POLITECNICO DI TORINO
Repository ISTITUZIONALE

Attacking DoH and ECH: Does Server Name Encryption Protect Users' Privacy?

Original
Attacking DoH and ECH: Does Server Name Encryption Protect Users' Privacy? / Trevisan, Martino; Soro, Francesca; Mellia, Marco; Drago, Idilio; Morla, Ricardo. - In: ACM TRANSACTIONS ON INTERNET TECHNOLOGY. - ISSN 1533-5399. - ELETTRONICO. - 23:1(2023), pp. 1-22. [10.1145/3570726]

Availability:
This version is available at: 11583/2979955 since: 2023-07-05T16:47:49Z

Publisher:
ACM

Published
DOI:10.1145/3570726

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

© ACM 2023. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in ACM TRANSACTIONS ON INTERNET TECHNOLOGY, http://dx.doi.org/10.1145/3570726.

(Article begins on next page)

16 May 2024

Attacking DoH and ECH: Does server name encryption protect users' privacy?

MARTINO TREVISAN, University of Trieste, Italy
FRANCESCA SORO, Austrian Institute of Technology, Austria
MARCO MELLIA, Politecnico di Torino, Italy
IDILIO DRAGO, Università di Torino, Italy
RICARDO MORLA, University of Porto, Portugal

Privacy in the Internet has become a priority and several efforts have been devoted to limit the leakage of personal information. Domain names, both in the TLS Client Hello and DNS traffic, are among the last pieces of information still visible to an observer in the network. The ECH extension for TLS, DNS over HTTPS (DoH) or over QUIC protocols aim to further increase network confidentiality by encrypting the domain names of the visited servers.

In this paper, we check whether an attacker able to passively observe the traffic of users could still recover the domain name of websites they visit even if names are encrypted. 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. We consider three attack scenarios, i.e., recovering the per-flow name, rebuilding the set of visited websites by a user, and checking which users visit a given target website. We next evaluate the efficacy of padding-based mitigation, finding that all three attacks are still effective, despite resources wasted with padding. We conclude that current proposals for domain encryption may produce a false sense of privacy, and more robust techniques should be envisioned to offer protection to end users.

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

Additional Key Words and Phrases: Privacy, Encryption, Network Traffic, Eavesdropping

1 INTRODUCTION

As a reaction to the ease of passive monitoring of network traffic [1], the Internet has massively embraced encrypted protocols [2, 3]. The simplicity of new methods to obtain and renew server certificates, the improvements on web protocols that support or mandate the use of TLS (e.g., HTTP/3 [4]), and the stricter security policies in web browsers have pushed the majority of services into the adoption of encryption [5]. These deployments are robust against in-network monitors and eavesdroppers and increase privacy and confidentiality.

The most notable omission in these efforts to enhance confidentiality has been the protection of the *host names* that are still exchanged in plain-text in several situations. Notably, DNS requests and responses carry Fully Qualified Domain Names (FQDNs) in clear, 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 fundamental when multiple websites are hosted behind a single server IP address, which is common in Content Delivery Networks (CDNs). The presence of plain-text host names in the traffic gives eavesdroppers access to rich information [6]. From host names, one can guess the type of devices, operating systems and browsers used by people by simply searching for specific strings. Similarly, eavesdroppers can build lists of services visited by users to build profiles and extract their interests [7]. Recent proposals aim at fixing these issues

Authors' addresses: Martino Trevisan, University of Trieste, Italy, first.last@dia.units.it; Francesca Soro, Austrian Institute of Technology, Austria, francesca.soro@ait.ac.at; Marco Mellia, Politecnico di Torino, Italy, marco.mellia@polito.it; Idilio Drago, Università di Torino, Italy, idilio.drago@unito.it; Ricardo Morla, University of Porto, Portugal, ricardo.morla@fe.up.pt.

by encrypting DNS traffic, e.g., running DNS over HTTPS (DoH) [8] or over QUIC [9, 10], and by encrypting the TLS Client Hello and SNI field, as in the Encrypted Client Hello (ECH) extension [11] currently under standardization.[1] In such a scenario, eavesdroppers would be left with only server IP addresses as sensitive information.

However, DoH and ECH are not yet widespread, since they require changes at both client and server sides. 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 already adopting the privacy-enhancing technologies can still be at stakes.

In this paper, we study to what extent the protection offered by the deployment of DoH and ECH can be attacked. More concretely, we evaluate whether an attacker observing the traffic of users adopting DoH and ECH can uncover the host names they contact using machine learning (ML) models trained using external traffic sources (e.g., unencrypted traffic). We here consider three attack scenarios: (i) recovering the per-flow host name (ii) rebuilding the set of websites visited by a user; (iii) checking which users visit a given target website. Clearly, the scenarios represent different threat levels. All however uncover sensitive information.

We assume the attacker can leverage different mechanisms to build a dataset for training models and mounting the attack: (i) eavesdropping traffic and leveraging the portion of traffic still carrying plain-text domain names; (ii) harvesting corporate/private traffic and DNS logs; or (iii) running active experiments.

For studying the effectiveness of the attack, we first rely on traces collected in a production network, thus simulating the eavesdroppers' case. We split users into two sets, for training and testing. We label the TLS traffic flows generated by these users in the training set with domain names found in the SNI field exposed during the TLS handshakes. In our traces only a negligible but increasing percentage ($\approx$ 2% in 2019, $\approx$ 5% in 2021 and $\approx$ 7.5% in 2022) of flows are protected by ECH, thus allowing us to extract names for the vast majority of the TLS flows. Using this labeled dataset, we build machine learning models to recover the domain names of the flows generated by users in the test dataset.

We show that a simple off-the-shelf ML solution is sufficient for performing all three attack scenarios.[2] We execute the attack using features such as flow duration, byte counters, packet sizes and packet inter-arrival times, which are known to be effective for traffic classification [12]. The machine learning models yield surprisingly good results, achieving F1-scores over 0.8 for 80% of the evaluated domain names. In other words, attackers can recover most of the domain names visited by users, and could build lists of users contacting given domain names with high precision and high recall.

The triviality of the attack calls for further actions to protect domain names. To this end, we evaluate to what extent standardized padding-based countermeasures [13] mitigate the privacy risks. Our experiments reveal that the effectiveness of the attacks decreases, but an eavesdropper would still be able to gain large amount of information from traffic. Even when using the most aggressive Maximal-Length Padding strategy (which makes all packets the same size), we observe F1-scores higher than 0.8 for 44% of the domain names. The cost of padding is sizable, adding 5% to 17% more traffic to different sites.

At last, we study whether datasets obtained using active crawlers are sufficient to mount the attack. Our results show that using such web crawlers is cumbersome due to the heterogeneity

[1] ECH was formerly named eSNI in the initial IETF drafts. While eSNI only encrypted the host name, ECH can protect all privacy-sensitive TLS handshake parameters.

[2] Source code and ML models are available at https://smartdata.polito.it/does-domain-name-encryption-increase-users-privacy/

of the traffic generated by a domain, which reflects the heterogeneity of possible configurations (device, network, user's habits, etc.) users have.

This paper extends our preliminary work [14]. First, we consider two additional attack scenarios, i.e., recovering the lists of domains visited by a user, and the extraction of website audience lists. Second, we study the additional protection offered by padding countermeasures. Finally, we explore different ways to mount the attack, evaluating the possibility to build the training data from automatic crawlers. The results presented in this paper show that only the encryption of domain names is not sufficient to protect users' privacy, and solutions like DoH and ECH could give users a false sense of protection.

The paper is organized as follows: Section 2 summarizes background and related work, while Section 3 defines our threat models and presents the employed dataset. In Section 4, we dissect the effectiveness of the threats, while Section 5 and 6 discuss the impact of padding-based mitigation and training data collection, respectively. Section 7 discusses limitations of the approach, and Section 8 concludes the paper. Finally, extra information about our datasets is provided in the appendixes.

2 BACKGROUND AND RELATED WORK

2.1 Countermeasures to traffic eavesdropping

Passive monitoring is the observation of the network traffic of a set of users. It can be done for various reasons, from legitimate network engineering to eavesdropping. Passive monitoring includes the gathering of protocol artefacts and headers, which may include personal identifiers (i.e., the client IP addresses) and sensitive data (e.g., accessed websites) and, as such, it is recognized to be critical for users' privacy [1].

To mitigate the effectiveness of passive monitoring, the Internet is running a race towards encryption. Starting from 2010, TLS started to be massively adopted [2], protecting the content of HTTP traffic. However, TLS includes the Server Name Indication (SNI) field in the Client Hello message, which is necessary for the server to present the appropriate certificate in case it serves multiple virtual hosts. Similarly, users still rely on plain-text DNS to resolve domain names, offering rich personal information to passive monitors.

Researchers and the industry proposed improvements based on encryption to the DNS and TLS protocols to prevent such information leakage. The first attempts to encrypt DNS propose to encapsulate it in TLS [15] or DTLS [16]. The most promising technology for confidential DNS is the so-called DNS over HTTPS (DoH) [8], supported by modern browsers (e.g., Chrome, Firefox and Safari) and by some of the Internet giants (e.g., Google and Cloudflare). Recently, DoH allows also to use QUIC as transport protocol [9, 10]. DoH carries DNS traffic over an HTTPS session and maps query-response pairs into HTTP request-response pairs. In our traces, we observe an increasing adoption trend for DoH, with adopters increasing from 2% in 2019 to 5% in mid-2021 and to 7.5% in late 2022.

Similar in spirit, the Encrypted Client Hello (ECH) extension of TLS [11] encrypts sensitive fields in the TLS handshake using an encryption key retrieved via DNS. Formerly, this proposal was named eSNI and targeted the SNI string exclusively. The current ECH proposal can be used to encrypt various handshake fields, such as the Application-Layer Protocol Negotiation (ALPN). ECH requires the user to retrieve encryption keys using a reliable DNS, and it shall be used in conjunction with a new type of DNS record currently under standardization [17]. In general, ECH still suffers from the need for a reliable network for key distribution, and, in our traces, we observe a negligible number of ECH flows as of 2021.

2.2 Related Work

Extensive prior work addresses the problem of inferring users' activity through packet eavesdropping. Already in 2002 authors of [18] 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 [19], multi-tab browsing [20] or different underlying encrypted protocols like Tor [21], HTTPS [22] and 802.11 [23]. Other works focus on specific websites belonging to diverse categories, like politics [24] and health [25], and make use of different machine learning techniques, such as deep neural networks [26, 27]. All these works rely on datasets collected in a controlled environment, where diversity is intrinsically limited. We follow a methodology similar to those works. We illustrate the feasibility of such attacks in a new scenario, uncovering the server host names of flows protected by ECH and DoH. Here, we use a dataset passively collected in a real environment, where thousands of domains are contacted by thousands of users, which significantly complicates the problem.

Researchers tested different website fingerprinting techniques focusing on the Tor network in [28, 29]. Other works proposed approaches for traffic classification using side-channels containing information such as plain-text DNS transactions [6, 30, 31]. Recent efforts to enhance privacy propose to encrypt DNS as a solution to limit the applicability of these approaches [8, 15]. The adoption of such privacy-enhancing techniques is yet to come, with some authors that already highlighted critical aspects of DoH deployments [32–34]. Authors of [35] recently pointed out how packet eavesdropping techniques used for website fingerprinting can be applied to encrypted DNS too, using metadata like packet sizes and timing. Authors of [36] achieved similar results by analyzing DNS-over-HTTPS (DoH) using features related to the size, timing and ordering of DNS packets. Differently, authors of [37–39] show that machine learning can be used to identify DoH flows with surprisingly high accuracy. In [40], authors show how it is possible to infer the domain from encrypted and padded DNS traces, achieving 86.1% accuracy in simple controlled experiments. We here target the scenario where server domain names are encrypted, but differently from previous works we fingerprint directly the TCP flows where domain information has been protected by the use of DoH and ECH.

To further improve privacy, some works suggest several countermeasures against eavesdropping: DoHoT [41] is an open project that proposes an enhancement of DoH using the Tor network without affecting latency. Authors of [42] introduce Oblivious DNS over HTTPs (ODoH) as an extension of DoH using an intermediate proxy node which performs the query for the original client. In [43], authors quantify the privacy gains and implications of ECH in terms of k-anonymity and dynamic changes on the IP addresses, claiming that the co-hosting of a large enough set of services, together with a more dynamic IP-domain mapping should contribute to the ECH reliability. Authors of [40] elaborate on the padding strategies suggested in RFC 8467 [13]. They confirm that padding alone is not enough to prevent eavesdroppers from performing website fingerprinting, as a classifier is able to shift from packet-based features to timing-based features. To counteract this phenomenon, authors suggest further practical mitigation strategies that take into account timing-based features: i) sending packets at constant time rates [44]; ii) adaptive padding [45], i.e., sending dummy traffic to conceal the actual data. In this paper, we confirm and further extend these claims by investigating the benefits of padding in our experiments. Here, we scale the attack to hundreds of domains and show how the attack can be mounted in real networks too.

3 METHODOLOGY

3.1 Threat models

We consider three threat models that we describe in Figure 1. In all cases, we assume the attacker monitors traffic of a set of users. Some of these users use only fully encrypted protocols to hide the domain names of servers they contact. In such a scenario, the attacker cannot obtain any insights from packet payload directly, beyond the source and destination IP addresses. We assume the attacker is interested in recovering the name of servers contacted by users. To this end, the attacker analyzes users' traffic grouping it into flows and extracting features from the packet characteristics.

In threat A, the attacker wants to label each flow with the contacted domain name. This is a classic traffic classification scenario, which would allow the attacker to characterize the traffic related to given services, such as to count the number of flows, packets and bytes directed to particular domains. Notice that there is a many-to-one association between flows and a domain name.

In threat B, the attacker wants to obtain the set of domains contacted by each user. In other words, the attacker wants to build users' profiles, defined as the set of domains each user has contacted during a time window. Such lists of domain names could reveal users' interests or uncover details of users' environment, such as the devices they use to browse the web.

In threat C, the attacker wants to obtain the set of users visiting a given domain, i.e., reconstructing the service audience. Similar to threat B, a single user will be part of the audience of multiple domain names. Notice that both threats B and C are strictly related to threat A, as the attacker first needs to label flows and then group flows by users or domain names.

Attackers need information about the characteristics of the traffic related to each domain name of interest. We study two alternatives to create such a *training* dataset: (i) obtaining data directly from passive measurements, leveraging the traffic of users that do not rely on encryption to protect server names; (ii) actively crawling those websites of interest while registering their traffic characteristics.

Figure 1 illustrates the first scenario. We assume the attacker has visibility on a group of users that still have not adopted DoH or ECH. These unprotected users exchange non-encrypted packets with servers, thus leaking the domain name associated with each flow either directly on the SNI field (for TLS flows) or indirectly on their DNS traffic. For the latter, the association of flows with the respective server domain name can be obtained by correlating flows with previously observed DNS requests and responses [6]. Note that users leak domain names in the network if they use *either* plain-text DNS *or* SNI, *or* both. From such unprotected traffic, the attacker can easily build a ground truth to learn traffic patterns for each domain name.

Additionally to domain names, the attacker could leverage the server IP address information to uncover services contacted by each user. As shown in [46], most IP addresses host a single domain name. However, these domain names are responsible for a minority of the traffic. In our dataset (introduced next) only 24% of the traffic comes from server IP addresses hosting only a single domain name. Most traffic comes from CDNs and hosting providers (e.g., Cloudflare or Cloudfront), which usually host multiple domain names behind few IP addresses. We here focus on the latter scenario and assume that the server IP address provides just information about the hosting company and no further hint to uncover the specific domain names.

3.2 Dataset

We first evaluate the attack using a dataset passively collected in July 2019 in a campus network, thus simulating the scenario in Figure 1. 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 [47] to obtain statistics from the traffic. Tstat is a passive traffic monitor that exports flow records, i.e., a

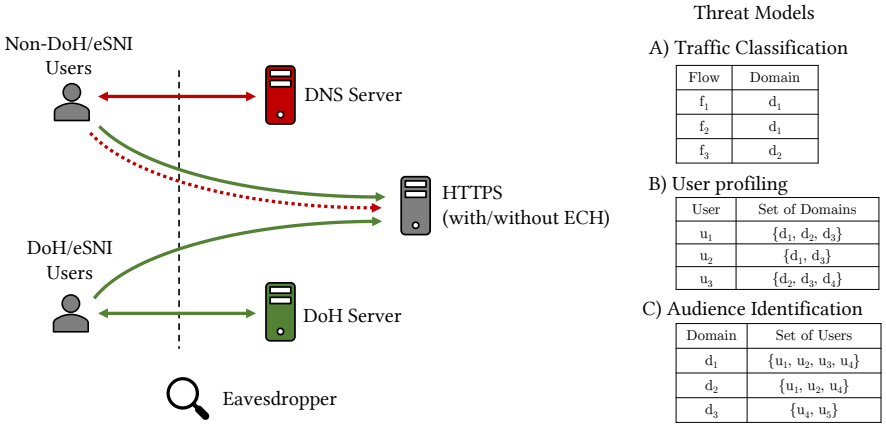


Fig. 1. Threat model: Some users leak information (red dashed) that eavesdroppers use to: A) infer domain names for flows protected by DoH and ECH (green solid), retrieve the set of B) domains per user and C) users per domain.

single entry for each TCP/UDP flow. Each flow record reports a rich set of statistics containing more than 200 features. 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 SNI, which we use to label the TLS 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 covers 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 to 380 thousand server IP addresses, referring to 933 thousand domain names. The majority of the 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). ECH is mostly absent in our data still in 2021, whereas DoH is used by less than 5% of the campus users.

We have taken different measures to protect users' privacy during this data collection. First, we anonymize the client IP address which is the only personally identifiable information (PII) during the data capture. Then, we carefully limit the collected information to flow-level statistics and domain names useful for this work. Our data collection is approved and constantly supervised by the university.

To complement the analysis, we assess the feasibility of using active crawlers to mount the attack. We run an experimental campaign targeting 6 websites that are among the most popular in our campus. We use BrowserTime,[3] a tool to instrument web browsers to automatically visit lists of web pages. Our goal is to recreate the characteristics and diversity of the passive data, where real users access websites with different browsers and devices. Thus, for each website, we instrument Google Chrome, Firefox and Microsoft Edge to access the home page and 100 internal pages. Indeed, the home page can largely differ from internal pages [48] and bias the outcome of the experiments. Each test is repeated 20 times, considering both cold and warm cache scenarios. In total, we collect more than 36 000 visit for each website.

Notice that we decide to test 6 websites only, focusing on some of the most popular ones in our campus, which allow us to validate the model with the passive traces. Moreover, we put our effort

[3]https://www.sitespeed.io/documentation/browsertime

in injecting diversity and realism in the experiments, by requesting multiple pages of the same sites. We will discuss later how this dataset has been used to in conjunction with the passively collected traces in our experiments.

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). These providers are the first to promote and support new privacy features.

In total, we focus on the traffic directed to 35 Autonomous Systems (ASes). These ASes are the most popular in our dataset, and their overall traffic represents 90% of all flows. They 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). For this filtering, we map each server IP address to the corresponding AS using a BGP View taken from the Route Views Project.[4]

Table 1 shows statistics of the traffic to these ASes for July 1st, 2019. We perform experiments with *popular* second-level domain names (SLDs) and *popular* FQDNs. We consider *popular* those names for which we observe at least 1 000 flows. In Section 6.2, we evaluate the impact of this threshold. Notice that we apply our threshold independently for the SLD and FQDN scenarios, and, as such, the number of FQDNs for an AS is not necessarily larger than the number of SLDs. We observe that the 64% of the included SLDs are present in the top-10k domains identified by the Cisco Umbrella Popularity list[5].

Considering traffic for July 1st, 2019, we select 708 popular FQDNs for our experiments. These FQDNs are responsible for 82% of the flows in that day. We see in Table 1 that some ASes include hundreds of FQDNs, while three ASes include just one popular FQDN. As such, the attack could be harder for the particular ASes where multiple FQDNs are hosted. Considering second-level domains, we obtain 409 popular names, representing 91% of the flows. Here, 11 ASes host only one second-level domain, which are thus trivially uncovered by attackers. For completeness, in Table 2 (in the Appendix), we report additional statistics of the dataset, including the total number of flows and unique FQDNs (and SLDs) for the considered ASes.

The rest of the traffic is composed by thousands of infrequent domain names 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 Attack methodology

We build a classifier that labels flows with server domain names. Therefore, the object of our classification is a *flow*. We here assume that a flow is the natural aggregation of traffic sharing the same class label – i.e., a flow cannot be associated with more than one domain. Clearly, this assumption limits the approach to scenarios in which users do not rely on tunnels (e.g., VPNs). We also avoid classifying single packets directly, as the per-packet classification would bring little advantage given the scarce features when encryption is in place, i.e., only L3 and L4 headers, packet size and packet timing would be available. Our flows, instead, are characterized by multiple

[4]http://www.routeviews.org/. We manually aggregate multiple ASes of the same company.
[5]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%)**

statistical features extracted from all flow packets. *Flow monitoring* [49], nowadays the prevalent method for network monitoring in high-speed networks, is often more scalable than traditional packet-based approaches. Indeed, it largely reduces the number of entries to track and process, while allowing flexibility on the choice of the per-flow features.

For threat A, we perform a correct classification when we give to a flow the right name among those in our selection, including the "other" class. For threat B and C, the attacker uses the flows labeled by the first classifier to build the sets of domains referring to the same user or service, respectively.

We design two sets of experiments. In the first set, we use the *FQDNs* as class labels. As such, when using the passive dataset as training set, our model has to learn how to classify flows into 709 classes (708 FQDNs and "others"). Note that classes are strongly imbalanced, i.e., the number of samples depends on the popularity of FQDNs, with the class "others" representing 18% of the flows. In the second experiment, we use the SLDs as class labels. Our model has to classify flows into 410 classes. Since SLD names aggregate more flows, the class "others" is less popular in this scenario, representing 9% of the total.

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. 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. At classification time, we use the server IP address to identify the AS and thus the classifier to be used.

As said, we initially assume the attacker uses the traffic of those clients not relying on DoH or ECH to create a training set and build a model. 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 client IP addresses 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 passive dataset both for training and testing. To check whether the classifier performance persists over time, we train a model based on data from the first day only, and test performance on data of the next days – always reserving 50% of users for the testing set. Finally, to assess the performance of classifiers trained using crawling, we build another set of models using the crawled dataset and test these models using again the 50% of the IP addresses reserved for testing in the passive dataset.

We use an off-the-shelf random forest classifier with 100 estimators and balanced class weights to limit class imbalance during training [50]. While other models may improve performance, our goal is not to find the best possible classifier, but instead, evaluate how practical the attack would be. As we will show, a standard random forest classifier is surprisingly good for this scenario. Regarding features, we deliberately avoid running feature selection when building the classifiers and consider all 200 numerical features present in our dataset. Yet, we provide results listing the most discriminative features, as Random Forests come with the advantage of getting the ranking of most important features for the classification.

3.5 Performance metrics

We use the F1-Score of each class to evaluate the classification goodness.[6] Given a class, F1-Score equals to 1 represents perfect performance. Considering each AS, we compute the distribution of each metric over all classes. We then analyze the distribution across ASes to gauge the effectiveness of the attack.

We assume the attacker uses the host names provided by the per-flow classifier for building the sets of flows per domain (threat A), domains per user (threat B) and users per domain (threat C). Given the real set S_R and the set built from classified flows S_C, the precision measures the share of elements of S_C that are present in S_R and is computed as $\frac{|S_R \cap S_C|}{|S_C|}$. Intuitively, it quantifies the reliability of elements in S_C. Conversely, the recall measures the share of elements of S_R that are correctly found in S_C, $\frac{|S_R \cap S_C|}{|S_R|}$. It quantifies the completeness of the set built using the classifier.

Notice that the metrics are defined for each user (threat B) or domain (threats A and C), and we study their distributions to understand the effectiveness of the attacks in building reliable and complete sets.

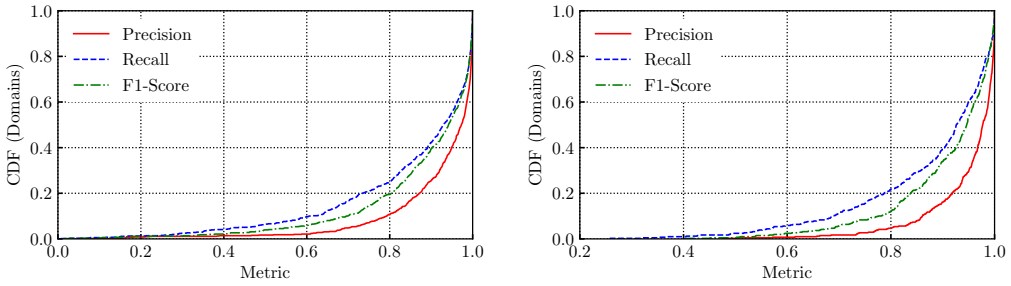
4 ATTACK EFFECTIVENESS

In this section we illustrate the overall performance obtained using the classifier for the three threats. We consider the attacker uses passive traces to build the training set. We then discuss the importance of the traffic features in the classification problem.

4.1 Threat A

We first discuss the effectiveness of threat A considering FQDN as target class. In Figure 2a we report the distribution of precision (red solid), recall (blue dashed) and F1-Score (green dashed line). Overall, around 80% of the domain names have F1-Score above 0.8, and 60% have F1-Score higher than 0.9. 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). Precision is higher than recall, hinting that, given a domain, false negatives are likely to be more frequent than false positives.

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



(a) FQDN. (b) SLD.

Fig. 2. F1-Score, precision and recall distributions for threat A, separately for the FQDN and second-level domain scenarios.

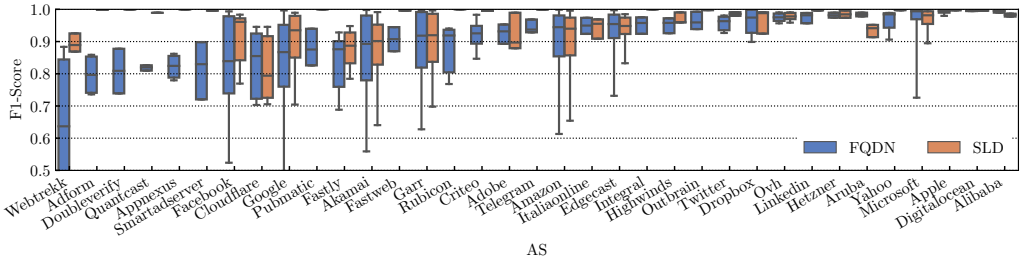
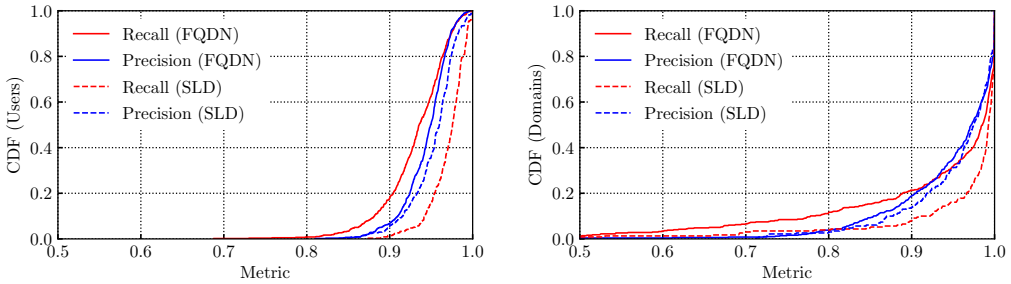


Fig. 3. Threat A: F1-Score distributions for fully qualified domain names (FQDN) and second-level domain names (SLD). Note that the y-scale starts from 0.5.

Figure 3 breaks down the performance on a per-AS level. The boxes report the distribution of F1-Scores. Black strokes represent the median, borders mark the 25^{th} and 75^{th} percentiles, and whiskers mark the 5^{th} and 95^{th} percentiles. Performance varies according to the AS. The 3^{rd} quartile of the distributions is above 0.75 for all ASes, and some ASes present median F1-Score close to 0.95. Curiously, the performance is not directly proportional to the number of labels – 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. All in all, the attack is very successful for the majority of ASes.

These experiments allow us to conclude that an attacker can easily recover the domain name of a flow by using simple traffic features. The results have been obtained with an off-the-shelf machine learning algorithm. In other words, the attack is very easy to be performed, as soon as ground truth data is available.

4.1.1 Second-level vs. full domain classification. We now evaluate the classification performance when using a coarse aggregation of class labels, i.e., limiting to the identification of the second-level domain names. This setup represents a scenario where attackers would be interested in identifying only specific services (e.g., `example.com`) regardless of the 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



(a) Threat B.

(b) Threat C.

Fig. 4. Precision and Recall for threats B and C. Note that the x-axes start from 0.5.

flows raises from 82% to 91%. This means that the attacker can now target more traffic, increasing the coverage of the attack.

Figure 2b shows the distribution of precision, recall and F1-Score in this second use case. Overall performance is better than what we obtain with full domain names. The ratio of classes having F1-Scores higher than 0.8 raises from 80% to 86%. Precision and recall improve consequently, with precision typically higher than recall again. The performance improvement can be associated with (i) the larger number of samples in the training set; (ii) grouping possibly similar flows with different FQDNs but same SLD into the same class, thus merging very similar classes which could be harder to classify. In Figure 3, the orange boxes break down the F1-Score by AS. For 11 of them, we find that all the traffic is served by a unique second-level name. As such, the classification becomes trivial and all flows are correctly identified. For the large majority of the remaining ASes, the performance with second-level domains is better than with full domains. 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.

4.2 Threats B and C

We now evaluate the effectiveness of the attack for threats B and C.

In threat B, the attacker aims to know the set of domains contacted by a given user, e.g., to build users' profiles [7]. The attacker uses the flows labeled by the classifier presented in the precious section, building the union of services visited by a given user. In Figure 4a we show the effectiveness of the attack in terms of precision and recall. The figure shows the distribution over the 594 users in the test set who contacted more than 100 domains during the target day. The solid lines refer to sets built for the FQDN scenario. The red line reports the recall: On 80% of cases recall is above 0.9 – i.e., the attacker's set includes more than 90% of the domains actually visited by 80% of users. Only for a handful of users recall is below 0.8. Precision (blue solid line) has even higher values, with 86% of users' sets above 0.9, meaning that 9 in 10 domains the attacker knows are actually associated with the user.

When the attacker is interested into second-level domains only (dashed lines) precision exhibits a similar distribution, while recall largely improves if compared to the FQDN scenario.

We conclude the attack is highly effective for threat B. An attacker can build accurate users' profiles based on the domains users contact, with only a few false positives and high completeness.

We now consider threat C, in which the attacker aims at building the set of users accessing a given domain. In this case, the attacker wants to enumerate the users visiting a sensitive website

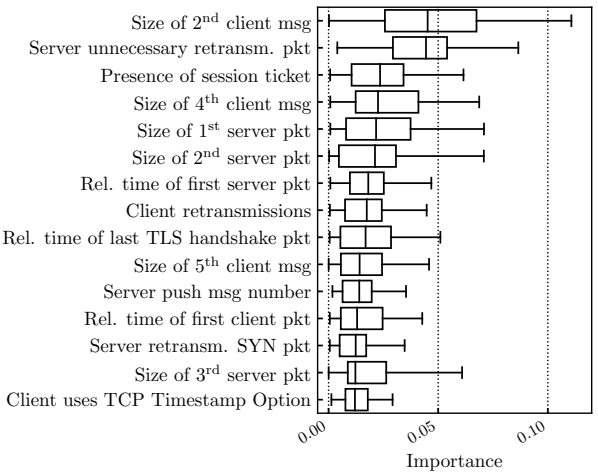


Fig. 5. Distribution of feature importance over different ASes for the top-15 features.

(e.g., pornography), or retrieving those devices running a particular software that communicates with a specific domain. In Figure 4b, we dissect the threat effectiveness showing the distribution of precision and recall over the 708 (409) FQDNs (second-level domain) we consider. Starting from the case of FQDNs (solid lines), we observe that precision and recall are above 0.9 roughly in 80% of cases. This means that for most domains the attacker can build a reliable and complete set of users. However, in a few cases the performance gets poor, showing that some domains are more challenging – in 9.5% of cases, recall is below 0.8, meaning that the attacker can retrieve less than 80% of users accessing the domain. Still, only 3.1% of domains present precision below 0.8, hinting that, in any case, the set of users for a domain is reliable (yet incomplete).

In the case of second-level domains, Figure 4b shows with dashed lines the distribution of precision and recall. With respect to the FQDN case (solid lines), precision has a little increase, while recall largely improves. In other words, the attacker can build a more complete set of users accessing a given second-level domain.

We conclude that also for threat C the attack is highly effective, and attackers can accurately build the set of users accessing a given domain.

4.3 Which features are the most informative?

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 features used by the classifier. As we use Random Forest models, we compute the feature importance by averaging the Gini impurity importance calculated over each tree [51]. The higher the Gini, the more information that feature brings.

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 the overall number of packets and bytes of each flow; (ii) *Advanced Flow Features*: Those computed from the analysis of the IP and TCP headers, and include per-flow packet loss, RTT, Time-To-Live, etc.; (iii) *First Flow Packets Features*: 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 TCP messages – i.e., groups of TCP segments delimited by a PSH TCP flag, often used by applications to delimit requests and responses.

Figure 5 details the top-15 most important features. Again, as we build an independent classifier for each AS, we report with boxplots the distribution of the importance of features across the models and rank them by the median values to ease the visualization.

The most important feature according to the above criterium is the size of the second client push-delimited message, likely carrying part of the TLS handshake. We find other six features based on packets or message size. To check if these top features would suffice to train a classifier, we run an experiment in which we use only the top 15 features and observe a marginal penalty in the overall performance, confirming packet size plays a major role.

Interestingly, we observe high importance for deployment-dependent features, such as the number of TCP retransmissions. That is, models seem to consider peculiarities of the network used to reach target ASes – transferring such models to different deployments may be hard and impair performance. Unfortunately, we do not have datasets collected in different networks to validate this conjecture, and thus leave the study of the spatial stability of models to future work.

Possible protections against these attacks require traffic signatures to be masked. Blurring packet sizes and packet inter-arrival times, e.g., by padding packets and adding random packet delays, could help to limit fingerprints. Our results show that such protections must be performed at the transport protocol level, and not only by modifying the DNS and TLS protocols. We investigate different padding countermeasures in the next section.

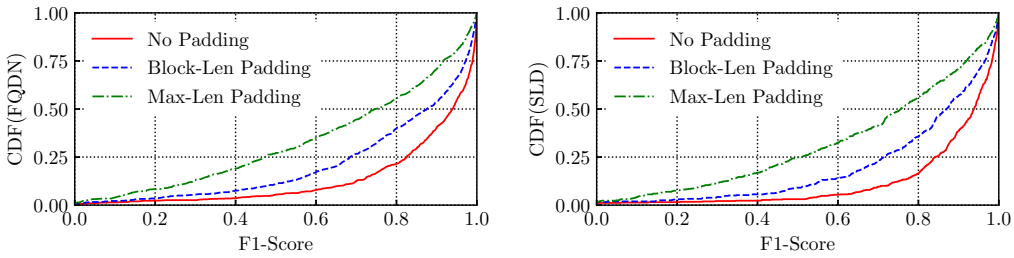
5 PADDING-BASED ATTACK MITIGATION

We now investigate to what extent padding-based mitigation techniques are effective against threats A, B, C. We evaluate two padding techniques contained in RFC 8467 [13], namely Maximal-Length Padding and Block-Length Padding. To this end, we post-process the collected Tstat [47] log files to compute the flow features that would have been observed in case padding was in place. In particular, we modify the sizes of the first 10 packets on server and client side according to the padding algorithm. Moreover, we estimate the remaining features (e.g., flow size) based on the new packet size values, assuming padding is employed for the whole flow lifespan.

5.1 Maximal-Length Padding

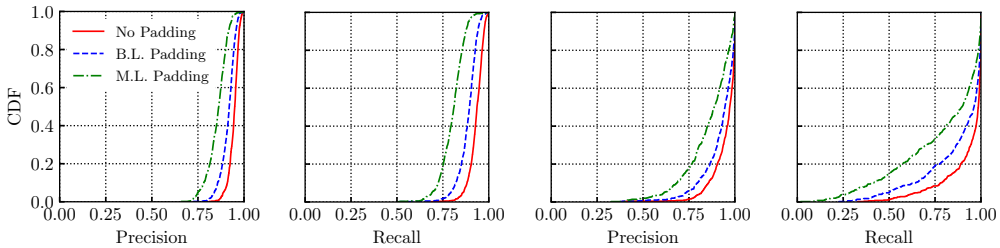
In Maximal-Length Padding, the sender pads every message to the maximum size allowed by the underline network, 1500 Bytes in our case. This strategy provides the highest level of confidentiality as no information can be derived from packet sizes. The drawback of this technique is the large overhead, as packets with maximum size would consume more resources. To evaluate its impact on threat A, we recompute all the features of our trace and then apply the same attack methodology employed before, assuming both training and testing are run on the padded flows.

We show the distribution of the F1-Score across domains in Figure 6 separately for the FQDN and second-level domain scenarios. For comparison, we report with red solid lines the results of the original non-padded features. Starting from the FQDN case (Figure 6a), we observe a sizable drop in performance. The median F1-Score is reduced from 0.94 to 0.75 and the number of FQDNs with scores higher than 0.8 decreases from 726 to 412. However, we still classify a consistent fraction of domain names with relatively good performance – 44 % (29 %) with F1-Score above 0.8 (0.9). Similar considerations hold for second-level domains (Figure 6b). Median F1-Score decreases from 0.93 to 0.76, but 44% of second-level domains can still be classified with a score above 0.8. Not shown here, precision is typically greater than recall, similarly to what we showed in Figure 2.



(a) FQDN. (b) SLD.

Fig. 6. F1-Score distributions for threat A, separately for the FQDN and second-level domain scenarios, with different padding settings.



(a) Threat B. (b) Threat C.

Fig. 7. Precision and Recall for threats B and C with and without Maximal-Length Padding (FQDN case).

Next, we observe feature importance when building the classifier with padded packets. In Figure 8 we show the top-15 features. Compare them with those of Figure 5. We observe how packet sizes are now completely neglected, as they bring no information. Conversely, the classifiers now leverage mostly packet timing, flow characteristics such as flow duration and other information coming from the TCP headers (e.g., the declared receiver window). Hence, if padding prevents a classifier from exploiting packet sizes, other flow characteristics still offer useful information.

As a consequence of the lower effectiveness of threat A, padding has an effect on threats B and C too. In Figure 7, we report the distributions of precision and recall, comparing the scenario with no padding (solid red lines) and Maximal-Length Padding (dashed green line). Here we restrict results to the FQDN scenario only. Starting from threat B (Figure 7a), we observe that both precision and recall still assume high values for most users. In 99% (94%) of cases, precision (recall) is above 0.7. Similar considerations hold for threat C (Figure 7b).

Moving on the overhead, Maximal-Length Padding comes with a high cost in terms of additional traffic. It would increase traffic volume by 13.4% downstream and 25.4% upstream – 16.7% overall.

We conclude that Maximal-Length Padding only partially helps to protect users from threat A, as the attacker can still obtain reasonable accuracy for a large portion of domains. Maximal-Length Padding has little positive effects for threats B and C, and the attacker could still build accurate users' profiles or site audiences even with padded traffic.

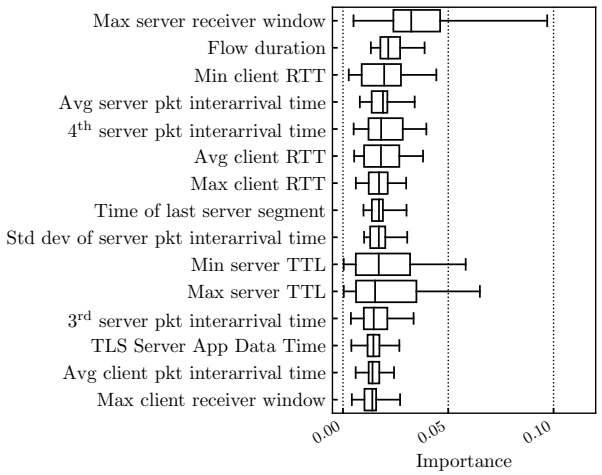


Fig. 8. Distribution of feature importance over different ASes for the top-15 features with Maximal-length Padding.

5.2 Block-Length Padding

We now evaluate a more relaxed padding strategy that comes with lower costs in terms of consumption of unnecessary resources. With Block-Length Padding, the sender should pad queries to the closest multiple of a fixed amount of Bytes. This strategy greatly reduces the variability of packet sizes, as the observable values are limited to multiples of the chosen constant.

Very large block lengths will result similar to the Maximal-Length Padding strategy, while small values allow large variability of packet sizes. We re-compute the traffic feature as if Block-Length Padding with 128B block size was in place. We show these results in Figure 6 with green dashed lines. We observe an intermediate performance between the non-padded and Maximal-Length Padding scenarios.

Considering the FQDN case (Figure 6a), the median F1-Score is 0.87, while 0.93 for the non-padded dataset and 0.75 in case of Maximal-Length Padding. Similar is the picture for second-level domains. Also Block-Length Padding increases resource consumption, even if less markedly than Maximal-Length Padding. In our dataset, traffic volume would increase overall by 5%.

Again, attacks are highly effective with Block-Length Padding for threats B and C. We conclude that Block-Length Padding can barely mitigate the threats, while still coming with a 5% additional network load. Note that other padding strategies, e.g., random padding, would offer worse protection than full size padding.

6 IMPACT OF TRAINING DATA

We now discuss the impact of training data from different points of view. We first quantify the impact of training data freshness and size. Next, we investigate the limits of using web crawlers to obtain a labeled dataset for training the classifiers. We illustrate the impact for threat A only.

6.1 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

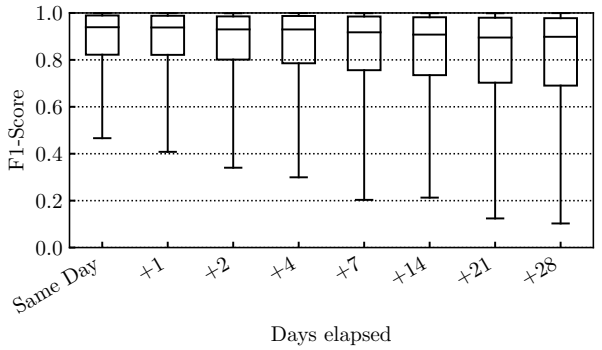


Fig. 9. F1-Score distributions (FQDNs) when varying the interval elapsed between training and testing data.

the freshness of training data is important to the classification performance. We train the classifiers on 50% of clients seen on the first day (FQDNs). Then we use the remaining days for testing, using only clients that were not part of the training.

Figure 9 shows the distribution of F1-Scores among domains when increasing the time interval between training and testing data. 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 25^{th} percentile drops from 0.83 (left-most boxplot) to 0.69 (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 decrease only slightly, from 0.94 (same day training and testing) to 0.90 (4-week gap between training and testing).

Notice that domain popularity does not radically change over time in our dataset, and, as such, the percentage of the overall traffic our 708 domains cover remains almost unchanged. Indeed, on the first day, the attack covers 82% of traffic (see Table 1), while the same set of domains accounts for 80% of traffic after 28 days. In sum, the set of popular names does not change as time passes, and the coverage of the attack remains very high. Yet, the attacker should update the models to keep high performance for some domains.

6.2 How many flows in the training set?

So far we have assessed performance after training the classifiers with at least 1 000 flows per name. We now quantify the impact of the number of observations on performance. To this end, we reduce the required number of observations to 50.

Figure 10 shows distributions of F1-Scores grouping domains by the numbers of observations in the training set. 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 build good models.

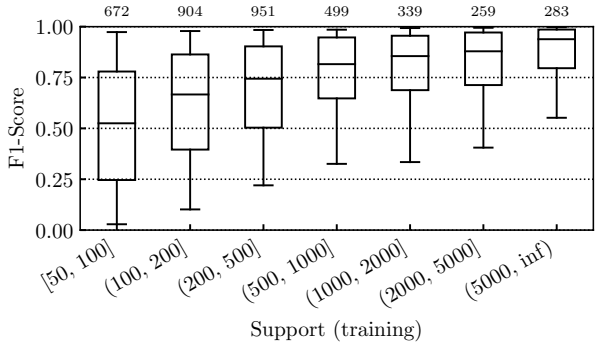


Fig. 10. F1-Score distributions according to the number of flows in the training set.

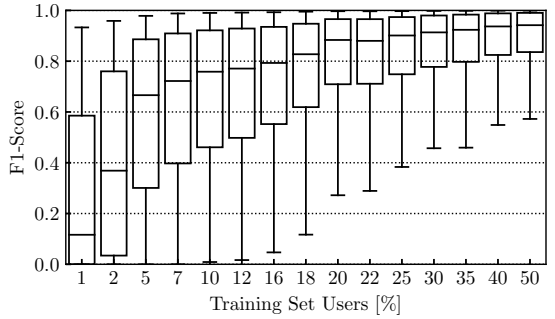


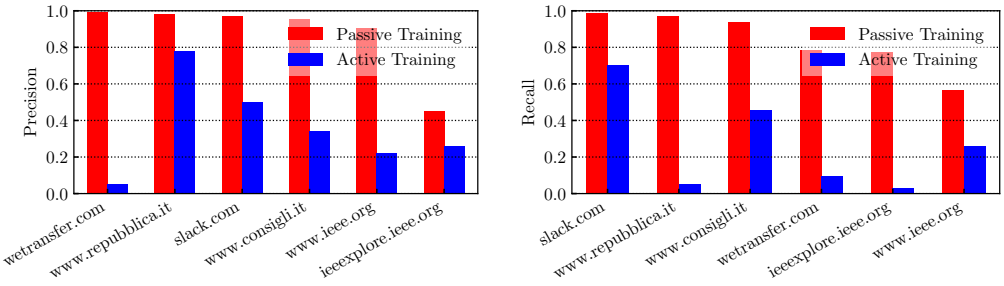
Fig. 11. F1-Score distributions when using a percentage of clients in the training set.

6.3 How many clients in the training set?

We complement the above analysis by studying the effects of varying the number of clients in the training set. We randomly select clients among those that we used in the previous sections. Remind that our dataset contains the traffic generated by 3 995 campus IP addresses.

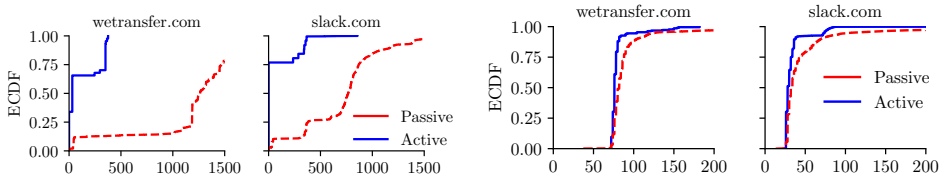
In Figure 11 we show the F1-Score distributions when limiting the number of clients in training set, randomly adding a batch of clients at each step. The left-most box reports the F1-Scores we obtain when restricting the training set to 1% of the clients – i.e., we consider the traffic from 40 IP addresses. The performance is radically worse with a median F1-Score of 0.11. However, some domains (14%) are already identified with a score greater than 0.8. The larger the set of clients in the training set is, the better performance is. With 200 clients (5% of the total) the median F1-Score is already 0.66, while half of the domains exhibit scores greater than 0.8 when including 720 (18%) of the clients in the training set. Further increasing the training set causes marginal benefits.

We conclude that the threat effectiveness depends on the number of clients the attacker leverages to build the training set. A few hundreds of clients are already enough to achieve high performance.



(a) Precision. (b) Recall.

Fig. 12. Precision and Recall for Threat A when building the training set through active crawlers on 6 websites.



(a) Size of 2nd client message. (b) Relative time of last TLS handshake packet.

Fig. 13. Distribution of the feature values for data collected passively and with active crawlers for two websites. Similar differences are observed for other features and also for other websites.

6.4 Training using Web Crawlers

Finally, we investigate the use of training sets built using crawlers that automatically visit the target websites. In this case, the attacker uses browser automation tools to generate requests to target domains and use the resulting traffic to train the classification models. The spectrum of possible techniques for this purpose is wide. Here, our goal is to quantify to what extent the use of well-known tools for automatic browsing is effective when building a training dataset.

We target 6 popular websites that are hosted on the Akamai and Amazon ASes. We visit each website homepage and 10 internal pages with BrowserTime while saving all network traffic observed during the crawling. To verify the impact of the browser cache, we run the crawler with cold cache first, and then repeat the visit with warm cache.

As done before, we train a separate classifier for Akamai and Amazon ASes using the traffic collected during crawling (including both cold and warm sessions). We then use the passive dataset containing half of the Campus users as the test set to evaluate the effectiveness of the attack considering threat A.

In Figure 12 we compare Precision and Recall obtained with the active training data (blue bars) to those obtained with the training set composed by half of the campus clients (red bars). Results are much poorer with active training. Only in one case precision is close 0.8 (`www.repubblica.it`). For `wetransfer.com` precision is close to 0. The remaining websites exhibit precision in the order of $0.2 - 0.5$. Instead, we obtain precision above 0.8 in 5 cases out of 6 cases when using passively collected training sets. The drop is even more pronounced when considering recall, with 3 out of 6

website with recall below 0.1. Passive training, instead, leads to recall close to 0.8 in all cases but `www.ieee.org`.

We conclude that using an off-the-shelf methodology to actively create a training set is far less effective than using data from real users. Although we run more than 36 000 visits, the effectiveness of the attack remains poor. We took care of extensively running active crawlers under different conditions, but apparently these steps have not been sufficient to recreate the heterogeneity and diversity of the passive dataset. In a nutshell, the features extracted from the dataset built with active crawlers are not representative of the actual traffic generated by real users.

To corroborate this intuition, in Figure 13 we compare the distribution of the values for 2 features that are among the most important ones in both the passive and active datasets. To ease visualization, we focus only on two websites, namely `wetransfer.com` and `slack.com`. Considering the size of the 2nd client message in Figure 13a, it is clear that the active crawler (solid blue line) is not able to reproduce the variability observed in the passive data (dashed red line). Similar considerations hold to the duration of the TLS handshake in Figure 13b. Here the active crawler runs on a wired network, where the access delay and loss probability are likely diverse from what is observed in the passive dataset, in which users access the internet from WiFi and wired links. As a result, the TLS handshake is generally shorter and more deterministic in the active case than in the passive data.

In a nutshell, passively collected traces naturally factor the heterogeneity that impacts the traffic generated by a browser. These factors include client configurations (device type, OS type, browser setup and eventual plugins, etc.), network configurations (access type, DNS configuration, performance and congestion, etc.) and user habits (first/second access, cookies, etc.). For instance, among the causes that may limit the effectiveness of active crawling, we suppose the privacy banners that a user has to accept before fully access the website content plays an important role. Tools like Priv-Accept [52] may help the attackers to build more realistic datasets. To mount a reliable attack, one shall consider all these factors, generating a mix of traffic as similar as possible to the traffic observed in the target network. Studying such techniques requires a different (and more complex) approach that we leave for future work.

7 DISCUSSION AND LIMITATIONS

Our experiments show that simple machine learning solutions allow one to mount an attack to unveil the domain name of flows protected by DoH and ECH. Our methodology exploits flow characteristics and packet-level features, which let the ML model achieve high performance when enough training data is available.

We have shown that the approach is feasible if attackers can observe traffic from unprotected users to train the classifier since users navigating without DoH or ECH are a rich source for attackers to build the ground truth for the training of ML models. However, attackers must be able to deploy passive monitoring systems, and such data collection must happen in vantage points where the traffic of several users is aggregated, e.g., at ISPs or large corporate networks. Attackers must be powerful enough to obtain access to such a privileged source of data.

Moreover, our experiments show that the amount of training data impacts the attack performance. In particular, the probability of correctly identifying the domain name of an encrypted flow is remarkably correlated to the support of that domain in the training set (see Figure 10). In fact, the training data come from passive measurements, so their mixture is not under the attacker's control. The need to observe a sufficient number (i.e., thousands) of requests to the target domains limits the attack to popular domains. That is, collecting data from passive measurements can be ineffective if the target website is unpopular.

Alternatively, attackers may build the training set using active web crawlers, facilitating the training for specific domains of interest. However, we showed that the effectiveness of such an

attack is limited because of the difficulties of reproducing the heterogeneous user habits with active crawlers. To overcome this limitation, attackers could rely on manual (and time-consuming) browsing or crowd-sourcing to mimic more realistic browsing habits. Building such a training set would again be very complex, so powerful actors could only use it.

8 CONCLUSION

We explored to what extent an eavesdropper can uncover the server domain names associated to traffic flows of users relying on DoH and/or ECH. We showed that such an attack is easy to mount if enough traffic of unprotected users is available to build a training set. Conversely, using active crawlers results less effective. We showed that website audience and users' domain history can be recovered with even higher accuracy using the same approach. The sizes and timings of the first packets of traffic flows are among the most useful features to mount the attack. Unfortunately, blurring the information offered by packet size (using padding) results insufficient to protect privacy as other features allow to build solid models.

Our results show the limits of privacy-preserving enhancements currently under standardization, such as DoH and ECH. Results call for further actions to protect the domain of servers contacted by users. These actions require deep changes in transport protocols, such as the inclusion of padding together with random inter-packet pacing, or the widespread adoption of solutions such as VPNs and ToR.

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] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack," Tech. Rep. 7528, RFC Editor, 2014.
[2] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," Proc. of the CoNEXT, pp. 133–140, 2014.
[3] B. Anderson and D. McGrew, "TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior," Proc. of the IMC, pp. 379–392, 2019.
[4] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)," Internet-Draft draft-ietf-quic-http-34, Internet Engineering Task Force, Feb. 2021. Work in Progress.
[5] T. Böttger, F. Cuadrado, G. Antichi, E. Fernandes, G. Tyson, I. Castro, and S. Uhlig, "An Empirical Study of the Cost of DNS-over-HTTPS," Proc. of the IMC, pp. 15–21, 2019.
[6] I. Bermudez, M. Mellia, M. Munafò, R. Keralapura, and A. Nucci, "DNS to the Rescue: Discerning Content and Services in a Tangled Web," Proc. of the IMC, pp. 413–426, 2012.
[7] L. Vassio, D. Giordano, M. Trevisan, M. Mellia, and A. Silva, "Users' Fingerprinting Techniques from TCP Traffic," Proc. of the Big-DAMA, pp. 49–54, 2017.
[8] P. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," Tech. Rep. 8484, RFC Editor, 2018.
[9] C. Huitema, S. Dickinson, and A. Mankin, "DNS over Dedicated QUIC Connections." RFC 9250, May 2022.
[10] M. Kosek, L. Schumann, R. Marx, T. V. Doan, and V. Bajpai, "Dns privacy with speed? evaluating dns over quic and its impact on web performance," in *Proceedings of the 22nd ACM Internet Measurement Conference*, pp. 44–50, 2022.
[11] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Internet-Draft draft-ietf-tls-esni-13, Internet Engineering Task Force, Aug. 2021. Work in Progress.
[12] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE network*, vol. 26, no. 1, pp. 35–40, 2012.
[13] A. Mayrhofer, "Padding Policies for Extension Mechanisms for DNS (EDNS(0))," Tech. Rep. 8467, Oct. 2018.
[14] M. Trevisan, F. Soro, M. Mellia, I. Drago, and R. Morla, "Does domain name encryption increase users' privacy?," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 3, pp. 16–22, 2020.
[15] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," Tech. Rep. 7858, RFC Editor, 2016.

[16] S. Dickinson, D. K. Gillmor, and T. Reddy.K, "Usage Profiles for DNS over TLS and DNS over DTLS." RFC 8310, Mar. 2018.

[17] B. M. Schwartz, M. Bishop, and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)," Internet-Draft draft-ietf-dnsop-svcb-https-07, Internet Engineering Task Force, Aug. 2021. Work in Progress.

[18] A. Hintz, "Fingerprinting Websites using Traffic Analysis," Proc. of the PET, pp. 171–178, 2003.

[19] Y. Shi and S. Biswas, "Website Fingerprinting using Traffic Analysis of Dynamic Webpages," Proc. of the GLOBECOM, pp. 557–563, 2014.

[20] X. Gu, M. Yang, and J. Luo, "A Novel Website Fingerprinting Attack against Multi-tab Browsing Behavior," Proc. of the CSCWD, pp. 234–239, 2015.

[21] D. Arp, F. Yamaguchi, and K. Rieck, "Torben: A Practical Side-Channel Attack for Deanonymizing Tor Communication," Proc. of the ASIA CCS, pp. 597–602, 2015.

[22] R. Gonzalez, C. Soriente, and N. Laoutaris, "User Profiling in the Time of HTTPS," Proc. of the IMC, pp. 373–379, 2016.

[23] S. Feghhi and D. Leith, "A Web Traffic Analysis Attack Using Only Timing Information," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1747–1759, 2016.

[24] M. Lescisin and Q. Mahmoud, "Tools for Active and Passive Network Side-Channel Detection for Web Applications," Proc. of the WOOT, 2018.

[25] B. Miller, L. Huang, A. Joseph, and J. Tygar, "I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis," Proc. of the PET, pp. 143–163, 2014.

[26] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated Website Fingerprinting through Deep Learning," Proc. of the NDSS, 2018.

[27] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning," Proc. of the PET, pp. 292–310, 2019.

[28] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning," Proc. of the CCS, pp. 1928–1943, 2018.

[29] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting," Proc. of the USENIX Security, pp. 143–157, 2014.

[30] D. Plonka and P. Barford, "Flexible Traffic and Host Profiling via DNS Rendezvous," Proc. of the SATIN, pp. 1–8, 2011.

[31] T. Mori, T. Inoue, A. Shimoda, K. Sato, S. Harada, K. Ishibashi, and S. Goto, "Statistical Estimation of the Names of HTTPS Servers with Domain Name Graphs," *Computer Communications*, vol. 94, pp. 104–113, 2016.

[32] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu, "An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come?," Proc. of the IMC, pp. 22–35, 2019.

[33] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, "Comparing the effects of dns, dot, and doh on web performance," in *Proceedings of The Web Conference 2020*, pp. 562–572, 2020.

[34] T. V. Doan, I. Tsareva, and V. Bajpai, "Measuring dns over tls from the edge: adoption, reliability, and response times," in *International Conference on Passive and Active Network Measurement*, pp. 192–209, Springer, 2021.

[35] R. Houser, Z. Li, C. Cotton, and H. Wang, "An Investigation on Information Leakage of DNS over TLS," Proc. of the CoNEXT, 2019.

[36] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted DNS–> Privacy? A Traffic Analysis Perspective," Proc. of the NDSS, 2020.

[37] D. Vekshin, K. Hynek, and T. Cejka, "Doh insight: Detecting dns over https by machine learning," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pp. 1–8, 2020.

[38] K. Hynek and T. Cejka, "Privacy illusion: Beware of unpadded doh," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0621–0628, IEEE, 2020.

[39] J. Cheng, R. He, E. Yuepeng, Y. Wu, J. You, and T. Li, "Real-time encrypted traffic classification via lightweight neural networks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[40] J. Bushart and C. Rossow, "Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*, 2020.

[41] "Alec Muffett. Dohot: making practical use of dns over https over Tor." https://github.com/alecmuffett/dohot. Accessed: 2021-02-15.

[42] S. Singanamalla, S. Chunhapanya, M. Vavruša, T. Verma, P. Wu, M. Fayed, K. Heimerl, N. Sullivan, and C. Wood, "Oblivious dns over https (odoh): A practical privacy enhancement to dns," *arXiv preprint arXiv:2011.10121*, 2020.

[43] N. P. Hoang, A. Akhavan Niaki, N. Borisov, P. Gill, and M. Polychronakis, "Assessing the privacy benefits of domain name encryption," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 290–304, 2020.

[44] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE symposium on security and privacy*, pp. 332–346, IEEE, 2012.

[45] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *European Symposium on Research in Computer Security*, pp. 18–33, Springer, 2006.
[46] M. Trevisan, I. Drago, M. Mellia, and M. Munafo, "Towards Web Service Classification using Addresses and DNS," Proc. of the TRAC, pp. 38–43, 2016.
[47] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, and D. Rossi, "Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 163–169, 2017.
[48] W. Aqeel, B. Chandrasekaran, A. Feldmann, and B. M. Maggs, "On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement," in *Proceedings of the ACM Internet Measurement Conference*, pp. 680–695, 2020.
[49] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.
[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, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[51] C. Strobl, J. Malley, and G. Tutz, "An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests.," *Psychological methods*, vol. 14, no. 4, p. 323, 2009.
[52] N. Jha, M. Trevisan, L. Vassio, and M. Mellia, "The internet with privacy policies: measuring the web upon consent," *ACM Transactions on the Web (TWEB)*, vol. 16, no. 3, pp. 1–24, 2022.

APPENDIX

For completeness, we report in the following table the number of flows for the considered ASes together with the number of observed unique FQDNs and SLDs. The table includes numbers for the first day of our dataset since most analyses of the paper are carried out over that period.

Table 2. Total number of flows and overall number of FQDNs and SLDs for the considered ASes.

AS	Flows	FQDN	SLD	AS	Flows	FQDN	SLD
Adform	21 340	14	3	Google	2 110 966	5 743	1 147
Adobe	16 056	438	153	Hetzner	45 456	1 185	839
Akamai	658 229	4 517	1 848	Highwinds	26 898	255	152
Alibaba	26 395	359	100	Integral	66 804	7	2
Amazon	1 044 532	14 572	5 380	Italiaonline	21 515	90	13
AppNexus	95 127	22	7	LinkedIn	27 677	57	5
Apple	420 841	205	12	Microsoft	1 476 443	3 470	534
Aruba	26 475	1 176	966	OVH	49 674	1 587	1 171
Cloudflare	110 041	5 845	4 069	Outbrain	21 809	14	3
Criteo	103 048	33	4	PubMatic	16 094	12	1
DigitalOcean	44 995	605	488	Quantcast	12 562	12	2
DoubleVerify	15 048	5	2	Rubicon	38 171	13	1
Dropbox	120 941	5 254	4	SmartAdServer	16 952	13	5
Edgecast	95 220	678	312	Telegram	23 312	18	6
Facebook	854 450	305	23	Twitter	20 830	21	2
Fastly	147 863	2 590	823	Webtrekk	21 266	48	14
Fastweb	13 764	243	153	Yahoo	32 277	83	13
GARR	214 702	715	145	**TOTAL**	**4 457 982**		