of targeted samples: \(FMR_{mc} = \frac{FM_{mc}}{FM_{mc} + TP_{mc} + FN_{mc}}\).

Fig. 7 plots the \(FMR_{mc}\), FPR\(_{mc}\) (fraction of non-targeted samples that are incorrectly classified), and TPR\(_{mc}\) (= \(\frac{TP_{mc}}{TP_{mc} + FN_{mc}}\), fraction of targeted samples that are fingerprinted correctly), for different number of non-targeted training samples (50k non-targeted testing samples). We find:

- FPR\(_{mc}\) is low—even with just 10k non-targeted training samples, only 4% (of 50k) non-targeted testing samples are misclassified with DuckDuckGo.
- \(FMR_{mc}\) decreases when a larger number of non-targeted training samples are used—the trend is more obvious with Google, for which the \(FMR_{mc}\) reduces to nearly 0 when 190k non-monitored training samples are used.
- Finally, TPR\(_{mc}\) also decreases along with \(FMR_{mc}\), for larger number of non-targeted training samples—this implies that \(FN_{mc}\) must be increasing, and a larger number of targeted samples get classified as non-targeted (consistent with Fig. 5d).

**Conclusion** Based on our multi-level and multi-class evaluations, we reach the following conclusions:

---

\(^{35}\)Results with different number of non-targeted testing samples are in Appendix C.

\(^{36}\)Note that the attacker has the ability to achieve a different balance between the FPR\(_{ml}\) and FNR\(_{ml}\) of the binary classifier, by tuning classifier thresholds (similar to Fig. 4c).

---
Determining which targeted keywords is searched for by a user, in the presence of large numbers of non-targeted samples, is challenging but not impossible. For example, with DuckDuckGo, an attacker is able to correctly classify more than 80% targeted keywords when training with 10k non-targeted samples.

Incorporating more non-targeted samples during training can help decrease the false positive rate, but also decreases the true positive rate and increases the false negative rate. To achieve a different balance between these metrics, attackers can either adjust the decision threshold in binary classification or consider “top-k” predictions in multi-class classification [31, 65, 50].

Consistent with closed-world evaluations, DuckDuckGo samples are more vulnerable than Google samples to be fingerprinted in the presence of non-targeted samples. A better performance is achievable when the attacker trains two classifiers using multi-level classification instead of one classifier with the “one-step” multi-class classification scheme, especially with Google samples. For example, when training with 190k non-targeted samples, an attacker is able to identify 90% of targeted keywords and correctly classify them with 50% accuracy in multi-level classification. However, the attacker is only able to correctly classify around 20% of targeted samples with multi-class classification.

VII. DISCUSSION

A. Regular Web page Visit vs. Search Query

One challenge remaining is whether it is possible for attackers to distinguish search query from regular web page visit. Although Oh et al. [48] attempted to study the problem in Tor, the results are not conclusive due to the use of two datasets collected using different platforms at different times. Due to the access to per-connection headers and IP addresses, the problem in HTTPS are much easier. We crawled the web pages in https://www.yahoo.com on Jan. 17th, 2020 and obtained in total 73,605 urls served from yahoo—among them, 24,016 are search query result pages, including 16,175 web search, 3,211 total 73,605 urls served from yahoo.com

B. Effect of Cache

In this section, we investigated the cache policy of each of the four search engines from its traffic results. Below is a summary of cache policies adopted by search engines:

1) When Google/Bing return the search results, the index.html is never cached (cache control = private, max-age = 0). Furthermore, all images in Google and most in Bing that are displayed on the returned page are encoded in the index.html, instead of sending GET request to fetch the object from another server. The only few contents being cached are general Google elements such as the googlemic figure, activityindictor gif and favicon.ico.

2) For DuckDuckGo, the max-age for index.html is set to 1 second (smaller than time to type a new keyword). DuckDuckGo results generally contain less images and are served from external-content.duckduckgo.com.

3) For Yahoo, index.html is not cached while almost all images are served from yimg.com and cache is applied. But attackers can get rid of the caching effect by considering only TCP connections to yahoo.com—which has been shown (Section V-D) to improve the accuracy of Yahoo.

VIII. LIMITATION AND CONCLUDING REMARKS

In this paper, we study the vulnerability of keyword fingerprinting in HTTPS traffic with a large-scale dataset of nearly 4 million search samples, and study the impact of several factors. Our evaluation methodology differs from [48] in the several ways: (i) we consider keyword fingerprinting in HTTPS traffic (versus Tor traffic); (ii) we study the impact of several factors on keyword fingerprinting—including client browser diversity, web search engine, time, as well as noisy traffic features; and (iii) our evaluation dataset is larger by several orders of magnitude. Our key findings include:

- Search engines differ in the vulnerability of their users to keyword fingerprinting—Bing is the least vulnerable (up to 45%), followed by Google (up to 80%); while Yahoo and DuckDuckGo (up to 96%) users are the most vulnerable.
- An attacker can achieve high fingerprinting accuracy: (i) by ignoring traffic going to secondary domains, other than the search engine contacted by a user; (ii) by training on data collected using diverse client browser platforms; and (iii) by re-training their classifiers on data collected every 2-3 days.
- Search query fingerprinting is indeed a potential privacy concern even in open-world scenarios in which large scale unseen samples may be encountered—attackers are able to identify specific targeted search queries with 80% recall and 85% precision with 10k non-targeted samples and 50k non-targeted testing samples from DuckDuckGo/Chrome (Section VI).

We find our observations alarming about the possibility of fingerprinting search keywords being used by current Internet users. We believe this topic should receive significant and immediate attention from the research community. Some open issues that need to be addressed are:

- Countermeasures: Given the dominance of HTTPS in world-wide Internet traffic, our results urge for the study of efficient countermeasures against keyword fingerprinting.
In Appendix [2] we study the efficiency of two countermeasures designed for website fingerprinting with HTTPS traffic: PadToMTU and HTTPPOS [40]. Our preliminary results suggest that these two countermeasures can help decrease the classification accuracy, albeit, after incurring significant bandwidth overhead. Thus, we consider an extensive study of the design and evaluation of efficient countermeasures (e.g., [22], [40], [71]) as important future work.

- **Mobile Web Browsers and Voice-based Searches:** An increasing number of users are relying on mobile devices, rather than PCs, to access the Internet. Meanwhile, voice services such as Amazon Alexa, Google Home, and Apple Siri has gained a lot of popularity in recent years. As part of future work, we plan to study the vulnerability of search query fingerprinting for users with mobile devices.

- **Traffic Segmentation:** This paper assumes that a user’s traffic has already been cleanly segmented on a per-search session basis—such an assumption has been used in nearly all prior work on website/webpage/keyword fingerprinting, both in HTTPS and Tor [37], [41], [22], [75], [74], [51], [60], [62], [81], [74], [41]. In practice, traffic from a user is likely to contain overlapping connections from multiple tabs, multiple browsers, as well as traffic from several web requests multiplexed onto pipelined HTTPS connections—segmenting such a mix into traffic corresponding to individual web requests remains an important open problem in this field.

- **Deep Learning:** The application of deep learning for website fingerprinting in Tor has recently been explored in [1], [57], [66]—this body of work has shown that deep learning is able to achieve comparable or even better performance compared to traditional machine learning classification frameworks, without the need for manual feature engineering. It is important for deep learning to also be explored for fingerprinting HTTPS traffic—as future work, we plan to do so for keyword fingerprinting.

**REFERENCES**

[1] K. Abe and S. Goto, “Fingerprinting attack on tor anonymity using deep learning,” Proceedings of the Asia-Pacific Advanced Network, vol. 42, pp. 15–20, 2016.

[2] (2014) What is adserver.adtechus.com? [Online]. Available: [https://www.quora.com/What-is-adserver-adtechus-com](https://www.quora.com/What-is-adserver-adtechus-com)

[3] H. F. Alan and J. Kaur, “Client diversity factor in https webpage fingerprinting,” in Proceedings of the 9th ACM Conference on Data and Application Security and Privacy (CODASPY). ACM, 2019.

[4] ———, “Can android applications be identified using only tcp/ip headers of their launch time traffic?” in Proceedings of the 9th ACM conference on security & privacy in wireless and mobile networks. ACM, 2016, pp. 61–66.

[5] I. Alexa Internet. (2019, Jan.) The top 500 sites on the web. [Online]. Available: [https://www.alexa.com/topsites](https://www.alexa.com/topsites)

[6] (2019) Atwola, how to remove? [Online]. Available: [https://www.2-spyware.com/remove-atwola.html](https://www.2-spyware.com/remove-atwola.html)

[7] (2019) Microsoft azure. [Online]. Available: [https://azure.microsoft.com/en-us/](https://azure.microsoft.com/en-us/)

[8] M. Barbaro and T. Zeller Jr. (2006, Aug.) A face is exposed for aol searcher no. 4417749. [Online]. Available: [https://www.nytimes.com/2006/08/09/technology/09aol.html](https://www.nytimes.com/2006/08/09/technology/09aol.html)

[9] D. Barradas, N. Santos, and L. Rodrigues, “Effective detection of multimedia protocol tunneling using machine learning,” in 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 169–185.

[10] (2010, Sep.) Google blacklist - words that google instant doesn’t like. [Online]. Available: [https://www.2600.com/googleblacklist/](https://www.2600.com/googleblacklist/)

[11] L. Breiman, “Random forests,” Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[12] R. Bucci. (2011) Are yahoo! and bing now the same? [Online]. Available: [https://getstat.com/blog/are-yahoo-bing-now-the-same/](https://getstat.com/blog/are-yahoo-bing-now-the-same/)

[13] X. Cai, N. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, “A systematic approach to developing and evaluating website fingerprinting defenses,” in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 227–238.

[14] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, “‘Touching from a distance: Website fingerprinting attacks and defenses,” in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 605–616.

[15] S. Chen, R. Wang, X. Wang, and K. Zhang, “Side-channel leaks in web applications: A reality today, a challenge tomorrow,” in 2010 IEEE Symposium on Security and Privacy. IEEE, 2010, pp. 191–206.

[16] (2014, Dec.) chinese-keywords: sensitive chinese keywords from various sources. [Online]. Available: [https://github.com/jasonqng/chinese-keywords](https://github.com/jasonqng/chinese-keywords)

[17] A. Chris. (2019, Jan.) Top 10 search engines in the world. [Online]. Available: [https://www.reliablesnetsoft.net/top-10-search-engines-in-the-world/](https://www.reliablesnetsoft.net/top-10-search-engines-in-the-world/)

[18] S. E. Coull and K. P. Dyer, “Traffic analysis of encrypted messaging services: Apple imessage and beyond,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 5, pp. 5–11, 2014.

[19] S. Das, “Filters, wrappers and a boosting-based hybrid for feature selection,” in Icml, vol. 1, 2001, pp. 74–81.

[20] (2018, Jan.) Does page load time really affect bounce rate? [Online]. Available: [https://royal.pingdom.com/page-load-time-really-affect-bounce-rate/](https://royal.pingdom.com/page-load-time-really-affect-bounce-rate/)

[21] D. L. Donoho et al., “High-dimensional data analysis: The curses and blessings of dimensionality,” AMS Math Challenges Lecture, vol. 1, p. 32, 2000.

[22] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail,” in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 332–346.

[23] K. Finley. (2017, Jan.) Half the web is now encrypted. that makes everyone safer. [Online]. Available: [https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/](https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/)

[24] P. Geurts, D. Ernst, and L. Wehenkel, “ Extremely randomized trees,” Machine learning, vol. 63, no. 1, pp. 3–42, 2006.

[25] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant, “Detecting influenza epidemics using search engine query data,” Nature, vol. 457, no. 7232, p. 1012, 2009.

[26] R. Gonzalez, C. Soriente, and N. Laoutaris, “User profiling in the time series,” in Proceedings of the 10th ACM SIGSAC Conference on Computer and Communications Security. ACM, 2003, pp. 1157–1182.

[27] S. Das, “Filters, wrappers and a boosting-based hybrid for feature selection,” in Icml, vol. 1, 2001, pp. 74–81.

[28] (2018, Jan.) Does page load time really affect bounce rate? [Online]. Available: [https://royal.pingdom.com/page-load-time-really-affect-bounce-rate/](https://royal.pingdom.com/page-load-time-really-affect-bounce-rate/)

[29] Google. (2019, Jan.) HTTPS encryption on the web. [Online]. Available: [https://google.com/https/overview?hl=en](https://google.com/https/overview?hl=en)

[30] J. Gu, J. Wang, Z. Yu, and K. Shen, “Walls have ears: Traffic-based side-channel attack in video streaming,” in IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018, pp. 1538–1546.

[31] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” Journal of machine learning research, vol. 3, no. Mat, pp. 1157–1182, 2003.

[32] J. Hayes and G. Danezis, “k-fingerprinting: A robust scalable website fingerprinting technique,” in USENIX Security Symposium, 2016, pp. 1187–1203.

[33] D. Herrmann, R. Wendolsky, and H. Federrath, “Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier,” in Proceedings of the 2009 ACM workshop on Cloud computing security. ACM, 2009, pp. 31–42.

[34] (2017) Https usage statistics on top 1m websites. [Online]. Available: [https://statcounter.com/research/https-usage-statistics-on-top-websites/](https://statcounter.com/research/https-usage-statistics-on-top-websites/)
— “Walkie-talkie: An efficient defense against passive website fingerprinting attacks,” in 26th [USENIX] Security Symposium ([USENIX] Security 17), 2017, pp. 1375–1390.

[78] G. Weinberg. (2018) Why should i use duckduckgo instead of google? [Online]. Available: https://www.quora.com/Why-should-I-use-DuckDuckGo-instead-of-Google

[79] Wikipedia. (2017, Aug.) Average precision. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=793358396#Average_precision

[80] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, “Uncovering spoken phrases in encrypted voice over ip conversations,” ACM Transactions on Information and System Security (TISSEC), vol. 13, no. 4, p. 35, 2010.

[81] J. Yan and J. Kaur, “Feature selection for website fingerprinting,” Proceedings on Privacy Enhancing Technologies, vol. 2018, no. 4, pp. 200–219, 2018.

[82] T. Yen et al., “Browser fingerprinting from coarse traffic summaries: Techniques and implications,” in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2009, pp. 157–175.

[83] S. Yu, W. Zhou, W. Jia, and J. Hu, “Attacking anonymous web browsing at local area networks through browsing dynamics,” The Computer Journal, vol. 55, no. 4, pp. 410–421, 2012.

APPENDIX

A. Related Work: Broader Discussion

Keyword Fingerprinting The possibility of identifying users’ search query has draw attentions from researchers. Chen et al. points out that sensitive information is being leaked out from several web applications, including search engines, despite the protection of HTTPS [15]. They consider the auto-suggestion/auto-complete features implemented by search engines as the main reason for query word leakage since for different combination of letters, list of packet sizes responded by search engines for each type-in is unique. More specifically, Sharma et al. demonstrates that for every character typed in Google search box, the exchanged packet is followed by a fixed pattern—the request size is increased by one byte for every character typed in while the response size reveals the suggestion made by the search engine [64]. In their threat model, the attacker sends 26 requests for each character (a-z) and captures the size of suggestion for comparison after collecting the sequence of packet sizes from networking trace. Thus to find a string of size n, the attacker needs to send 26^n automated search requests. Later, Schaub et al. study how to use stochastic algorithm to deal with variable packet lengths considering that Google has supported variable packet lengths for a given query with payload randomization and Gzip compression since 2012 [58]. For a given length, they create a prefix tree to represent the set of all possible words based on a chosen dictionary and preform hierarchical matching based on the observed size of responded packets. Oh et al. first extend standard website fingerprinting attacks to fingerprint individual keywords that contains more than one word in Tor and further apply deep learning to the threat model [50].

Website/Webpage Fingerprinting Website fingerprinting refers to the task of learning which website/webpage is being visited based on the information available from the TCP/IP headers in network traffic. Researchers have demonstrated the possibility to fingerprint websites/webpages in several communication scenarios including HTTPS (e.g., [45], [73], [41], [31]), encrypted tunnel (e.g., [37], [41], [52], [59], [22]) and Tor (e.g., [14], [83], [75], [13], [51], [76]). We refer readers to [81] for a detailed discussion about website/webpage fingerprinting in each communication scenario.

More recently, Abe et al.[11], Rimmer et al. [57] and Sirinam et al. [65] explore the application of deep learning to further boost the classification accuracy of website fingerprinting without manual feature selection. Jansen et al. [83] explore traffic analysis attacks on Tor with middle relays rather than with relays from entry or exit positions.

Other Learning-based Traffic Analysis Topics Other than website/webpage fingerprinting, machine learning has been applied in other fields of traffic analysis. Specifically, Wright et al. [80] shows the possibility to identify the phrase spoken within encrypted VoIP calls using knowledge of the phonetic pronunciation of words and Hidden Markov Model. Coull et al. [18] shows that information from instant messaging services such as users actions, the language of messages and even the length of the message can be learnt with more than 96% accuracy using only the sizes of encrypted packets. Alan et al. [4] and Vincent et al. [71] study the possibility of identifying applications installed on a smartphone using side-channel data such as packet size and direction. Schuster et al. [60] and Gu et al. [28] demonstrate the possibility of fingerprinting streaming videos by modeling their unique burst patterns using machine learning. Barradas et al. [9] explores the application of semi-supervised and unsupervised machine learning techniques to identify multimedia protocol tunneling systems including Facet, CovertCast and Deltashaper.

B. Distribution and Examples of search queries

Fig. 8 plots the distribution of number of characters in each search keyword. As can be seen, most of keywords are composed of multiple characters.

Fig. 8: Number of Characters in Search Keywords.

TABLE: X shows 10 examples of the harvested non-targeted keywords. The non-targeted keyword list is representative since: (i) it is composed of more than 200K popular web search queries, (ii) it contains more than one language, and (iii) it covers different topics.

TABLE X: 10 Examples of non-targeted search queries.

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | iron man 3 lego 7 cool things to buy | 6 | gmail windows app |   |   |
| 2 | $5 pizza hut deals 7 cool things to buy | 7 |   |   |   |
| 3 | (1 - x^2)^3 8 python |   |   |   |   |
| 4 | 澳门教會 | 9 | 스크래치 |   |   |
| 5 | proceso de fecundación | 10 | comcast specials |   |   |
C. Impact of Typing speeds

In this section, we study how typing speeds of entering search queries affect the traffic trace and the keyword classification accuracy in closed-world scenario. We collect a monitored dataset with 4 search engines (i.e., Google, Bing, Yahoo, DuckDuckGo) and 2 browsers (Chrome, Firefox), and vary the intervals between typing of consecutive characters in the search query from 0 to 1.2 seconds (representing different typing speeds). In total, we collect around 315,000 samples and analyze the following three aspects of the traffic traces collected using different typing speeds:

1) **General statistical information such as the total incoming/outgoing packet number and bytes**. Fig. 9a presents the results with Google/Chrome: the average inter-arrival time between subsequent incoming/outgoing packets does increase as typing speed decreases. However, there is no obvious variations in terms of packet number and bytes. The same trend is observed across other search engines and browsers.

2) **Linear discriminant analysis to understand how separable are traces collected with different typing speeds in their feature space**. We consider around 6,000 samples for each typing speed and use packet size count as features considering its good performance in prior experiments. Fig. 10 shows the result when projecting each feature space onto the three-dimensional subspace. As can be seen, the accuracy does not consistently increase or decrease as the typing speed varies.

Based on the above investigations, we did not observe any obvious evidence to indicate that a users’ typing speed impacts their vulnerability of being fingerprinted.

D. Wfin++: Forward Feature Selection

The Wfin methodology for feature selection ranks features based on their importance for classification, and then groups the features that account for 99% importance into semantically-relevant feature categories—all of these feature categories are then used to perform website fingerprinting.

We believe that the above methodology does not consider the significant amount of noise that can be retained when a large number of features are used by a classifier—we believe that a better classification performance can be achieved by reducing features further in a manner similar to forward selection. 

![Fig. 9: Impact of Typing Speeds](image)

![Fig. 10: Linear discriminant analysis for traces with different typing speeds using PSC](image)
We consider the final ranked list of feature categories returned by Wfin, and compute the validation accuracy by considering only the top-N feature categories. The goal is to find the N that yields the best validation accuracy. Fig. 11 plots the validation accuracy versus N, when the top-N feature categories are used for keyword classification with the four search engines accessed using the Chrome browser (Extra-Trees classifier, number of trees: 700). A similar trend is observed with all search engines—after the initial increase in classification accuracy, adding more features causes the accuracy to drop.

Thus the final feature list in Wfin++ is selected as the one that achieves the highest validation accuracy—just like Wfin, this feature list differs across the search engines. For instance, the final number of features selected for training with Yahoo/Chrome is around 5,820, while with Google/Chrome is around 7,200.

![Validation Accuracy](image)

Fig. 11: Validation Accuracy: top-N features used for classification (Chrome, homepage searching mode)

### HTTPS vs. Tor

Tor traffic has attracted lots of attentions for traffic analysis in website/keyword fingerprinting [50], [73]. Hence we compare the performance of Tor and HTTPS traffic in face of keyword fingerprinting to understand: (i) How vulnerable HTTPS traffic is compared with Tor? (ii) Can classifiers designed for Tor be directly applied onto HTTPS traffic?

To minimize the impact of irrelevant variations (e.g., data collection platform, visiting time and targeted keywords etc.) on classification accuracy, we collect another dataset with 2 search engines (Google and DuckDuckGo) and 2 browsers (Chrome and Tor (version 8.5.5)) in a round-robin way using the same 100 targeted keywords as Oh et al. [49]. Each keyword contains at least 100 samples, with the first 100 samples used for training/validation and the last 10 for testing. TABLE X shows the testing accuracy with the PSC, k-FP and EtResp/SvmResp. Besides, we train with SvmResp using the code provided by Oh et al. [49] to compare the performance of Extra-Trees and SVM. We find that:

- With the same set of features, Extra-Trees achieves comparable and slightly better performance compared with SVM.

### F. Vulnerability of Search Engines with Edge

TABLE XII shows the classification accuracy achieved using four different search engines with the Edge browser. The targeted dataset is split into three disjoint subsets for training, validation, and testing, with the ratio of 8:1:1: for every 10 consecutive visits of a search query, the first 8 samples are used for training, the 9th for validation, and the 10th for testing. For each targeted keyword, we consider 32 training samples, 4 validation samples and 4 testing samples during evaluation. To obtain the test accuracy shown in TABLE XII, we use 36 samples for training (including both samples from training and validation dataset) and 4 for testing. We observe that:

- Wfin++ and PSC consistently achieve better accuracy than EtResp and k-FP.
- Among all four search engines, DuckDuckGo makes the user the most vulnerable to keyword fingerprinting, while Bing offers the least vulnerability in homepage searching mode.
- After filtering out homepage traffic in address searching mode, classification accuracy of Yahoo increased by about 10%—which makes Yahoo traffic the most vulnerable. The classification accuracy of DuckDuckGo and Bing did not increase much, while that of Google decreased by about 4%.

### G. Informative Features Selected by Wfin++

TABLE XIII lists the top 20 informative feature categories selected by Wfin++ with DuckDuckGo (Chrome) in address-bar searching mode.
TABLE XII: Classification Accuracy Achieved: Edge.

| Search Mode       | Bing | Yahoo | Google | DuckDuckGo |
|-------------------|------|-------|--------|------------|
| Wfin++            | 35.10 ± 0.12 | 77.52 ± 0.15 | 64.09 ± 0.25 | 78.75 ± 0.08 |
| PSC               | 28.81 ± 0.36 | 72.22 ± 0.13 | 59.31 ± 0.32 | 76.15 ± 0.19 |
| EtResp            | 5.57 ± 0.10  | 0.49 ± 0.03  | 10.60 ± 0.09 | 8.17 ± 0.08  |
| k-FP              | 9.53 ± 0.07  | 1.11 ± 0.07  | 13.85 ± 0.14 | 11.56 ± 0.11 |

TABLE XIII: Top 20 informative feature categories with Wfin++

| Feature Category | Accuracy (%) |
|------------------|--------------|
| 0 unique packet size | 24.39 |
| initial 30 outgoing packets | 10.76 |
| packet size count | 6.33 |
| first 300 incoming packets preposition | 3.82 |
| first 300 outgoing packets preposition | 3.70 |
| first 300 outgoing packets | 3.62 |
| first 300 incoming packets | 2.97 |
| initial outgoing bursts | 2.86 |
| average outgoing inter-arrival time | 2.74 |
| initial 30 packets | 2.33 |
| initial 30 incoming packets | 2.24 |
| unique burst size | 1.93 |
| first 20 largest outgoing bytes per TCP conn | 1.82 |
| initial incoming bursts | 1.70 |
| ratio of incoming bytes per TCP conn | 1.56 |
| initial 30 outgoing in first TCP conn | 1.37 |
| burst size count | 1.27 |
| first 20 largest outgoing bytes per hostname | 1.26 |
| outgoing bytes per TCP conn | 1.07 |
| outgoing bytes per TCP conn w.r.t. Port 443/80 | 0.97 |

H. Parameter tuning in Extra-Trees Classifier

We use the implementation of Extra-Trees classifier from sklearn in python3 and tune two parameters (criterion and n_estimators) based on the validation accuracy achieved with the monitored dataset. For other parameters, we use the default value in sklearn.

1) Gini Index vs. Information Gain: The selection of features at each node of the tree to split the data (split criterion) directly affects the performance with decision tree-based ensemble methods. Two widely used split criterion is Gini Index and Information Gain. A lot of research was dedicated to understand which of them produce the best decision tree for a given dataset. Although most of empirical studies concluded that there is no significant differences between those two criteria and the disagreement is generally no higher than 2% of all cases, we show the validation accuracy achieved with different feature sets using different criteria with Extra-Trees classifier (n_estimators = 700) in Fig. 12 for Google/Chrome (the overall trend is consistent across different search engines and browsers).

Based on the outcome in Fig. 12, Gini Index achieves better accuracy compared with Information Gain with the monitored dataset and the gap ranges from around 2% to 21% across different feature sets with Google/Chrome. Thus we choose Gini Index as the split criterion in Extra-Trees classifier.

2) Number of Trees: As Breiman stated in 11, the behavior of prediction error for randomization methods is a monotonically decreasing function of number of trees in the ensemble. Thus the more the trees, the better the accuracy and the higher the computational overhead. In Fig. 13 we show the average error rate obtained by Wfin++ and the classification time when ranging the number of trees from 100 to 1,000 with Google/Chrome. As the results suggested, the error rate tends to stabilize and the classification time increase significantly as we keep increasing the number of trees – the error rate decrease around 0.1% while the classification time is increased by nearly 2/3 from 700 to 800. The same trend is observed with different search engines and browsers. In both closed-world and open-world experiments, therefore, we set the number of trees to 700.

I. Impact of Number of Targeted Keywords

In this section, we study how the accuracy changes as more targeted keywords are considered for classification by varying the number of keywords from 100 to 1,400. For each keyword, 45 samples are used for training and 9 for testing as in Section V-A. Fig. 14 shows the accuracy with four search engines using Wfin++ and PSC with Chrome in addressbar searching mode. As the results indicate, the performance keeps decreasing as more targeted keywords are considered although the rate of decline differs among search engines. For example, when number of targeted keywords is increased from 100 to 1,440, the accuracy drops significantly from around 67% to 45% with Bing/Chrome, but slightly from 99% to 96% with DuckDuckGo/Chrome. Thus when the number of targeted keyword keeps growing, more advanced machine learning techniques such as deep learning may help boost the performance.

J. Addressbar Searching Mode: Generated Traffic Pattern

Fig. 15 displays the traffic pattern generated by four different search engines for the same search query in Fig. 2 in

37A consistent trend is also observed with Firefox samples but the results are omitted due to space constraints.
addressbar searching mode. TABLE XIV shows the second-level server name contained in the traffic trace in Fig. 15 with Yahoo (Chrome).

TABLE XIV: Second-level Sever Names: Yahoo with Chrome in addressbar searching mode

| 2019-02-25-16:48:03 | 2019-02-25-17:39:38 |
|---------------------|---------------------|
| yahoo.com 4         | yahoo.com 5         |
| yimg.com 1          | yimg.com 1          |
| bing.com 1          | addthis.com 1       |
|                    | google.com 1        |
|                    | bing.com 1          |

| 2019-02-25-18:30:39 | 2019-02-25-19:20:47 |
|---------------------|---------------------|
| yahoo.com 5         | bing.net 6          |
| yahoodns.net 5      | yahoo.com 5         |
| yimg.com 1          | krxd.net 1          |
| nax.cac.com 1       | addthis.com 1       |
| google.com 1        |                    |
| bing.com 1          |                    |

K. Safari and Chrome in homepage searching mode

TABLE XV shows the performance of Wfin++ and PSC in cross-browser scenario with Google and Firefox samples. Similar trend is observed as in addressbar searching mode—they accuracy degrades drastically when the training and testing browser do not match.

TABLE XV: Classification Accuracy: Cross-browser Attacks in Homepage Searching

|                        | Firefox   | PSC       | Chrome    | PSC       |
|------------------------|-----------|-----------|-----------|-----------|
| Firefox                | 91.95 ± 0.10 | 92.23 ± 0.02 | 1.22 ± 0.09 | 1.82 ± 0.05 |
| Chrome                 | 0.22 ± 0.01 | 0.97 ± 0.05 | 96.15 ± 0.01 | 96.33 ± 0.04 |
| Fire/Chr               | 91.40 ± 0.08 | 92.14 ± 0.06 | 96.30 ± 0.04 | 96.04 ± 0.04 |
| Google                 | 75.56 ± 0.07 | 76.75 ± 0.02 | 0.38 ± 0.07 | 1.42 ± 0.04 |
| Chrome                 | 0.07 | 0.98 ± 0.13 | 60.32 ± 0.07 | 57.72 ± 0.05 |
| Fire/Chr               | 75.17 ± 0.13 | 75.88 ± 0.13 | 61.91 ± 0.31 | 57.12 ± 0.14 |
| Yahoo                  | 58.06 ± 0.25 | 49.87 ± 0.13 | 15.61 ± 0.35 | 12.42 ± 0.21 |
| Chrome                 | 4.01 ± 0.39 | 2.47 ± 0.04 | 64.60 ± 0.21 | 57.85 ± 0.26 |
| Fire/Chr               | 57.56 ± 0.12 | 48.73 ± 0.18 | 64.22 ± 0.22 | 56.23 ± 0.05 |

L. Time Effect in homepage searching mode

TABLE XVI summarizes the impact of time gap between training and test samples, in closed-world evaluations with the homepage searching mode (Chrome browser).

|                        | test-3 | test-4 | test-8 | test-10 | test-14 |
|------------------------|--------|--------|--------|---------|---------|
| Wfin++                  | 92.20 ± 0.07 | 90.56 ± 0.02 | 81.22 ± 0.35 | 71.79 ± 0.05 | 71.70 ± 0.04 |
| PSC                    | 92.47 ± 0.07 | 90.84 ± 0.02 | 80.2 ± 0.35 | 69.74 ± 0.05 | 69.39 ± 0.04 |
| EtResp                 | 26.65 ± 0.07 | 22.04 ± 0.02 | 11.92 ± 0.35 | 7.61 ± 0.05 | 6.58 ± 0.04 |
| k-FP                   | 28.82 ± 0.07 | 27.07 ± 0.02 | 20.98 ± 0.35 | 14.62 ± 0.05 | 14.00 ± 0.04 |

M. Cross-browser attack with Edge and Safari

TABLE XVII shows the accuracy achieved when using 36 Edge samples for training and 4 Firefox samples for testing in both homepage searching mode and addressbar searching mode. Consistent with the results obtained in Section V.C, the classification accuracy degrades severely when using samples from different browser/device for training and testing. For instance, when testing with Safari samples the accuracy is dropped to less than 0.2% compared with around 78% with Edge samples.

TABLE XVII: Classification accuracy (%) in cross-browser attack with Edge and Safari. (DDG: DuckDuckGo)

|                        | Edge | Safari |
|------------------------|-----|--------|
| Wfin++                  | 78.75 ± 0.07 | 76.15 ± 0.02 |
| PSC                    | 0.19 ± 0.04 | 0.15 ± 0.02 |
| Google                 | 64.10 ± 0.07 | 59.31 ± 0.02 |
| Wfin++                  | 0.19 ± 0.05 | 0.12 ± 0.02 |
| Edge                   | 59.54 ± 0.07 | 60.33 ± 0.07 |
| Safari                 | 0.46 ± 0.06 | 0.07 |

N. Eliminating Noise from “Other” Domains

TABLE XVIII summarizes the classification accuracy when connections to only top-level domains are considered by the attacker.

TABLE XVIII: Classification accuracy (%) when only considering top domains in addressbar searching mode

|                        | Firefox | Chrome |
|------------------------|---------|--------|
| Wfin++                  | 93.30 ± 0.06 | 93.42 ± 0.06 |
| PSC                    | 1.79 ± 0.20 | 1.99 ± 0.14 |
| Edge                   | 1.00 ± 0.14 | 0.96 ± 0.09 |
| Safari                 | 96.63 ± 0.03 | 96.62 ± 0.04 |
| Wfin++                  | 78.20 ± 0.10 | 80.53 ± 0.16 |
| PSC                    | 5.13 ± 0.11 | 1.30 ± 0.17 |
| Fire/Chr               | 2.21 ± 0.06 | 1.80 ± 0.05 |
| Google                 | 62.39 ± 0.10 | 59.16 ± 0.08 |

O. Multi-level Classification: Impact of Number of Non-targeted Test Samples

Fig. 16 shows the performance of Wfin++ and PSC in multi-level classification, for different number of non-targeted
testing samples. We observe that:

- More non-targeted samples are incorrectly classified as targeted ones \((FPR)\), as the number of non-targeted testing samples increases.
- The likelihood of incorrectly classifying a targeted sample \((FNR)\) is not affected by the number of non-targeted testing samples.

\[\text{Fig. 15: Traffic generated using Chrome: Addressbar Searching (caption below each sub-figure indicates time of search)}\]

\[\text{Fig. 16: Multi-level Classification: Impact of Number of Non-targeted Test Samples (50k Non-targeted Training Samples)}\]

\[\text{P: Countermeasures}\]

In this section, we evaluate keyword fingerprinting in the presence of countermeasures in closed-world scenarios. By examining the most-informative features yielded by Wfin++, we find that most of these are extracted from packet sizes (e.g., unique packet size and packet size count) and packet ordering (e.g., initial 30 outgoing packets and first 300 incoming/outgoing packets preposition). We next consider two prominent HTTPS countermeasures that obfuscate actual packet size or packet ordering—PadToMTU \(^{38}\) and HTTPOS \(^{40}\). PadToMTU aims at hiding actual packet sizes, which is one of the most informative features for website fingerprinting in HTTPS \(^{37}, 31, 51, 22, 81, 85\), by padding each packet to MTU bytes. HTTPOS is a browser-side defense to obfuscate traffic by exploiting several TCP and HTTP features—including, MSS negotiation, advertised window, HTTP Range, and HTTP Pipelining \(^{40}\). Specifically,

\(^{38}\)Top 30 informative features categories selected by Wfin++ in addressbar searching mode with DuckDuckGo (Chrome) are listed in Appendix \(\text{G}\).

\(^{39}\)Advanced countermeasures designed for encrypted tunnels or Tor (e.g., \(\text{L3}, 27\)) are not considered since they are not directly applicable in HTTPS.
it injects dummy requests within the user traffic in order to obfuscate actual traffic patterns, and modifies advertised window size to make servers pack responses into blocks of MSS-byte packets (and hide actual packet size). Luo et al. [40] shows that HTTPOS is able to effectively prevent attackers from inferring information from 1,000 Google search queries over HTTPS.

In order to implement PadToMTU, we replace the sizes of all incoming and outgoing packets in our traces with MTU bytes. In order to implement HTTPOS, we pad the packet size of each outgoing packet (request) \( (p_{\text{size}}) \) to a value chosen randomly from the discrete uniform distribution \([p_{\text{size}}, \text{MTU}]\); for each incoming packet (response), we pad the packet size to a value chosen randomly from the discrete uniform distribution \([p_{\text{size}}, 3\times\text{MSS}]\), and segment into multiple packets with MSS bytes in each. Specifically, we set MTU to 1,500 and MSS to 1,000, as described in [40].

### TABLE XIX: Classification Accuracy Against Countermeasures: Addressbar Searching

| PadToMTU       | Bing | Yahoo | Google | DuckDuckGo |
|----------------|------|-------|--------|------------|
| Wfin++         | 9.26 ± 0.17 | 2.34 ± 0.06 | 27.93 ± 0.03 | 44.22 ± 0.01 |
| PSC            | 0.90 ± 0.03 | 0.11 ± 0.02 | 1.45 ± 0.06 | 3.38 ± 0.04 |
| BW overhead    | 209.6% | 522.8% | 193.9% | 193.4% |

| HTTPOS         | Bing | Yahoo | Google | DuckDuckGo |
|----------------|------|-------|--------|------------|
| Wfin++         | 7.86 ± 0.03 | 4.54 ± 0.01 | 27.72 ± 0.07 | 40.72 ± 0.04 |
| PSC            | 0.85 ± 0.01 | 0.11 ± 0.03 | 1.78 ± 0.05 | 2.43 ± 0.05 |
| BW overhead    | 220.2% | 330.1% | 207.3% | 218.7% |

| Firefox PadToMTU       | Bing | Yahoo | Google | DuckDuckGo |
|------------------------|------|-------|--------|------------|
| Wfin++                 | 7.42 ± 0.12 | 2.11 ± 0.08 | 25.53 ± 0.02 | 36.23 ± 0.05 |
| PSC                    | 0.29 ± 0.07 | 0.11 ± 0.00 | 0.25 ± 0.01 | 0.32 ± 0.02 |
| BW overhead            | 403.3% | 391.8% | 387.5% | 387.4% |

| Firefox HTTPOS         | Bing | Yahoo | Google | DuckDuckGo |
|------------------------|------|-------|--------|------------|
| Wfin++                 | 6.54 ± 0.02 | 4.05 ± 0.06 | 24.23 ± 0.07 | 33.96 ± 0.15 |
| PSC                    | 0.26 ± 0.00 | 0.08 ± 0.03 | 0.28 ± 0.02 | 0.26 ± 0.04 |
| BW overhead            | 420.5% | 562.9% | 405.1% | 414.3% |

TABLE XIX shows the accuracy of Wfin++ and PSC in the presence of PadToMTU and HTTPOS for fingerprinting 1,440 targeted keywords—when only connections to top domains are used (Section V-D) in addressbar searching mode. The training, validation, and testing dataset are the same as in Section V-A—for each query, 36 samples are used for training, 9 for validation, and 9 for testing. The accuracies of PSC and Wfin++ are significantly lower in the presence of PadToMTU and HTTPOS (compared to Tables VIII and XVIII)—for instance, for samples collected with DuckDuckGo/Chrome, the accuracy achieved by Wfin++ decreases from around 96% to 44% or 36%, while the accuracy achieved by PSC decreases from 96% to 3% or 0.3%, respectively, in the presence of PadToMTU and HTTPOS.

However, both PadToMTU and HTTPOS incur significant bandwidth overhead—for instance, with Google/Chrome the overhead is as high as 193% for PadToMTU, and 387% for HTTPOS. The PadToMTU overhead is due to padding each packet to MTU bytes; the HTTPOS overhead is due to insertion of dummy requests/responses. We leave the design and evaluation of additional countermeasures against keyword fingerprinting as important future work.

The integration of HTTPOS into different browsers to perform automated large scale data collection with Selenium is considered as a future work.

---

[40] The integration of HTTPOS into different browsers to perform automated large scale data collection with Selenium is considered as a future work.