Comprehensive Study Of Malware Analysis on DNS over HTTPS(DoH)

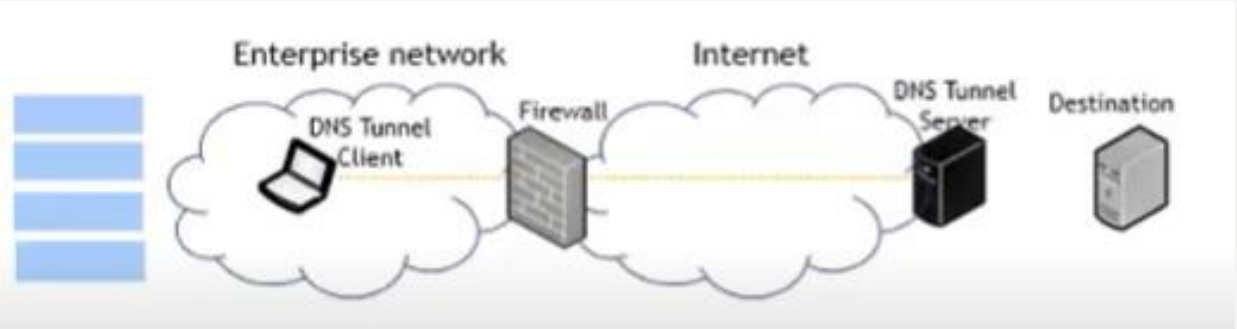
GROUP : LOG4J

SUBMITTED BY :

ABHISHEK SAHU	(21111002)
ASHITOSH VANKATRAO MORE	(21111017)
BINAYA KUMAR SUNA	(21111021)
SHIVAM KHARWAR	(21111058)
TANIKELLA SAI KIRAN	(21111061)
MAJ ASHISH AHLUWALIA	(21111073)

Introduction

- **Domain Name System (DNS) -** Provides a mapping hostnames and Internet Protocol (IP) addresses.



- **Major issues**.

  - **DNS tunneling:** encapsulating data transmission between a client and a server.
  - Susceptible to various **active and passive attacks**, such as man-in-the-middle attacks (MitM) and eavesdropping.

- **DoH (DNS OVER HTTPS)** : Prevents eavesdropping and manipulation of DNS data through MitM attacks.

- **DoH wraps DNS records** : Providing encryption and authentication of the server, changing the connection-less aspect of DNS.

- DoH primarily serves two purposes –

  - preventing interference and maintaining Security.

  - Allowing web applications to access DNS information via existing browser APIs.

Table of Contents

Problem Statement

DoH has already been criticized by many security re-searchers for making **DNS tunnels** harder to detect and mitigate.

- Since the DoH wraps the DNS traffic in HTTPS, the DNS traffic is imperceptible to the network infrastructure between the client (malware) and the DoH server.
- This effectively makes detection methods that rely on examining the DNS packets obsolete for the firewalls.
- Since HTTP/2 is the minimum version of HTTP that DoH standard recommends for using with DoH, Malware can utilize the HTTP/2 connection to send several DoH request, without creating a separate connection (or packet) for each request.
- The same also applies to the responses that DoH server is sending to the malware.
- Malware can hide the frequency of their DNS resolutions, further reducing the number of methods that can detect DNS tunnels
- Our goal is to train a model which can **detect DNS tunneling for DNS over HTTPS(DoH)**

Data Collection And Feature Extraction

Data Collection

We have collected different PCAP file from CIRA-CIC-DoHBrw-2020 which contains HTTPS traffic flows with two levels of distinct labels i.e DoH and Non-DoH.

Browser/Tool	DoH Server	Packets	Flows	Type
Google Chrome	AdGuard	5609K	105141	HTTPS (Non-DoH and Benign DoH)
	Cloudflare	6117K	132552	
	Google DNS	5878K	108680	
	Quad9	10737K	199090	
Mozilla Firefox	AdGuard	4943K	50485	
	Cloudflare	4299K	90260	
	Google DNS	6413K	138422	
	Quad9	4956K	92670	
dns2tcp	AdGuard	1281K	5459	Malicious DoH
	Cloudflare	3694K	6045	
	Google DNS	28711K	17423	
	Quad9	8750K	138588	
DNSCat2	AdGuard	1301K	5369	
	Cloudflare	12346K	9230	
	Google DNS	48069K	11915	
	Quad9	19309K	9108	
Iodine	AdGuard	3938K	11336	
	Cloudflare	5932K	14110	
	Google DNS	73459K	12192	
	Quad9	22668K	8975	

Dataset Details

Data Collection And Feature Extraction

Data Preprocessing And Feature Extraction

- We have read the captured traffic in PCAP format which is created by tools such as tcp dump or Wireshark and extract the features.
- The traffic captured in the dataset in form of PCAP files as input and extracts features for each flow in the input.
- This process is done in **image@cse.iitk.ac.in** and it took roughly 20 days to extract the files.

Parameter	Feature
F1	Number of flow bytes sent
F2	Rate of flow bytes sent
F3	Number of flow bytes received
F4	Rate of flow bytes received
F5-F12	Packet Length (F5: Mean, F6: Median, F7: Mode, F8: Variance, F9: Standard deviation, F10: Coefficient of variation, F11: Skew from median, F12: Skew from mode)
F13-20	Packet Time (F13: Mean, F14: Median, F15: Mode, F16: Variance, F17: Standard deviation, F18: Coefficient of variation, F19: Skew from median, F20: Skew from mode)
F21-F28	Request/response time difference (F21: Mean, F22: Median, F23: Mode, F24: Variance, F25: Standard deviation, F26: Coefficient of variation, F27: Skew from median, F28: Skew from mode)

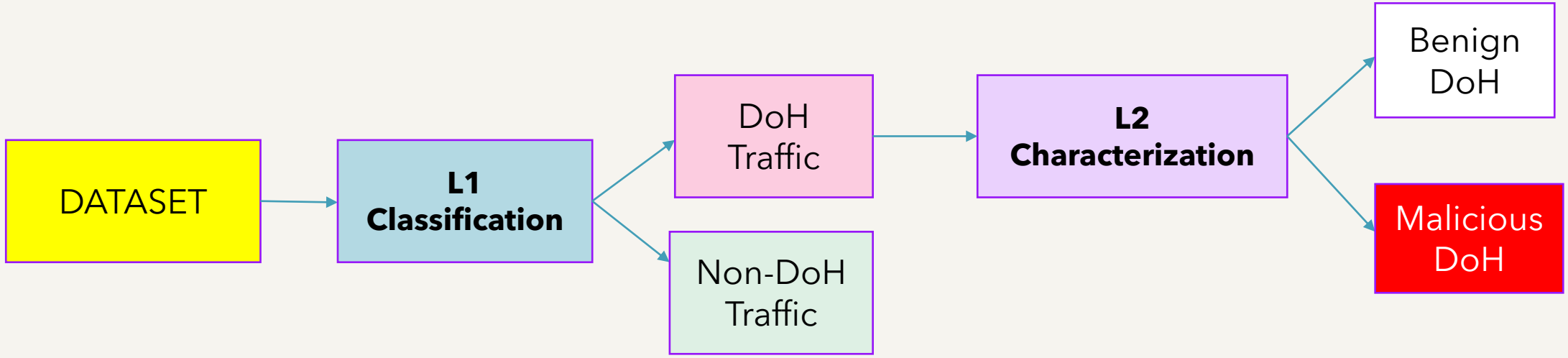
- Results are saved in a CSV file, where each row in the output CSV file would specify a flow in the input traffic.
- Also, we have done the labelling of the outputs CSV into Benign or Malicious based on the label of the PCAP files.
- Then we have selected a subset of features based on our intuition for our training.
- We have splitted our dataset into two parts training and testing in ratio 70:30.

Implementation

We have developed a **two-layer classification** approach.

In the first layer, we have developed a classification model which will divide our training data into **DoH** and **Non-DoH** packets.

In the second layer, we characterized the DoH packets detected in layer 1 into **Benign-DoH** and **Malicious-DoH**.



Level 1- Classification

At layer 1 we will classify the packets in to DoH and Non-DoH using these following classifiers

1. **Random Forest (RF):** With parameters gini index, best splitter, min samples split =2, *n-estimators=100*
2. **Decision Tree (DT):** With parameters gini index, best splitter, min samples split=2, max depth=10, random state=10
3. **Gaussian Naive Bayes (NB):** With parameters var smoothing = 1e − 9
4. **CNN Classifier:** With parameters binary cross entropy loss, Adam optimizer and Activation functions are Sigmoid activation function and Rectifier linear unit, accuracy metric and kernel size=3
5. **DNN Classifier:** With parameters binary cross entropy loss, Adam optimizer and Activation functions are Sigmoid activation function and Rectifier linear unit, accuracy metric and kernel size=3

Model	Precision	Recall	Accuracy
Random Forest	0.993	0.993	0.993
Decision Tree	0.993	0.993	0.993
Naïve Bayes	0.84	0.834	0.833
DNN	0.97	0.97	0.97
CNN	0.98	0.98	0.98

TESTING RESULT

Level 2- Characterization

After layer 1 classification, we have performed characterization of DoH packets. Here we will classify the DoH packets detected in level 1 them into Benign DoH and Malicious DoH.

1. **Random Forest (RF):** With parameters gini index, best splitter, min samples split =2, *n-estimators=100*
2. **Decision Tree (DT):** With parameters gini index, best splitter, min samples split=2, max depth=10, random state=10
3. **Gaussian Naive Bayes (NB):** With parameters var smoothing = 1e − 9
4. **CNN Classifier:** With parameters binary cross entropy loss, Adam optimizer and Activation functions are Sigmoid activation function and Rectifier linear unit, accuracy metric and kernel size=3
5. **DNN Classifier:** With parameters binary cross entropy loss, Adam optimizer and Activation functions are Sigmoid activation function and Rectifier linear unit, accuracy metric and kernel size=3

Model	Precision	Recall	Accuracy
Random Forest	0.999	0.999	0.999
Decision Tree	0.999	0.999	0.999
Naïve Bayes	0.836	0.833	0.832
DNN	0.98	0.98	0.98
CNN	0.99	0.99	0.99

TESTING RESULT

Limitations and improvements

Limitations

1. Some tools create the dataset that we have used to train our model, so it might not capture the same traffic flow generated by DoH tunneling traffic
2. For testing our model, we don’t have any live data
3. There might be some other areas of DoH protocol that still inherit DNS vulnerabilities.
4. We have done static analysis of our model

Improvements

1. Real-time dataset can be used to train our model for finding better non-linearity
2. We can perform dynamic analysis of our model once we have more data

Conclusion

- **DNS-over-HTTPS (DoH)** makes DNS more private by encrypting the DNS packets through the HTTPS protocol.
- DoH protocol makes **detection** of DNS tunnels a concern
- **Pre-processing a dataset** (containing non-DoH HTTPS traffic, benign-DoH traffic and malicious-DoH traffic). The Statistical features to detect DoH connections and malicious-DoH activity (DoH tunneling) in a two-layered binary classification approach.
- **First layer :** Distinguishes DoH traffics from Non-DoH traffics. **Second layer :** Determines malicious characterized and harmless DoH flows.
- **Simple Machine learning models**(Random Forest Algorithm) performs better result compared to Deep Neural Network model with better accuracy.
- **Malicious-DoH traffic can be accurately detected** by using the proposed two-layer binary classification architecture with a precision of more than 99% by using Decision Tree (DT) and Random Forest (RF) ML algo.

Contributions

Abhishek – Data collection, preprocessing, testing, final model

Maj Ashish – L1 classification, L2 characterization

AshItosh – L1 classification, L2 characterization

Binaya – L1 classification, L2 characterization

Shivam Data collection and preprocessing, testing

Kiran – Data collection and preprocessing, L2 characterization, final model

CODE DEMO