An Investigation on Information Leakage of DNS over TLS

Rebekah Houser
University of Delaware
rlhouser@udel.edu

Zhou Li
University of California, Irvine
zhou.li@uci.edu

Chase Cotton
University of Delaware
ccotton@udel.edu

Haining Wang
Virginia Tech
hnw@vt.edu

ABSTRACT

DNS over TLS (DoT) protects the confidentiality and integrity of DNS communication by encrypting DNS messages transmitted between users and resolvers. In recent years, DoT has been deployed by popular recursive resolvers like Cloudflare and Google. While DoT is supposed to prevent on-path adversaries from learning and tampering with victims' DNS requests and responses, it is unclear how much information can be deduced through traffic analysis on DoT messages. To answer this question, in this work, we develop a DoT fingerprinting method to analyze DoT traffic and determine if a user has visited websites of interest to adversaries. Given that a visit to a website typically introduces a sequence of DNS packets, we can infer the visited websites by modeling the temporal patterns of packet sizes. Our method can identify DoT traffic for websites with a false negative rate of less than 17% and a false positive rate of less than 0.5% when DNS messages are not padded. Moreover, we show that information leakage is still possible even when DoT messages are padded. These findings highlight the challenges of protecting DNS privacy, and indicate the necessity of a thorough analysis of the threats underlying DNS communications for effective defenses.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

DNS Privacy, Website Fingerprinting, Traffic Analysis

ACM Reference Format:
Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. 2019. An Investigation on Information Leakage of DNS over TLS. In *The 15th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '19), December 9–12, 2019, Orlando, FL, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3359989.3365429

1 INTRODUCTION

As users increasingly rely on the Internet for diverse, daily interactions, the demand for secure online communications with privacy protection increases. The main threat to secure and private communication is "pervasive monitoring" announced in an RFC by the IETF [27], which describes the network adversaries who can monitor Internet communications and infer users' private information. While network encryption provides a certain degree of protection by obscuring the content of packets, metadata like packet size and timing are still exposed. This information can be exploited for the purpose of website fingerprinting, i.e., learning which websites a user has visited, and a number of prior works have explored this idea [19, 26, 32, 33, 48, 49, 62, 63].

While those works have demonstrated that website fingerprinting is an effective attack vector, they all focus on the protocol that is directly related to a visit to a website, *HTTP(s)*. In this work, we study another protocol that is relevant to a website visit in an indirect way, *DNS*. As stated in [16]: "Almost every activity on the Internet starts with a DNS query (and often several). Its use has many privacy implications[.]" However, exploiting DNS for website fingerprinting was a less attractive research idea as most DNS messages are sent in plain text, which is rather different from the deployment of encryption on HTTP.

Recently, this situation has been changing. As advocated by the IETF, the research community, and industry, there is a strong push to apply network encryption on DNS messages. In fact, several RFCs about DNS encryption have been established, and a number of DNS service and software providers have implemented the RFCs in their products. As such, we believe that now is the time to investigate the protection of DNS encryption on user privacy, especially whether and how it can defend against website fingerprinting attacks. In this work, we focus on DNS over TLS (DoT) that has been both standardized and extensively implemented [23, 35].

Although the information from the encrypted DNS and HTTP messages could be similar under website fingerprinting, DNS poses greater challenges in achieving the same level of effectiveness, since the packet size of DNS is usually much smaller than that of HTTP, which could result in a higher similarity between different website visits. We tackle this issue by incorporating a list of traffic features found in existing works (such as time intervals and ordering groups) and new features unique to DoT (such as the number of DNS messages within a TLS record). We evaluated our approach using Random Forest and Adaboost classifiers. The evaluation results indicate that website fingerprinting is still a serious threat under DoT. In particular, we collected DoT traces of 98 sensitive websites under three categories (health insurance, dating, and gambling). When DNS messages are not padded, the false negative rate (FNR) and false positive rate (FPR) of category classification are less than 7% and 5%, respectively. For classifying individual pages, the FNR

CoNEXT '19, December 9-12, 2019, Orlando, Florida, USA

ACM ISBN 978-1-4503-6998-5/19/12…$15.00
https://doi.org/10.1145/3359989.3365429

and FPR are less than 17% and 0.5%, respectively, without padding. Even with padding, classification still works relatively well, with the FNR or FPR as low as zero when identifying individual websites. The result is consistent across different public DoT resolvers we tested, including Google, Cloudflare, and Surfnet. More surprisingly, we found that though DNS messages can be padded, i.e., through enabling `EDNS(0)` option, reasonable fingerprinting accuracy can still be achieved. Finally, we make recommendations on effective defenses against DoT fingerprinting attacks.

The rest of the paper is organized as follows. Section 2 presents the background of DNS, especially DNS privacy and DNS over TLS. Section 3 defines our threat model. Section 4 details our proposed approach for DoT traffic fingerprinting. Section 5 describes our evaluation methodology and results. Section 6 further discusses limitations of our work and effective defenses. Section 7 reviews relevant studies. Finally, Section 8 concludes the paper.

2 BACKGROUND

In essence, DNS comprises a hierarchical database of resource records distributed across a global network of nameservers. This distributed system serves to map a domain name (e.g., *example.com*) to one or more associated IP addresses, and vice versa.

Typically, to initiate a DNS lookup for a host name, a user first needs to communicate with a stub resolver and/or recursive resolver. The resolver acts as a proxy, and completes the DNS name resolution for the user. If the record requested is in the resolver's cache, the cached response will be used to reply to the user. Otherwise, the resolver will communicate with other DNS servers to obtain the desired information. A stub resolver (which usually runs on the user's machine) will send the query to a recursive resolver. A recursive resolver will traverse the hierarchical tree of nameservers, starting from the root until the resolver has the authoritative answer to the user's request. Recursive resolvers are typically operated by service providers or organizations to serve multiple users. A user may also run a recursive resolver on his or her own machine, and communicate directly with authoritative nameservers. This approach is less common, as shared caching at the recursive resolver is key to decreasing the latency experienced by clients and the traffic at nameservers [36, 56].

2.1 DNS Privacy

When the DNS standard was proposed, privacy issues were not considered [43, 44]. By default, DNS messages are sent in plain text, allowing any party capable of monitoring network traffic to eavesdrop on DNS messages [16]. DNS messages contain a great deal of information about which domains users visit, what services or applications they use, and with whom they communicate [20, 31, 64]. A person may have various reasons for keeping such information private, including circumventing censorship and avoiding surveillance. This situation has led to the development of secure protocols and standards for providing privacy for DNS communications. In this work, we focus on one of these protocols: DoT.

DoT attempts to provide privacy via encryption for the most vulnerable portion of a DNS lookup: the segment between a user and his/her recursive resolver. Using DoT, a client on the user's machine will establish a TLS session with a recursive resolver, and then use this session to exchange encrypted DNS queries and responses. This segment is the most vulnerable to attacks on privacy, as traffic in this segment can be more easily associated with individual users. DoT may also be used between the recursive resolver and nameservers. Organizations operating public resolvers and nameservers have also done experiments using DoT [21].

While DoT is supported by many groups and tools, DNS over HTTPS (DoH) has also gained a substantial following. Both DoT and DoH encrypt DNS communications. One of the major differences between these two protocols lies in that DoT has its own port, Port 853, but DoH uses Port 443, which is the standard port for HTTPS traffic.

The protocol creates opportunities for significant changes in how DNS is handled. As noted in the RFC, DoH "is more than a tunnel over HTTP" [34]. DoH allows DNS records to be integrated into the HTTP ecosystem of "caching, redirection, proxying, authentication, and compression" [34]. While various groups have been developing tools for DoH for months, the potential for control and customization introduced by DoH has yet to be fully explored. Until these changes take place, we believe the insights gained from DoT will also apply to DoH. We confirm this by conducting a simple test, which is presented in Appendix A.

2.2 DNS over TLS

2.2.1 Implementation Status. DoT deployment has been growing steadily over the past few years. The IETF published the RFC specifying DoT in 2016 [35]. The list of DoT implementations maintained by the DNS Privacy Project includes stub and recursive resolvers, forwarders, command line tools, and a browser [23]. Currently, several public recursive resolvers provide DoT with varying levels of support for features such as padding and query name minimization. The implementation and adoption of DoT is ongoing. In early 2019, Google announced support for DoT in its public resolvers [25], and developers of Stubby (the main stub resolver associated with DoT) have provided new packages and installers [24].

2.2.2 DoT and Traffic Analysis. Encrypting DNS does not entirely eliminate concerns regarding the privacy of DNS. An adversary may still gain information about a user's activities by analyzing DoT traffic. The specification for DoT notes the possibility for such an attack against encrypted DNS [35].

Additional measures may thus be necessary to ensure the privacy of DoT. Padding encrypted DNS messages has already been proposed and implemented as one such protection measure, although it remains to be determined how effective this approach is. For DNS, one might expect padding to be highly effective, since DNS messages tend to be short and their size can be hidden in a relatively easy manner. Still, no prior research has conducted a thorough analysis on the privacy implications of DoT deployments. Our work aims to fill this gap, assess whether the communication privacy is adequately protected, and provide recommendations for more effective defense.

2.2.3 Insights of DoT Traffic. The traffic analysis of DoT shares the same general approach as previous works involving HTTPS traffic. However, the nature of the underlying systems makes the traffic analysis on DoT, in some ways, a more difficult problem.

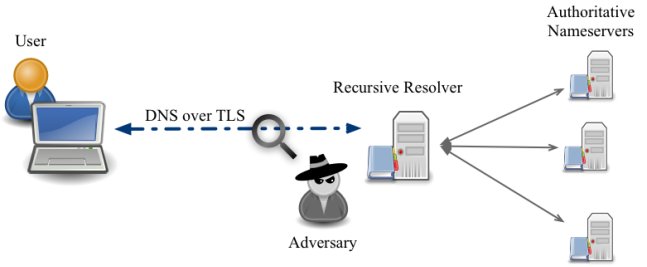


Figure 1: Threat model: passive eavesdropper between a user and recursive resolver.

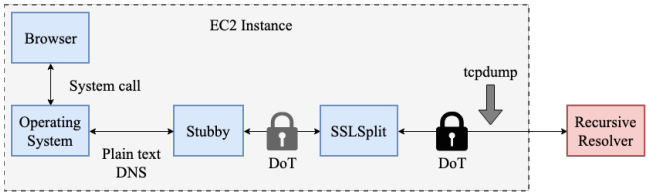


Figure 2: Data collection.

The challenge for DNS lies largely in its relatively low traffic volume and small message sizes. To load a webpage, a browser will issue several GET requests to retrieve all the objects of the page. Not all of these requests require a unique DNS lookup, as multiple objects may be stored on the same host. One measurement study found that the median number of objects required to load a page was 40, but for 60% of pages, at most 20 servers were contacted [17]. Since DNS queries will be largely driven by the number of servers contacted, this finding suggests that the number of DNS messages sent when loading a page will tend to be lower than the number of HTTP messages.

Moreover, the size of a DNS message is normally small. HTTP requests can include a variety of fields that lead to significantly different message sizes [47]. The sizes of the objects on a webpage may span orders of magnitude, from less than 100 bytes up to thousands of bytes [17]. By contrast, DNS messages have historically been constrained to be less than 512 bytes when transmitted over UDP (which is by far the most common method) [44]. DNS Extensions now allow DNS messages over 512 bytes [41], and DNSSEC often generates messages over 512 bytes [57]. However, the DNSSEC deployment is still relatively low [7, 8].

3 THREAT MODEL

We consider a scenario, illustrated in Figure 1, in which a user is using DoT for DNS privacy, while an adversary attempts to infer that user's Internet activities from his/her DoT traffic. The adversary has two primary goals. First, the adversary attempts to learn if the victim is visiting a particular *category of websites*, which the adversary deems sensitive or controversial. Second, the adversary is interested in identifying if the victim visits a *specific website*.

In our threat model, we assume that the adversary has the capability to eavesdrop on the network traffic between the victim user and the recursive resolver.

However, the adversary is limited to accessing one vantage point and cannot collect data from the path between the recursive resolver and nameservers, where eavesdropping is typically much more difficult. Based on the terminology defined by Díaz et al. [22], our adversary is passive and local. As suggested by [33], this kind of adversaries widely exist in the real world.

4 PROPOSED DOT FINGERPRINTING

The design of our proposed DoT fingerprinting is based on the observations described in Section 2.2.3. To conduct DoT fingerprinting, the adversary first needs to collect DoT traces by visiting websites of interest using an environment similar to that of the victim, e.g., using the same browser family. Then, the adversary will extract features from the DoT traces based on the specific objective, i.e., category or website classification, and derive fingerprinting models (or classifiers) based on machine learning. After the training stage mentioned above, the adversary will need to collect the DoT traces of the targeted victim (e.g., by monitoring the traffic flowing to destination port 853) and apply the trained classifier to predict whether the victim has visited a website of interest.

Following this strategy, we collected a large number of DoT traffic instances related to website visits to train and test models. In Section 4.1, we describe the details of our testing setup and data collection. In Section 4.2, we elaborate on how features are extracted from DoT traffic. Finally, in Section 4.3, we describe how the classifiers are chosen and implemented.

4.1 Environment and Data Collection

4.1.1 Test Setup. We set up our testing environment on a virtual machine rented from Amazon EC2 to generate website visits and capture the associated DoT traffic. The instance was located in the `us-east-1` availability zone and ran Ubuntu 18.10 as the operating system. Each website visit is executed by Firefox (Extended Release Support version 60.4.0) [9]. Given that a large number of DoT traces need to be collected, we automate the task by using Python Selenium [46] to drive Firefox actions in *headless mode*, under which Firefox runs without showing a GUI [10].

One thing to note is that at the time of this writing, Firefox does not integrate a DoT stub resolver. As a result, we needed to configure an external DoT stub resolver and use it to relay Firefox DNS queries. To this end, we used a forwarder named `Stubby` [6], which is a local, non-caching stub resolver developed under the `getdns` project. `Stubby` listens to the loopback address and forwards the plain-text DNS messages it receives to the recursive resolver using DoT. Relaying DNS messages through DoT prevents us from viewing the DNS traces for analysis. To address this issue, we set up a network proxy named `SSLSplit` [53] between `Stubby` and the recursive resolver. `SSLSplit` allows us to decrypt DoT traffic and recognize the fields. We note that `SSLSplit` is only used for debugging and labeling. None of the decrypted information is leveraged for features of classifiers, as they should not be available to our on-path adversary. Finally, `SSLSplit` communicates with

Table 1: Summary of datasets

Dataset long[1]	Padded	Number of instances: Target	Top	Background
Surfnet 0	Yes	3,443	2,780	1,429
	No	3,818	3,782	972
Surfnet 1	Yes	1,294	3,314	806
	No	1,256	2,922	1,154
Google	Yes	1,334	3,561	1,496
	No	1,302	3,967	735
Cloudflare	Yes	1,330	3,862	1,509
	No	1,350	3,873	879

the recursive resolver, and the traffic between them is captured by `tcpdump` [12]. Figure 2 illustrates the data collection setup.

Regarding the tested recursive resolvers, we chose Google, Cloudflare, and Surfnet. Surfnet resolvers are defaults for `Stubby`. Google and Cloudflare are well-known resolvers. For most tests, we chose the Surfnet resolver because its padding policy depends on the `EDNS(0)` padding option from the client, allowing us to assess the impact of padding on the inference results. This decision and the resolvers' padding policies are further discussed in Section 5.2.4.

4.1.2 Dataset. We chose websites to be visited in three groups: *target* (visits to the sites are sensitive and monitored by adversaries), *top* (popular sites users are likely to visit) and *background* (less popular sites visited during testing but not training). For the target group, we selected 98 websites under three sensitive categories: health insurance, dating, and gambling websites. We decided to use those websites after searching keywords related to each category and using the Alexa Top Sites categorical lists for gambling and dating [13]. The top group includes 100 websites taken from the 150 most popular sites ranked by Alexa after filtering out localized versions of domains (i.e., if example.com and example.us both appeared on the list, we would only keep example.com) and pages that had problems loading in pre-tests. The background group contains 1,650 less popular websites, selected from the Alexa Top Sites list in groups of 75 pages at intervals of 5,000, starting at rank 5,000. We obtained the lists for the top and background groups using the Alexa Top Sites API to get pages ranked by popularity in the United States [5]. We removed the duplicates shared by at least two lists to ensure there is no overlap between different groups.

For each website, we directed Firefox to visit its homepage and captured the associated DoT traffic. Before loading a page, we started running `tcpdump`, and we stopped `tcpdump` after the page was considered to be loaded in order to create separate `pcap` files for different webpage visits. In particular, each page is given 30 seconds to load, and we waited an additional 5 seconds between webpage visits. This approach models a scenario in which an attacker is able to determine the exact start and end of DoT messages associated with a single webpage. In practice, the attacker may make such a determination by using heuristics related to intervals between queries as in [58], or more sophisticated clustering methods as in [15]. Our testing environment (Firefox and Stubby) does not cache the DNS responses locally to avoid interference between visits. For each page load, we also saved its HTML code to a disk. We used the HTML file information (i.e., its size and keywords in the title field) to filter out failed page loads. This filtering procedure yields a set of "good" instances that we used for further analysis. Table 1 summarizes the datasets after filtering.

We ran three rounds of data collection over two months. We collected data from the Surfnet resolvers in early February and late March 2019, and from the Google and Cloudflare resolvers in early April 2019. Regarding the tests for the top and target groups, we loaded each homepage 80 times: 40 times without padding queries and another 40 with padded queries. Our data collector visits the pages in a round-robin fashion, in that all pages are visited sequentially before the next round. Sometimes, the DNS caching at the recursive resolver might introduce prominent changes to the subsequent visit. To address this issue, we shuffled the order of webpage visits per round to let as many caches expire as possible. Since the background group is mainly for balancing the training and testing dataset with irrelevant instances, we loaded each webpage in the group only twice (once padded and once non-padded).

4.1.3 Ethics of Data Collection. The webpage visits are executed by our automated data collector so that no human subjects are involved. While the sites in the target group are considered sensitive to human users, none of them are illegal in United States; therefore, visiting them does not break the law. The main ethical concern of data collection is that our visits could introduce excessive overhead on the public recursive resolver. To address this concern, we limited the number of webpage visits in a given time period to a rate similar to what is expected for a human user browsing the Internet. We gave each visit 30 seconds to finish, consistent with a measurement study of webpage visits collected by European ISP [58], which suggests that 50% visits take 30 seconds or less.

4.2 Features

From an adversary's point of view, the DoT traces associated with one webpage visit can be considered as a time-sequence of packets. Due to the encryption performed by DoT, only the size, timestamp, and direction are useful to the adversary (the destination port,853, is a constant and the destination IP belongs to the recursive resolver). As such, the traces for one webpage visit can be represented as $[(t_0, l_0, d_0), (t_1, l_1, d_1), ..., (t_n, l_n, d_n)]$, where t_i is the timestamp, l_i is the length of packet, and d_i is the direction (DNS query or response). Given that a webpage usually combines content from many different origins, one webpage visit usually yields a varying number of DNS requests and responses with different t_i, l_i, and d_i. As such, we extract statistical feature values from the packet sequence and use them for classification.

In particular, we include *minimum, maximum, median, mean, deciles*, and *count* on a subset or the whole packet sequence. The feature values are decimal numbers, which can be directly consumed by machine-learning models. Below we give details of each feature we use. We acknowledge that some of these features are inspired by previous works on website fingerprinting (though the protocols targeted by those works are mainly HTTPS or Tor, which are different from ours), and we reference those features in the description below. We also derive new features based on our observations of DNS traffic characteristics.

4.2.1 Query Length and Response Length. These two features represent the length (in bytes) of TLS records carrying DNS queries and responses. We found the maximum, median, mean, minimum, and deciles of the query and response lengths in a trace. This length is distinct from the TCP payload size of a single packet, as the TCP payload may contain multiple TLS records. In addition, this field might not be equal to the DNS message size, as a single TLS record may contain multiple DNS queries or responses. Individual message sizes have been often used in similar studies [1, 19, 33, 48, 49, 62].

4.2.2 Total Number of Queries and Responses. How resources are included by a webpage (e.g., JavaScript files, iFrame content, images, and Adobe Flash video) usually has a significant impact on the number of queries and responses [16]. Such numbers can be distinctive among pages for different websites and website categories. For example, a news website can integrate many online advertisements, resulting in many DNS resolutions. By contrast, a government website can have far fewer queries because content from external sources is less likely to be included. This feature is equivalent to the packet count features in [32, 39, 48, 49, 62].

4.2.3 Cumulative Bytes of Queries and Responses. While it is obvious that the sum of bytes of queries and responses can be distinctive for a webpage visit, given that their number and individual record sizes are different, we also found that the size and rate over time can be distinctive as well. For example, a webpage might load all the needed external resources in the very beginning, while another webpage might load the resources intermittently, triggered by events or timeouts. We constructed this feature by calculating the cumulative bytes sent and received at each point a packet is captured, and then by finding the maximum, median, mean, and deciles of these values. This method is similar to the one used by [48]. We also found the ratio of cumulative bytes received to cumulative bytes sent. To construct this feature, we divided the trace into tenths and found the ratio of bytes transmitted to bytes received at the end of each segment, as well as the mean of the ratios calculated at each point a packet is captured.

4.2.4 Time Intervals. Similar to [32], we extracted the features from the intervals between packets. Firstly, we considered the intervals between two consecutive queries and two consecutive responses. Secondly, we considered the intervals between an adjacent pair of a query and response (and vice versa). We found the maximum, minimum, median, mean, and deciles of both types of intervals in a trace.

4.2.5 Total Transmission Time. We found that some webpages keep loading resources, resulting in long transmission times, but there are webpages that finish resource loading quite early.

We computed the total time elapsed between the timestamps of the first and last TLS record containing application data (sent or received) in the trace, and used the result as the feature. This feature is also used in [26, 62].

4.2.6 Ordering Group. The response is not always successive to the corresponding query. In fact, the client may send multiple queries before receiving a response. Additionally, multiple responses may arrive without a query inserted in between, possibly due to a prior burst of queries. We call a series of the uninterrupted queries or responses an *ordering group*, and we computed the maximum, median, and mean over the query and response groups. Similar features have been used in [32, 49, 62].

4.2.7 Number of DNS Messages within a TLS record. We observed that a recursive resolver may place multiple DNS messages in a TLS record. Though the number of messages is not explicitly available from a field in the TLS record, we found that this can be inferred if padding is performed by the recursive resolver. To obtain this feature, we found all TLS records in a trace containing multiple DNS messages, and then calculated the maximum and median number of messages contained in these records. This feature only applies when responses are padded.

4.2.8 Queries per Second. While the sequence order features tell how many queries or responses were seen in ordering groups, they do not reflect how close in time the queries and responses are observed. In fact, one ordering group can take a few seconds to finish the transmission while another with the same number of queries and responses can only take a few milliseconds. To measure such timing distribution, we computed the number of queries or responses in non-overlapping, 1-second windows, and then found the maximum, median, mean, minimum, and deciles of these values. The same feature was used by [32].

4.2.9 Time to receive first N bytes. The time to receive a response can reflect whether the recursive resolver has the response cached. Caching may reflect aspects of a page, such as popularity or geographical location of its nameserver. Since not every query is closely followed by its response, determining the timing duration for all query-response pairs is impossible. As an alternative approach, we focus on the queries and responses at the beginning of a trace, which include the query for the domain under which the webpage is hosted. While a browser may issue several other queries before the query for the target domain, these will follow a pattern, and it is possible to choose N, such that the first N queries include the one for the target domain. Based on our observations of the patterns generated by our browser, we set N to 3,000 bytes when responses are not padded, 4,000 when responses are padded with a block size of 128 octets, and 5,000 when responses are padded with a block size of 468 octets. Features focused on the packets at the beginning of a trace are also included in [49, 62].

As a final step of feature extraction, we removed features with zero variance from the dataset. When padding is applied, most of the features involving message response lengths are dropped. Most of the other features dropped, with or without padding, related to time intervals.

4.3 Classifiers

To determine which classifiers to use, we tested a number of widely used machine-learning classifiers, including Naive Bayes, Simple Logistic, SMO, J48 Decision Tree, and Random Forest. Note that this exploratory analysis is conducted on a dataset having no overlap with the datasets we used for evaluation.

From the preliminary results, we found that Random Forest performs best, so we continued to use this algorithm for the evaluation.

Table 2: Results for Random Forest classifiers identifying DoT traffic instance category. Values given are mean ±SEM

	Category	F1	FNR	FPR
Padded 0	Dating	0.783±0.002	29.21 ±0.27%	4.31±0.03%
	Gambling	0.782±0.001	30.65 ±0.17%	3.39±0.02%
	Health Insurance	0.839±0.001	21.31 ±0.2%	3.38±0.05%
Padded 1*	Dating	0.874±0.001	17.1 ±0.12%	2.93±0.05%
	Gambling	0.875±0.001	17.69 ±0.11%	2.53±0.05%
	Health Insurance	0.924±0.001	9.55 ±0.16%	2.04±0.03%
Unpadded	Dating	0.976±0.0	3.85 ±0.08%	0.33±0.02%
	Gambling	0.964±0.001	6.35 ±0.12%	0.23±0.01%
	Health Insurance	0.959±0.001	6.27 ±0.08%	0.54±0.02%

*The second set of results for padding use the technique of inferring the number of DNS messages per TLS record.

In later tests, we found that AdaBoost works better in some scenarios, so we also used this algorithm. Both Random Forest and AdaBoost are ensemble-based classifiers, which construct multiple weaker learners (that may be other classifiers) and combine their outputs to generate the final prediction [52].

For the task of identifying the webpage category, we found that the Random Forest classifier performs slightly better than AdaBoost. On the other hand, when we attempted to identify an individual page, Random Forest performs worse than AdaBoost. Based on the work in [30], we speculate that AdaBoost might be better suited to processing imbalanced datasets.

We used implementations of Random Forest and AdaBoost from the Scikit-learn libraries [50], leaving most of the default parameters unchanged. The exceptions are that we used 200 *estimators* (weak learners) and set the random state. The random state provides the seed for the random number generator that the classifier uses. Setting the random state to a fixed number yields a constant result for the same dataset.

5 EVALUATION

5.1 Evaluation Methodology

Unless otherwise noted, in our experiments we used 10-fold cross validation [2] and adjusted the validation process slightly when creating test sets as follows. We first used the datasets for top and target webpages to generate folds and train the classifiers. In the testing stage, we then added instances of background pages to the test set. Thus, classifiers are trained on folds containing instances of top and target pages, and tested on folds containing instances of top, target, and background pages. Background pages allow us evaluate how well the classifiers will work when presented with DoT traffic for those websites not observed in the training. Such a scenario would occur if a victim visits random websites that the adversary has not observed in the training. The number of background instances added to a test set is equal to a ratio (0, 5%, 10%, 15%) times the number of target and popular instances in the test set. We chose the ratio of background pages based on the fact that the popularity of websites follows a power law [14, 59], which suggests that most web requests are for a few popular websites so that an adversary can observe most websites that (>80%) a user is likely to visit during the training.

[2]Some pages have less than 10 instances, and we removed these in tests using 10-fold cross validation.

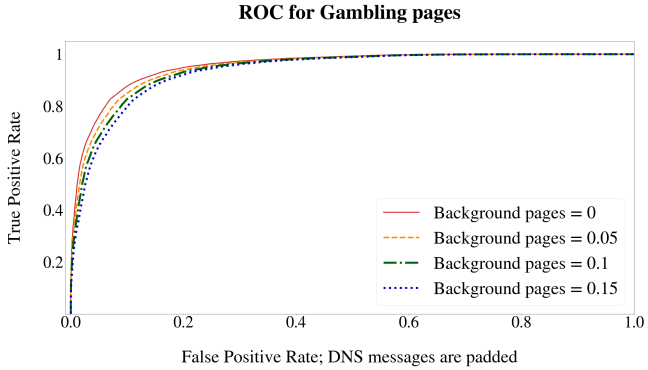


Figure 3: ROC curves for gambling pages.

To measure the classifier's performance, we primarily used FNR, FPR, and F1 score. We also examined the receiver operating characteristic curve (ROC) and equal error rate (EER). We repeated cross validation 10 times, using classifiers built with different seeds for the pseudo-random generator that is used to generate the splits, and computed the standard error of the mean (SEM) of the results.

5.2 Evaluation Results

5.2.1 Classification of Groups of Websites. We first considered the scenario where an adversary attempts to determine if a victim is visiting any of the group of websites of interest. For example, given a sample of DNS traffic generated by the user, the attacker predicts if the user visited any of popular a group of gambling sites.

For this experiment, we used a Random Forest classifier, and explored the effects of including varying levels of background websites in test sets. Table 2 summarizes the results for the base case, where no background websites have been included. When DNS messages are not padded, the FNR is less than 7%, and the FPR is less than 1% for all categories. When DNS messages are padded, the FNR is less than 31%, and the FPR is less than 5%, without using the technique of inferring the number of DNS messages in a TLS record. With the additional processing, the results with padding improve substantially, yielding the FNR of less than 18% and the FPR of less than 3%.

To compare the results with different ratios of background pages, we use EER and AUC. Table 3 shows results for all three target categories under different padding conditions, as the portion of background pages varies. Figure 3 shows results for the gambling category[3]. As expected, the classifier performance drops as the number of background pages in the test set increases; still the AUC remains above 0.92, and the EER under 15% in all tests. With

[3]We used threshold averaging as defined in [28] to average the curves generated by 10 classifiers. The SEM is at least an order of magnitude smaller than the TPR and FPR, so error bars are not shown here.

Table 3: Area under ROC and EER for Experiment 1

		Background: 0		Background: 0.05		Background: 0.1		Background: 0.15	
	Category	AUC	EER	AUC	EER	AUC	EER	AUC	EER
Unpadded	Dating	0.999 ±0.0	1.4 ±0.04%	0.996 ±0.01	2.27 ±0.03%	0.993 ±0.01	3.41 ±0.02%	0.991 ±0.01	4.23 ±0.03%
	Gambling	0.999 ±0.0	1.16 ±0.03%	0.997 ±0.0	2.31 ±0.02%	0.995 ±0.0	3.48 ±0.04%	0.992 ±0.0	4.45 ±0.02%
	Health Insurance	0.998 ±0.0	1.85 ±0.04%	0.997 ±0.0	2.71 ±0.05%	0.994 ±0.0	3.46 ±0.03%	0.992 ±0.01	4.16 ±0.04%
Padded 0	Dating	0.949 ±0.03	12.2 ±0.08%	0.942 ±0.03	13.24 ±0.07%	0.935 ±0.03	14.2 ±0.08%	0.928 ±0.03	14.92 ±0.1%
	Gambling	0.956 ±0.03	11.02 ±0.11%	0.949 ±0.03	12.05 ±0.1%	0.942 ±0.03	12.9 ±0.08%	0.934 ±0.03	13.61 ±0.08%
	Health Insurance	0.974 ±0.01	8.36 ±0.08%	0.970 ±0.02	8.93 ±0.07%	0.964 ±0.02	9.68 ±0.1%	0.961 ±0.02	10.19 ±0.08%
Padded 1*	Dating	0.978 ±0.02	7.45 ±0.05%	0.973 ±0.02	8.45 ±0.06%	0.967 ±0.02	9.31 ±0.04%	0.963 ±0.02	10.14 ±0.07%
	Gambling	0.979 ±0.01	7.3 ±0.07%	0.974 ±0.01	8.38 ±0.06%	0.968 ±0.02	9.36 ±0.05%	0.964 ±0.01	10.12 ±0.04%
	Health Insurance	0.992 ±0.01	4.03 ±0.05%	0.989 ±0.01	4.65 ±0.03%	0.986 ±0.01	5.3 ±0.07%	0.983 ±0.01	5.84 ±0.05%

* The second set of results for padding use the technique of inferring the number of DNS messages per TLS record .

padding, and using the maximum ratio (0.15) of background pages, the classifiers achieve a TPR of at least 99%, with an FPR of at most 42%. Without padding, a TPR over 99% can be achieved, with an FPR of less than 15%, even with the maximum ratio of background pages.

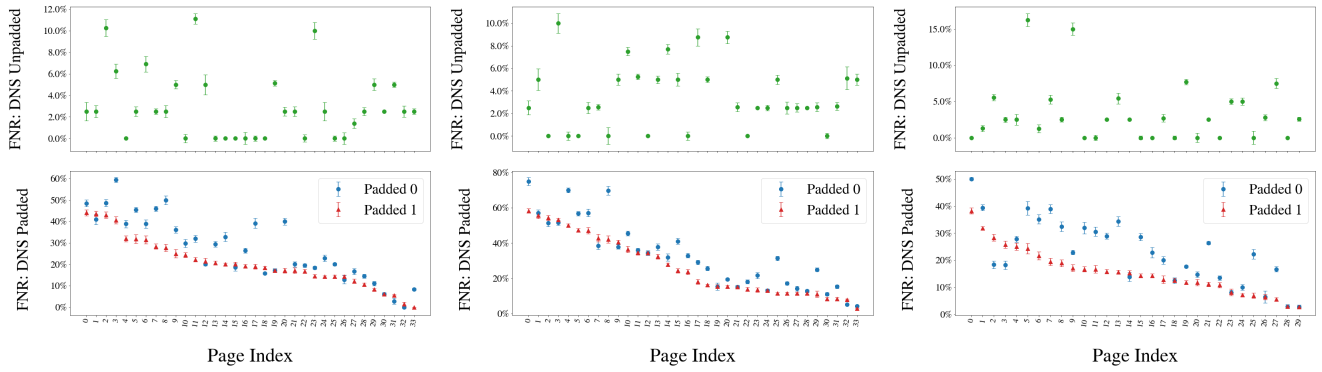
5.2.2 Classification of Individual Websites. We next explored the scenario in which an adversary attempts to determine if the user is visiting a specific website. In these experiments, we used an AdaBoost Classifier. Figure 4 summarizes the FNR for identifying the homepage of individual websites.

In tests where DNS messages are not padded, more than half of the pages have the FNR of less than 3%, and the FPR of less than 0.5%, even when the maximum ratio of background pages is added. The two pages with the highest FNR (16.25% and 15%), both of which are in the health insurance category, were run by the same provider using a shared template, and appeared almost identical. For the other four pages with an FNR of more than 10%, the queries sent were not consistent. Three pages did not generate the same set of queries each time; at least one-fourth of the queries that appeared in any of the DoT traces for one of these pages appeared in less than 50% of its traces. For the other pages, the set of queries was relatively consistent, but several of the queries sent in all instances were sent twice as many times in a few instances. Inspection of the values for the top 10 features of the incorrectly classified instances shows that for most, the number of bytes received was substantially higher than the median for all instances. This finding suggests that the incorrectly classified instances contained extra queries. Some pages without a low FNR have variations in queries; thus, while change in the content and number of queries is the underlying cause in those cases with a high FNR, these variations do not result in a high FNR for all pages.

With padding, the FNR and FPR increase in general, although for some pages, they are still quite low. The median FNR increases from 2.5% without padding to 26% when padding is applied, although the FNR drops to 17% when applying processing to infer the number of DNS messages per TLS record. The results suggest that padding does provide some defense, but it is not adequate for all pages. We may still identify several pages with an FNR or FPR as low as 0.

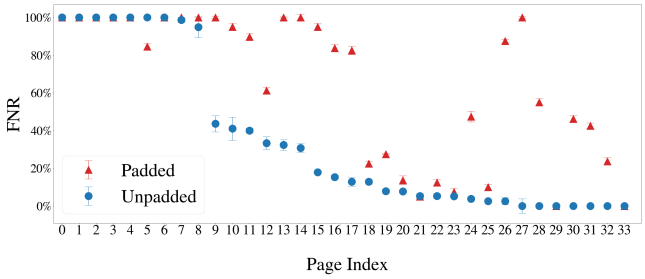
5.2.3 Classification with Time Stability. Over time, changes in a webpage or in the infrastructure supporting the DNS resolutions will affect the profile of the DoT traffic, such that classifiers trained on older data may no longer be able to identify traffic instances. To explore the impact of these changes upon the ability to identify webpages, we conducted additional experiments. Five weeks after the end of the initial data collection from the Surfnet resolver, we collected a second set of DoT traffic for the homepages of the gambling sites and the top sites from the same resolver. In these experiments, we did not use 10-fold cross validation, since we have distinct datasets for training and testing. Instead, we built 10 classifiers by splitting the target and popular pages from the early dataset into training as if doing cross validation. Then, after building the AdaBoost classifier on a training set, we tested on the entire new dataset. We focused on the case without background pages being added. In comparison to the results obtained with cross validation within one dataset, the FNR and FPR increase for most pages, with or without padding. To understand what changes in DNS traffic might cause the increase of incorrect classifications, we examined the number of DNS queries generated when a page was loaded. The changes in the number of queries will affect most other features. For each page, we found the relative change between the two datasets: $\Delta_{NQ} = |NQ_0 - NQ_1|/NQ_0$, where NQ_0 and NQ_1 are the median number of queries in the DoT traces for a page in the first and second datasets, respectively. As Figures **??** and **??** show, those pages with a higher FNR also tend to have higher values of Δ_{NQ}.

A shift in the number of queries indicates a change in the content of a webpage, which may be characterized by the hostnames in DNS queries. Thus, we further evaluated the DNS queries to understand these changes. For each page in the gambling category, we found the queries that appeared in at least 90% of the traces in the earlier datasets (February) but did not appear in at least 90% of the traces in the later datasets, and vice versa. These we call *unstable* queries. We also found queries that appeared in over 90% of the traffic instances

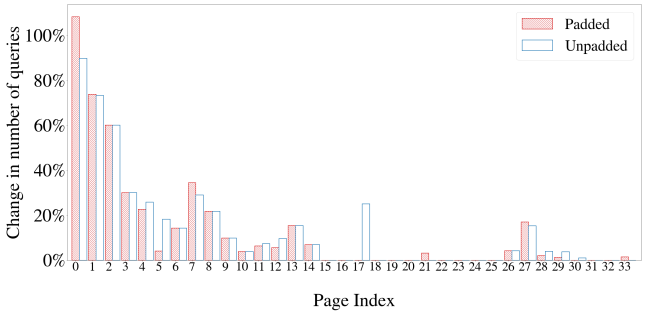


(a) **Dating Pages** (b) **Gambling Pages** (c) **Health Insurance Pages**

Figure 4: FNR for identification of individual webpages. Results for Padded 0 and Padded 1 both used the same set of padded DNS. Additional processing was used for Padded 1 to map TLS record sizes to the number of included DNS messages.



5a **FNR with with 5 weeks between training and test data collection**



5b **Change in queries with 5 weeks between training and test data collection**

for both datasets. These we call *stable* queries.[4] For each query, we used Virus Total [4] to find the Threat Seeker categories [11] for the second level domain (e.g., for gambling.example.com, we would find the category for example.com). We found that the largest category for unstable queries is advertisements; 26% of unstable queries were advertisements compared to only 15% of stable queries. Appendix B includes further details about the categories of the queries.

[4]In each dataset used to calculate the query stability, less than 1% of the traces could not be decrypted. We do not use these when computing stability.

5.2.4 Comparison of Recursive Resolvers. The ability to identify DoT traffic instances may vary with respect to the recursive resolver being used, especially its padding policy. To explore this issue, we ran a set of tests with different resolvers. In our tests, we observed three basic approaches to padding among several recursive resolvers: 1) always pad responses, 2) never pad responses, and 3) pad responses when the client sends the EDNS(0) option, indicating padding (12) [41]. For recursive resolvers that pad, we found two padding approaches: using fixed block size padding (128 or 468 octets) or variable amounts of padding. Table 4 summarizes our findings. Measurements were taken for 22 providers and 36 servers.

We attempted to study the effects of having all DNS messages either padded or unpadded. Thus, the tests discussed in the previous sections used the data from a recursive resolver that padded responses only if the client sent the EDNS(0) option for padding. The other recursive resolvers we chose to test either always padded (Cloudflare), or never padded (Google). As the purpose of this experiment is to compare resolvers, we only explored the base case and did not include background pages in these tests. For the Surfnet results, we used processing to enable us to infer the number of DNS messages per TLS record when messages are padded. We did not apply this processing to the Google or Cloudflare traffic or use the feature *Number of DNS Messages within a TLS record*, since the Google resolvers do not pad, and we observed that Cloudflare does

Table 4: DoT Resolver Padding Strategies.

Pads Responses	Padding (octets)	Providers	Servers
Never	0	14	21
Always	128	1	2
	468	3	3
When prompted	468	2	5
	Variable	2	5

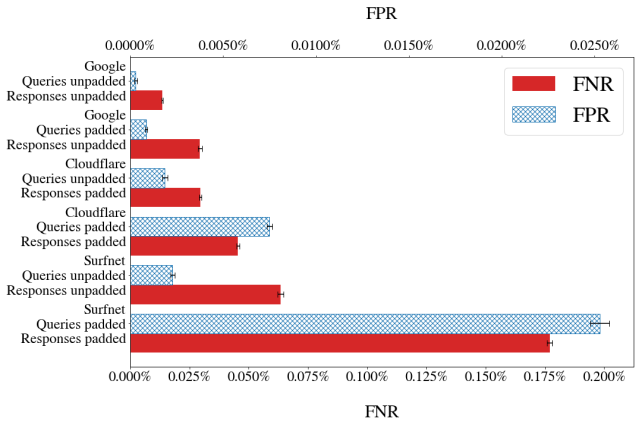


Figure 5: Comparison of recursive resolvers.

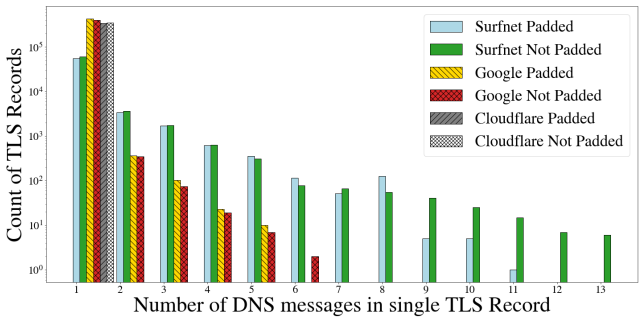


Figure 6: Comparison of recursive resolvers' behaviors of including multiple DNS responses in a single TLS record.

not place multiple messages in one TLS record. We used Random Forest classifiers for each resolver. Figure 5 summarizes the results.

The FNR and FPR are much higher when using the Surfnet resolver and padding DNS messages than when using either of the other two resolvers in any scenario. The low FNR and FPR for the Google resolvers are expected, since responses are never padded. Although the results for Google show that padding queries decreases the classifier's performance, the change is relatively small.

The difference in performance between the Cloudflare and Surfnet resolvers is caused by, at least in part, the padding block size. The Surfnet resolver padded responses to 468 octets, while the Cloudflare resolver padded responses to 128 octets. The difference in performance does not necessarily imply that the larger block size should be used. While 468 octets as the padding block size for responses has been recommended based on one study [42], there is no padding approach appropriate to every scenario. Additional privacy provided by larger block sizes comes at the cost of extra bandwidth and energy consumption. The RFC's providing recommendations for padding policies highlights the need for special considerations of scenarios where resources such as battery life or bandwidth are constrained [42].

Padding is not the only factor contributing to the difference in performance between different resolvers. Another difference between resolver behaviors is the inclusion of multiple DNS responses in a single TLS record. Figure 6 shows the number of TLS records that contain different numbers of DNS responses. The maximum number contained in any message is six for Google, one for Cloudflare, and 13 for Surfnet resolvers. As the inclusion of multiple messages per record undermines the stability of features, such as the number and size of responses, it will decrease the ability to use these features for classification. Thus, the larger number of messages per record in the Surfnet traffic also helps to explain the difference in performance.

5.3 Feature Importance

Finally, we would like to understand how much individual features or groups of features contribute to the ability to classify webpages. The Random Forest and AdaBoost classifiers in Scikit-learn both assign feature importance scores based Gini impurity.

To find importance, we selected the classifiers used in the scenarios described in Sections 5.2.1 and 5.2.2. For each category or page, we first found the mean score assigned by all of the learners built. We then found the median and mean scores over all categories or pages. Appendix C provides the list of the top 10 features for the experiments in Sections 5.2.1 and 5.2.2.

As expected, the query and response lengths are the most important features when DNS messages are not padded, while when messages are padded, the classifiers rely on more features. In particular, the intervals between responses followed by queries or vice versa (RQ/QR interval) is important for individual pages. Information about the overall amount of data transmitted is a key feature, more so for the category-level classification than per page.

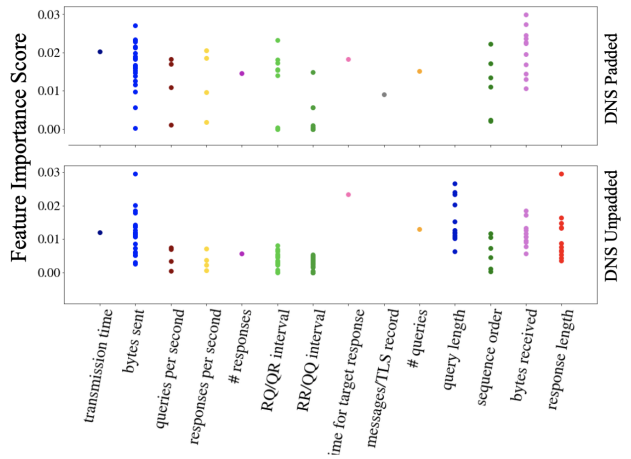
In addition to considering the top 10 features, we examined the overall impact of all features. Figures **??** and **??** show the median importance assigned to each feature, with features plotted in groups. These graphs highlight that features related to query and response lengths and bytes exchanged, tend to dominate when DNS messages are not padded. When messages are padded, the model tends to rely more evenly on a larger set of features.

6 DISCUSSION

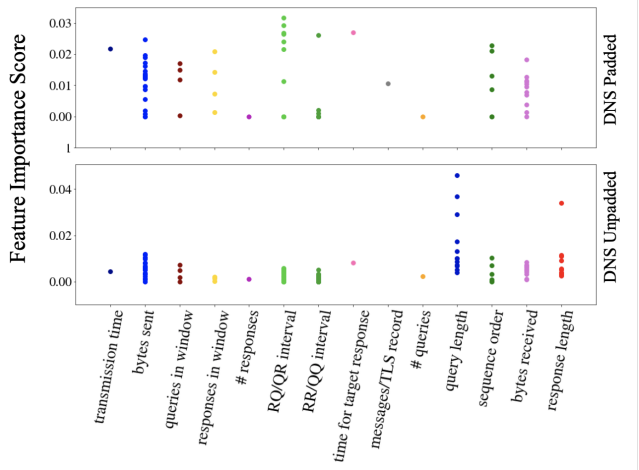
6.1 Limitations and Future Work

The results of this study can be recognized as an upper-bound on the effectiveness of the website fingerprinting attack against DoT traffic, due to its three major limitations. (1) Caching effects at the client browser are not considered. (2) Our experiments are not conducted in a true open-world scenario, and previous works have shown that it is more challenging to achieve successful website fingerprinting in more realistic, true open-world scenarios [37, 48]. (3) Our experiments explore a relatively narrow set of parameters, and how the results can be generalized to other settings requires further investigation. Below we elaborate on these limitations.

The primary concern regarding our attack is how the browser cache on the victim side impacts our fingerprinting method. In a real-world scenario, DNS requests related to a webpage visit would vary based on which domain resolutions are already cached by the user's browser or operating system. The effects of caching and varying page content would degrade the attacker's ability to identify a user's visits to a target webpage. While such an impact needs



7a **Feature importance for per-category classification**



7b **Feature importance for per-page classification**

to be assessed, we argue that it should not be significant enough to prevent DoT information leakage, mainly due to the following two observations. (1) The largest common TTL for a DNS record is 48 hours, but the most common TTLs are currently set to much smaller values such as one hour [45]. (2) There is an increasing usage of privacy-preserving techniques to obfuscate users' network traffic.

For example, to eliminate side-channel attacks exploiting a shared cache in browsers, major browsers such as Safari and Chrome either have implemented or are implementing cache partitioning [2, 3, 54]. The policy for partitioning (e.g., per domain, per URL, or a combination of factors) will determine the probability that the browser cache has been reset when a user visits a page. In all cases, however, partitioning will increase this probability and the chance that our attack succeeds. Similarly, users who employ incognito mode will generate DNS traffic that tends to be generic.

Thus, the empty-cache scenario we study could often occur in many real-world scenarios.

While the attack scenario modeled is not unrealistic, we note that the results presented here represent an early exploration of this area, and we would like to emphasize that similar limitations are embodied in other research focusing on the privacy of network traffic (e.g., Tor traffic). Many of the prior works also considered scenarios without caching [1, 32, 49, 63]. Those that did explore caching suggested that the attacks were still reasonably successful with warmed caches [19, 33]. Some early works focused on closed-world scenarios [1, 33], and most considered an individual page as representative of a site. A few works explored fingerprinting websites [19, 48]. While the feasibility of these attacks in real-world scenarios has not been fully understood, they did motivate Tor developers to add features to combat website fingerprinting [51]. In a similar fashion, we believe our research could motivate new designs of DoT to defend against website fingerprinting.

We plan to address the limitations of this work in the future primarily by exploring caching, dynamic content, and open world scenarios, but also by expanding the parameters of the experiments and considering additional threat models. Specifically, our future work would consider additional resolvers, vantage points, and browsers, including regular (not headless) browsers.

For this study, to understand the impact of using a headless browser, we did a simple test to compare DNS queries generated by a headless browser to those of a browser with a GUI. Details of the test are included in Appendix D. We found that the numbers of queries sent on both cases are comparable, and we expect results to be similar when using a browser with a GUI. Other parameters to consider involve the classifiers. We may further improve results by using other classifiers or tuning hyper-parameters. Finally, threat models that allow an attacker additional capabilities would gain better results. In this work, we have intentionally excluded HTTP traffic from the fingerprinting in order to better understand the dynamics of DoT. This approach models an attacker on the path between the user and recursive resolver but outside the user's local network. If we assume the attacker has access to other user traffic, we expect the ability to fingerprint websites would increase. Our future work could explore this scenario further and evaluate the contribution of DoT traffic to other fingerprinting attacks. Also, while we only consider a passive attacker, an active attacker may be able to create conditions that will increase the chances of a successful fingerprinting attack. For example, an attacker can issue DNS requests to a resolver shared with the victim so that the resolver's cache can be impacted in a way favorable to the attacker. We leave a systematic evaluation on these factors as our future work.

6.2 Defense

Based on our results, padding as a baseline defense does impact the effectiveness of adversarial traffic analysis, but other methods might also be needed to fully mitigate the threat. We observed that the padding block size is important, with a larger block size disguising the traffic better. However, as discussed in Section 5.2.4, the choice of padding size must be balanced against other factors, such as overhead.

While padding is effective at preventing recognition of some pages, other methods are necessary to prevent identification in all scenarios. Additional methods of defending against traffic analysis could include disguising the amount of data exchanged, and ensuring uniform time intervals. For the former defense, the stub should send the same number of queries and responses for each page, or each page within a group. Groups would comprise pages that normally send a similar number of queries. Variations of such an approach have been proposed for defenses against website fingerprinting attacks on Tor [18, 26, 61]. To disguise timing, extraneous packets may be introduced to maintain desired intervals, similar to the approach proposed in [38]. Delaying packets to maintain a desired profile may also be an approach to disguise timing, but would not be recommended, given the demands on DNS for low latency.

7 RELATED WORK

There are two categories of related works. First, we discuss previous works that have specifically explored DNS privacy protection or traffic analysis of DNS protection mechanisms. Then we describe previous works in the area of website fingerprinting. Since the objective of website fingerprinting is similar to that of our threat model, our work is based on several of the concepts or approaches they proposed.

7.1 DNS Privacy

Some early proposals to address DNS privacy rely on obscuring DNS queries largely by using noise, i.e., the inclusion of extra queries to hide the true interests of a user. Variations on this approach, called range queries, have been proposed in several prior works [20, 29, 31, 40, 64, 65]. In most of the proposed designs, DNS messages are still transmitted in plain text. Thus, while some of the proposed attack scenarios are similar (i.e., privacy leaks via eavesdropping), their approaches are fundamentally different from those we have explored for DoT. The works that consider adversarial approaches closest to ours are the two that suggest using mixers (e.g., Tor) for DNS exchanges [29, 31]. Both works leverage the possibility that an adversary could use traffic analysis to guess a user's activity. Garcia-Alfaro et al. [31] discussed attacks using the correlation between traffic at different nodes, rather than fingerprinting encrypted traffic. The latter subject is addressed by Federrath et al. [29], who recommended padding.

While the aforementioned works focus on the protection of DNS privacy, there are other proposed approaches that one might use to undermine protections. In [60], the authors explored the ability to use plain text DNS to identify which websites a user accesses. The technical aspects of the attack are significantly different from ours, given that the attacker has access to the plain text of DNS queries. In a study more closely related to ours, Shulman examined the potential privacy of encrypted DNS [55]. Shulman's experiments explore issues of privacy, compatibility, and increased overhead related to the use of encrypted DNS. However, the adversary's goal in this work is to determine the identity of the nameserver being contacted. In [55], Shulman noted that since nameservers often host multiple zones, knowing the target nameserver does not necessarily provide much information about what content the victim requested. Thus, the information available to an adversary and the features used to perform identification in Shulman's work are very different from those we consider. In our threat model, the adversary does not have access to information about any nameservers contacted except the recursive resolver. Moreover, in Shulman's study, defense mechanisms such as padding are not considered.

7.2 Website Fingerprinting

Our threat model is similar to those scenarios presented in the studies on website fingerprinting for identifying which websites a user has visited based on encrypted HTTP traffic.

Liberatore and Levine [1] used two classification methods, Naive Bayes and one based on Jaccard's coefficient, representing traffic instances to capture the size and direction of packets. Herrmann et al. [33] used a Multinomial Naive Bayes classifier with various text-mining transformations to characterize packet frequency and size as features but timing and order were not considered. Panchenko et al. [49] developed a richer feature set and used an SVM for website fingerprinting in onion routing-based anonymization networks. Dyer et al. [26] evaluated multiple algorithms with Panchenko's augmented features, and explored the use of a Naive Bayes classifier with their own feature set, which uses information regarding timing, size, direction, and bandwidth. Cai et al. [19] proposed a new method for website fingerprinting by leveraging edit distance between traces to create a kernel matrix for SVM. Wang and Goldberg [63] developed a similar approach of using SVM, with two different methods for calculating edit distance. In both works, traces are represented as vectors of integers indicating packet sizes. Wang et al. [62] proposed a new approach to deriving a more complex feature set, and used these features to calculate the distance between instances for classification by utilizing the K-Nearest Neighbor algorithm. In [48], Panchenko et al. proposed an alternative approach to capturing much of the same information without requiring manual selection, and again they tested the proposed features by using SVM.

The work perhaps most relevant to ours is that of Hayes and Danezis [32]. They used a list of features that explicitly capture information about the traffic and what the traffic represents (loading a web page). This work also uses Random Forest, but the output of the classifier is not used directly for prediction. In addition, they did not test on the encrypted DNS traffic.

8 CONCLUSION

In this work, we have conducted an investigation on the information leakage of DNS over TLS (DoT) through traffic analysis. We have developed a novel DoT fingerprinting method, and then we have demonstrated that this proposed approach is effective for identifying these websites a user visits. More specifically, we have found that when DNS messages are not padded, we are able to identify whether a user has visited one of a group of websites in sensitive categories (health insurance, dating, and gambling) with the FNR and FPR of less than 7% and 5%, respectively. Further, we are able to identify whether the user has visited specific websites with the FNR of less than 17% and the FPR of less than 0.5%, respectively. Even when DNS messages are padded, our proposed DoT fingerprinting is still able to achieve relatively low FNR and FPR, even

zero for some individual websites. Since the security vulnerability of DoT has not yet been fully investigated, we expect that our results represent a baseline for the classification of DoT traffic. More importantly, our findings will help future research to develop more effective defense mechanisms against traffic analysis of DoT, and help public DNS resolvers to use DoT for DNS communications in a more secure manner.

9 ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1247394. This work was also partially supported by the National Science Foundation Grants CNS-1618117 and DGE-1821744. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] [n.d.]. Inferring the source of encrypted HTTP connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, publisher = ACM, author = Liberatore, Marc and Levine, Brian N., year = 2006,*. Alexandria, Virginia, USA.
[2] [n.d.]. Mass XS-Search using Cache Attack. https://terjanq.github.io/Bug-Bounty/Google/cache-attack-06jd2d2mz2r0/index.html
[3] [n.d.]. Optionally partition cache to prevent using cache for tracking. Optionallypartitioncachetopreventusingcachefortracking
[4] [n.d.]. VIRUSTOTAL. https://www.virustotal.com/gui/home/upload
[5] 2017. Alexa Top Sites. https://docs.aws.amazon.com/AlexaTopSites/latest/index.html
[6] 2018. About Stubby. https://github.com/getdnsapi/stubby
[7] 2019. DNSSEC Validation Rate by country. https://stats.labs.apnic.net/dnssec
[8] 2019. Estimating IPv6 & DNSSEC Deployment SnapShots. https://fedv6-deployment.antd.nist.gov/snap-all.html
[9] 2019. Firefox Extended Support Release. https://www.mozilla.org/en-US/firefox/organizations/
[10] 2019. Headless mode. https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Headless_mode
[11] 2019. Master Database URL Categories. https://www.forcepoint.com/product/feature/master-database-url-categories
[12] 2019. TCPDUMP and LIBPCAP. https://www.tcpdump.org
[13] 2019. The top 500 sites on the web. https://www.alexa.com/topsites/category
[14] 2019. What's going on with my Alexa Rank? https://support.alexa.com/hc/en-us/articles/200449614-What-s-going-on-with-my-Alexa-Rank-
[15] A. Bianco, G. Mardente, M. Mellia, M. Munafo, and L. Muscariello. 2009. Web User-Session Inference by Means of Clustering Techniques. *IEEE/ACM Transactions on Networking* 17, 2 (April 2009), 405–416.
[16] S Bortzmeyer. 2015. *DNS Privacy Considerations.* RFC 7626. RFC Editor. 1–17 pages. https://tools.ietf.org/html/rfc7626
[17] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar. 2011. Understanding Website Complexity: Measurements, Metrics, and Implications. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC '11).* ACM, 313–328.
[18] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES '14).* ACM, 121–130.
[19] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12).* Raleigh, North Carolina, USA.
[20] Sergio Castillo-Perez and Joaquin Garcia-Alfaro. 2008. Anonymous Resolution of DNS Queries. In *On the Move to Meaningful Internet Systems: OTM 2008.* 987–1000.
[21] Manu Chantra. 2018. *DNS over TLS_ Encrypting DNS end-to-end - Facebook Code.pdf.*
[22] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2003. Towards Measuring Anonymity. In *Privacy Enhancing Technologies (PET'03).* 54–68.
[23] John Dickinson and Sara Dickinson. 2019. DNS Privacy Implementation Status. https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Implementation+Status
[24] Sara Dickinson. 2019. Windows installer for Stubby. https://dnsprivacy.org/wiki/display/DP/Windows+installer+for+Stubby
[25] Chris Duckett. 2019. Google Public DNS gets DNS-over-TLS treatment. https://www.zdnet.com/article/google-public-dns-gets-dns-over-tls-treatment/
[26] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *2012 IEEE Symposium on Security and Privacy.* IEEE, San Francisco, CA, USA, 332–346.
[27] S. Farrel and H. Tschofenig. 2014. *Pervasive Monitoring Is an Attack.* RFC 7258. RFC Editor. 1–6 pages. https://tools.ietf.org/pdf/rfc7258.pdf
[28] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (June 2006), 861–874.
[29] Hannes Federrath, Karl-Peter Fuchs, Dominik Herrmann, and Christopher Piosecny. 2011. Privacy-Preserving DNS: Analysis of Broadcast, Range Queries and Mix-Based Protection Methods. In *ESORICS 2011.* 665–683.
[30] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 4 (July 2012), 463–484.
[31] Joaquin Garcia-Alfaro, Michel Barbeau, and Evangelos Kranakis. 2009. Evaluation of Anonymized ONS Queries. *arXiv:0911.4313 [cs]* (Nov. 2009). arXiv: 0911.4313.
[32] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium (USENIX Security 16).* USENIX Association, Austin, TX, 1187–1203.
[33] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW '09).* Chicago, Illinois, USA.
[34] P. Hoffman and P. McManus. 2018. *DNS Queries over HTTPS (DoH).* Technical Report RFC8484. RFC Editor. RFC8484 pages. https://www.rfc-editor.org/info/rfc8484
[35] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS).* RFC 7858. RFC Editor. 1–19 pages. https://tools.ietf.org/html/rfc7858
[36] Jaeyeon Jung, E. Sit, H. Balakrishnan, and R. Morris. 2002. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on Networking* 10, 5 (Oct 2002), 589–603.
[37] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14).* ACM, 263–274.
[38] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an Efficient Website Fingerprinting Defense. In *ESORICS 2016.* 27–46.
[39] Shuai Li, Huajun Guo, and Nicholas Hopper. 2018. Measuring Information Leakage in Website Fingerprinting Attacks and Defenses. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18).* ACM, 1977–1992.
[40] Yanbin Lu and Gene Tsudik. 2009. Towards Plugging Privacy Leaks in Domain Name System. *arXiv:0910.2472 [cs]* (Oct. 2009). arXiv: 0910.2472.
[41] A. Mayrhofer. 2016. *The EDNS(0) Padding Option.* Technical Report RFC7830. RFC Editor. RFC7830 pages. https://www.rfc-editor.org/info/rfc7830
[42] A. Mayrhofer. 2018. *Padding Policies for Extension Mechanisms for DNS (EDNS(0)).* RFC 8467. RFC Editor. 1–9 pages. https://tools.ietf.org/pdf/rfc8467
[43] P Mockapetris. 1987. *Domain Names - Concepts and Facilities.* Technical Report RFC 1034. RFC Editor. 1– 55 pages. https://www.rfc-editor.org/rfc/pdfrfc/rfc1034.txt.pdf
[44] P.V. Mockapetris. 1987. *Domain names - implementation and specification.* Technical Report RFC1035. RFC Editor. 1–55 pages. https://www.rfc-editor.org/info/rfc1035
[45] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. 2019. Cache Me If You Can: Effects of DNS Time-to-Live (extended). In *Proceedings of the ACM Internet Measurement Conference.* ACM, Amsterdam, the Netherlands.
[46] Baiju Muthukadan. 2018. Selenium with Python. https://selenium-python.readthedocs.io/#
[47] B. Newton, K. Jeffay, and J. Aikat. 2013. The Continued Evolution of Web Traffic. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems.* 80–89.
[48] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. 2016. Website Fingerprinting at Internet Scale. In *Proceedings 2016 Network and Distributed System Security Symposium.* Internet Society, San Diego, CA.
[49] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society (WPES '11).* Chicago, Illinois, USA.
[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[51] Mike Perry. 2011. Experimental Defense for Website Traffic Fingerprinting. https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting
[52] R. Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine* 6, 3 (2006), 21–45.
[53] Daniel Roethlisberger. 2018. SSLsplit - transparent SSL/TLS interception. https://www.roe.ch/SSLsplit
[54] Sharma Shivani and Josh Karlin. 2019. *HTTP Cache Threat Model - Partitioning the cache.* Technical Report.
[55] Haya Shulman. 2014. Pretty Bad Privacy: Pitfalls of DNS Encryption. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES '14).* ACM, 191–200.
[56] Haya Shulman. 2015. Pretty Bad Privacy Pitfalls of DNS Encryption. https://www.ietf.org/proceedings/93/slides/slides-93-irtfopen-1.pdf
[57] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. 2014. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14).* ACM, 449–460.
[58] Luca Vassio, Idilio Drago, Marco Mellia, Zied Ben Houidi, and Mohamed Lamine Lamali. 2018. You, the Web, and Your Device: Longitudinal Characterization of Browsing Habits. *ACM Transactions on Web* 12, 4, Article 24 (Sept. 2018), 24:1–24:30 pages.
[59] Juan Vera, Soumen Chakrabarti, and Alan Frieze. 2006. The Influence of Search Engines on Preferential Attachment. *Internet Mathematics* 3, 3 (1 1 2006).
[60] Kai Wang, Liyun Chen, and Xingkai Chen. 2019. Website Fingerprinting Attack Method Based on DNS Resolution Sequence. In *International Conference on Applications and Techniques in Cyber Security and Intelligence 2018.* Vol. 842. 1227–1233.
[61] Tao Wang. 2013. Comparing Website Fingerprinting Attacks and Defenses.
[62] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium.* USENIEX Association, San Diego, CA.
[63] T. Wang and I. Goldberg. 2013. Improved website fingerprinting on TOR. In *Proceedings of ACM Conference on Computer and Communications Security (CCS'13).* Berlin, Germany.
[64] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. 2007. Analysis of Privacy Disclosure in DNS Query. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07).* IEEE, Seoul, Korea, 952–957.
[65] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. 2007. Two-Servers PIR Based DNS Query Scheme with Privacy-Preserving. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007).* IEEE, Korea, 299–302.

A DNS OVER HTTPS

We ran a test to explore the effectiveness of the proposed analysis methods against DNS over HTTPS (DoH). The major difference between the DoH and DoT experimental setups was that we did not use `SSLSplit` or `Stubby` for DoH. Since Firefox supports DNS over HTTPS, and provides a method to obtain the key for decrypting the traffic, these tools were not needed. As Firefox does not provide settings to configure padding, we did not perform different tests for different padding configurations, and we used a recursive resolver provided by Cloudflare.

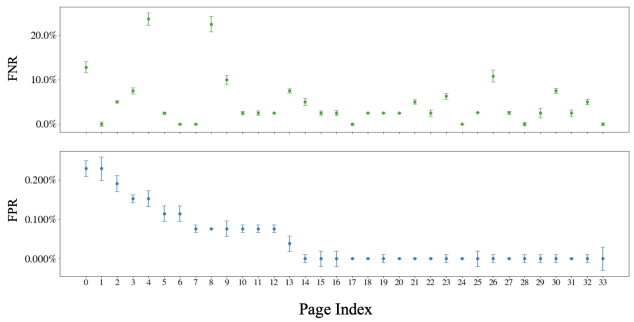


Figure 7: DNS over HTTPS classification.

Table 5: Categorization of Queries

Category	Percentcage of Queries	
	Stable	Unstable
Advertisements	15.36	26.4
Web analytics	10.72	20.0
Information technology	21.16	16.0
Business and economy	6.09	8.0
Uncategorized	2.32	5.6
Gambling	23.48	4.8
Streaming media	0.29	4.8
Search engines and portals	4.64	3.2
Social web - Facebook	0.87	2.4
Social web - Twitter	2.61	1.6
Social web - Youtube	0.58	1.6
Travel	1.45	1.6
Web infrastructure	2.03	1.6
Games	2.32	0.8
Text and media messaging	0.00	0.8
Web hosting	0.58	0.8
Illegal or questionable	0.29	0.0
Parked domain	0.58	0.0
Hosted business applications	0.58	0.0
Application & sw download	0.29	0.0
Sports	0.58	0.0
Gambling; potentially unwanted sw.	0.58	0.0
Content delivery networks	1.45	0.0
Web and email marketing	0.29	0.0
Web collaboration	0.87	0.0

After collecting the data, we performed one basic test: classification for individual pages in the gambling category. We excluded features that were tuned specifically for DoT traffic: time to receive first N bytes, and features involving the number of DNS messages per TLS record. Figure 7 shows the FNR and FPR. These results are comparable to those for DoT, suggesting that the analysis proposed in this paper would perform well on DoH traffic. The performance for classification of DoH traffic instances may also be improved through tuning specifically for DoH.

B QUERY CATEGORIZATION

As described in Section 5.2.3, we define an unstable query as one that appears in 90% of DoT traces for a page in one dataset, but less than 90% of the traces for that page in a second dataset. A stable query is one that appears in at least 90% of the traces for a page in both datasets. We did not find stability for queries related to Firefox services and the EC2 network, as well as three sets of queries where a different alphanumeric string is appended to a single second-level domain in all or almost all instances. Queries in this last group appear as unstable queries, even though they represent consistent query patterns. We used Virus Total to categorize the second-level domains of the queries. The distribution of queries over different categories is shown in Table 5. The biggest difference in the distributions between stable and unstable queries is in the advertisements category. Advertisements comprise a substantially larger percentage of unstable queries than that of stable queries

Table 6: Top 10 features over all classifiers built to identify individual pages.

	DNS Padded			DNS Unpadded		
Rank	Feature	Mean	Median	Feature	Mean	Median
0	RQ interval max	0.0321	0.03	qry length min	0.0666	0.05
1	RQ interval decile 0.9	0.0371	0.03	qry length mean	0.0527	0.04
2	QQ interval max	0.0297	0.03	resp length min	0.0494	0.03
3	RQ interval decile 0.8	0.0301	0.03	qry length decile 0.1	0.0459	0.03
4	time for target response	0.0280	0.03	qry length decile 0.4	0.0227	0.02
5	RQ interval decile 0.7	0.0299	0.03	bytes sent decile 0.8	0.0081	0.01
6	r in window 1 sec max	0.0267	0.02	resp length decile 0.3	0.0138	0.01
7	bytes sent median	0.0229	0.02	RQ interval decile 0.9	0.0078	0.01
8	qry burst order mean	0.0248	0.02	bytes received median	0.0104	0.01
9	transmission time	0.0273	0.02	bytes received max	0.0143	0.01

Table 7: Top 10 features for classifiers built to identify groups of pages.

	DNS Padded			DNS Unpadded		
Rank	Feature	Mean	Median	Feature	Mean	Median
0	bytes received mean	0.0302	0.0299	resp length min	0.0310	0.0296
1	bytes received decile 0.8	0.0268	0.0273	ratio bytes received/bytes sent median	0.0276	0.0295
2	bytes sent mean	0.0274	0.0270	qry length mean	0.0245	0.0265
3	bytes received decile 0.9	0.0249	0.0245	qry length min	0.0245	0.0241
4	bytes received median	0.0241	0.0236	qry length decile 0.1	0.0282	0.0234
5	bytes sent median	0.0234	0.0234	time for target response	0.0212	0.0233
6	RQ interval max	0.0215	0.0233	qry length decile 0.9	0.0210	0.0202
7	bytes sent decile 0.8	0.0233	0.0233	bytes sent max	0.0200	0.0200
8	bytes sent decile 0.9	0.0210	0.0229	ratio bytes received/bytes sent mean	0.0195	0.0185
9	bytes received decile 0.7	0.0225	0.0226	bytes received max	0.0189	0.0184

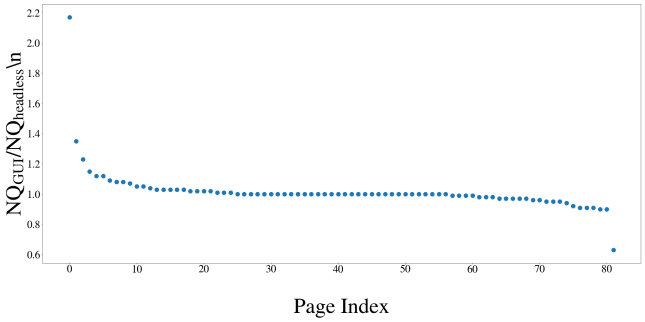


Figure 8: Comparison of the number of queries sent to load pages with a headless or normal browser.

(26% vs. 15%). Streaming media is also noticeably higher for unstable queries than for stable queries. We found that all unstable queries in the streaming media category came from instances for a single domain, and were all related to the same video sharing service.

C TOP FEATURES

Random Forest and AdaBoost provide estimates of feature importance based on the Gini impurity. Random Forest gives the average of the importance scores provided by the estimators, while AdaBoost returns a weighted average. Tables 6 and 7 show the top 10 features for the tests described in sections 5.2.1 and 5.2.2. Query and response lengths make up most of the top features when DNS messages are not padded. Further, the highest scores assigned to features related to length are greater than the top scores of features when padding is used. This suggests that when messages are not padded, the classifiers rely heavily on individual message lengths, but when messages are padded, the classifiers rely more on a wider set of features.

D HEADLESS VS. NORMAL BROWSER

In our tests, we use one browser (Firefox) in headless mode. We consider the degree to which the results obtained using a headless browser are representative of the ability to fingerprint the DNS traffic of a real user. Although the general operation between a headless browser and one that uses a GUI is the same, we expect some differences in timing and possibly content of DNS traffic. To understand the impact of such browser settings, we did a simple test to compare DNS queries generated by a headless browser to those of a browser with a GUI. To evaluate the difference between the DoT traffic generated by a headless browser versus a browser with a GUI, we conducted an experiment where we fetched pages using both configurations. We conducted this experiment using a VM running Ubuntu 16.04.5 on a local machine and used the same basic test setup as described in Section 4. We used pages in the target group and loaded each page twice: once in headless mode, and once with the GUI enabled. We then compared how many queries were sent in each mode. After filtering for failed loads, we had 82 pairs of DoT traffic. For each pair, we found the number of queries sent from

the browser in headless mode ($NQ_{headless}$) and that of when using a GUI (NQ_{gui}). We then found the ratio ($NQ_{gui}/NQ_{headless}$). As Figure 8 shows, for all but 9 of the pairs, $NQ_{gui}/NQ_{headless}$ was between 0.9 and 1.1.

For the one page with a ratio over 2, the difference in queries appears to be related to advertisements. We already expected that for some pages, at least, aspects of the browser, user history, or location could affect the DNS traffic. Future work will explore these aspects further. For the current study, the fact that the number of queries in different browser modes is similar overall is sufficient to justify the use of the headless browser for our tests.