Does Domain Name Encryption Increase Users' Privacy?

Martino Trevisan,
Francesca Soro, Marco Mellia
Politecnico di Torino, Italy
first.last@polito.it

Idilio Drago
University of Turin, Italy
idilio.drago@unito.it

Ricardo Morla
University of Porto, Portugal
ricardo.morla@fe.up.pt

ABSTRACT

Knowing domain names associated with traffic allows eavesdroppers to profile users without accessing packet payloads. Encrypting domain names transiting the network is, therefore, a key step to increase network confidentiality. Latest efforts include encrypting the TLS Server Name Indication (eSNI extension) and encrypting DNS traffic, with DNS over HTTPS (DoH) representing a prominent proposal. In this paper, we show that an attacker able to observe users' traffic relying on plain-text DNS can uncover the domain names of users relying on eSNI or DoH. By relying on large-scale network traces, we show that simplistic features and off-the-shelf machine learning models are sufficient to achieve surprisingly high precision and recall when recovering encrypted domain names. The triviality of the attack calls for further actions to protect privacy, in particular considering transient scenarios in which only a fraction of users will adopt these new privacy-enhancing technologies.

CCS CONCEPTS

• **Networks** → **Network privacy and anonymity**; **Web protocol security**; **Network measurement**.

KEYWORDS

Privacy, Encryption, Network Traffic, Eavesdropping

1 INTRODUCTION

As a reaction to the ease of passive monitoring of network traffic [6], the Internet has massively moved towards encrypted protocols [1, 19]. The simplicity of new methods to obtain and renew server certificates, the improvements on web protocols that now support or mandate the use of TLS, and the stricter security policies in web browsers have pushed popular services into secure deployments [5]. These deployments are robust against in-network monitors and eavesdroppers, thus overall increasing privacy and confidentiality.

The most notable omission in the initial efforts to enhance confidentiality has been the *domain names*, which are still exchanged in plain-text in different situations. Notably, DNS requests and responses carry Fully Qualified Domain Names (FQDNs) in clear in and TLS includes the non-encrypted Server Name Indication (SNI) field to allow clients and servers to negotiate the certificate used in each connection. This step is required when multiple websites are hosted behind a single server IP address, which is common in Content Delivery Networks (CDNs). The presence of plain-text domain names in the traffic gives eavesdroppers access to rich information [3]. This aspect gives the eavesdropper the possibility to obtain a dataset of traffic flow statistics, having as labels their associated domain names. Eavesdroppers can therefore build lists of the websites visited by users, with clear implications for users' privacy [29]. Recent proposals aim at fixing the issue by encrypting DNS traffic, e.g., running DNS over HTTPS (DoH) [12] and by encrypting the TLS SNI [22], i.e., deploying eSNI. In such a scenario, eavesdroppers would be left with only IP addresses as sensitive information.

However, DoH and eSNI deployments are not widespread. Whereas one can expect an increasing trend in their adoption, a significant portion of users will likely not start using these technologies any time soon. This partial deployment raises questions on whether the privacy of users adopting the privacy-enhancing technologies are still protected if *some other users* remain unprotected. Moreover, even with a complete shift towards DoH and eSNI, it would still be possible for an attacker to build a labeled dataset by actively visiting the set of websites of interest.

In this paper, we study to what extent the privacy gains with the deployment of DoH and eSNI can be reverted. More concretely, we evaluate whether an attacker observing the traffic of some users exposing domain names they contact can uncover the domain names contacted by the remaining users – i.e., breaking the protection intended by DoH and eSNI. We assume that the attacker can refer to different sources to build a dataset of flow level statistics labeled with their domain names: (i) eavesdropping traffic of users relying on plain-text protocols; (ii) harvesting corporate/private traffic and DNS logs; or (iii) running active experiments. Here, we rely on large-scale traces collected in a production network. We randomly split users into two sets, for training and testing. We label flows with their associated domain names, focusing on TLS traffic only. We observe that only a negligible percentage of flows are currently protected by eSNI, thus allowing us to extract reliable names for the vast majority of the TLS flows. Using traffic observed in the training dataset we build models that can uncover the domain names in the testing dataset.

We show that an off-the-shelf machine learning approach is sufficient to execute the attack. The source code is openly available and we can share the dataset upon request. [1] We extract traffic features for the most popular domain names observed in the training set, such as flow duration, byte counters as well as packet sizes and packet inter-arrival times. The machine learning models yield surprisingly good results, delivering F1-scores over 0.8 for 80% of the evaluated domain names. We analyze the performance of the classifier under different conditions. We conclude that simple statistics of the first few IP packets of the connections (i.e., packet lengths and inter-arrival times) already provide sufficient information to seed the attack. The triviality of the attack calls for further actions to protect domain names.

[1] https://smartdata.polito.it/does-domain-name-encryption-increase-users-privacy/

2 RELATED WORK

Extensive prior work addresses the problem of inferring users' activity through packet eavesdropping. Already in 2002 authors of [11] showed that it is possible to understand which website a user is visiting by simply matching the volume of downloaded data against a set of known web pages. More recently, other works targeted the same problem from diverse perspectives, such as dynamic content [24], multi-tab browsing [10] or different underlying encrypted protocols like ToR [2], HTTPS [9] and 802.11 [7]. Other works focus on websites belonging to diverse categories, like politics [15] and health [17], and make use of different machine learning techniques, such as deep neural networks [4, 23]. All these works rely on datasets collected in a controlled environment, where diversity is intrinsically limited. We follow a methodology similar to those works. However, we illustrate the feasibility of such attacks in a new scenario, i.e., against hostnames protected by eSNI and DoH. Moreover, we use a passive dataset collected in a real environment, where thousands of domains are contacted by thousands of users.

Different website fingerprinting techniques are tested on a TOR network in [26, 30]. Other works proposed approaches for traffic classification using side-channels containing valuable information, such as plain-text DNS transactions [3, 18, 21]. Recent efforts toward privacy propose encrypted DNS as a solution to limit the applicability of these approaches [12, 14]. However, authors of [16] already highlighted some critical aspects of many DoH deployments. Moreover, authors of [13] recently pointed out how the packet eavesdropping techniques used for website fingerprinting can be applied to encrypted DNS too, using metadata like packet sizes and timing. Authors of [25] achieve similar results by analyzing DNS-over-HTTPS (DoH) using features related to the size, timing, and ordering of DNS packets. In this paper, we target the scenario where DNS traffic is encrypted, but differently from these works we fingerprint directly TCP connections where domain information is hidden by DoH and eSNI.

3 METHODOLOGY

3.1 Threat model

The considered threat model is depicted in Figure 1. We assume eavesdroppers monitor traffic of a set of users. Some of these users utilize only fully encrypted protocols to hide the domain names of servers they contact. In such scenario, attackers cannot obtain any insights from network packets directly, beyond the source and destination IP addresses. Attackers want to know whether these users access to particular services or websites – i.e., attackers have a list of domain names of interest and want to detect the users contacting these domain names.

The network hosts also users that do *not* rely on DoH or eSNI. These users exchange non-encrypted packets with servers leaking their domain names. This allows the attacker to easily build a ground truth. Attackers, therefore, learn patterns that characterize the domain names of interest from these users. We do not assume anything else about the traffic. Attackers can easily associate flows with the respective server domain names, e.g., by correlating with the previously observed DNS requests and responses [3], or directly using the SNI information. Note that users already leak domain names in the network if they use *either* plain-text DNS *or* SNI.

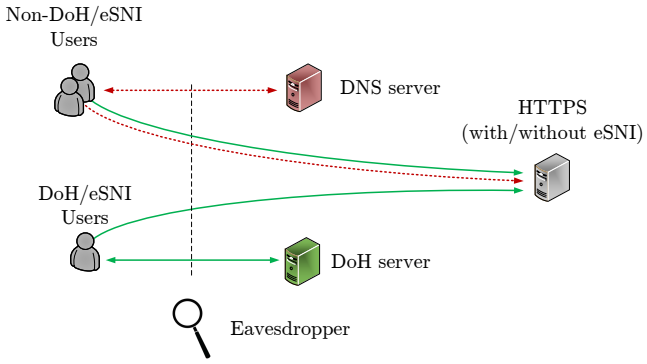

Figure 1: Threat model: Some users leak information (red dashed) that eavesdroppers use to infer domain names of the users relying on DoH and eSNI (green solid).

In some cases, server IP addresses are a rich source of information to uncover domain names. As shown in [27], most IP addresses host a single domain name. However, these domain names are responsible for a minority of the traffic. For instance, in our dataset (described in the next section) only 24% of the traffic comes from server IP addresses hosting only a single domain name. Most traffic comes instead from large services (e.g., YouTube [8]), CDNs, or hosting providers (e.g., Cloudflare or Cloudfront), which usually aggregate multiple domain names behind few IP addresses. We here focus on the latter scenario and assume that server IP addresses provide only coarse information about the hosting company and no further hint to uncover the specific domain names.

3.2 Dataset

We evaluate the attack using a dataset collected in July 2019 in our campus network. We rely on a passive probe to capture traffic in the edge router that aggregates all traffic exchanged by the University with external servers. We use Tstat [28] to obtain statistics from the traffic. Tstat is a passive traffic monitor that exports flow records, i.e., a single entry for each TCP/UDP stream in the network. Each flow record is composed of a rich set of statistics. Beside classical flow-level fields, such as IP addresses, port numbers, packet and byte-wise counters, Tstat extracts the domain names seen in the TLS SNIs, which we use to label the TCP flows. We export also basic information from the first 10 packets of each flow direction (from client to server and vice versa), including packet sizes and inter-packet arrival times.

Our dataset spans over 28 days and includes all flows originating from campus users and reaching external servers. In total, we obtain 340 million TLS flows from 3 995 client IP addresses. They contacted 380 thousand server IP addresses, and we observed 933 thousand domain names in the SNI field. The majority of TLS flows uses TLS version 1.2 (77.6%), while we observe TLS 1.3 in the 16.3% of cases. Using the Application-Layer Protocol Negotiation (ALPN) field of TLS, we note that HTTP versions 1.1 and 2.0 appear with a similar share (46.5% and 53.5% of the flows, respectively). eSNIs are mostly absent in our data.

We have taken different measures to protect users' privacy during the data collection. First, we avoid recording personally identifiable information (PII). We carefully limit the collected information to flow-level statistics and domain names, discarding any other fields not useful for the study, like plain-text HTTP records. Second, IP addresses are anonymized by the probes, thus limiting the identification of the owner of the station. Finally, data collection is approved and constantly supervised by the responsible IT teams.

3.3 Selection of domain names

We restrict our analysis to popular domain names. Our rationale is twofold. First, we filter out the long-tail of domain names with a negligible number of flows and traffic. Second, we focus on large company infrastructures that host popular services (e.g., Facebook, Google) or offer their servers to third parties (CDNs and cloud providers).

We focus on the traffic of the 37 ASes listed in Table 1. To this end, we map each server IP address to the corresponding AS using a BGP View taken from the Route Views Project.[2] These ASes are the most popular in our dataset, and their overall traffic represents the 90% of all flows. These ASes include popular content providers (e.g., Google, Microsoft and Dropbox), CDNs (e.g., Cloudflare, Akamai and Edgecast) and cloud providers (e.g., Amazon and OVH).

Table 1 shows statistics of the traffic to these ASes for July 1st, 2019. We perform experiments with both second-level domain names (SLDs) and FQDNs. We consider *popular* names all FQDNs or SLDs that contribute with at least 1 000 flows in our training set. In Section 4.5, we evaluate the impact of this filtering threshold. Note that the 64% of the SLDs we included are present in the Top-10k domains identified by the Cisco Umbrella Popularity list[3].

In total, we select 708 FQDNs, which are responsible for 82% of the flows for the selected ASes. Some ASes include hundreds of FQDNs, while in three cases only one FQDN passes our threshold. Considering second-level domains, we obtain 409 unique names, representing 91% of the flows. Note that 11 ASes host only one second-level domain, thus representing the totality of their traffic. These ASes host infrastructure dedicated to particular services, for which classification becomes a trivial task. The selected domains include popular sites, such as news and weather forecast websites, and domain names supporting third-party services, such as advertising and tracking domains. The remainder of the traffic is composed by thousands of infrequent domains generating little traffic, and for which it is hard to build reliable models. Naturally, we do not exclude that their popularity may change when collecting data for a longer period. All flows of these unpopular domains are kept in the dataset, in a special category "others". As such, while we are not interested in identifying domain names of these flows, they still contribute with noise, making it harder to execute the attack.

3.4 Testing methodology

We build a classifier to label flows with their respective server domain name. Therefore, the object of our classification is a *flow*.

[2]http://www.routeviews.org/. We manually aggregate multiple ASes of the same company.

[3]http://s3-us-west-1.amazonaws.com/umbrella-static/index.html

Table 1: Statistics of full and second-level names in our dataset. Statistics refer to 1st July 2019. The popular domains appear in at least 1 000 flows. Percentages in brackets represent the share of traffic for popular names.

AS	FQDN	SLD	AS	FQDN	SLD
Adform	5 (94%)	1 (100%)	Google	136 (91%)	42 (97%)
Adobe	4 (42%)	5 (75%)	Hetzner	4 (43%)	5 (45%)
Akamai	99 (68%)	77 (84%)	Highwinds	4 (71%)	5 (75%)
Alibaba	2 (34%)	3 (75%)	Integral	4 (100%)	1 (100%)
Amazon	130 (55%)	117 (74%)	Italiaonline	4 (68%)	4 (94%)
Apple	30 (86%)	4 (100%)	Linkedin	2 (74%)	1 (100%)
Appnexus	6 (100%)	1 (100%)	Microsoft	109 (89%)	33 (96%)
Aruba	1 (13%)	4 (35%)	Outbrain	4 (93%)	3 (100%)
Cloudflare	9 (25%)	12 (30%)	Ovh	6 (33%)	10 (43%)
Criteo	18 (98%)	2 (100%)	Pubmatic	3 (91%)	1 (100%)
Digitalocean	1 (31%)	2 (31%)	Quantcast	3 (85%)	1 (100%)
Doubleverify	3 (100%)	1 (100%)	Rubicon	5 (96%)	1 (100%)
Dropbox	9 (90%)	3 (100%)	SmartAd	4 (100%)	1 (100%)
Edgecast	11 (64%)	13 (75%)	Telegram	2 (91%)	1 (100%)
Facebook	32 (98%)	10 (99%)	Twitter	6 (100%)	2 (100%)
Fastly	25 (49%)	20 (68%)	Webtrekk	5 (71%)	5 (100%)
Fastweb	1 (8%)	4 (31%)	Yahoo	6 (70%)	1 (100%)
Garr	15 (82%)	12 (93%)	**TOTAL**	**708 (82%)**	**409 (91%)**

We perform a correct classification when we give a flow the right name among those in our selection, including the "other" class.

We focus only on HTTPS flows for which we recover the domain name by inspecting the SNI (i.e., our ground truth).[4] We design two sets of experiments. In the first set, we use the *full domain names* as class labels. As such, our model has to learn how to classify flows into 709 classes (708 names and "others"). Note that classes are strongly unbalanced, i.e., the number of samples depends on the popularity of the domains. In the second experiment, we use the *second-level names* as class labels. Our model has to classify flows into 410 classes. Since second-level names aggregate more flows, the class "others" is less popular in this scenario, representing 9% of the totality.

Learning a single model to classify such a large number of classes is a complex problem. To simplify the learning, we train one classifier for each AS in Table 1. We confirm that each domain name in our list appears solely in one AS. Each classifier is thus responsible for the identification of a disjoint set of domain names. We use the server IP address to identify the AS and then the classifier to use.

Attackers build machine learning models for the domain names of interest using the traffic of the users not relying on DoH or eSNI. We simulate this scenario by randomly choosing 50% of the client IP addresses for training, reserving the remaining 50% of the IP addresses for testing. As such the training and testing sets include around 2 000 clients each. This procedure allows us to test classifiers on clients whose data has not been previously observed for training, thus limiting over-fitting due to specific client characteristics – e.g., browsers, operating systems and user habits. For most experiments, we use data from the first day of our dataset (50% of the IP addresses for training, 50% for testing). To check whether the classifiers' performance persists over time, we perform experiments in which we apply the methods built with data from the first day to the data captured in the remaining days (testing set).

[4]Similarly we could focus on TCP flows with domain names coming from the DNS.

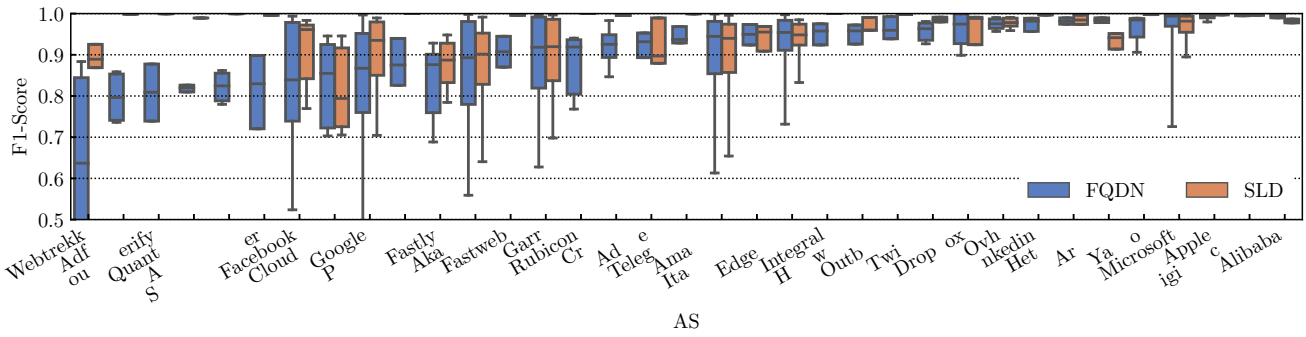

Figure 2: F1-Score distributions for fully qualified domain names (FQDN) and second-level domain names (SLD). Note that the y-scale starts from 0.5.

We use an off-the-shelf random forest classifier with 100 estimators and balanced class weights to limit class imbalance during training [20]. While other models may improve performance, our goal is *not* to find the best possible classifier, but, instead, evaluate how practical the attack is. As we will show, a random forest classifier is surprisingly good for this scenario. Regarding features, we deliberately avoid running feature selection when building the classifiers and consider more than 200 numerical features. Yet, we provide results listing the most discriminative features in our data.

Finally, we use the F1-Score of each class to evaluate the classification performance.[5] Given a class, an F1-Score of 1 represents perfect performance. Considering each AS, we compute the distribution of F1-Scores over all classes. We then analyze the distribution of the F1-Scores across ASes to gauge the effectiveness of the attack.

4 ATTACK EFFECTIVENESS

We first illustrate the overall performance obtained using the random forest classifier. Next, we assess the impact on classification results when changing the experimental setup.

4.1 Classification performance

Figure 2 depicts the performance of the classifiers. The blue boxes, report separately by AS, the distribution of F1-Scores obtained when classifying FQDNs. Black strokes represent the median.

Performance varies according to the AS. The 1st quartile is above 75% for all ASes, with some ASes presenting median F1-Score close to 95%. Curiously, the performance is not directly proportional to the number of classes – more classes could represent a harder scenario. For example, Akamai, Amazon, Google and Microsoft are the ASes with the largest number of classes but have a median F1-Score larger than 0.85. In a nutshell, 50% of full domain names can be almost perfectly recovered.

Overall, around 80% of the full domain names used as class labels have F1-Score above 80%. Recalling that the F1-Score for a class is the harmonic mean between its precision and recall, F1-Scores close to one require both metrics to have high values. That means that classifiers not only correctly identify the domain names (high precision), but also retrieve most of the flows related to the given names (high recall).

We conclude that an attacker can easily recover the domain name by using just traffic features. These results have been obtained with an off-the-shelf, not-optimized machine learning algorithm. In other words, the attack is very easy to be executed. Next, we investigate some factors influencing the classification results.

4.2 Second-level vs. full domain classification

We now evaluate the classification results when using a coarser aggregation of class labels, i.e., using second-level domain names. This setup represents a scenario where attackers are interested in identifying only specific websites (e.g., `example.com`) regardless sub-domains contacted by users (e.g., `images.example.com` or `css.example.com`). Recall that by using second-level domain names we reduce substantially the frequency of the class "others". In fact, the overall percentage of labeled flows raises from 57% to 73%.

Red boxes in Figure 2 show results. Overall performance is similar to what we obtained with full domain names. The ratio of classes having F1-Scores higher than 0.8 raises from 80% (full names) to 86% (second-level names). The performance improvement can be associated with (i) the larger number of samples in the training set; (ii) the smaller number of classes. In particular, for 11 ASes we find that all the traffic is served by a unique second-level name, and, as such, the classification becomes trivial and all flows are correctly identified. In sum, attackers willing to uncover only second-level names can (i) increase the coverage from 82% to 91% of the flows; (ii) reach better performance without increasing the complexity of the classifiers.

4.3 Which traffic features are needed?

Designing eventual protections against these attacks require protocol designers to understand which side-channels contribute to the domain name inference. To this end, we analyze the importance of the flow features that we use in the classifiers.

Recall that Tstat exports hundreds of per-flow features, that we can coarsely aggregate into 3 groups: (i) *Basic Flow Features*: Those exported by simple flow meters such as Cisco's NetFlow – i.e., the

[5]The F1-Score is the harmonic mean between *Precision* and *Recall* of a class.

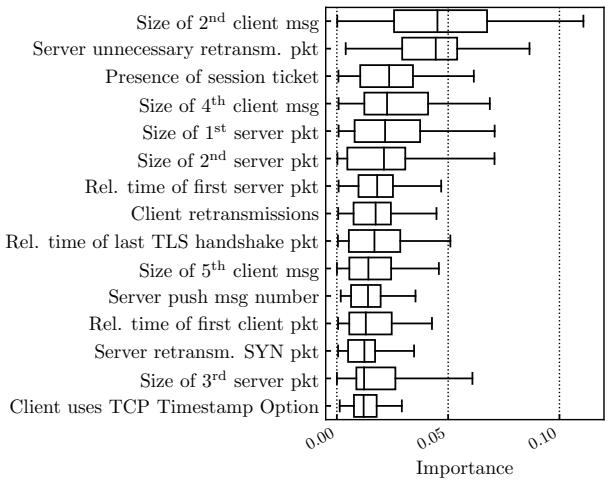

Figure 3: Distribution of feature importance over different ASes for the top-15 features.

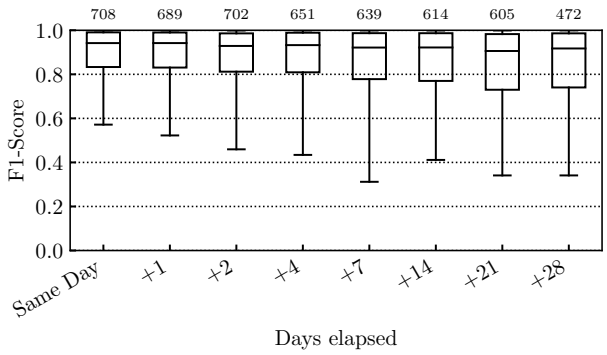

Figure 4: F1-Score distributions (full names) when varying the interval elapsed between training and testing data.

overall number of packets and bytes of each flow; (ii) *Advanced Flow Features*: Those computed from the analysis of the IP and TCP headers such as per-flow packet loss, RTT, Time-To-Live, etc.; (iii) *First Flow Packets*: The size and inter-packet arrival time for the first 10 packets of each flow direction; Tstat also reports the size of the first 10 push-delimited messages – i.e., sequences of segments delimited by a packet with the PSH flag set.

Figure 3 shows the top-15 most important features. As we use Random Forest models, we compute the feature importance by averaging the Gini impurity importance calculated over each tree.[6] The higher the Gini, the more information that feature brings. We build an independent classifier for each AS, and, as such, we report with a boxplot the distribution of the importance of a feature across the models and rank them by the median values.

The most important feature is the size of the second client push-delimited message, likely containing part of the TLS handshake. The importance of size information is confirmed by the presence of other two message-based features among the top-15. In other six cases we find features based on sizes, as well as timings of the first packets. This result allows us to conjecture that such kind of features is sufficient to obtain good classification performance. To confirm it, we run an experiment in which we use only these features in isolation, and we observe a small penalty in the overall performance.

Flow-wise counters are almost absent, represented only by overall number of push-delimited messages. This result suggests that mounting an attack with simple flow-level features is not trivial. We also find deployment-dependant features, such the number of TCP-level retransmissions, appearing under different forms in three different features. That is, models seem to incorporate some the peculiarities of the targeted network. Transferring such models to different deployments may impair performance. Unfortunately, we do not have datasets collected in different networks, and we leave the study of the spatial stability of learning as future work.

[6]https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity

Possible protection against this kind of attacks would require traffic signatures to be masked. On the one hand, blurring packet sizes and packet inter-arrival times, e.g., by padding packets and adding random packet delays, could help to limit fingerprints. This has to be done at the TCP level, on all flows, and not only by modifying the DNS and TLS protocols.

4.4 How long models persist?

In the prior experiments we use data from the first day of our dataset for both training and testing, randomly splitting the client IP addresses into the sets. Here we want to quantify to what extent the freshness of training data is important to the classification performance. We train the classifiers on 50% of clients seen in the first day (full domains). Then we use the remaining days for testing, using only clients that were not part of the training.

Figure 4 shows the overall distribution of F1-Scores when increasing the interval between training and testing. The performance slightly degrades for some domains as time goes on. This effect is clearer when the time difference between the datasets is longer than 1 week. Note how the 25th percentile drops from 0.83 (left-most boxplot) to 0.73 (right-most boxplot). However, for most domains the classifier is still able to correctly classify flows even after 28 days. Median values for F1-Scores decreases only slightly, from 0.95 (same day training and testing) to 0.92 (4-week gap between training and testing).

To help understanding the drop in performance, numbers on top show how many FQDNs present in the training still exceed 1 000 flows in the testing set. We can see that some domain names that are popular in the first capture day are no longer popular at the end of the capture. Other names initially in the "others" category increase in popularity (not shown in the figure). In sum, the set of popular names changes as time passes. Attackers therefore must update their models accordingly.

4.5 How many flows in the training set?

So far we have assessed performance after training the classifiers with at least 1 000 flows. We now quantify the impact of the number of observations on performance. To this end, we relax the threshold,

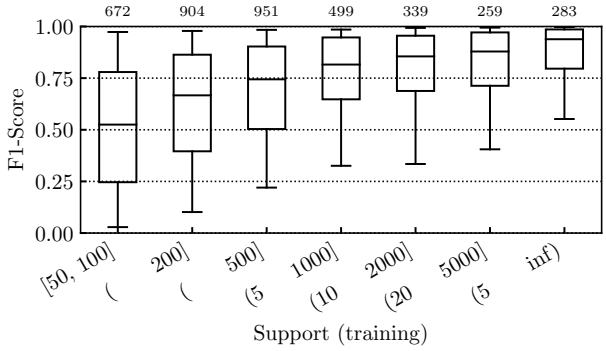

Figure 5: F1-Score distributions according to the number of flows in the training set.

reducing the required number of observations from 1 000 to 50. We use the first day of data for both training and testing.

Figure 5 shows distributions of F1-Scores for the names grouped by the numbers of observations in the training set. The left-most boxplot reports the distribution for names seen [50, 100] times during training. Numbers on the top report the total domain names in each bin.

As expected, training with more samples improves performance. While only 23% of the names seen [50, 100] times during training have F1-Score larger than 0.75 (left-most boxplot), 75% of names seen more than 5 000 times have F1-Score larger than 0.8 (right-most boxplot). In sum, learning a model for the most popular domains is easy. However, even a small number of observations suffices to learn good models.

5 CONCLUSION

In this paper we explored to what extent an eavesdropper can uncover the server domain name of network flows generated by users making use of DoH or eSNI. We showed that such an approach is feasible when enough observations are provided. We used a large dataset coming from an operational network. After training a random forest classifier, an eavesdropper can classify 80% of domain names with F1-Score higher than 0.8. The size and timing of the first packets of the flows are among the most useful features.

Our results show the limits of the effectiveness of privacy-preserving enhancements currently under standardization, such as DoH and eSNI. They call for further actions to avoid the name of servers to be retrieved. Unfortunately, these actions require deep changes in transport protocols, such as the inclusion of padding and random inter-packet pacing, or the widely deployment of advanced solutions such as VPNs and/or TOR.

Future work includes extending the set of classification approaches to understand whether performance can be further improved, and exploring other threat models – e.g. obtaining training data from an active testbed.

ACKNOWLEDGMENTS

The research leading to these results has been funded by the European Union's Horizon 2020 research and innovation program under grant agreement No. 871370 (PIMCity) and the SmartData@PoliTO center for Big Data technologies.

REFERENCES

[1] B. Anderson and D. McGrew. 2019. TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior *(Proc. of the IMC)*. 379–392.
[2] D. Arp, F. Yamaguchi, and K. Rieck. 2015. Torben: A Practical Side-Channel Attack for Deanonymizing Tor Communication *(Proc. of the ASIA CCS)*. 597–602.
[3] I. Bermudez, M. Mellia, M. Munafò, R. Keralapura, and A. Nucci. 2012. DNS to the Rescue: Discerning Content and Services in a Tangled Web *(Proc. of the IMC)*. 413–426.
[4] S. Bhat, D. Lu, A. Kwon, and S. Devadas. 2019. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning *(Proc. of the PET)*. 292–310.
[5] T. Böttger, F. Cuadrado, G. Antichi, E. Fernandes, G. Tyson, I. Castro, and S. Uhlig. 2019. An Empirical Study of the Cost of DNS-over-HTTPS *(Proc. of the IMC)*. 15–21.
[6] S. Farrell and H. Tschofenig. 2014. *Pervasive Monitoring Is an Attack*. Technical Report 7528. RFC Editor.
[7] S. Feghhi and D. Leith. 2016. A Web Traffic Analysis Attack Using Only Timing Information. *IEEE Transactions on Information Forensics and Security* 11, 8 (2016), 1747–1759.
[8] D. Giordano, S. Traverso, L. Grimaudo, M. Mellia, E. Baralis, A. Tongaonkar, and S. Saha. 2015. YouLighter: An Unsupervised Methodology to Unveil YouTube CDN Changes. In *Proc. of the 2015 27th International Teletraffic Congress*. 19–27.
[9] R. Gonzalez, C. Soriente, and N. Laoutaris. 2016. User Profiling in the Time of HTTPS *(Proc. of the IMC)*. 373–379.
[10] X. Gu, M. Yang, and J. Luo. 2015. A Novel Website Fingerprinting Attack against Multi-tab Browsing Behavior *(Proc. of the CSCWD)*. 234–239.
[11] A. Hintz. 2003. Fingerprinting Websites using Traffic Analysis *(Proc. of the PET)*. 171–178.
[12] P. Hoffman and P. McManus. 2018. *DNS Queries over HTTPS (DoH)*. Technical Report 8484. RFC Editor.
[13] R. Houser, Z. Li, C. Cotton, and H. Wang. 2019. An Investigation on Information Leakage of DNS over TLS *(Proc. of the CoNEXT)*.
[14] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS)*. Technical Report 7858. RFC Editor.
[15] M. Lescisin and Q. Mahmoud. 2018. Tools for Active and Passive Network Side-Channel Detection for Web Applications *(Proc. of the WOOT)*.
[16] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu. 2019. An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come? *(Proc. of the IMC)*. 22–35.
[17] B. Miller, L. Huang, A. Joseph, and J. Tygar. 2014. I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis *(Proc. of the PET)*. 143–163.
[18] T. Mori, T. Inoue, A. Shimoda, K. Sato, S. Harada, K. Ishibashi, and S. Goto. 2016. Statistical Estimation of the Names of HTTPS Servers with Domain Name Graphs. *Computer Communications* 94 (2016), 104–113.
[19] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste. 2014. The Cost of the "S" in HTTPS *(Proc. of the CoNEXT)*. 133–140.
[20] 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.
[21] D. Plonka and P. Barford. 2011. Flexible Traffic and Host Profiling via DNS Rendezvous *(Proc. of the SATIN)*. 1–8.
[22] E. Rescorla, K. Oku, N. Sullivan, and C. Wood. 2019. *Encrypted Server Name Indication for TLS 1.3*. Technical Report draft-ietf-tls-esni-04. RFC Editor.
[23] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen. 2018. Automated Website Fingerprinting through Deep Learning *(Proc. of the NDSS)*.
[24] Y. Shi and S. Biswas. 2014. Website Fingerprinting using Traffic Analysis of Dynamic Webpages *(Proc. of the GLOBECOM)*. 557–563.
[25] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso. 2020. Encrypted DNS–> Privacy? A Traffic Analysis Perspective *(Proc. of the NDSS)*.
[26] P. Sirinam, M. Imani, M. Juarez, and M. Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning *(Proc. of the CCS)*. 1928–1943.
[27] M. Trevisan, I. Drago, M. Mellia, and M. Munafo. 2016. Towards Web Service Classification using Addresses and DNS *(Proc. of the TRAC)*. 38–43.
[28] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, and D. Rossi. 2017. Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned. *IEEE Commun. Mag.* 55, 3 (2017), 163–169.
[29] L. Vassio, D. Giordano, M. Trevisan, M. Mellia, and A. Silva. 2017. Users' Fingerprinting Techniques from TCP Traffic *(Proc. of the Big-DAMA)*. 49–54.
[30] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting *(Proc. of the USENIX Security)*. 143–157.