Privacy Leakage of DNS over QUIC: Analysis and Countermeasure

Guannan Hu
Sokendai
Tokyo, Japan
kokannan@nii.ac.jp

Kensuke Fukuda
NII / Sokendai
Tokyo, Japan
kensuke@nii.ac.jp

***Abstract*—Original DNS packets are unencrypted, which leads to information leakage while users visit websites. The adversary could monitor the DNS communication and infer the users' Internet preferences, which may contain private content, such as health, finance, and religion. Although several encrypted DNS protocols have been proposed: DNS over HTTPS (DoH), DNS over TLS (DoT), and DNS over QUIC (DoQ), recent research shows that the adversary could still infer the category of websites even using DoT and DoH.**

This paper studies the privacy leakage problem of DoQ protocol with two different DNS recursive resolvers (NextDNS and Bind). We show that the classification performance of the websites is high both in NextDNS and Bind resolvers for identifying whether the category of websites is sensitive. More particularly, we indicate that discriminative features are mainly related to the inter-arrival time of packets and packet length. Therefore, we further investigate two countermeasures that could affect the classification results: adding random delay in responses and padding the DNS payload. 1) We find that mean F1 scores decrease as the delays increase. Specifically, it decreases the classification performance by 21% with NextDNS and 19% (0-300 ms) with Bind. Also, 2) DNS padding decreases the classification performance by 10%. We further investigate the combination of the two countermeasures: both adding random (0-60ms and 0-100ms) delays and padding the DNS payload. We confirm that the combined method could greatly reduce the classification performance, on average 25% in Bind. These results indicate that adding random time and padding can protect users' information from the website fingerprinting attack, though the random delay might affect the user experiences.

***Index Terms*—DNS over QUIC, Privacy Leakage, Website Fingerprinting**

I. Introduction

With the increasing dependence of humans on the Internet, the privacy leakage of users when browsing the Internet has been widely discussed. "Pervasive Monitoring" [1] is the main attack in that the adversary could monitor the network communication and infer the users' private information by analyzing the packet (i.e., packet length, count, and time). HTTPS is often used to encrypt communication using Transport Layer Security (TLS) or Secure Socket Layer (SSL). While these encryption protocols (i.e., HTTPS, TLS, and QUIC) protect the contexts, some packet information, such as packet arrival time and size, is still exposed.

Our focus in the paper is on another protocol related to website access: DNS. The DNS is a global service for mapping between hostnames and IP addresses. The DNS packets are unencrypted and easily be eavesdropped by the adversary. To protect users' privacy, some encrypted DNS protocols have been standardized, i.e., DNS over TLS (DoT) [2], DNS over HTTPS (DoH) [3], and DNS over QUIC (DoQ) [4]. More particularly, DoQ is a new protocol to encrypt DNS messages using QUIC as an underlying transport. QUIC is a new transport protocol based on UDP and standardized by IETF RFC 9000 [5]. Currently, various public DNS services support the encrypted DNS protocol. For example, Google [6], Cloudflare [7], and Quad9 [8] have already support the DoT and DoH. Only AdGuard [9] and NextDNS [10] could support DoQ. Recent studies investigated the information leakage by DoT [11] and DoH [12], [13] found that information leakage is still possible even though DoT with DNS padding [14].

We discuss two possible countermeasures to mitigate the privacy leakage in DoQ for website fingerprinting attacks: adding random delay for DNS responses and padding DNS payload. For this, we first split the websites from Alexa's top websites list [15] into two categories: sensitive and non-sensitive. Then, for each website visit, we collect DoQ traffic and extract 174 flow features corresponding to inter-arrival time, packet number, and packet length. After training four supervised machine learning algorithms, we measure the F1 score regarding different configurations and DNS resolvers. The contributions of this paper are as follows:

- We confirm that the information leakage against website fingerprinting attack exists in DoQ, as well as DoT and DoH.
- We find high discriminative features to infer the website which is mainly related to inter-arrival time and packet length.
- We investigate two countermeasures that affect the classification performance of DoQ: adding random delays and padding the DNS payload [14]. Adding random delay decreases the classification performance by 19% (0.9 to 0.71) with Bind and 21% (0.9 to 0.69) with NextDNS.
- DNS padding also mitigates the classification performance by 10% (0.9 to 0.8).
- We finally demonstrate that combining the two countermeasures decreases the performance by 25%; they are useful in protecting user's privacy.

II. Related Works

The adversary could monitor the network traffic between the victim and the web server, known as a website fingerprinting (WFP) attack. The adversary intends to classify the websites the victim visited by analyzing the traffic patterns. Several previous works [12], [16]–[20] analyzed the WFP attacks on the HTTP and extracted the features related to the packet order, size, inter-arrival time, and burst behavior with different machine learning classifiers (i.e., SVM and Random Forest). For example, in Ref. [17], the authors applied a novel fingerprinting technique to mitigate the WFP attack for OpenSSH, OpenVPN, CiscoVPN, Stunnel, and Tor using a Multinomial Naive-Bayes classifier. They found the text mining technique had no protection against the WFP attack. These works showed that the WFP attack is possible against some privacy-enhance services, such as Tor, IPsec, and VPN. Also, some encrypted protocols (HTTPS, TLS/SSL, and QUIC) have been proposed to protect the users' private information. Other researchers [21], [22] also analyzed the website fingerprinting attack on the encrypted traffic: HTTPS and QUIC have been more studied. A work [21] investigated whether the QUIC is more difficult to fingerprint than TCP. They selected the top 300 features and used k-fingerprinting and deep fingerprinting classifiers. The results indicated that the QUIC is not more difficult to fingerprint than TCP. Zhan et al. [22] also studied the vulnerability between HTTPS and QUIC (Google and IETF QUIC) in different scenarios. They found that Google and IETF QUIC were more vulnerable than HTTPS to the WFP attack. In our work, we investigate the WFP attack on encrypted DNS instead of web traffic traces.

Original DNS queries are unencrypted, sent in plain text, and vulnerable to eavesdropping, which leads to information leakage. Intending to protect the privacy of information, the DNS encryption techniques had been standardized. Recent papers [23] described the deployment of DNS-Over-Encryption servers, and a comparison of different encrypted DNS protocols: DoT, DoH, DoQ, DNSCrypt, and DTLS. Ref. [24] presented the measurement, comparison, and analysis of the DoH, DoT, and DoQ in three global organizations. Deccio et al. [25] measured DoT and DoH in many different open DNS resolvers and authoritative DNS servers. In Ref. [26], authors summarized the current deployment of encrypted DNS techniques (i.e., DoT, DoH, and DoQ), then surveyed the analysis for detecting encrypted DNS protocols by analyzing the encrypted DNS traffic. Other works from [27] intended to classify and recognize the DoH traffic with five machine learning algorithms: KNN, Decision Tree, Random Forest, Naive Bayes, and AdaBoost. They found that the accuracy of recognition DoH is over 99%.

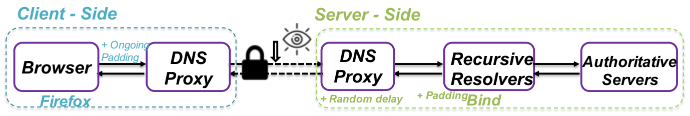
Prior works are more focused on the WFP attack with DoT and DoH. For example, Ref. [12] examined whether encrypted DNS traffic could protect users and analyzed the DoH traces with different environments (i.e., location, client application, platform, or DNS resolver). They found monitoring and censorship were feasible even using DoH, and some features used to attack HTTPS were not appropriate for DoH. In Ref. [7], authors analyzed the WFP attacks of DoH traffic using three groups of features: size, timing, and ordering on Google and Cloudflare DNS resolvers. The results suggested that encrypted DNS messages should be padded to protect privacy. The work of [11] analyzed the WFP attack on DoT and compared the classification performances with padded or unpadded DNS. The work from Jonas et al. [28] measured the impact of padding strategies (128 B / 468 B block padding) on DoT and DoH. They observed padding cannot mitigate the WFP attack on DoH and DoT. These works demonstrated that information leakage is still possible in encrypted protocols (HTTPS, QUIC, and DNS-Over-Encryption). In our work, we investigate two countermeasures that could affect the classification performances of DoQ: adding random delay and padding the DNS payload.

III. Methodology

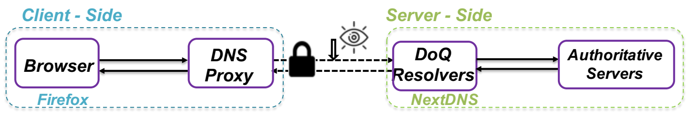
A. Overview

We introduce the overview of two possible configurations of encrypted DNS to simulate the website fingerprinting attack.

Here, our adversary scenario consists of three components; web browser (client), recursive resolvers, and authoritative servers. We assume the adversary can monitor the DNS traffic between the victim (client browser) and the DNS resolver. This scenario is easier than web traffic monitoring, especially for high-speed networks. We consider two configurations to simulate the website fingerprint attack, depending on DNS software, as shown in Fig.1. We require a DNS proxy to encrypt the DNS traffic and send it to the DNS resolver when the resolver does not support the encrypted DNS. For the first configuration Fig.1 (a), we set up the AdGuard DNS proxy [29] on the client and local recursive resolver (Bind9 [30] on Raspberry Pi). The DNS proxy is just a forwarder without the cache and encrypts DNS packets with different protocols. The second configuration is for DNS resolvers (i.e., AdGuard [9] and NextDNS [10]) that support the encrypted DNS protocols. When the user visits the website, the encrypted DNS queries are sent to these DNS resolvers directly, as shown in Fig.1 (b). We choose the NextDNS for our experiment. On the client side, we use the popular browser: Firefox [31]. We capture the encrypted DNS traffic (dotted lines) between the client and server.



(a)



(b)

Fig. 1. Two configurations to deploy the DoQ

B. Category Selection

Next, we explain the process of dataset collection: determine the category of websites, select the websites for each category, and capture the traffic data. For selecting visiting websites, we determine the category of Alexa's Top 300,000 websites based on the FortiGuard Web Filtering [32]. Then, we select the Top 400 for each category popular in websites [15] from January 2020. We build a Python script to access the website in the list automatically, and only one website is loaded each time on the client side. For each website, we only visit its homepage. While the access is complete (timeout 30s), we close the browser and save the traffic data as pcap files. In this process, we use Selenium [33] webdriver to drive the Firefox (ver. 117.0.1) browser.

We treat the same domain (different TLD) as one sample, i.e., if example.com and example.net are both in the list, we prefer to choose a more popular one in the Alexa ranking.

After we prepare the targeted datasets, the next step is to split the dataset into two labels for the binary classification. We choose 30 [1] categories of websites and split these categories into two labels: *'Non-Sensitive'* and *'Sensitive'*. Sensitive category is related to personal information such as health, finances, business, and government. For example, websites in the health category may indicate that users intend to access to obtain medical information for some specific symptoms. Similar to past literature [11], we select dating, health, and gambling categories as the sensitive. In addition, we select seven (Finance and Banking, Global Religion, Government and Legal Organization, Illegal or Unethical, Other Adult Materials, Phishing, and Political Organization) categories as sensitive. In our experiment, we consider the binary classification: *'Non-Sensitive'* and *'Sensitive'*. We select 10 categories as sensitive, and the remaining 20 categories as non-sensitive as shown in Table I. We evaluate the classification performance of two datasets: balanced and imbalanced datasets, due to the uneven distribution of the number of sensitive and non-sensitive categories. Also, we intend to confirm whether the imbalanced dataset (more realistic situation) could get better performances. In addition, we use the oversampling technique (SMOTE [34]) to handle the imbalanced data.

C. Feature Extraction

Features are variables or attributes that describe the instance of the class in machine learning. We extract 174 bidirectional flow features in traffic traces such as packet number, inter-arrival time, and packet length between consecutive queries and responses packets, mainly followed by the past literature [21], [26], [35].

We list broad categories of traffic features as follows:

[1]The categories are: Advertising, Art, Brokerage and Trading, Business, Dating, Domain Parking, Education, Entertainment, File sharing and Storage, Finance and Banking, Freeware and Software Downloads, Gambling, Game, General Organizations, Global Religion, Government and Legal Organization, Health and Wellness, Illegal or Unethical, Information Technology, Internet Radio and TV, Job Search, Malicious Websites, Meaningless Content, Newly Observed Domain, News and Media, News groups and Message boards, Other Adult Materials, Personal Vehicles, Phishing, Political Organizations.

TABLE I
SPLIT THE DATASET (30 CATEGORIES)

	Non-Sensitive	***Sensitive***
Balanced Data	*Business/Education/Entertainment/ File sharing and Storage/ Job Search/Newly Observed Domain/ News and Media/Personal Vehicles/ General Organizations/ Internet Radio and TV*	*Dating/Gambling/ Finance and Banking/ Global Religion/ Government and Legal Organization/ Health and Wellness/ Illegal or Unethical/ Other Adult Materials/ Phishing/ Political Organizations*
Imbalanced Data	*Advertising/Art/Brokerage and Trading/ Business/Domain Parking/Education/ Entertainment/File sharing and Storage/ Freeware and Software Downloads/ Game/General Organizations/ Information Technology/Job Search/ Malicious Websites//Meaningless Content/ Internet Radio and TV Newly Observed Domain/News and Media/ News groups and Message boards/ Personal Vehicles*	

Packet count: The number of in/out packets.

Packet length: The length (bytes) of query and response packets. We use the maximum, minimum, median, mean, standard deviation, variance, coefficient variation, and deciles for this feature.

Inter-arrival time: The features from a consecutive pair of query and response packets. It includes the maximum, minimum, median, mean, standard deviation, variance, coefficient variation, and deciles of the inter-arrival value.

Cumulative bytes: The cumulative bytes of query and response packets. We also consider the rate of bytes being sent and received in a trace.

Entropy value: Shannon's normalized entropy of inter-arrival time and packet length.

Throughput: The average in packets and bytes.

Duration: The total transmission time in the trace.

Time to receive first N bytes: As in [35], the time that is received from the DNS resolver could reflect whether the resolver cached. We set the value of N to 3000 and 5000.

In the final step, we remove the features with zero variance.

D. Classifiers

For evaluating the classification performance, we split the datasets into sets: training and testing. Then, we use the dataset to train the supervised machine learning models we prefer and evaluate the model with F1 scores.

We split 80% of the dataset as a training set and 20% as a testing set randomly. We also adopt the 10-fold cross-validation [36] method and repeat it ten times for each test to evaluate the performance of our experiments. We train all binary classifiers and tune parameters using the RandomizedSearchCV [37] function from the scikit-learn library [38]. The RandomizedSearchCV function can help us to find the best estimator for each classifier. We obtain the mean F1 score to evaluate the performance of each classifier. A higher value of the F1 score (close to one) means better, and a lower (close to zero) indicates a worse classification performance. In our context, a lower F1 score is desirable because the attacker cannot infer the website categories precisely. We use well-known four supervised binary classification algorithms;

Random Forest (RF), AdaBoost (AB), XGBoost (XB), and LightBM (LB).

IV. Baseline Classification Performance

We first evaluate baseline classification for DoQ traffic in the two configurations, then discuss the important features for the classification.

A. Baseline Classification Performance

At first, we focus on the baseline classification performance for two resolvers (Bind and NextDNS) as explained in the previous section. Specifically, we discuss the F1 score [2] of the classification and the resulting top 10 important features.

Here, we crawl the encrypted traffic without cleaning the cache in the Bind. As shown in Fig.2 (left), overall the binary classification performances are very high for the four classifiers, in the balanced and imbalanced dataset. Thus, these classifiers are good enough to infer the website for the local resolver.

Next, we select NextDNS as the DNS resolver. Fig.2 (right) shows the results of NextDNS. We again find a high classification performance; the F1 scores are all around 0.90 with imbalanced and balanced data. The performances of imbalanced data are slightly higher than balanced data, about 2-3% more. Thus, we demonstrate that the classification performance is high enough to infer the website regarding DoQ, whether on NextDNS or Bind resolver.

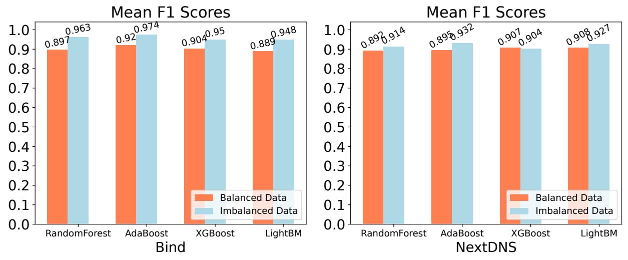


Fig. 2. Classification Performance (Bind and NextDNS)

B. Feature Importance

We intend to understand what features have the most influence on the results for discussing possible countermeasures. We measure the feature importance by calculating the mean degradation of the gini impurity with the Random Forest algorithm. For investigating the discriminative powers of the feature, we repeat ten times to obtain the average value for each feature. The range of feature important is 0 to 1, the higher value means more influential for the classification.

We show the top-10 discriminative features in Table II. We find that significant features are mainly related to inter-arrival time. For the balanced data, the entropy of inter-arrival time between a pair of query and response packets is the most significant. We also notice that only one feature out of the top-10 is related to another feature, packet length.

[2]F1 score is the harmonic mean of precision and recall.

For NextDNS, we obtain consistent results with those in Bind. Thus, regardless of the DNS software, we should decrease the classification performance by controlling these effective features, for protecting users' privacy.

TABLE II
Top-10 Discriminative Features (Bind)

Rank	Balanced Data		Imbalanced Data	
	Feature	Mean	Feature	Mean
1	EntropyQRIntervalTime	0.022	QRIntervalTimeMean	0.024
2	EntropyRRIntervalTime	0.018	QRIntervalTimeMedian	0.018
3	EntropyQQIntervalTime	0.016	EntropyDFIntervalTime	0.016
4	QueryIntervalTime	0.015	QRIntervalTimeMin	0.014
5	EntropyResponseLength	0.011	RQIntervalTimeDeciles4	0.012
6	EntropyDFIntervalTime	0.0103	RQIntervalTimeDeciles3	0.0114
7	RQIntervalTimeDeciles3	0.0103	QueryIntervalTimeStandard	0.0113
8	RQIntervalTimeDeciles4	0.0102	EntropyRRIntervalTime	0.0104
9	ResponseIntervalTimeMin	0.010	ResponseLengthMax	0.01001
10	QueryIntervalTimeMin	0.009	QueryLengthDeciles3	0.009

TABLE III
Top-10 Discriminative Features (NextDNS)

Rank	Balanced Data		Imbalanced Data	
	Feature	Mean	Feature	Mean
1	QueryIntervalTimeDeciles1	0.086	ResponseIntervalTimeMin	0.082
2	QueryIntervalTimeMedian	0.082	QueryIntervalTimeMedian	0.078
3	ResponseIntervalTimeMin	0.070	QRIntervalTimeCoefficientVariation	0.068
4	QueryIntervalTimeMin	0.069	QueryIntervalTimeVariance	0.057
5	ResponseIntervalTimeMedian	0.068	ResponseLengthDeciles1	0.054
6	QRIntervalTimeMean	0.060	QRIntervalTimeMedian	0.053
7	QueryIntervalTimeMean	0.057	QRIntervalTimeMin	0.050
8	ResponseIntervalTimeMean	0.056	QRIntervalTimeMean	0.049
9	QRIntervalTimeMedian	0.043	ResponseIntervalTimeMedian	0.049
10	QRPktLength	0.038	QueryIntervalTimeMax	0.047

V. Countermeasures

From the previous baseline results, a promising approach is to control the inter-arrival distribution and packet length for the mitigation. To randomize the inter-arrival time, the easiest way is to add different random delays to the query or response. Also, as standardized in RFC 8467 [14], DNS payload padding is useful in controlling the packet length. Thus, we discuss these two countermeasures that affect the identification performance of DoQ.

A. Random Delay

We conduct an experiment to understand the random delay effect as the first countermeasure. We add a random delay on the returning path at the DNS proxy to the client, to ensure that the random delay is added to the encrypted traffic. The random delay is followed by uniformly distributed random values. The range of random delay varies from 0-3 milliseconds to 0-300 milliseconds.

As shown in Fig.3, we confirm the classification performance degradation (10% - 21%) with random delays. The degradation of NextDNS (21%) is a little higher than that of Bind (19%). This is likely because of the different geographical locations between the client and the resolver. NextDNS is a public DNS resolver that can be somewhere in the network, though Bind is the local resolver in the home network. Thus, in the more realistic situation that DoQ is provided by a public resolver, we expect better mitigation. The results demonstrate that adding random delays could decrease the performance. However, the effect is still considerably small. Also, the higher random delay might affect the user experience.

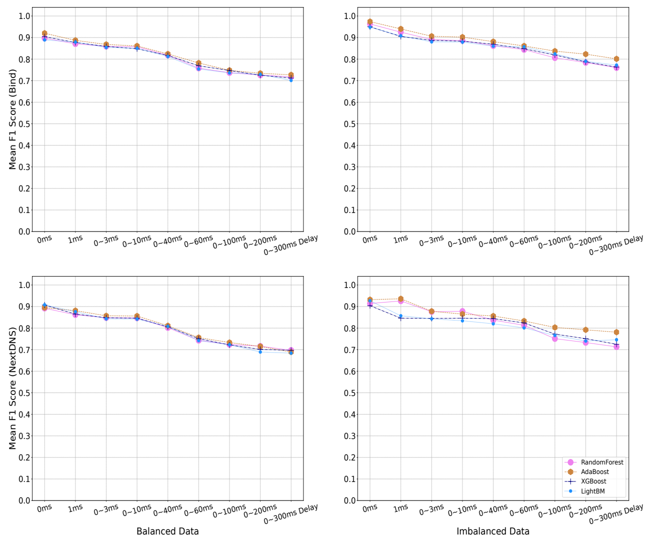


Fig. 3. Classification performance for random delays (Bind and NextDNS)

B. Payload Padding

Here, we focus on DNS payload padding standardized in RFC 8467 [14]. DNS padding has not been supported by all the DNS software. We notice that a newer version (ver. 9.19) of Bind supports the EDNS(0) padding option, though NextDNS does not. Thus, we take two evaluations with Bind: 1) Only padding at the ongoing direction (from the client to the resolver), and 2) Padding at both ongoing and returning direction (between the client and the resolver). The former corresponds to the case that the resolver does not support the padding, and the latter is the case that both the client and the resolver are aware of the padding.

As shown in Fig.4, we confirm that the padding affects the classification performance. For padding the DNS message both on the client and server, we obtain the lowest results; the performances of balanced data decrease from 0.9 to 0.8 with padding query and response. Also, about the imbalanced data, the results decrease from 0.95 to 0.88. However, the performance degradation is limited to the case of ongoing padding only. Thus, padding the DNS message for both directions is effective, though the wide deployment of padding is required for the caching resolvers.

C. Combination of Two Countermeasures

From the results of adding random delay and payload padding, we find they have some positive effects on the performance degradation. Finally, we add the random delay (0-60ms and 0-100ms) and pad the query/response with Bind. As shown in Fig.5, we find a significant change with padding and adding 100ms delay regardless of balanced and imbalanced data. The average reduction is about 25% with two countermeasures.

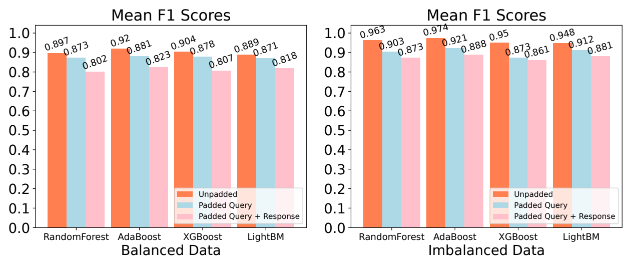


Fig. 4. Classification performance for padding (Bind)

The results conclude that adding some random delays and padding at the same time can protect users' information from attacks to a certain extent though the delay affects the user experience.

We also show the top-10 discriminative features in Table IV. We notice that the effective features change, not only related to inter-arrival time and packet length. Other features such as 'QueryPktCount' and 'RPerSecondMax' related to packet count also appear in the list. We also observe that the mean value of the top-10 important features decreases compared to the Bind (Table II) and NextDNS (Table III) without the countermeasures, regardless of balanced and imbalanced datasets. Thus, our approach mitigating the inference ability works as expected.

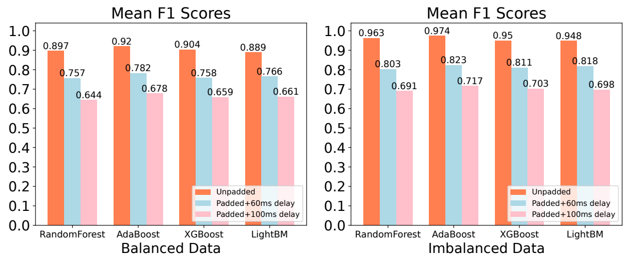


Fig. 5. Classification Performance for random delays and padding (Bind)

TABLE IV
TOP-10 DISCRIMINATIVE FEATURES (RANDOM DELAYS AND PADDING)

Rank	Balanced Data		Imbalanced Data	
	Feature	**Mean**	**Feature**	**Mean**
1	QueryIntervalTimeMax	0.015	QueryLengthDeciles8	0.015
2	QueryLengthDeciles7	0.013	QueryIntervalTimeMax	0.015
3	FlowSentRate	0.012	ResponseLengthMin	0.013
4	QRIntervalTimeStandard	0.011	ResponseLengthDeciles3	0.012
5	QueryLengthMode	0.011	QueryPktCount	0.010
6	QRIntervalTimeCoefficientVariation	0.010	ResponseLengthDeciles5	0.009
7	QueryIntervalTimeStandard	0.009	Duration	0.009
8	QueryPktCount	0.009	RPerSecondMax	0.009
9	QRIntervalTimeMax	0.009	QueryLengthMax	0.009
10	ResponseIntervalTimeDeciles5	0.008	DFIntervalTimeDeciles2	0.008

VI. LIMITATION

We discuss the limitations of our experiments. The first limitation of our work is that the physical measurement point of the experiment is close to the client. Also, we conducted these experiments in only one location (Tokyo, Japan). For the general scenario, we will consider more platforms and sites to measure classification performance in future work.

Another limitation of our work is that our padding experiments are only conducted with the bind resolver, because other public DNS resolvers (i.e., NextDNS, Google) do not support the padding, at the time of writing.

Finally, we only consider the binary (sensitive or non-sensitive) classification performance of our work. The binary classification is easier than the multi-class classification for the adversary. In this sense, we tackled the more serious case for users. For a more realistic situation, we will investigate the performance of multi-classification in the future.

VII. Conclusion

Privacy leakage on the web is a serious problem with the current Internet. In this work, we intended to understand whether the DoQ protocol could protect user privacy while visiting the webpage.

We investigated the information leakage by analyzing the DoQ traffic with Bind and NextDNS resolvers to discuss the countermeasure to mitigate this threat. We show that the classification performances of the websites are very high both in these two resolvers to infer whether the category of websites is sensitive. We pointed out that discriminative features are mainly related to the inter-arrival time of packets and the packet length. On the basis of this finding, we investigated two countermeasures: adding the random delay and padding the DNS message. Adding the random delay decreased the classification performance by 19% (from 0.9 to 0.71) with Bind and 21% (from 0.9 to 0.69) with NextDNS. When we padded the DNS payload with Bind, the performances decreased by 10% (from 0.9 to 0.8). In the end, we confirmed reasonable degradation (approx 25%) with padding and adding a 100ms delay regardless of balanced and imbalanced data.

Our analysis clarified that both countermeasures are promising to mitigate privacy harm from users, having a 0.25 reduction in the F1 score with Bind. However, adding a delay is a tradeoff between the privacy and user experience. Also, DNS padding has not yet been widely deployed in public DNS. For the more realistic situation (i.e., NextDNS), we hope to get more decrease. Therefore, in future work, we will consider adding an appropriate delay and building a tool to pad DNS queries or responses in the server.

References

[1] S. Farrell, "Pervasive Monitoring Is an Attack," RFC 7258, 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7258

[2] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," RFC 7858, 2016.

[3] P. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, 2018.

[4] C. Huitema, "Specification of DNS over Dedicated QUIC Connections," RFC 9250, 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc9250/

[5] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[6] [Online]. Available: https://developers.google.com/speed/public-dns/

[7] [Online]. Available: https://cloudflare-dns.com/

[8] [Online]. Available: https://www.quad9.net/

[9] [Online]. Available: https://adguard.com

[10] [Online]. Available: https://nextdns.io/

[11] R. Houser, Z. Li, C. Cotton, and H. Wang, "An Investigation on Information leakage of DNS over TLS," in *ACM CoNEXT'19*, Orlando, Florida, USA, 2019, pp. 123 – 137.

[12] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted DNS $\Rightarrow$ Privacy? A Traffic Analysis Perspective," in *NDSS'20*, San Diego, CA, USA, 2020.

[13] S. D. Siby, M. Juárez, N. Vallina-Rodriguez, and C. Troncoso, "DNS Privacy not so private: the traffic analysis perspective," in *HotPETs'18*, 2018.

[14] A. Mayrhofer, "Padding Policies for Extension Mechanisms for DNS (EDNS(0))," RFC 8467, 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc8467

[15] 2020. [Online]. Available: https://www.alexa.com

[16] J. Hayes and G. Danezis, "K-fingerprinting: a Robust Scalable Website Fingerprinting Technique," in *USENIX Security'16*, Austin, TX, 2016, pp. 1187 – 1203.

[17] D. Herrmann, R. Wendolsky, and H. Federrath, "Website Fingerprinting: attacking popular privacy enhancing technologies with multinomial naïve-bayes classifier," in *ACM Workshop on Cloud Computing Security (CCSW'09)*, Chicago, Illinois, USA, 2009, pp. 31 – 42.

[18] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *ACM CCS'12*, Raleigh, NC, USA, 2012, pp. 605 – 616.

[19] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *ACM CCS'14*, Scottsdale, AZ, USA, 2014, pp. 227 – 238.

[20] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "A Real-time Defense against Website Fingerprinting Attacks," *arXiv*, vol. abs/2102.04291, 2021.

[21] J.-P. Smith, P. Mittal, and A. Perrig, "Website Fingerprinting in the age of QUIC," *Computer Networks*, vol. 200, pp. 48 – 69, 2021.

[22] P. Zhan, L. Wang, and Y. Tang, "Website Fingerprinting on Early QUIC Traffic," *arXiv preprint arXiv:2101.11871*, 2021.

[23] 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," in *ACM IMC'19*, Amsterdam, Netherlands, 2019, pp. 22 – 35.

[24] S. García, K. Hynek, D. Vekshin, T. Čejka, and A. Wasicek, "Large Scale Measurement on the Adoption of Encrypted DNS," in *arXiv 2107.04436*, 2021.

[25] C. Deccio and J. Davis, "DNS Privacy in Practice and Preparation," in *ACM CoNEXT'19*, Orlando, FL, USA, 2019, pp. 138 – 143.

[26] M. Lyu, H. H. Gharakheili, and V. Sivaraman, "A Survey on DNS Encryption: Current Development, Malware Misuse, and Inference Techniques," *arXiv preprint arXiv:2201.00900*, 2022.

[27] D. Vekshin, K. Hynek, and T. Cejka, "DoH Insight: Detecting DNS over HTTPS by Machine Learning," in *ARES'20*, New York, USA, 2020, pp. 1 – 8.

[28] 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.

[29] A. Meshkov, 2021. [Online]. Available: https://github.com/AdguardTeam/dnsproxy

[30] [Online]. Available: https://www.isc.org/bind

[31] [Online]. Available: {https://www.mozilla.org}

[32] [Online]. Available: https://fortiguard.com/webfilter

[33] B. Muthukadan, 2018. [Online]. Available: https://selenium-python.readthedocs.io/

[34] [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[35] K. Carmen, J. Paul, Q. Shela, W. Cathy, and B. Cecylia, "Exploring Simple Detection Techniques for DNS-over-HTTPS Tunnels," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet (FOCI'21)*, ser. FOCI'21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 37 – 42. [Online]. Available: https://doi.org/10.1145/3473604.3474563

[36] [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[37] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[38] D. Cournapeau, 2007. [Online]. Available: https://scikit-learn.org