Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS

Jonas Bushart
CISPA Helmholtz Center for Information Security

Christian Rossow
CISPA Helmholtz Center for Information Security

Abstract

DNS over TLS (DoT) and DNS over HTTPS (DoH) encrypt DNS to guard user privacy by hiding DNS resolutions from passive adversaries. Yet, past attacks have shown that encrypted DNS is still sensitive to traffic analysis. As a consequence, RFC 8467 proposes to pad messages prior to encryption, which heavily reduces the characteristics of encrypted traffic. In this paper, we show that padding alone is insufficient to counter DNS traffic analysis. We propose a novel traffic analysis method that combines size and timing information to infer the websites a user visits purely based on encrypted *and padded* DNS traces. To this end, we model *DNS Sequences* that capture the complexity of websites that usually trigger dozens of DNS resolutions instead of just a single DNS transaction. A closed world evaluation based on the Tranco top-10k websites reveals that attackers can deanonymize test traces for 86.1 % of all websites, and even correctly label all traces for 65.9 % of the websites. Our findings undermine the privacy goals of state-of-the-art message padding strategies in DoT/DoH. We conclude by showing that successful mitigations to such attacks have to remove the entropy of inter-arrival timings between query responses.

1 Introduction and Background

DNS is one of the most fundamental protocols on the Internet because it is the starting point of nearly all Internet traffic. It translates memorable names into cryptic IP addresses. One critical path for privacy is between the client and the resolver where several entities, such as Wi-Fi access points or Internet Service Providers (ISPs) can eavesdrop. They can use this information for advertisement or to create browsing profiles of the victims. Upon visiting a website by the user, the client sends one or multiple DNS queries to the resolver, which can either answer them from the cache or they perform the iterative lookup to query the Authoritative Name Server (AuthNS). This means that DNS leaks almost all user behavior to any eavesdropper.

Users are understandably worried about privacy, yet for years DNS did not have any means to protect the integrity and confidentiality of DNS messages. Partial solutions like DNSSEC [1, 7] exist, but do not provide confidentiality. DNS over TLS (DoT) [14] and DNS over HTTPS (DoH) [12] close the gap and finally provide confidential DNS and are supported in resolvers [6], browsers [2, 22, 29] and operating systems [17, 18].

Yet, similar to other privacy-preserving communication systems that leverage encryption (e.g., HTTPS [25, 33] or Tor [23, 28, 31]), DoT and DoH are susceptible to traffic analysis. Gillmor's empirical measurements [10] show that passive adversaries can leverage the mere size of a single encrypted DNS transaction to narrow down the queried domain. RFC 8467 [21] then follows Gillmor's suggestions to pad DNS queries and responses to multiples of 128 B / 468 B.

In this paper, we study the privacy guarantees of this widely-deployed DoT/DoH padding strategy. We assume a passive adversary (e.g., ISP) who aims to deanonymize the DNS resolutions of a Web client. Modern Web services are intertwined, such that visiting a website, creates many DNS queries. While this padding strategy destroys the size entropy of messages, we assess to what extent DNS resolution *sequences* (e.g., due to third-party content) allow adversaries to reveal the Web target. Such sequences utilize DNS *message sizes and timing information* between DNS transactions.

We then leverage a k-Nearest Neighbors classifier to search for the most similar DNS transaction sequences in a previously-trained model. Our closed world evaluation shows that an attacker can deanonymize 86.1 % of all test traces and we show how DNS provides a better basis for performing subpage-agnostic domain classification. These findings undermine the privacy goals of state-of-the-art message padding strategies in DoT/DoH, which is highly critical.

Two previous papers [13, 27] also address DoT/DoH. [13] addresses DoT by using statistical features of sizes and timing. In their most similar setting, they report 83 % correct classifications for a dataset of only 98 websites. [27] attacks DoH without utilizing timing, which we found to be very

important. For a comparable dataset (1500 domains) they report 94.0 % precision, but they also report that DoH is easier to classify. Compared to both papers we use the largest dataset (10k domains) and address *subpage-agnostic classification* and evaluate *countermeasures*.
Summarizing, we provide the following contributions:

1. We illustrate a traffic analysis attack that leverages DNS transaction *sequences* to reveal the website a client visits.
2. We provide an extensive analysis of the privacy guarantees offered by DNS message padding (RFC 8467 [21]) against our attack in a Web browsing context. We demonstrate severe privacy losses even against passive adversaries that sniff on encrypted and padded DNS traffic and *extend* the analysis beyond just the index page.
3. We are the first to evaluate alternative padding strategies and constant-rate communication systems against our proposed attack. We reveal that even perfect padding cannot mitigate traffic analysis, and show that any promising countermeasure needs to obfuscate timing.

2 Traffic Analysis and DNS Padding

We now present how an adversary can use traffic analysis to infer the browsing target purely based on encrypted DNS traffic. Encrypted DNS only leaves three characteristics, which can be used to infer the communication content, namely (i) counts, such as packets, (ii) sizes, such as overall transmitted bytes, and (iii) time. These three dimensions provide valuable hints about the communication content. Two standards describe how to add padding to reduce size information: RFC 7830 [20] describes how to add padding to DNS messages and RFC 8467 [21] recommends a padding scheme of padding all queries to a multiple of 128 B, while all responses are padded to a multiple of 468 B.

Encrypted DNS only protects against eavesdroppers between client and resolver, but assumes the resolver is trusted. Upstream communication of the resolver towards the AuthNSs is not covered in this threat model. The resolver will see all communication in plaintext and has to be trusted to uphold privacy. We follow this threat model and assume a passive attacker which can observe the communication between client and resolver, yet not delay, alter, or inject data into the traffic. The attacker is allowed to initiate their own network connections from the same network-topological location as the victim.

2.1 DNS Sequences

The padding recommendations are based on single query/response pairs. We show that attackers can leverage a *sequence* of DNS query/response pairs, to increase the uniqueness compared to individual DNS transactions. Web browsing causes the sequences of DNS queries for many reasons, such as redirects, loading of third-party resources, or resources from subdomains. As we will show, these sequences characterize a user's website visit quite well.

Consider a visit to "wikiquote.org" which triggers four DNS requests/responses upon visit. It fetches the IP address for the domain, then after 287 ms for the "www" subdomain, and after another 211 ms for both the "meta" and "upload" subdomains simultaneously. The resulting DNS Sequence (which ignores requests) looks like `Msg(1), Gap(8), Msg(1), Gap(7), Msg(1), Msg(1).` The DNS Sequence encodes the four DNS responses (each 468 B long, i.e., one padding block). If there is a time gap between two responses, we note this gap and its magnitude using a millisecond log scale (e.g., $\lfloor log_2(287) \rfloor = 8$). This make it less susceptible to timing variations, e.g., due to network jitter. We remove all time gaps with a numerical value ≤ 0 (i.e., those shorter than 1 ms), such as between the last two Msgs.

The DNS Sequence only includes DNS responses, since we found the DNS queries to have almost no entropy and their timing is highly correlated with the replies. We chose to represent message sizes and timestamps abstractly as this provides higher flexibility and generalizes over different implementations and events outside of our control (e.g., network performance, jitter). Overall, this simplifies our design while keeping most features.

DNS Sequence Extraction: We derive the DNS Sequences from encrypted and padded traffic. We identify a DNS carrying connection using ports (853 for DoT), IP addresses (e.g., `9.9.9.9`), the TLS handshake [15, 16], or DNS-like characteristics (like packet sizes). Then we reassemble these TCP streams and extract "application data" TLS records. After some data cleanup, like merging consecutive records and ignoring overhead such as certificates, we only keep the message sizes and inter-arrival times of the DNS messages.

2.2 DNS Sequence Classifier

Our classifier uses k-Nearest Neighbors (k-NN) to assign labels to DNS Sequences based on the labels of the nearest neighbors, i.e., the DNS Sequences most like the unlabeled one. This assumes that similar DNS Sequences belong to the same website. The k parameter specifies how many neighbors should be searched and the plurality of the k found labels determines the output classification.

We base the distance function upon the Damerau-Levenshtein/edit distance [5, 19]. It counts the edit operations required to turn one sequence into another using the four operations insertion, deletion, substitution, and transposition. We assign each operation a different cost based on the importance of each operation. For example changes to the volatile timing information have a lower cost than to the rather stable size information. We optimize these constants using a hyper-parameter search over a subset of our closed world dataset.

3 Evaluation

We will now evaluate the efficacy of our proposed methodology to classify DNS Sequences that we obtain from traffic captures. We shortly describe our dataset generation setup and then introduce two evaluation scenarios. The closed world scenario shows the baseline classification performance, while the subpage-agnostic domain classification extends the classification beyond the index page.

3.1 Measurement Setup

We create our dataset by visiting the top websites in the Tranco list [30] and recording the DNS traffic. We use a server with a 10 GB/s network interface running Debian 9.11. For the closed world scenario we also collect data using a Raspberry Pi 3 running Raspbian 10. The low power Pi allows us to measure the effect of hardware performance on dataset collection and classifier.

For each website we spawn a Docker container running Firefox 72 and Unbound 1.9.4 as the DNS stub resolver. We configure Unbound to forward all DNS queries using DoT to Cloudflare's resolvers at `1.0.0.1` and `1.1.1.1`. Unbound's DNS cache is preloaded with the `NS` entries for all TLDs since we assume a user will have these records cached due to past resolutions. We control Firefox using Selenium. Measurements are repeated up to two times if we detect errors, such as missing DNS traffic or HTTP errors. Lastly, we convert the network traffic into a DNS Sequence as described in Section 2.1 and label it based on the website. We group identical websites under a single label, such as websites using different TLDs for their localized versions.

3.2 Evaluation Results

We evaluate our classifier in two settings. First, the *closed world*, which highlights the best case for an attacker, and provides a baseline for the performance and second, we perform *subpage-agnostic domain classification*, where we classify more than just the index page.

3.2.1 Closed World Scenario

The *closed world* scenario is the easiest for the attacker, since all websites a client can visit are known in advance. The attacker only needs to decide which known website is visited.

Our dataset consists of the top 10000 websites from the Tranco list [24, 30] (2019-08-27) for which we collect ten samples each. For 9235 websites we could collect this data, the others repeatedly caused errors, and we could not collect ten DNS Sequences. A seconds set of DNS Sequences are collected on the Raspberry Pi, for which we collect four traces per website. Data for 7699 websites was collected successfully.

This fully labeled dataset allows us to use cross-validation to measure our classifier. We use 10-fold cross-validation between all the DNS Sequences, since we have exactly ten traces per website and measure the percentage of correctly classified DNS Sequences and show the results in Table 1. Our k-NN classifier achieves up to 86.1 % on the server dataset and 80.9 % for the Raspberry Pi dataset, both with $k = 1$. We notice that higher k's slightly reduce the performance of the classifier, but not significantly, so even with $k=9$ the performance only falls to 82.9 % for the server dataset.

Table 1: Percentage of correctly classified DNS Sequences in the closed world scenario with different classifiers and datasets.

Classifier	Server	Raspberry Pi
k-NN ($k=1$)	86.1 %	80.9 %
k-NN ($k=3$)	85.6 %	79.0 %
NN	81.4 %	63.5 %

Table 2: Per website results for fixed $k = 1$ in the closed world scenario. The table show the percentage of websites we can re-identify in how many of the 10 data points/traces.

$^{n}/_{10}$ traces	$^{1}/_{10}$	$^{2}/_{10}$	$^{3}/_{10}$	$^{4}/_{10}$	$^{5}/_{10}$	$^{6}/_{10}$	$^{7}/_{10}$	$^{8}/_{10}$	$^{9}/_{10}$	$^{10}/_{10}$
Precision	92.2	91.8	91.3	90.8	89.9	88.5	86.9	84.4	79.6	65.9

The per-website results in Table 2 show how well we can classify websites instead of DNS Sequences. We achieve a perfect classification (i.e., $^{10}/_{10}$) for about two thirds of all websites. If we relax the requirements to 90 % correct classification per website, we can classify about 80 % correctly—purely based on *encrypted and padded* DNS traffic.

We build a second type of classifier using Neural Networks (NNs), also shown in Table 1. It performs in the same ballpark for the server dataset (81.4 %), but falls to only 63.5 % for the Raspberry Pi data. Since the results are worse than the k-NN classifier, we do not consider it further.

3.2.2 Subpage-Agnostic Domain Classification

One challenging aspect of Website Fingerprinting (WF) is subpage-agnostic domain classification, which is also a more realistic scenario. Instead of performing classification only in the index page, the attacker is faced with an arbitrary subpage of the target domain. The challenge in this setup is the high variability of the subpages, for example due to different embedded images or different amounts of texts, and the much larger space, as usually a domain has many subpages.

Our encrypted DNS-based classifier has a distinct advantage here, as DNS requests are more stable than HTTP(S)-based WF. Intuitively, DNS requests mainly depend on third-party domains (e.g, for JS libraries or fonts). These are commonly defined in templates and thus are identical for many subpages. HTTP(S) traffic is more noisy, as the amount and concrete images often changes from subpage to subpage.

We compare our method to Panchenko et al. [23, Fig. 11 (b)] by calculating our own subpage confusion matrix. The test consists of a closed world evaluation with

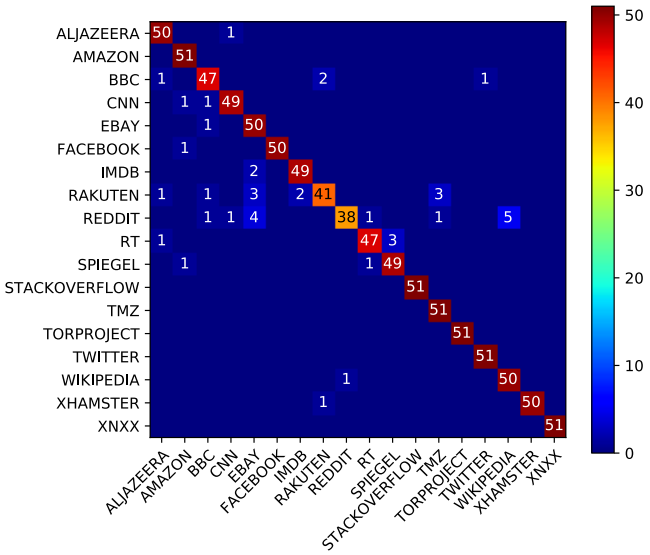

Figure 1: Subpage confusion matrix between different domains and their 51 distinct subpages.

20 domains and 51 subpages each. For the same set of 20 domains as Panchenko et al. we pick 51 random URIs from the Common Crawl [4] dataset from October 2019. Only domains which resolve to a 200 status code are eligible.

Compared to Panchenko et al. [23, Fig. 11 (b)] we have a much lower classification error in Fig. 1. Out of the total of 918 subpages only 43 (4.7 %) are wrongly classified, whereas Panchenko et al. wrongly classify 175 (19.1 %) subpages. Overall, our methodology performs better on 15 of the 18 domains. The most challenging domains are RAKUTEN and REDDIT in which our classifier performs worse. One reason why we can perform better is that the variance of DNS traffic is lower than for HTTP(S) traffic.

4 Countermeasures

We showed that attacks based solely on encrypted DNS traffic are feasible and can be successful. We now analyze in which directions countermeasures need to be developed by understanding the impact of the two major feature types (size and timing) of our classifier. Based on these insights, we then describe potential mitigations.

4.1 Evaluating Perfect Mitigations

Our classifier uses two feature types: packet sizes and timing information. To build a better countermeasure, we first need to understand which feature contributes more entropy, because mitigating that will have the largest effect. To this end, we perform a thought experiment in which we assume perfect mitigations, i.e., a perfect padding scheme and a perfect timing defense. We simulate these by removing the Msg elements, for perfect padding, or the Gap elements from the DNS Sequences.

Table 3: Comparison between our classifier, a simulated perfect padding scheme, and a simulated perfect timing defense.

	$k=1$	$k=3$	$k=5$	$k=7$	$k=9$
Baseline	86.1%	85.6%	84.9%	83.9%	82.9%
Perfect Padding	84.5%	84.2%	83.9%	83.2%	82.4%
Perfect Timing	4.2%	4.0%	4.0%	3.9%	3.8%

We re-run the closed world classification with these modifications and compare the results in Table 3. We see only a minuscule difference for a perfect padding defense, however, we see a large drop in performance for the timing defense. This indicates that the existing padding mechanism is already close to optimal. In contrast, a perfect timing-based defense destroys the classification results. The inter-arrival timings of the DNS responses in the sequences carry significant entropy that we use to classify perfectly-padded traffic.

From these observations we can derive important novel insights. First, the currently proposed padding strategy [10, 21] is indeed a good compromise between overhead and the maximum privacy guarantees that an optimal padding could guarantee. Yet, second, even an optimal padding strategy does not decrease the trace's entropy and does not suffice to preserve the user's privacy. Third, countermeasures should also take into account timing information, as timing has proven to contribute significant entropy in DNS Sequences.

4.2 Evaluating Practical Mitigations

Based on these observations, we now implement two practical mitigations and measure their efficacy and efficiency.

Constant-Rate (CR) schemes [8] send a packet on a fixed schedule every x ms. The packet is filled with payload, if some is waiting, otherwise with padding data. CR entirely removes timing information as everything is constant. However, CR has a significant bandwidth and latency overhead, since packets *must* be sent, even if no payload is waiting, and packets have to wait until the next scheduled transmission time thus increasing latency. A larger x reduces the bandwidth overhead but creates a larger latency overhead. Finally, CR requires a termination condition to avoid infinite transmissions. We define a probability p, which specifies the likelihood that a dummy packet is sent after the end of the stream.

Adaptive Padding (AP) [26] mitigates timing side-channels by masking the statistical timing features on the client-side. AP sends dummy traffic with indistinguishable timing from real traffic by switching between creating bursts and waiting for the next burst. The burst sizes, inter-burst and intra-burst timings must be from realistic distributions, so we extract them from our closed world dataset.

Comparing AP with CR: To compare them, we create an experimental setup to measure them in varying configurations. We implement a DoT proxy that we place between Unbound and the DoT server, providing the mitigations on the client

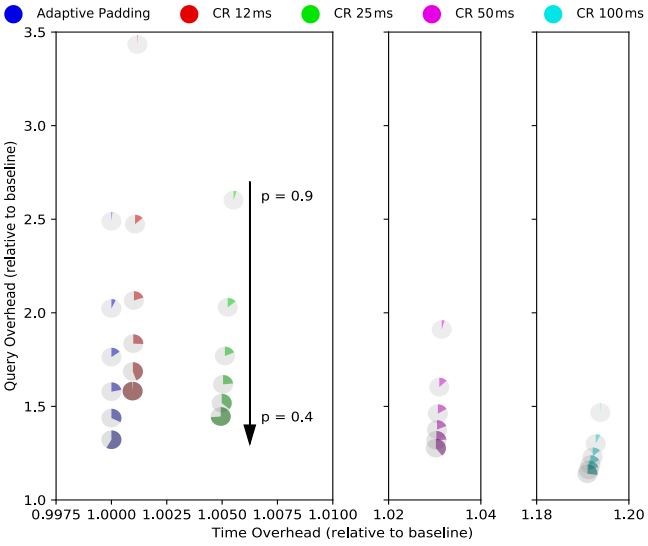

Figure 2: Comparison of time overhead (x-axis) and packet overhead (y-axis) between AP and CR. A full circle represents 10% correctly labeled domains ($\geq 5/10$ traces correct).

side. Technically, we simulate the effects on DNS Sequences, which allows for quicker testing without regenerating traces.

We measure the mitigation effects with our 10-fold cross-validation setup for varying packet rates x ms (for CR) and the probabilities p (for CR and AP). We test four packet rates from 12 ms to 100 ms and six probabilities p from 0.4 to 0.9. We measure the bandwidth overhead in additional DNS messages, the increase in resolution time, ignoring dummy packets, and the impact on the classification results.

Figure 2 shows the classification results. It is color coded with AP being blue (left-most column of circles) and CR in the remaining colors. The x-axis shows the time overhead as a factor compared to the baseline of no modification. Similarly, the y-axis measures the overhead in the number of DNS messages with 1 being the baseline. The size of the colored slice of each circle represents the classification results. A full circle is equal to a 10 % classification success in the $5/10$ traces correctly classified setup.

All variations are successful in mitigating our classifier. For a similar defense strength AP has a higher bandwidth overhead than low-rate CR, however, lower than fast-rate CR. For interactive use cases AP is likely better, because it has no timing overhead, yet when bandwidth is of concern, CR with a slow sending rate is probably preferred over AP.

5 Discussion

Our work helps to understand the privacy threats that DNS users face and how they can be protected. Analyzing new side-channel attacks becomes more important, as there is a general trend for more encryption that aims to mitigate (obvious) privacy breaches.

This leaves DNS as an important target for snooping on the privacy of users, since it is the first step of connecting to the Internet. Given that we constrained ourselves to *encrypted and padded* DNS traffic, we find the provided classification results quite alarming. We can partially deanonymize 92.2 % of websites and correctly classify 86.1 % of DNS Sequences. The foremost goal of our study, assessing the privacy guarantees of encrypted DNS, was thus successful, as we have shown drastic privacy problems. The classification accuracy can be further boosted by combining our approach with existing WF methodologies. Having said this, there are some limitations to our proposed methodology, which we will describe next.

Datasets: We use a rather extensive dataset with 9235 websites / 92350 DNS Sequences in the closed world dataset. Related WF attack papers regularly use a much smaller number of websites in their datasets, in the range of tens to hundreds [11, 23, 28]. A larger dataset causes precision and recall to decline [23, Fig. 10], which means our results would shine better on datasets of identical size to these papers.

DNS Traffic Identification and Extraction: The approach to extract a DNS Sequence from network captures assumes DNS servers are identifiable as such. DoT helps us here, by using a dedicated TLS port, and most DoH servers have dedicated IP addresses.

Similarly, we assumed that we can extract message sizes from the TLS stream, which is possible for the current implementations. They try to use small TLS records to transmit the data, by either using a single TLS record for a DNS reply. In principle, DNS messages could be transmitted in many tiny or equally sized TLS records, hampering attempts of exact message size extraction, but increasing the cost due to more processing and network overhead.

User Modeling: For our experiments, we modeled a certain user behavior, which however might deviate in practice. First, we assumed the client is waiting until the website has fully loaded without any background DNS traffic. Second, the evaluations were performed with an empty browser cache and DNS cache, which only included the effective TLDs. In practice users may have different states for their DNS cache and browser cache, which can result in fewer DNS requests sent. However, all major browsers implement strategies to partition the browser cache based on first party domain [3, 9, 32]. This prevents the browser from re-using common resources across domains and is implemented to prevent user tracking from websites. We think that simulating an empty cache is therefore a fair approximation for many websites, given that our attack can only work on the first page visit of a stay on a website. While partial caching may decrease the classifier's accuracy, one could argue that a stateful adversary can use our attack to model the DNS cache state of a user. Knowing that, the adversaries can adapt the training datasets by retraining on the traces that are expected with a certain cache state. We plan to perform such analyses and an according adaptation to our classifier in the future.

6 Related Work

DoT [14] and DoH [12] were created to provide confidential DNS transactions. The standardized padding policies [21], required to combat traffic analysis, are based on Gillmor's work analyzing individual query/response pairs [10]. We assess this general threat model to encrypted DNS in a Web setting, in which we can leverage dependencies between multiple DNS requests. This argument is agnostic to the specific proposal and applies to future schemes.

Parallel to our work two papers from Houser et al. [13] and Siby et al. [27] were published. Their work is closely related to ours, and we will highlight the relevant differences.

Houser et al. [13] analyze DoT by using a machine learning (ML) classifier on statistical features of a time series of DNS messages. Their threat model is identical to ours. The feature set is built from a time series of DNS messages, containing the timestamp, length of the encrypted message, and the traffic direction. From this, they extract higher level features, such as the query or response length, total number of packets, time intervals, or queries per seconds. For each higher level feature, they calculate a range of statistics (minimum, maximum, median, mean, deciles, and count) which they feed into the ML classifier. They use two self-learning classifiers, random forests and Adaboost. For domain category inference (i.e., domain is dating or gambling related) they report 93.65 % to 96.12 % correct classification for unpadded data. This drops to only up to 78.7 % when using padded data. Identifying individual websites with padding, like our paper, achieves only 83 % correct classifications. These results are for a dataset with only 98 sensitive websites.

Siby et al. [27] analyzed the privacy of DoH, focusing on unpadded traffic but evaluating the padding impact. The main feature is a sequence of bytes with the sign indicating the traffic direction and the value either representing the TLS record lengths or the combined size of a burst. This sequence is then converted into bi-grams using a sliding window and fed into a random forests classifier. Importantly, they do not use any timing information, however, in Table 3 we found timing to be very important. The dataset ranges from 700 to 1500 domains for the closed world and 5000 for the open world. They report a precision of 94.0 % when evaluating on a closed world dataset comparable to ours. By their own report DoT is much harder to classify than DoH, having a 0.3 reduction in the F1 score (see [27, Table VII]). The strength of the paper lies in the diverse evaluation, testing with different clients, resolvers, and padding configurations.

Dataset: Compared to both papers we use more domains for our dataset with 9235 domains in the closed world. Smaller datasets benefit the classification results, as the possibility for wrong classifications and the diversity in the dataset is lower. Even with the larger dataset, we beat Houser et al.'s performance when classifying individual websites with 86.1 % compared to their 83 %. The data collection pipeline is similar in all three papers.

Feature Set: Houser et al. use the most extensive feature set, by including time, sizes, and directionality, while Siby et al. use sizes and directionality. We use sizes and timing, but ignore directionality, as we only use the downstream traffic, since we found the upstream to contain almost no entropy.

Houser et al.'s use of inspectable self-learning classifiers allows them to list the most important features. In the case of individual website classification with padding they are different timing features, like our results in Section 4.1. Interestingly, when performing domain category inference timing plays almost no role, even with padding. Siby et al. only use low level byte counts and no timing, therefore the feature importance does not apply for their classifier.

Unique Contributions: Most importantly, we are the only ones to implement and test countermeasures specific to encrypted DNS fingerprinting (see Section 4). Both papers measure the impact of padding on their classification results, whereas we only evaluat the impact of our classifier on padded data with the standardized 128 B/468 B block padding. This padding is also the most aggressive one tested in the related papers. Siby et al. test two additional configurations, a perfect padding in which all sizes are indistinguishable analog to our work in Section 4.1 and the effect of tunneling DNS over Tor. Houser et al. only discuss the high level ideas of defense, namely hiding exact size and timing information.

We are the only ones measuring subpage agnostic classification (see Section 3.2.2) in which we visit webpages beyond the index page, an inherently more complicated task due to the variety of pages.

7 Conclusions

Our work underlines the importance of carefully studying the possibility of traffic analysis against encrypted protocols, even if message sizes are padded. While there is a plethora of literature on Website Fingerprinting based on HTTPS and Tor traffic, we turned to encrypted DNS—an inherently more complex context, given the low entropy due to short sequences and small resources. We show that passive adversaries can inspect sequences instead of just single DNS transactions to break the widely deployed best practice of DNS message padding. We hope that our observations will foster more powerful defenses in the DNS setting that can withstand even more advanced traffic analysis attacks like ours.

Acknowledgment

We thank our anonymous reviewers whose useful comments helped us to improve the quality of our paper. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the BMBF project 16KIS0656 (CAMRICS).

References

[1] APNIC. Use of DNSSEC validation for world. https://stats.labs.apnic.net/dnssec/XA, Last Accessed: 2019-02-05.

[2] Bromite, 2019. https://www.bromite.org/.

[3] Chrome: Partition the HTTP cache, November 2019. https://chromestatus.com/feature/5730772021411840.

[4] Common crawl CC-MAIN-2019-43, October 2019. https://commoncrawl.s3.amazonaws.com/cc-index/collections/CC-MAIN-2019-43/indexes/cdx-00000.gz.

[5] Fred Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 1964.

[6] DNS privacy implementation status, January 2019. https://dnsprivacy.org/wiki/pages/viewpage.action?pageId=23035950.

[7] DNSSEC deployment report, February 2019. https://rick.eng.br/dnssecstat/, Last Accessed: 2019-02-05.

[8] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *IEEE Symposium on Security and Privacy*, 2012.

[9] Firefox: Top-level origin partitioning, October 2019. https://bugzilla.mozilla.org/show_bug.cgi?id=1590107.

[10] Daniel Kahn Gillmor. Empirical DNS padding policy, March 2017. https://dns.cmrg.net/ndss2017-dprive-empirical-DNS-traffic-size.pdf.

[11] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium*, 2016.

[12] Paul E. Hoffman and Patrick McManus. DNS Queries over HTTPS (DoH). RFC 8484, October 2018.

[13] Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. An investigation on information leakage of DNS over TLS. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019.

[14] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016.

[15] Martin Husák, Milan Cermák, Tomás Jirsík, and Pavel Celeda. Network-based HTTPS client identification using SSL/TLS fingerprinting. In *10th International Conference on Availability, Reliability and Security*, 2015.

[16] Martin Husák, Milan Cermák, Tomás Jirsík, and Pavel Celeda. HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting. *EURASIP Journal on Information Security*, 2016.

[17] Tommy Jensen, Ivan Pashov, and Gabriel Montenegro. Windows will improve user privacy with DNS over HTTPS, November 2019. https://techcommunity.microsoft.com/t5/Networking-Blog/Windows-will-improve-user-privacy-with-DNS-over-HTTPS/ba-p/1014229.

[18] Erik Kline and Ben Schwartz. DNS over TLS support in Android P developer preview, April 2018. https://android-developers.googleblog.com/2018/04/dns-over-tls-support-in-android-p.html.

[19] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[20] Alexander Mayrhofer. The EDNS(0) Padding Option. RFC 7830, May 2016.

[21] Alexander Mayrhofer. Padding Policies for Extension Mechanisms for DNS (EDNS(0)). RFC 8467, October 2018.

[22] Patrick McManus. Improving DNS privacy in Firefox, June 2018. https://blog.nightly.mozilla.org/2018/06/01/improving-dns-privacy-in-firefox/.

[23] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *23rd Annual Network and Distributed System Security Symposium*, 2016.

[24] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *26th Annual Network and Distributed System Security Symposium*, 2019.

[25] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Beauty and the burst: Remote identification of encrypted video streams. In *26th USENIX Security Symposium*, 2017.

[26] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *11st European Symposium on Research in Computer Security*, 2006.

[27] Sandra Siby, Marc Juarez, Claudia Diaz, Narsea Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS → privacy? A traffic analysis perspective. In *27th Annual Network and Distributed System Security Symposium*. The Internet Society, 2020.

[28] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[29] The Chromium Developers. DNS over HTTPS (aka DoH), November 2019. https://www.chromium.org/developers/dns-over-https.

[30] Tranco list id G63K, August 2019. https://tranco-list.eu/list/G63K/10000.

[31] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[32] WebKit: Optionally partition cache to prevent using cache for tracking, November 2019. https://bugs.webkit.org/show_bug.cgi?id=110269.

[33] Charles V. Wright, Scott E. Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the Network and Distributed System Security Symposium*, 2009.