Identifying Malicious DNS Tunnel Tools from DoH Traffic Using Hierarchical Machine Learning Classification

Rikima Mitsuhashi[1,2], Akihiro Satoh[3], Yong Jin[4], Katsuyoshi Iida[2], Takahiro Shinagawa[1], and Yoshiaki Takai[2]

[1] The University of Tokyo, Tokyo, Japan
mitsuhashi@os.ecc.u-tokyo.ac.jp, shina@ecc.u-tokyo.ac.jp
[2] Hokkaido University, Hokkaido, Japan
{iida, ytakai}@iic.hokudai.ac.jp
[3] Kyushu Institute of Technology, Fukuoka, Japan
satoh@isc.kyutech.ac.jp
[4] Tokyo Institute of Technology, Tokyo, Japan
yongj@gsic.titech.ac.jp

Abstract. Although the DNS over HTTPS (DoH) protocol has desirable properties for Internet users such as privacy and security, it also causes a problem in that network administrators are prevented from detecting suspicious network traffic generated by malware and malicious tools. To support their efforts in maintaining network security, in this paper, we propose a novel system that identifies malicious DNS tunnel tools through a hierarchical classification method that uses machine-learning technology on DoH traffic. We implemented a prototype of the proposed system and evaluated its performance on the CIRA-CIC-DoHBrw-2020 dataset, obtaining 99.81% accuracy in DoH traffic filtering, 99.99% accuracy in suspicious DoH traffic detection, and 97.22% accuracy in identification of malicious DNS tunnel tools.

Keywords: DNS over HTTPS (DoH) · Network traffic classification · Suspicious DoH traffic · DNS tunnel · Malicious DNS tunnel tool identification.

1 Introduction

There is growing momentum to encrypt DNS traffic on the Internet for privacy and security concerns. A promising method to encrypt DNS traffic is DNS over HTTPS (DoH), which uses SSL/TLS protocols for encryption and has been standardized in RFC8484 [5]. DoH has already been implemented in the latest versions of major web browsers, such as Firefox and Google Chrome. In a client system, DoH can be used by installing proxy software such as cloudflared [4], doh-proxy [10], dnscrypt-proxy [8] and doh-client [9] for all DNS domain name resolutions. For OS support, DoH is available in the insider preview build provided by the Windows Insider Program [15,16]. The structure of domain name

resolution using DoH is depicted in Fig. 1. DoH encrypts DNS traffic by using the HTTPS protocol between the client and the DoH server that works as a DNS full-service resolver, and the DoH server uses the conventional DNS protocol with authoritative DNS servers on the Internet for domain name resolutions.

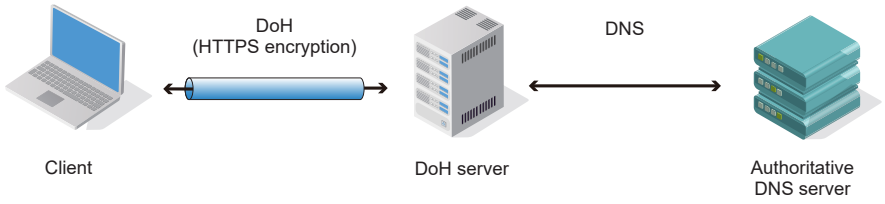

Fig. 1. Domain name resolution using DoH.

By encrypting DNS traffic using DoH technology, Internet users no longer have to worry about privacy violations where someone can eavesdrop on domain name resolutions when they visit websites, and the risk is reduced of their being directed to an unintended website due to tampering by DNS cache poisoning attacks [33]. On the other hand, DoH technology has a problem in that it prevents network administrators from monitoring network traffic for providing network security services. When malware communicates with a command and control (C&C) server on the Internet by using DoH technology, the network administrators cannot detect the communication even if they are monitoring the entire network. The existence of malware that uses DoH to communicate with C&C servers has been confirmed [11]. Many large-scale malware attacks have also been reported [18] and cyber-attacks will not stop on their own. Therefore, the fact that network traffic cannot be monitored because of encryption is a critical issue for network administrators. To deal with this problem, a DoH server could be set up in an organization's network. The network administrator can monitor the DNS traffic as usual since the DNS traffic between the DoH server and the authoritative DNS server uses the conventional DNS protocol without encryption. However, if the clients do not use that DoH server and instead use public DoH servers provided by Internet service providers such as Google and Cloudflare, again the network administrators cannot monitor the DNS traffic.

To determine proper network security solutions, network administrators need to identify the original application programs that generate the malicious DoH traffic, such as malicious DNS tunnel tools. Consequently, it is possible to block traffic to the external websites to which the compromised internal computers access and download the DNS tunnel tools and create rules that prohibit the use of DNS tunnel tools or identify clients in which the DNS tunnel tools are used. In the literature, several approaches have been proposed to detect malicious DoH traffic [29,34]. However, the authors only proposed methods for detecting malicious DoH traffic; their methods cannot identify the programs that gener-

ated the traffic. Therefore, network administrators cannot block the particular problematic traffic generated by compromised computers or stop the spread of vulnerabilities within their organization's network.

Machine-learning technology is a useful way to identify DNS tunnel tools, because it can automatically classify DoH traffic according to its characteristics. However, when the technology is learning the features of DNS tunnel tools, the DNS tools must be running and a DoH proxy has to be prepared to convert the DNS traffic into DoH traffic. Moreover, a large amount of data must be gathered over a long period to generate the training traffic flows since the amount of data in a single packet is so small that it is difficult to determine whether or not the traffic is malicious by verifying an individual packet. Furthermore, because the DoH traffic is encrypted, the features for identifying DNS tunnel tools should be found in the limited clues contained in the DoH, such as the packet header, packet number, packet length, packet direction, and arrival interval between packets. Thus, using machine-learning technology to identify DNS tunnel tools takes time and effort. However, once the machine-learning model has been trained and is up and running, it can analyze DoH traffic automatically.

In this paper, we propose a novel system that uses machine-learning technology on DoH traffic to identify malicious DNS tunnel tools that generate encrypted DNS traffic. As shown in Fig. 2, we use a hierarchical network traffic classification. The unique process in the proposed system is the 3rd stage, which is the identification of malicious DNS tunnel tools. We have made parameter-tuned models suitable for each stage of the classification, which enable us to identify DNS tunnel tools with more accuracy.

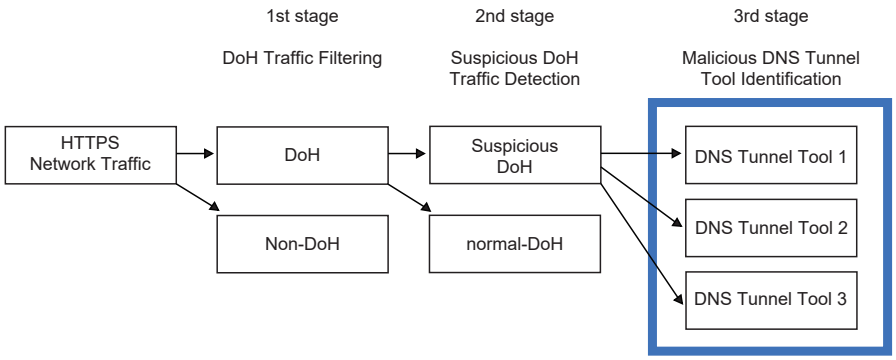

Fig. 2. Concept of hierarchical network traffic classification.

We evaluated the proposed system on the CIRA-CIC-DoHBrw-2020 dataset [3] and found that the DoH traffic filtering had an accuracy and F-score of 99.81% and 99.87%. As far as we know, this F-score is higher than any of the previously reported ones. Regarding detection of suspicious DoH traffic, the accuracy and

F-score were 99.99% and 99.99%, respectively. The accuracy of identifying malicious DNS tunnel tools was 97.22% and the F-score was 95.19%. To the best of our knowledge, this is the first report to show the possibility of identifying malicious DNS tunnel tools from an analysis of DoH traffic.

Our contributions are as follows.

- We propose a novel system that uses machine-learning technology on DoH traffic to identify malicious DNS tunnel tools through a hierarchical classification method.
- We show that the proposed system can identify malicious DNS tunnel tools with enough accuracy that it can support security maintenance efforts by network administrators.
- Experiments conducted on the CIC-DoHBrw-2020 dataset indicate that the proposed system can distinguish DoH traffic from other HTTPS traffic more accurately than the previous methods can.

The rest of this paper is organized as follows. Section 2 presents related work on network traffic classification. Section 3 describes the proposed system design of hierarchical network traffic classification. Section 4 shows the experimental evaluation. Section 5 concludes the paper.

2 Related Work

2.1 Network Traffic Classification

Network traffic classification is a very active research area. In particular, a number of approaches to network traffic classification using machine-learning technology have been proposed [30]. As well, there are many reports on classification of encrypted network traffic [19]. However, since DoH technology has a short history and is still in the process of deployment as a practical application, few research reports on DoH traffic classification are included in the survey papers.

Looking at the recently reported research on DoH network classification, D. Vekshin et al. [36] identified DoH network traffic by classifying HTTPS traffic by using machine learning. They also identified DoH clients such as Chrome, Firefox, and Cloudflare by classifying DoH traffic. For both classifications, they used the Ada-boosted decision tree model and obtained a classification accuracy of 99.9%. The dataset they used was the access data to the domain names taken from the top one million websites provided by Alexa [1]. M. MontazeriShatoori et al. [29] used machine-learning technology to classify HTTPS and DoH traffic, then benign and malicious DoH traffic. Both classifications used the random forest model; the former yielded a 99.3% F-score and the latter a 99.9% F-score. They used the CIRA-CIC-DoHBrw-2020 dataset for the evaluation. S. K. Singh et al. [34] improved the accuracy of classifying benign and malicious DoH traffic on the CIRA-CIC-DoHBrw-2020 dataset. They used the gradient boost model and obtained 100% classification accuracy with the holdout method.

2.2 DNS Tunnel Detection

Many studies have been reported on attack methods and countermeasures against DNS [20,24,25,26], and the research field of DNS tunnel detection has received particular attention recently. Because domain name resolution based on DNS is one of the most basic and indispensable services on the Internet, attackers exploit the characteristics of DNS to build tunnels. A DNS tunnel is a common technique attackers use to establish C&C nodes and to exfiltrate data from networks [21].

Regarding recent reports on DNS tunnel detection, P. Yang et al. [38] tried to detect DNS covert channels by using a stacking model. The DNS traffic was generated by the collection of tools, including dns2tcp, dnscat2, DeNiSe and Heyoka. They used a stacking model that is an ensemble of three different algorithms (K-nearest neighbors (KNN), support vector machine (SVM) and random forest). A. L. Buczak et al. [21] also detected DNS tunnels by analyzing network traffic. They extracted features from a penetration testing effort and trained random forest classifiers to distinguish normal DNS activity from DNS tunneling activity. D. Lambion et al. [28] detected malicious DNS tunnels by using a convolutional neural network (CNN), random forest, and ensemble classifiers for DNS traffic. They assessed the classifiers' performance and robustness by exposing them to one day of real-traffic data. Y. Chen et al. [22] proposed a framework for DNS tunnel detection using long short-term memory (LSTM), gated recurrent unit (GRU) and CNN. A. Chowdhary et al. [23] presented two methods for detecting DNS tunneling queries. The first method uses cache misses in a DNS full-service resolver and the second method utilizes machine-learning technology to classify a given DNS query. K. Wu et al. [37] introduced a three-stage DNS tunnel detection method based on a character feature extraction, called FTPB, which uses feature extraction to filter out the domain names resolved by the DNS tunnels.

In summary, no studies on either network traffic classification or DNS tunnel detection have reported on DoH traffic classification for identifying malicious DNS tunnel tools. In contrast, in experiments conducted on the CIRA-CIC-DoHBrw-2020 dataset, we have achieved the same or better accuracy than those of previous methods of classification in the 1st and 2nd stage. In addition, we also implemented DNS tunnel tool identification in the 3rd stage.

3 Design

In Section 2, we introduced some related work regarding the network traffic classification and investigated the methods of DNS tunnels detection. For network administrators to maintain network security, they need to identify any malicious tools communicating in the DoH traffic. In this section, we describe the design of our system.

3.1 System Overview

To be able to identify the malicious tools used in the DoH communication, it is necessary to analyze the characteristics of network traffic. We introduce a hi-

erarchical classification method to identify malicious tools. The key idea is to determine the best machine-learning model for each stage of the traffic classification. As shown in Fig. 3, the traffic data classification consists of three blocks: 1) DoH traffic filtering, 2) suspicious DoH traffic detection, and 3) malicious DNS tunnel tool identification. In the following subsections, we explain the details of each block.

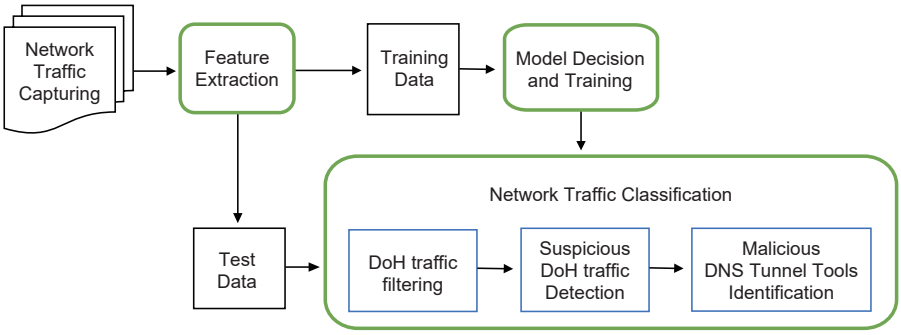

Fig. 3. Overview of proposed system to identify malicious DNS tunnel tools.

3.2 Capturing and Extracting the Features of Network Traffic

As DoH encrypts DNS traffic by using the SSL/TLS protocol, the proposed system takes HTTPS traffic as input data. Although there are likely many different types of traffic in a network, we can determine if the traffic is generated by HTTPS from the source or destination port number of the packet. The HTTPS traffic generated when the client connects to the web server or DoH server is collected at the capture points shown in Fig. 4. The purpose of the client's connection to the web server is to retrieve web content. The client connects to the normal DoH server to resolve domain names, but the malicious DNS tunnel tools on the client might connect with a suspicious DNS server to receive attack instructions or send sensitive information.

To classify the acquired network traffic with machine-learning models, statistical features are extracted from HTTPS traffic of two-way communications. Each traffic is determined by the source IP address, destination IP address, source port number, and destination port number. This information is included in the header of the packet and can be used because it is not encrypted. Statistical features of the traffic are extracted using a series of packets, e.g., number of packets, packet direction, packet arrival time, and packet length, etc. The payload of packets in the HTTPS traffic is encrypted, but these external characteristics can be ascertained.

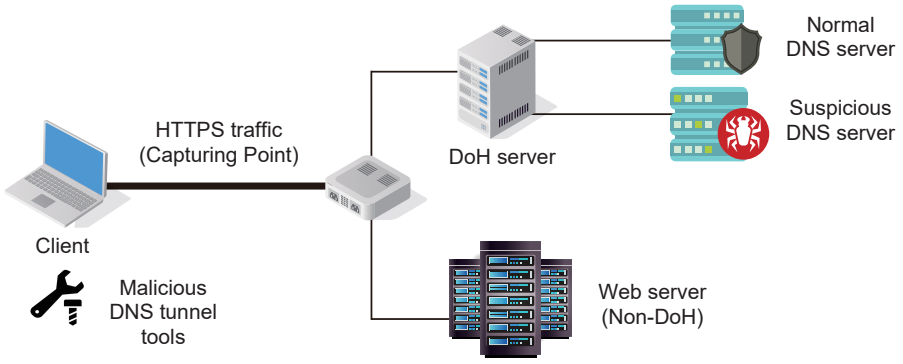

Fig. 4. Network connections and capture point of HTTPS traffic.

3.3 Model Decision and Training

In this subsection, we describe how to determine the model to be used at each stage to implement a hierarchical classification for identifying malicious DNS tunnel tools. In the proposed system, we use the XGBoost [35], LightGBM [27], and CatBoost [31] libraries using the gradient boosting decision tree (GBDT) algorithm. According to S. R et al. [32], these GBDT libraries have substantial flexibility and considerably faster training times compared to other machine learning algorithms at present. They also describe these libraries are widely used in competitive machine learning contests like Kaggle [13] because of their expected high classification accuracy. Generally, boosting is a general ensemble technique that produces a strong classifier from a large number of weak classifiers. As for GBDT, the learning process is as follows. First, a very simple tree that predicts a single number is used. Next, the residual error (observed - predicted) of the tree is calculated. Then, the next decision tree is added to reduce the residual error. If there is still a significant amount of error remaining, another decision tree is added to decrease the error. By repeating this process, a strong classifier is produced.

In addition to the use of high-performance machine-learning libraries, parameter tuning suitable for the dataset is also important to obtain high classification accuracy. We present a method to determine a suitable machine-learning model for traffic classification in Fig. 5. We first train the training data against the parameter-tuned model. Next, we use the trained models to classify the validation data. By comparing the accuracy obtained from the classification of the validation data, we can determine the best parameter-tuned model for the dataset. The determined model is then trained again on the training and validation data and used as a classifier for the test data. This process is carried out in each of the three stages, resulting in the determination of three classifiers. Here we note the problem of overfitting. Overfitting means that a machine-learning model which is closely related to a particular dataset cannot accurately classify additional data. If the parameter tuning overfits the model to the validation data, the model

does not classify the test data with sufficient accuracy. Therefore, the results of parameter tuning need to be analyzed by using not only the classification results of the validation data, but also those of the test data.

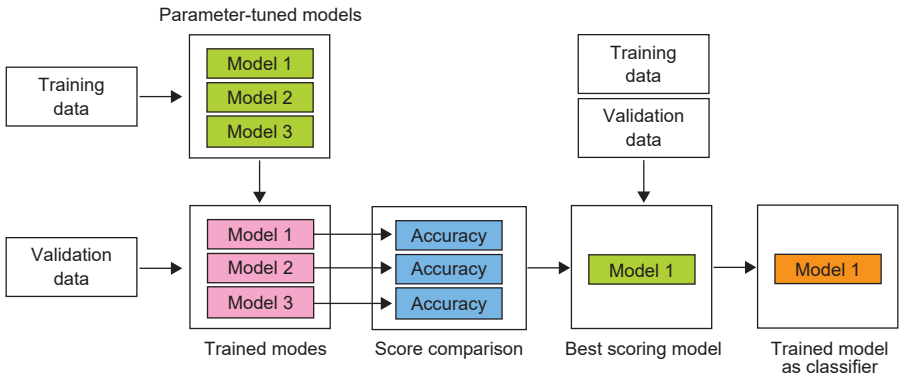

Fig. 5. Model decision process of network traffic classification.

Table 1 shows the parameters used for tuning each model. The parameters are described in the documentation of each model [17,14,2] as being effective in improving classification accuracy. We create specific parameter values by spreading out from the default values in a certain range. We then use a grid search to find the combination that classifies the validation data with the highest accuracy. Here, XGBoost has 35 models, LightGBM has 56 models, and CatBoost has 48 models, so a total of 139 different models are available.

Table 1. Parameters of grid search (underline indicates default parameters).

XGBoost	LightGBM	CatBoost
max_depth:	num_leaves:	max_depth:
2, 4, <u>6</u>, 8, 10, 12, 14	7 ,15, <u>31</u>, 63, 127, 255, 511	2, 4, <u>6</u>, 8, 10, 12, 14, 16
max_bin:	max_bin:	l2_leaf_reg:
128, <u>256</u>, 512, 1024, 2048	127, <u>255</u>, 511, 1023, 2047, 4095 8191, 16383	1, 2, <u>3</u>, 4, 5, 6

3.4 Network Traffic Classification

In order to make it possible to identify malicious tools used in DoH communications, the proposed system classifies network traffic in three stages. Analyzing the network traffic in detail at each stage enhances the possibility of achieving

classification with better accuracy. The classifier to be used in each stage is the one determined by the system in Section 3.3. Fig. 6 shows a diagram of the classification, in which the 1st stage classifies the HTTPS traffic data into DoH and non-DoH; the 2nd stage classifies the DoH traffic into normal DoH and suspicious DoH; and the 3rd stage identifies the suspicious DNS tunnel tools that generated the DoH traffic. In terms of the 3rd stage, numerous DNS tunnel tools are available, such as pick pocket, ozymands, DeNiSe, Heyoka, and many more. From the perspective of a risk-based approach, it is effective to start by identifying the high-risk ones and increase the number of targets step by step. Hence, in this paper, we focus on identifying the well-known and frequently used DNS tunnel tools: dns2tcp [6], dnscat2 [7] and iodine [12].

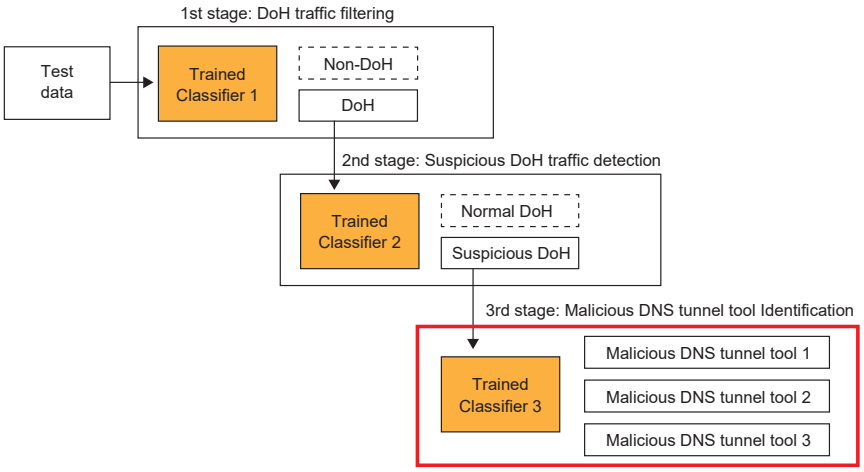

Fig. 6. Hierarchical classification to identify malicious DNS tunnel tools.

4 Evaluation

In Section 3, we presented the overall picture of the proposed system, explained how to determine the machine-learning model to be used in the hierarchical classification, and described the classification targets at each stage. In this section, we evaluated the classification performance of an implementation of the proposed system and analyzed the important features.

4.1 Implementation

On the basis of the design proposed in Section 3, we implemented the proposed system as follows. In terms of the hardware environment, we used a machine

with an Intel Xeon Silver 4210R CPU, 96-GiB memory and Nvidia GeForce RTX 3080 GPU. The software environments were Ubuntu 20.04 with singularity 3.7.3 and Nvidia TensorFlow Release21.02 Container. The machine-learning libraries that we ran and parameter-tuned were XGBoost 1.3.3, LightGBM 3.2.1, and CatBoost 0.25.1. Note that XGBoost and CatBoost were run on the GPU, while LightGBM was run on the CPU.

4.2 Dataset

The experimental evaluations used the CIRA-CIC-DoHBrw-2020 dataset. The number of labels and traffic included in this dataset is shown in Table 2. Since the dataset has a bias in the amount of traffic in each stage, it is important to understand not only the results of the overall classification but also the results of the classifications with a small amount of traffic. We extract 28 statistical traffic features from the dataset as shown in Table 3. We use all the statistical traffic features in each of the three stages.

Table 2. Labels and amount of traffic in the dataset.

	Labels	Traffic
1st stage (HTTPS)	Non-DoH traffic	897494
	DoH traffic	269643
2nd stage (DoH)	Normal DoH traffic	19807
	Suspicious DoH traffic	249836
3rd stage (Malicious DNS tunnel tool)	dns2tcp	167486
	dnscat2	35770
	iodine	46580

We used stratified 10-fold cross-validation to classify the network traffic of the dataset; thus, in each test evaluation, the training and test data were split in the ratio of 9:1. We used accuracy, recall, precision, and F-score as the metrics to measure the overall classification results and those of classifications with a small amount of traffic. The formulas for each of these metrics are as follows. Note that, for the multi-class classification in the 3rd stage, we also used the macro-average of each metric.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Table 3. List of statistical traffic features.

1 Number of flow bytes sent	15 Mode Packet Time
2 Rate of flow bytes sent	16 Variance of Packet Time
3 Number of flow bytes received	17 Standard Deviation of Packet Time
4 Rate of flow bytes received	18 Coefficient of Variation of Packet Time
5 Mean Packet Length	19 Skew from median Packet Time
6 Median Packet Length	20 Skew from mode Packet Time
7 Mode Packet Length	21 Mean Request/response time difference
8 Variance of Packet Length	22 Median Request/response time difference
9 Standard Deviation of Packet Length	23 Mode Request/response time difference
10 Coefficient of Variation of Packet Length	24 Variance of Request/response time difference
11 Skew from median Packet Length	25 Standard Deviation of Request/response time difference
12 Skew from mode Packet Length	26 Coefficient of Variation of Request/response time difference
13 Mean Packet Time	27 Skew from median Request/response time difference
14 Median Packet Time	28 Skew from mode Request/response time difference

$$Recall = \frac{TP}{TP + FN}$$

$$F\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

where TP means true positive, FP means false positive, FN means false negative, and TN means true negative. Furthermore, we use mean-time-between-false-alarms (MTBFA) as a metric of classification. MTBFA is the average time between false alarms in monitoring, which is calculated by the following equation. As the number of false alerts (false positives) increases due to the insufficient classification accuracy of the system, MTBFA becomes shorter.

$$MTBFA = \frac{Monitoring\ hours}{Number\ of\ false\ alarms}$$

4.3 Model Decision

Table 4 shows the best accuracy and parameters obtained by performing a grid search on validation data. As all the parameters shown here are larger than the

minimum value of the search range and smaller than the maximum value, we concluded that we did not need to extend the search range any further. By comparing the accuracy at each stage, we decided to use the parameter-tuned XGBoost for DoH traffic filtering in the 1st stage, the parameter-tuned LightGBM for suspicious DoH traffic detection in the 2nd stage, and the parameter-tuned CatBoost for malicious DNS tunnel tool identification in the 3rd stage. Note that, in some of the experiments, the system resources were insufficient. As for the combination of max_depth: 14 and L2_leaf_reg: [1,2,3,4,5,6] in the 3rd stage, CatBoost used the CPU because of a shortage of GPU memory.

Table 4. Best score and parameters obtained by grid search.

	1st stage: DoH traffic filtering		2nd stage: Suspicious DoH traffic detection		3rd stage: Malicious DNS tunnel tool identification	
XGBoost	accuracy	0.9981	accuracy	0.99998	accuracy	0.9719
	max_depth	12	max_depth	4	max_depth	6
	max_bin	1024	max_bin	512	max_bin	1024
LightGBM	accuracy	0.9981	accuracy	0.99997	accuracy	0.9721
	num_leaves	255	num_leaves	15	num_leaves	63
	max_bin	511	max_bin	255	max_bin	8191
CatBoost	accuracy	0.9979	accuracy	0.99999	accuracy	0.9690
	max_depth	14	max_depth	4	max_depth	10
	L2_leaf_reg	5	L2_leaf_reg	2	L2_leaf_reg	4

4.4 Results of Malicious DNS Tunnel Tool Identification

The results of network traffic classification using test data and classifiers that were selected in Section 4.3 are shown in Table 5. The results of identifying malicious DNS tunnel tools in the 3rd stage were 97.22% in accuracy and 95.19% in F-score. Moreover, the results of filtering DoH traffic in the 1st stage were 99.81% in accuracy and 99.87% in F-score, while the results of detecting suspicious DoH traffic in the 2nd stage were 99.99% in accuracy and 99.99% in F-score. These results indicate that the performance of the proposed system is sufficient to support network administrators in their efforts to maintain network security. In addition, the results of our model were better or equal to those of the other finalist models, which means that the overfitting problem associated with the parameter tuning did not occur.

Looking at MTBFA in the 2nd stage, it was much longer than that in the 1st and 3rd stages. This is due to the high classification accuracy of the 2nd stage, which is acceptable in a large-scale real network environment. In terms of MTBFA in the 3rd stage, it was half an hour and shorter than that in the

2nd stage. To use the proposed system in a real network, it is desirable to be a little longer. This can be achieved by improving the classification accuracy in the 3rd stage. As for the three sequential stages, the metrics were calculated by reflecting the false positives and false negatives of the 1st and 2nd stage to the 3rd stage. The classifiers were XGBoost in the 1st stage, CatBoost in the 2nd stage, and LightGBM in the 3rd stage. Since the accuracy of classification in the 1st and 2nd stage was relatively high, the metrics in the 3rd stage were affected slightly.

Table 5. Results of network traffic classification using test data.

	Classifiers	Accuracy	Precision	Recall	F-score	MTBFA
1st stage:	XGBoost	0.9981	0.9981	0.9994	0.9987	181 min
DoH traffic filtering	LightGBM	0.9981	0.9980	0.9995	0.9987	111 min
	CatBoost	0.9979	0.9978	0.9995	0.9986	101 min
2nd stage:	CatBoost	0.9999	1.0	0.9999	0.9999	80683 min
Suspicious DoH traffic	LightGBM	0.9999	0.9999	0.9999	0.9999	48410 min
detection	XGBoost	0.9999	0.9999	0.9999	0.9999	30256 min
3rd stage:	LightGBM	0.9722	0.9497	0.9543	0.9519	33 min
Malicious DNS tunnel	XGBoost	0.9706	0.9473	0.9518	0.9495	32 min
tool identification	CatBoost	0.9691	0.9446	0.9494	0.9469	29 min
Three sequential stages:		0.9703	0.9487	0.9503	0.9494	31 min

A performance comparison between the proposed system and the systems of the previous studies is shown in Table 6. All previous studies have used the holdout method, which divides the dataset into training and test data, and then evaluates them once. In contrast, the 10 fold cross-validation we used performed 10 evaluations using the training and test data, and then calculated the average of these evaluations. To align the comparisons, we picked the best results from the 10 evaluations. In the 1st stage, which is DoH traffic filtering, our system had the highest precision, recall, and F-score. In the 2nd stage, which is suspicious DoH traffic detection, the results of tour system, like those of the previous studies, reached 1.0 in precision, recall, and F-score. It should be noted that the holdout method may decrease the values of these metrics depending on how the samples are selected when splitting the dataset into training and test data. In the cross-validation results of our system, the precision, recall, and F-score were 1.0, 0.9999 and 0.9999, respectively, with no significant loss in classification accuracy. As far as we know, this is the first attempt to identify malicious DNS tunnel tools; thus, there are no other studies that ours can be compared with.

Table 6. Comparison with previous studies using the holdout method.

	Classifiers	Precision	Recall	F-score
1st stage: DoH traffic filtering	Random Forest [29]	0.993	0.993	0.993
	XGBoost (ours)	**0.9982**	**0.9995**	**0.9989**
2nd stage: Suspicious DoH traffic detection	Random Forest [29]	0.999	0.999	0.999
	Gradient Boost [34]	1.0	1.0	1.0
	CatBoost (ours)	**1.0**	**1.0**	**1.0**
3rd stage: Malicious DNS tunnel tool identification	LightGBM (ours)	0.952	0.956	0.954

4.5 Consideration of Important Features

To analyze the background that enabled the hierarchical traffic data classification, the most important features used by each classifier are listed in Table 7. For filtering DoH traffic in the 1st stage, XGBoost used "Mode Packet Length" as the most important feature, followed by "Mean Packet Time". "Mode Packet Length" means the packet length that appears most often in the traffic flow, while "Mean Packet Time" refers to the average inter-arrival time of packets in the traffic flow. The value of "Mode Packet Length" is much larger than that of "Mean Packet Time", indicating that the former feature is very important. The average size of "Mode Packet Length" in the 1st stage is 164.0 for the non-DoH traffic and 68.0 for the DoH traffic. This difference is due to the fact that the non-DoH traffic contains a lot of data provided by the web server.

Regarding detecting suspicious DoH traffic in the 2nd stage, CatBoost considered "Mode Packet Length" to be the most important feature, followed by "Median Packet Length". Here, "Median Packet Length" means the packet length that separates the higher half of the traffic flow from the lower half. The average size of "Mode Packet Length" in the 2nd stage is 74.1 for the normal DoH traffic and 67.5 for the suspicious DoH traffic. This difference is due to the fact that normal DoH traffic contains a lot of SSL/TLS key exchange data between the client and DoH server. In contrast, suspicious DoH traffic has less of that data, because most malicious DNS tunnel tools stay connected with the DoH server for a long time.

Regarding identifying DoH tunnel tools in the 3rd stage, LightGBM considered "Median Request/response time difference" to be the most important feature, followed by "Skew from median Request/response time difference", "Mode Request/response time difference", and "Skew from mode Request/response time difference". The top-four features are related to "Request/response time difference", which means the inter-arrival time of received packets in the traffic flow. The remaining part of "Skew from median Request/response time difference" means the value defined by the equation: $3 \cdot (mean - median)/standard\ deviation$, while "Skew from mode Request/response time difference" means the value calculated by the following equation: $(mean - mode)/standard\ deviation$. The av-

Table 7. Most important features in hierarchical traffic data classification.

	Classifiers	Important features	Value
1st stage: DoH traffic filtering	XGBoost	Mode Packet Length	0.7757
		Mean Packet Time	0.0819
2nd stage: Suspicious DoH traffic detection	CatBoost	Mode Packet Length	68.9465
		Median Packet Length	13.5604
3rd stage: Malicious DNS tunnel tool identification	LightGBM	Median Request/response time difference	3694
		Skew from median Request/response time difference	3304
		Mode Request/response time difference	2963
		Skew from mode Request/response time difference	2290

erage size of "Median Request/response time difference" in the 3rd stage is 0.2 for dns2tcp, 2.7 for dnscat2, and 1.4 for iodine. Since these malicious DNS tunnel tools were developed by separate organizations, we assume that the difference in processing load on the suspicious DNS server caused the difference in response time. We also considered the possibility that the geographical distance from the client to the suspicious DNS servers could be responsible for the difference in response time, but rejected this hypothesis because, according to the description of the CIRA-CIC-DoHBrw-2020 dataset [3], all the malicious DNS tunnel tools used a single suspicious DNS server in common.

4.6 Discussion

We list up some consideration points regarding the evaluation performed. First of all, we used the most popular and famous DNS tunnel tools in the evaluation considering the high possibility of use by attackers. In Section 3.4, we have distinguished three DNS tunnel tools with high accuracy. We also agree that there are many other types of malicious DNS tunnel tools, and the number may increase. In this case, even if there are some different factors in the new tools, some similarities may remain. Therefore, we expect that the proposed system will also be effective to those new varieties and the specific evaluation will be performed in future work.

Secondly, we performed the evaluations on a local network environment and confirmed the effectiveness of the proposed system. In Section 4.5, we showed that the request-response time feature of network traffic can be used to distinguish between three malicious DNS tunneling tools. It should be noted that in the data we used in our experiments, the malicious DNS servers that each DNS tunnel tool connects to are in a common network and have similar performance specifications. Therefore, we consider that in an environment with similar conditions, the proposed classification using the request-response time feature will work well. Regarding the evaluation in a real network environment, we plan to deploy the proposed system on our campus network and confirm its effectiveness.

On the other hand, in case attackers modify parts of well-known DNS tunnel tools or add new features to them, those tools may be out of the target of the proposed system. Furthermore, the length of connection to the DoH server and the DoH server capacity change based on the different operators. Therefore, we consider that the increase of features specifying these factors will be necessary for the deployment in a real network.

5 Conclusion

DoH technology has been developed to provide security and privacy for Internet users by encrypting the DNS traffic. However, DoH has a significant disadvantage because it prevents network administrators from analyzing network traffic for ensuring network security. Although many studies on encrypted network traffic classification and DNS tunnel detection have been reported, DoH is a new protocol to which previous research results cannot be directly applied.

In this study, we attempted to identify DoH traffic generated by malicious DNS tunnel tools. The payload of DoH traffic is encrypted; thus, its content cannot be accessed. Therefore, we decided to use the statistical features of the packets to analyze the traffic in detail. Our approach is a hierarchical traffic classification in which each stage uses a parameter-tuned model that is suitable for DoH network traffic. We designed, implemented, and evaluated our system with three levels of network traffic classification. For the prototype, we prepared 139 different models by tuning the parameters of the XGBoost, LightGBM, and CatBoost machine-learning libraries, which are expected to have high classification accuracy. To prove that our system can identify malicious DNS tunnel tools and evaluate its performance, we conducted a series of experiments using the CIRA-CIC-DoHBrw-2020 dataset. The results showed that our system can identify malicious DNS tunnel tools with 97.22% accuracy. They also showed that it can filter DoH traffic from normal HTTPS network traffic with 99.81% accuracy and detect suspicious DoH traffic from normal DoH traffic with 99.99% accuracy. We also showed the features that the machine-learning model considered to be important during the classification and discussed the conditions under which high classification accuracy can be achieved by using these features. Then, we discussed several consideration points regarding the evaluation performed.

References

1. Amazon Alexa Voice AI, https://developer.amazon.com/en-US/alexa/. Last accessed 17 July 2021
2. CatBoost Documentation - Parameters, https://catboost.ai/docs/concepts/python-reference_parameters-list.html. Last accessed 16 Jun 2021
3. CIRA-CIC-DoHBrw-2020, https://www.unb.ca/cic/datasets/dohbrw-2020.html. Last accessed 15 Jun 2021
4. cloudflared, https://developers.cloudflare.com/cloudflare-one/connections/connect-apps. Last accessed 10 July 2021

5. DNS Queries over HTTPS (DoH) - Request For Comments 8484, https://tools.ietf.org/html/rfc8484. Last accessed 15 Jun 2021
6. dns2tcp, https://github.com/alex-sector/dns2tcp. Last accessed 3 Jul 2021
7. dnscat2, https://github.com/iagox86/dnscat2. Last accessed 3 Jul 2021
8. dnscrypt-proxy, https://github.com/DNSCrypt. Last accessed 10 July 2021
9. doh-client, https://docs.rs/crate/doh-client/1.1.5. Last accessed 10 July 2021
10. doh-proxy, https://github.com/facebookexperimental/doh-proxy. Last accessed 10 July 2021
11. First-ever malware strain spotted abusing new DoH (DNS over HTTPS) protocol, https://www.zdnet.com/article/first-ever-malware-strain-spotted-abusing-new-doh-dns-over-https-protocol/. Last accessed 10 July 2021
12. iodine, https://code.kryo.se/iodine/. Last accessed 3 Jul 2021
13. Kaggle, https://www.kaggle.com/. Last accessed 16 Jun 2021
14. LightGBM Documentation - Parameters, https://lightgbm.readthedocs.io/en/latest/Parameters.html. Last accessed 16 Jun 2021
15. Windows Insiders can now test DNS over HTTPS, https://techcommunity.microsoft.com/t5/networking-blog/windows-insiders-can-now-test-dns-over-https/ba-p/1381282. Last accessed 10 July 2021
16. Windows Insiders gain new DNS over HTTPS controls, https://techcommunity.microsoft.com/t5/networking-blog/windows-insiders-gain-new-dns-over-https-controls/ba-p/2494644. Last accessed 10 July 2021
17. XGBoost Documentation - Xgboost Parameters, https://xgboost.readthedocs.io/en/latest/parameter.html. Last accessed 16 Jun 2021
18. Acar, A., Lu, L., Uluagac, A.S., Kirda, E.: An Analysis of Malware Trends in Enterprise Networks. In: Lin, Z., Papamanthou, C., Polychronakis, M. (eds.) Proceedings of Information Security. pp. 360–380 (2019)
19. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges. IEEE Transactions on Network and Service Management **16**(2), 445–458 (2019)
20. Ajmera, S., Pattanshetti, T.: A Survey Report on Identifying Different Machine Learning Algorithms in Detecting Domain Generation Algorithms within Enterprise Network. In: Proceedings of 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–5 (2020)
21. Buczak, A.L., Hanke, P.A., Cancro, G.J., Toma, M.K., Watkins, L.A., Chavis, J.S.: Detection of Tunnels in PCAP Data by Random Forests. In: Proceedings of the 11th Annual Cyber and Information Security Research Conference (2016)
22. Chen, Y., Li, X.: A High Accuracy DNS Tunnel Detection Method Without Feature Engineering. In: Proceedings of 2020 16th International Conference on Computational Intelligence and Security (CIS). pp. 374–377 (2020)
23. Chowdhary, A., Bhowmik, M., Rudra, B.: DNS Tunneling Detection using Machine Learning and Cache Miss Properties. In: Proceedings of 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). pp. 1225–1229 (2021)
24. Ichise, H., Jin, Y., Iida, K.: Analysis of DNS TXT Record Usage and Consideration of Botnet Communication Detection. IEICE Transactions on Communications **E101**(1), 70–79 (2018). https://doi.org/10.1587/transcom.2017ITP0009
25. Ichise, H., Jin, Y., Iida, K., Takai, Y.: NS record History Based Abnormal DNS traffic Detection Considering Adaptive Botnet Communica-

tion Blocking. IPSJ Journal of Information Processing **28**, 112–122 (2020). https://doi.org/10.2197/ipsjjip.28.112
26. Iuchi, Y., Jin, Y., Ichise, H., Iida, K., Takai, Y.: Detection and Blocking of DGA-based Bot Infected Computers by Monitoring NXDOMAIN Responses. In: Proceedings of 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). pp. 82–87 (2020)
27. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: Proceedings of Advances in Neural Information Processing Systems. vol. 30 (2017)
28. Lambion, D., Josten, M., Olumofin, F., De Cock, M.: Malicious DNS Tunneling Detection in Real-Traffic DNS Data. In: Proceedings of 2020 IEEE International Conference on Big Data (Big Data). pp. 5736–5738 (2020)
29. MontazeriShatoori, M., Davidson, L., Kaur, G., Habibi Lashkari, A.: Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic. In: Proceedings of 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). pp. 63–70 (2020)
30. Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., Aguilar, J.: Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey. IEEE Communications Surveys Tutorials **21**(2), 1988–2014 (2019)
31. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: CatBoost: unbiased boosting with categorical features. In: Proceedings of Advances in Neural Information Processing Systems. vol. 31 (2018)
32. R, S., Ayachit, S.S., Patil, V., Singh, A.: Competitive analysis of the top gradient boosting machine learning algorithms. In: Proceedings of 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN). pp. 191–196 (2020)
33. Siby, S., Juarez, M., Diaz, C., Vallina-Rodriguez, N., Troncoso, C.: Encrypted DNS → Privacy? In: Proceedings of Network and Distributed Systems Security (NDSS) Symposium 2020 (2020)
34. Singh, S.K., Roy, P.K.: Detecting Malicious DNS over HTTPS Traffic Using Machine Learning. In: Proceedings of 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies. pp. 1–6 (2020)
35. Tianqi, C., Carlos, G.: XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 785–794 (2016)
36. Vekshin, D., Hynek, K., Cejka, T.: DoH Insight: Detecting DNS over HTTPS by Machine Learning. In: Proceedings of the 15th International Conference on Availability, Reliability and Security (2020)
37. Wu, K., Zhang, Y., Yin, T.: FTPB: A Three-Stage DNS Tunnel Detection Method Based on Character Feature Extraction. In: Proceedings of 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 250–258 (2020)
38. Yang, P., Wan, X., Shi, G., Qu, H., Li, J., Yang, L.: Naruto: DNS Covert Channels Detection Based on Stacking Model. In: Proceedings of the 2020 The 2nd World Symposium on Software Engineering. p. 109–115 (2020)