Detecting DoH-Based Data Exfiltration: FluBot Malware Case Study

***Abstract*—FluBot is a highly sophisticated Android banking Trojan that has been actively used in attacks during 2021 and 2022. This paper proposes an approach for detecting FluBot malware using a machine learning-based DNS-over-HTTPS (DOH) classifier. Our approach leverages the fact that FluBot communicates with its command and control server using DOH, a secure protocol that encrypts DNS queries and responses. We collected a dataset of network traffic traces containing FluBot malware samples and benign traffic and used it to train a machine-learning model to classify DOH traffic as either malicious or benign. Our experimental results show that our DOH classifier achieves high accuracy and detection rates in identifying FluBot malware while maintaining a low false positive rate.**

***Index Terms*—malware analysis, DNS-over-HTTPS, covert channel, traffic classification, DGA detection, FluBot**

I. Introduction

DNS-over-HTTPS (DoH) is a recently developed protocol that enables unencrypted DNS messages to be transmitted securely through an encrypted HTTPS channel. While this protocol provides enhanced security for online activities, it is also attractive to malware seeking to use it as a command and control (C2) communication tool. The challenge with DoH is that it blends in with other HTTPS traffic, making it difficult for security solutions to detect and analyze it effectively. It can easily bypass network filters that are designed to block non-HTTPS traffic. Attackers can use DoH to establish covert, undetected communication channels, allowing them to maintain control of compromised devices and exfiltrate sensitive data. Compounding the issue, legitimate service providers such as web browsers and operating systems are increasingly adopting DoH, providing a cover for malicious traffic. Malware can use the same encrypted channel as legitimate traffic, making identifying and blocking malicious traffic challenging without accidentally blocking it.

One concrete example of such malware is FluBot [1]. It is a targeted Android banking Trojan that steals sensitive information such as login credentials and payment card details. To contact its C2 server, FluBot uses a Domain Generation Algorithm (DGA). In version 4.9, FluBot resolved the IP addresses of DGAs and communicated directly with the server using HTTPS port 443. However, in version 5.0, the authors of FluBot changed their approach to use DoH for covert communication with the server. By using DoH, FluBot is able to exfiltrate data and conceal the content of its communication, making it difficult to identify as covert communication.

Several works were recently published to distinguish DoH in HTTPS communication and malicious DoH traffic [2, 3, 4]. The proposed methods either consist of only a classifier for identifying DoH [5], or provide additional detectors for malware activities hidden within DoH [2, 3, 4]. The approaches are based solely on machine learning classification working with independent network connections. These proposals detect different kinds of malware [2] or focus on specific strains such as DGA-based [3].

In this paper, we are proposing a two-layer detection method to identify malware's network activity. The first layer utilizes machine learning to detect DNS-over-HTTPS (DoH) from Netflow records of network traffic. However, our approach differs from previous methods by incorporating a second layer that employs a modification of the Netflow-based DGA algorithm developed initially by Grill et al. [6] to identify infected machines. It is important to note that this detection method is still a work in progress. However, we have already observed that this approach is successful in applying the DGA detection algorithm to DoH traffic, even though it was originally designed for plaintext DNS. Additionally, our method considers not only single flows but also groups the traffic generated by the hosts, enabling us to detect malicious behavior that may be hidden across multiple DoH connections.

II. Methodology

Our work proposes a two-layer approach to detect malware activities. In the first layer, we classify DNS-over-HTTPS (DoH) traffic using a machine learning classifier. The second layer identifies malware within the identified DoH traffic by using statistical traffic analysis with previously developed malware detection algorithms designed for plaintext DNS traffic [6]. The advantage of this approach is that it allows the same malware detection algorithm to be used for different encrypted DNS protocols (e.g., DNS-over-TLS) by plugging in different classifiers to the first layer.

A. Layer 1: Encrypted DNS traffic classification

This layer identifies DNS-over-HTTPS (DoH) traffic within HTTPS connections. To achieve this, we trained a DoH classifier that operates on bi-directional flow records as input. To extract relevant information from the flow records, we calculated statistical values such as standard deviation, mean, and variance from packet sizes and inter-packet arrival times. These values were then grouped into three categories: incoming, outgoing, and combined. After grouping, we used a robust scaler to scale the values according to the interquartile range and then used them as features for the model. The required data preparation and cleaning are described next within this section.

1) Data Preparation: The main dataset used for training and evaluation of the DoH classification models was the collection of DoH traffic provided in [7]. Network flows from the dataset were extracted using the IPFIXprobe tool, configured to collect the first 100 packets. To classify DNS-over-HTTPS (DoH) and non-DoH traffic, statistically significant HTTPS flows with at least 3 packets were filtered and labeled using an accompanied list of DoH resolvers IP addresses. Using this approach, we obtained a total of 150.10^3 generated DoH, 125.10^3 real-world DoH, and 75.10^3 real-world HTTPS traffic samples, respectively.

After filtering labeling, and data sampling we obtained a representative dataset containing 41% DoH and 59% non-DoH flows, which were later split into the train, test, and validation parts. By incorporating both generated and real-world traffic, we aimed to increase the diversity of the dataset and make it more representative of a real-world environment.

To validate the performance of our model, we created a separate dataset that was partly derived from the validation split of the design dataset (consisting of 5000 DoH and 5000 non-DoH flows), and partly from the benign traffic collected from the sandbox network during the model tuning phase of our study (also consisting of 5000 DoH and 5000 non-DoH flows).

By incorporating both types of data into the validation dataset, we were able to assess the performance of our model not only on the data it was trained on but also on data that it might encounter in a real deployment scenario. This approach created a more robust and accurate estimation of the model's performance in real-world settings.

2) TLS Handshake Influence Elimination: The first few packets of a DoH flow consist of a TLS handshake, and the extracted packet size metadata is more reflective of the HTTPS server deployment than the actual transmitted traffic. To improve the accuracy and discrimination power of the statistical features based on packet sizes, we hypothesized that reducing the influence of these packets on the computed features would be beneficial. This approach could also improve the model's generalization since the TLS handshake may vary depending on the resolver used. Overall, by minimizing the impact of TLS handshake packets on our statistical features, we aim to enhance the performance and accuracy of our DoH classifier.

The packet size of the TLS handshake can vary, and simply skipping these packets may result in losing important information. To address this, we devised a method to calculate the statistical features of these packets while considering their varying packet size. Instead of discarding or treating these packets equally, we applied weights to the first few packets based on their position in the handshake sequence. Specifically, we set a weight for N packets that linearly increases from 0.1 for the first packet to 1.0 for the $N+1$ packet. This ensures that discarding the packets or assigning a weight of 0.0 will result in the same outcome. By using this approach, we can calculate the statistical features of the TLS handshake packets while accounting for their varying packet size and ensure that no important information is lost in the process.

In order to determine the best approach for handling TLS handshake packets, we considered two pairs of integer parameters: $N_{lengths}$ and $M_{lengths}$ for packet sizes, and $N_{intervals}$ and $M_{intervals}$ for inter-packet arrival times. These parameters control which packets are skipped and which ones are assigned linearly increasing weights.

The values of N to M are used to assign weights linearly to packets starting from N to M, while all following packets starting from $M+1$ are assigned a weight of 1.0. Packets preceding N weight 0.

To summarize, the four integer parameters form a four-tuple $(N_{lengths}, M_{lengths}, N_{intervals}, M_{intervals})$, which are used to determine the best combination of skipping and applying weights to the TLS handshake packets (see Table I).

3) Classification: We trained classification models using two algorithms: Random Forest (RF) and Histogram-based Gradient Boosting. In addition, we used Logistic Regression as the baseline model. The classifiers were trained on the design dataset and then tuned to optimize their performance.

To evaluate the trained classifiers, we used a prepared validation dataset and checked several variations of the handshake influence elimination parameters. We used the area under the curve (AUC) as a metric for evaluation.

B. Layer 2: Traffic analysis for malware detection

In the second layer of our approach, we utilized a statistical method proposed by [6] to detect malicious activity in DNS traffic. This method involves calculating the ratio (1) of DNS requests and contacted IP addresses for each host in the local network.

Our idea was that this method, which was initially developed to detect domain generation algorithm (DGA)-based malware for plaintext DNS, could also be applied in the second layer as a malware detector for DoH traffic. Despite the possibility of classification errors, we believed that this method could be effective because it is based on the statistical characteristics of DoH traffic, rather than the content of the DNS queries or responses.

The ratio $\rho(a)$ represents the *specific gravity* of contacted IP addresses per DNS request, and can be used to detect attacks that involve a relatively high number of queries, including DGA-based malware, fast-fluxing-based malware, DNS tunneling, and exfiltration attacks. Of these attacks, DNS tunneling and exfiltration attacks are particularly relevant to DoH.

$$\rho(a) = \frac{\delta(a)}{\pi(a)+1} \tag{1}$$

In Equation 1, the a is the host in the network, $\delta(a)$ values - number of DNS requests created by the host a, $\pi(a)$ values - the number of unique IP addresses contacted by the a.

The overall number of DNS requests $\delta(a)$ is linearly dependent on the observed duration, while the rate of increase of the count of new IPs in the time window is logarithmically

decreases over time. We use $\ln \rho$ (2) in our experiments to make the detection algorithm more robust to the change of the observed time window duration.

$$\ln \rho(a) = \ln \frac{\delta(a)}{\pi(a) + 1} \tag{2}$$

C. Environment Setup

In order to simulate the deployment of our malware detection algorithm in a production network, we set up a sandbox network on a Proxmox server. The sandbox network comprised four virtual machines (VMs): three Android x86 emulators and one Kali Linux instance. To capture traffic at the network border, we connected these machines through an OpenWRT router equipped with the IPFIXprobe tool.

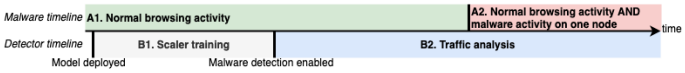

Fig. 1. Timeline of experiments

During the experiments, we utilized three Android VMs to generate network activity, with one of the machines intentionally infected with malware while the other two remained uninfected. The timeline for our experiments, as shown in Figure 1, consisted of two phases: the B1 phase, during which we fitted the scaler, and the B2 phase, during which we ran our malware detection algorithm.

D. Description of Experiments

To assess the proposed method's ability to detect malware in isolated malicious traffic and malicious mixed with benign, we conducted two experiments to detect the deliberately infected machine with FluBot malware:

Experiment 1 – Clean Room Experiment: For this experiment, we infected one Android VM with malware and configured it to generate isolated malicious traffic while the other two Android VMs remained uninfected. To generate benign traffic, we scripted the two uninfected VMs to browse the top 500 domains reported by Moz [8], with one of them enabling DNS-over-HTTPS (DoH) in the Chrome browser. We then applied the trained models to the collected traffic, testing several variations of TLS handshake elimination tuples, including a baseline with no elimination.

Experiment 2 – Real-World Scenario: In this experiment, we intentionally infected an Android machine with the FluBot malware and also generated benign traffic on the same machine by randomly browsing the top 500 domains [8]. The goal of this experiment was to verify whether our proposed method could effectively detect the infected machine in the presence of benign browsing interference. To conduct this experiment, we used the best-performing model from Experiment 1. We also applied an outlier detection technique based on ARIMA, which was trained on the first B1 stage. Confidence intervals were set based on the user's typical behavior profile to detect any anomalous behavior caused by the malware accurately.

III. Results and Discussion

In this section, the preliminary results are presented and discussed.

First, we built and evaluated a total of 45 models for the DoH classifier using the Logistic Regression, Random Forest, and Histogram-based Gradient Boosting algorithms. To optimize the performance of these models, we tested various variations of the packet weighting parameters, with values of N and M ranging from 0 to 6.

TABLE I
Top 3 models AUC, and the baseline model

Model	Packet skip/weight parameters $(N_{lengths}, M_{lengths}, N_{intervals}, M_{intervals})$	AUC
RF (baseline)	$(0, 0, 0, 0)$	0.9026
RF	$(2, 4, 0, 0)$	0.9355
RF	$(2, 4, 2, 4)$	0.9324
RF	$(1, 0, 0, 0)$	0.9278

Table I presents the AUC scores for the top 3 models and the baseline model. The analysis shows that the models which minimize the influence of the first few packets outperform the baseline model, confirming our hypothesis.

Next, the results of performed malware detection experiments are provided.

Experiment 1 – Clean Room Experiment: We collected 163 minutes of data from three Android devices for this experiment, with the model fitting phase (B1) lasting 42 minutes and the model running phase (B2) lasting 120 minutes (with 60 minutes spent in the infected phase A2). To evaluate the performance of the DoH detection algorithm, we utilized the best-performing models from Table I for DoH detection. Then, we calculated the $\ln \rho(a)$ (2) for hosts in the network as follows to determine infected machines.

First, the ratio for the whole period of collected data was calculated. To evaluate the algorithm performance with the given model, the $\Delta \ln \rho(a)$ (3) for each run of the experiment was calculated. The results are presented in Table II.

Initially, the ratio for the entire period of data collection was computed. To assess the performance of the algorithm with the selected model, we then calculated the difference of ratios between benign and malicious samples $\Delta \ln \rho(a)$ (3) for each run of the experiment:

$$\Delta \ln \rho = |\ln \rho(a_{infected}) - \ln \rho(a_{clean})| \tag{3}$$

Table II presents the resulting values. The difference between the benign and malicious classes ranges from 4 to 6 orders of magnitude, indicating that any of the selected models can effectively detect the malware. Based on the results, the $(2, 4, 0, 0)$ model exhibited the highest DoH detection accuracy and performed well in this experiment. However, other selected models can be considered, as their results for malware detection are similar.

Real-time malware detection was simulated by calculating the $\ln \rho(a)$ value for each host in 3-minute windows during the

TABLE II
DIFFERENCE BETWEEN $\ln \rho(a)$ OF INFECTED AND BENIGN HOSTS USING WHOLE COLLECTED DATASET

Model	Parameters	$\Delta \ln \rho(a)$ $sensitivity = 0.8$	$\Delta \ln \rho(a)$ $sensitivity = 0.9$
RF	$(0, 0, 0, 0)$	6.319705	6.441457
RF	$(2, 4, 0, 0)$	6.157668	6.440201
RF	$(2, 4, 2, 4)$	5.984703	6.487608
RF	$(1, 0, 0, 0)$	4.59773	6.264593

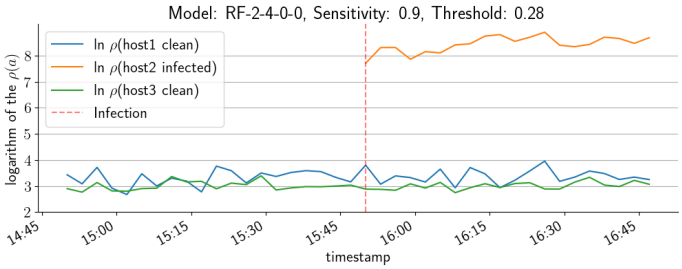

Fig. 2. Time-series visualization of $\ln \rho(a)$ for the Experiment 1

B2 period. The best-performing model (Random Forest algorithm with $(2, 4, 0, 0)$ packet weight parameters) is presented in Figure 2. The moment of infection of $host2$ is clearly visible in the time-series plot.

Experiment 2 – Real-World Scenario: In this experiment, two Android machines were browsing benign websites, and one was infected with FluBot during the experiment. A total of 165 minutes of data was collected, with 41 minutes in the model fitting (B1) phase and 124 minutes in the model running (B2) phase (62 minutes in the infected A2 phase). The best-performing DoH detector was used, and the same approach of detecting infected machines using 3-minute windows were applied as in the previous experiment.

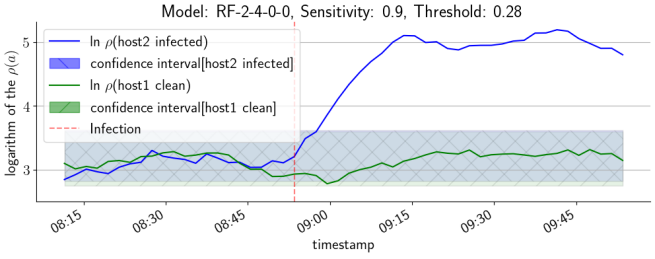

Fig. 3. FluBot infection moment of Experiment 2

Along with the scaler, in the B1 period, the ARIMA model was trained for both hosts to learn their normal behavior. The confidence interval $\alpha = 0.05$ was used in the B2 period to detect outliers, detecting the FluBot infection. The algorithm was able to detect the outlier, which is demonstrated in Figure 3. It can be seen that after the infection moment, the value of $\ln \rho$ rises above the confidence interval, which can be interpreted as an indicator of compromise by an automated system.

IV. CONCLUSION

DoH covert channels have become increasingly popular among malware, as they allow them to bypass traditional security solutions enabling undetected communication with their C2 servers. We presented a novel simple method for detecting FluBot malware using a two-layer classification approach. The capabilities of the method were demonstrated in experiments considering both a sandbox and a more realistic environment.

Future research aims to improve the detection performance and consider other malware utilizing DNS-based covert channels such as DoH, DoT, and DoQ. To achieve this, additional datasets of DNS covert malicious communications will be collected, and the ability to attribute such communications to specific malware families will be improved. Further experiments will be conducted to investigate the validity of the models and their applicability under realistic conditions.

REFERENCES

[1] H. Salsabila, S. Mardhiyah, and R. B. Hadiprakoso, "Flubot malware hybrid analysis on android operating system," in *2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*. IEEE, 2022, pp. 202–206.

[2] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. H. Lashkari, "Detection of doh tunnels using time-series classification of encrypted traffic," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing*. IEEE, 2020, pp. 63–70.

[3] R. Mitsuhashi, Y. Jin, K. Iida, T. Shinagawa, and Y. Takai, "Detection of dga-based malware communications from doh traffic using machine learning analysis," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 2023, pp. 224–229.

[4] M. Behnke, N. Briner, D. Cullen, K. Schwerdtfeger, J. Warren, R. Basnet, and T. Doleck, "Feature engineering and machine learning model comparison for malicious activity detection in the dns-over-https protocol," *IEEE Access*, vol. 9, pp. 129 902–129 916, 2021.

[5] D. Vekshin, K. Hynek, and T. Cejka, "DoH Insight: detecting DNS over HTTPS by machine learning," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3407023.3409192

[6] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak, "Detecting dga malware using netflow," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 1304–1309.

[7] K. Jeřábek, K. Hynek, T. Čejka, and O. Ryšavỳ, "Collection of datasets with dns over https traffic," *Data in Brief*, vol. 42, p. 108310, 2022.

[8] "Top 500 Most Popular Websites." [Online]. Available: https://moz.com/top500