2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Encrypt DNS Traffic: Automated Feature Learning Method for Detecting DNS Tunnels

Shuai Ding[1,2], Daoqing Zhang[1], Jingguo Ge[1,2], Xiaowei Yuan [1,2], Xinhui Du[1,2]
[1]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing, 100049, China*
E-mail: {dingshuai, zhangdaoqing, gejingguo, yuanxiaowei, duxinhui}@iie.ac.cn

***Abstract*—In recent years, attacks on the DNS continue to proliferate due to the lack of security mechanisms. DNS over HTTPS (DoH) is a standard developed for encrypting plaintext DNS to protect user privacy, but attackers may use this protocol to bypass enterprise firewalls and exfiltrate private data through the establishment of DNS tunnels. Traditional DNS tunnel detection methods based on packet inspection are no longer applicable because of DNS encryption. Although popular machine learning methods are widely used, it require expert knowledge to extract statistical feature sets which are complicated. In order to solve the problem of detecting encrypted DNS tunnels, we propose an end-to-end anomaly detection model based on a variational autoencoder which incorporates the attention mechanism. By modeling the raw flow sequence data at the flow-level, we use bidirectional GRU-based network to automatically learn the feature representations and detect anomalies via reconstruction error. We conducted experiments on the public dataset, which contains raw normal DoH traffic and abnormal DNS tunnel traffic. The results show that our model achieves excellent performance, and has the ability to identify unknown DNS tunnels.**

***Index Terms*—Encrypted DNS tunnels, DNS over HTTPS, Deep learning, Anomaly detection**

I. Introduction

Domain Name System (DNS) is the internet infrastructure and one of the most important network protocols, it translates the human-readable domain names into machine-usable IP addresses. Like many other Internet protocols, the DNS system was not designed with security in mind and contains several design limitations. These limitations have made it is vulnerable to many attacks, such as DNS cache poisoning, DNS hijacking, NXDomain attack, and DNS tunnel [1] [2]. It is worth noting that DNS protocol is not designed for data transmission, so people usually ignore that it may be used as a covert communication tunnel for data exfiltration and other malicious activities [3]. Many businesses have faced the DNS exfiltration and been affected by DNS tunnel. Some types of malware such as Feederbot [4] and Morto worm [5] have occurred. Cybercriminals rely on DNS protocol to establish command and control (C2) channel and then steal sensitive information of the compromised host. By default, DNS queries and responses are sent in plaintext (via UDP), which means they can be read by ISPs, or anybody able to monitor traffic. In recent years, DNS over HTTPS (DoH) [6] is an IETF-approved standard of encrypted DNS traffic to improve the privacy and prevent malicious parties to interpret the data between users and DNS resolver servers. Currently, Cloudflare, Google, Quad9 and other DNS service providers have implemented public DoH resolver, and popular browsers such as Mozilla Firefox and Google Chrome already supported DoH. DoH protocol provides privacy protection for users, but enterprises and organizations will face new obstacles for monitoring network traffic on their networks as attackers attempt to use encrypted DNS [7]. In 2019, 360 netlab has discovered a backdoor program called Godlua Backdoor [8], which uses DoH to obtain domain name resolution to ensure the secure communication between Bot and C2 Server.

The main motivation of our work is to detect malicious traffic of DoH, like DNS tunnels. In this paper, we assume the scenario where the DoH protocol is used based on the DNS communication between the compromised host and the public recursive resolver. Since the entire DNS packet is encrypted, deep packet inspection is not possible and domain name detection is also not feasible. And traffic statistical analysis based on the machine learning (ML) method needs manually extract predefined features with expert experience, which is complicated and time-consuming. For the ML model training, predefined statistical features are fed into a specific classification model which needs to be carefully selected to generate convincing results. In this paper, we propose an end-to-end deep learning method that combines the feature engineering and model training into a unified model. The deep neural model can automatically learn the feature representations from the raw flow-level traffic and detects DNS tunnels. We consider that normal DoH traffic reflect different network behavior patterns with DNS tunnel traffic in some original features of the raw traffic, such as packet length, packet direction and inter-arrival time. These fixed features will not be changed or invisible due to encryption. We use flow sequence to describe network traffic, because the essence of network traffic is a series of packets transmitted over time. The proposed deep neural model is based on Variational Autoencoder (VAE) [9] which is widely used in the field of anomaly detection. Since the recurrent neural networks (RNNs) perform well in processing sequential data tasks. The bidirectional Gate Recurrent Unit (Bi-GRU) is used as encoder and decoder in VAE. In addition, in order to make full use of the information carried by the entire input sequence, attention mechanism is utilized to generate latent variables of the VAE model. The main contributions of this paper can be summarized as follows:

978-0-7381-2646-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00056

- We propose an end-to-end model which uses a variational autoencoder to automatically learn the feature representations from the raw traffic and detects encrypted DNS tunnels based on the reconstruction of sequence.
- The proposed Bi-GRU based VAE model fusion attention mechanism can learn long sequential and structural information which is incapable of the traditional machine learning methods.
- The model learns the patterns of normal traffic based on the semi-supervised training method, which achieves the purpose of detecting unknown types of encrypted DNS tunnel traffic.

The rest of this paper is organized as follows. Section II summarizes the related work about detecting DNS tunnels and encrypted traffic identification. The DNS tunnel mechanisms and feature representations are described in Section III, and the detailed system architecture is proposed in Section IV. In section V, the related experiments evaluates our method. Finally, we conclude this paper in Section VI.

II. Related Work

In this section, we categorize the previous work on the detection of DNS tunnels and encrypted traffic identification. Due to the popularity of machine learning and deep learning methods, we pay more attention to these methods.

A. DNS Tunnel Detection

DNS tunnel detection methods can be divided into two types: domain name analysis and DNS traffic statistical analysis. Romaña et al. [10] proposed detection approaches based on the entropy of requested hostnames. Qi et al. [11] proposed a scoring mechanism that utilized bigram character frequency to distinguish normal domain names and random ones. Nadler et al. [12] presented a method for detecting data leaks on DNS. They extracted six features within a sliding window on each domain name, and used isolation forest algorithm for detecting anomaly by calculating an anomaly score. Deep learning methods are also used to detect DNS tunnels based on DNS subdomain names. Jiang et al. [13] first used character-level convolutional neural network (CNN) to detect malicious URLs in the field of security. Some researchers who also used character-level and word-level CNN [14] to detect malicious URLs, and Chen et al. [15] adopted long short-term memory (LSTM) model to detect tunnel-based malicious URLs. All these deep learning-based methods achieve high performance, but they need to extract domain name string features that will fail on encrypted DNS traffic.

For traffic statistical analysis, traditional machine learning methods are widely used for detecting DNS tunnels. DNS requests have distinguishing features, such as number of distinct DNS requests per hour, number of MX record queries per hour, etc [16]. Liu et al. [17] used four kinds of behavior features including time-interval features, request packet size features, record type features and subdomain entropy features to detect DNS tunnels. They evaluated the performance with SVM, Decision Tree and Logistical Regression, and experiments detection accuracy can reach 99.96%. In addition, Liu et al. [3] also proposed a packet-based deep learning method. They represented DNS packets with bytes. The embedding technology converted bytes into vectors and finally used CNN to detect the DNS tunnels. However, the DNS packet encryption can cause that the packet byte-level features are obfuscated. So it is difficult to extract valid features.

B. Encrypted Traffic Identification

When the DNS tunnel communication is encrypted, that is, when the DoH protocol is used, the problem turns encrypted traffic identification. Houser et al. [18] developed a DNS over TLS (DoT) based website fingerprinting method to analyze DoT traffic generated by visiting websites. They extracted statistical features from the packet sequence and use them for classification. Siby et al. [19] also introduced a novel set of features, consisting of n-grams of TLS record lengths in a trace. They selected Random Forests (RF) algorithm for website fingerprinting tasks. Vekshin et al. [20] used five ML classification models to classify the DoH traffic by selecting 19 traffic features, but they only classified DoH client traffic such as Google Chrome and Firefox. Popular deep learning methods are already used in the field of encrypted traffic identification, such as CNN [21] and LSTM [22]. The first n bytes of a flow as input and the model automatically learn features from raw binary data. In addition, some complex models have been proposed. For example, FS-Net [23] was an end-to-end classification model by using flow sequence, MIMETIC [24] used multimodal deep learning for classifying mobile encrypted traffic.

III. Encrypted DNS Tunnels Analysis

In this section, we describe the use of DoH protocol and the detection of encrypted DNS tunnels. To protect user privacy, many traffic protection methods, like traffic obfuscation and encryption technologies, are introduced. DoH protocol is approved as standard RFC8484 [6] by the Internet Engineering Task Force (IETF). The RFC document regulations that a DoH client encodes a single DNS query (DNS wire format [25]) into a HTTPS request using either the HTTP GET or POST method. Since every technology has two sides [26], DoH protocol may theoretically be abused by attackers in a variety of ways. DNS tunnel for data leakage is a kind of malicious traffic which becomes more difficult to detect under the shelter of the encrypted DoH protocol. In this work, we label encrypted DNS tunnel traffic as abnormal traffic.

The communication principle of DNS tunnel is divided into IP direct tunnel and relay tunnel. The IP direct tunnel is easy to be detected and blocked through the IP blocklist mechanism. The main discussion in this paper is the relay tunnel that uses DoH protocol communication. The encrypted DNS Tunnel communication architecture is shown in the Fig. 1. There is an internal host controlled by the attackers in the local network. The compromised host makes a request for a domain name lookup and encapsulates the transmitted data in the DNS packet. The requested domain name will

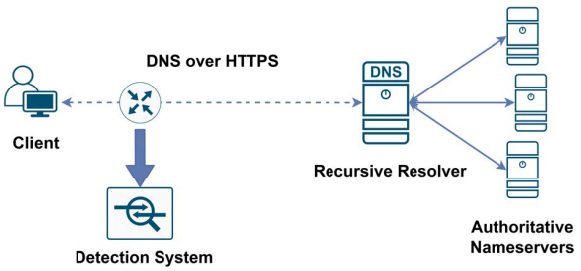


Fig. 1. DNS communication using DoH protocol between a client and recursive resolver, and detect system deployed at the gateway

not hit the local cache, and the DNS request will be sent to the public recursive DNS server by using DoH client. The public recursive DNS server then queries the authoritative server (under the control of attackers), and the latter sends a response (control commands) to the compromised host, which provides the attacker with a covert channel communication between compromised hosts.

In order to reduce the risk of network attacks as much as possible in many internal networks, the security policies are generally configured where the DoH client can only communicate with trusted public DoH resolvers. For our detection system, the detection of abnormal traffic is divided into two steps. Firstly, DoH flows are filtered out from HTTPS traffic based on the IP address of the public resolver, then the DoH flows are processed and sent to the deep model for malicious traffic detection. It can be deployed at the network exit gateway of the enterprise to detect DoH traffic of internal hosts.

IV. Proposed Method

A. Overview

In this subsection, we describe the overall structure of our method on the detection of encrypted DNS tunnels. As shown in Fig. 2, the framework consists of two parts: The solid lines denote offline training model, and the dash lines indicate the procedure of online detection. When the traffic is captured at the detection location, it needs to be parsed into flow sequences. The Data Preprocessing module processes the flow sequences into a standard data format and inputs them to the neural network for offline training, and the appropriate threshold is selected. In the online detection mode, after the flow sequences are preprocessed, they are input into the trained model, which produces the final result according to the threshold.

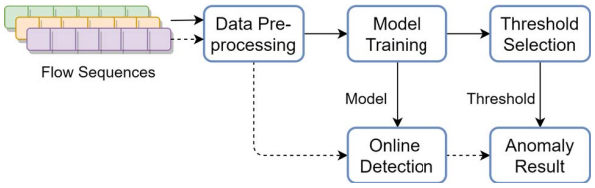


Fig. 2. Overall structure of our method

B. Problem Definition

The essence of the DoH traffic can be considered as a series of packets transmitted over time. Due to the encryption performed by DoH, only the Packet Length (PL), Packet Direction (PD) and Inter-Arrival Time (IAT) are useful to us because they can imply some information about the traffic behavior patterns. So the combination of these features will inevitably make the traffic more detailed. In several previous works, these features have already used these features for trafic classification [23] [27]. Therefore, to improve the ability of monitoring network traffic, we use the packet sequence features formed by a three-dimensional observation element (PL, PD and IAT). Assume that there are N samples and given the p-th sample sequence for one bidirectional flow can be represented as follow:

$$X_p = [(l_1, d_1, t_1), (l_2, d_2, t_2), ..., (l_{n_p}, d_{n_p}, t_{n_p})] \quad (1)$$

where n is the length of the p-th sequence, l_i is the packet length and d_i is the direction at time step i, and d_i is the IAT between the ith packet and $(i+1)th$ packet. We aim to train a deep neural model to minimize the reconstruction error $||X_p - g_\theta(f_\phi(X_p))||$, where f_ϕ is encoder and g_θ is decoder. ϕ and θ are their parameters respectively.

C. Variational Bi-GRU model with attention mechanism

We design a variational Bi-GRU autoencoder model with attention to process the sequence data. Our model architecture is shown in the Fig. 3. The encoder and decoder of our VAE model each consist of a Bi-GRU neural network. The input sequence is vectorized by the embedding layer, and fed to the encoder. The hidden vectors which are output by the encoder, then pass through the attention layer. The attention layer will generate the distribution of latent variables. The latent variables is sampled from the distribution. Finally, the latent variables are decoded by a decoder which will restructure the input data. After calculating the loss, network will backpropagate the gradient and update the network parameters.

1) Embedding layer: Inspired by word embedding in natural language processing (NLP), we embed each three-dimensional element in the flow sequence to a fixed length vector. The embedding layer can be viewed as a matrix $E \in \mathbb{R}^{K \times d}$, where $K = 3$ is the size of one element. We set the dimension of embedding vector $d = 128$. The sequence element (l_i, d_i, t_i) is converted into a d-dimensional vector e_i. Finally, the embedding sequence $E_p = [e_1, e_2, ..., e_{n_p}]$ is input to the Bi-GRU network. The reason why we use embedding layer is that, first of all, it can perform feature representations after the fusion of heterogeneous data. Non-numeric values such as packet direction can be fused and expressed, which is convenient for calculation. The embedding vector representations expands the data dimension and enriches the information saved in each element of one sequence. This is more conducive to the encoder to generate effective latent variables.

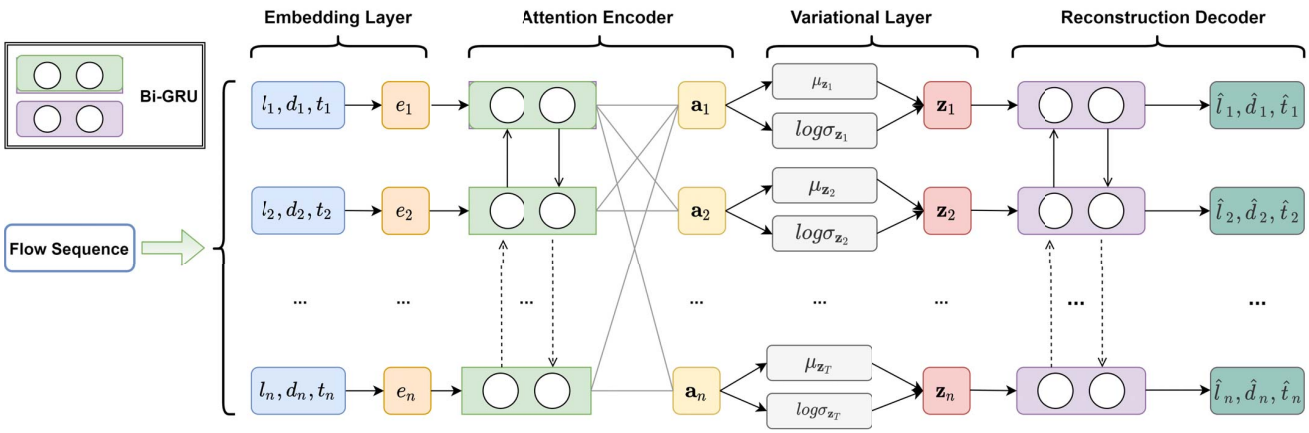


Fig. 3. Variational Bi-GRU Autoencoder with Attention.

2) Attention encoder: Recurrent neural networks (RNNs) have achieved great success in many fields. The GRU is a variant of an RNN that can overcome the issues of vanishing gradients and gradient explosions in traditional RNNs. More importantly, GRUs can learn both long-term and short-term dependencies, and compared to LSTM with fewer parameters, the training speed is faster. Since flow sequence is also a time-related sequence type of data, it is very suitable for processing with GRUs. Simultaneously, in NLP, attention mechanism is proposed to consider different positions of a single sequence when it is employed to compute a representation of the sequence [28]. Attention mechanisms enable an extra performance improvement in some NLP tasks. The intuition behind our choice of is that bidirectional GRU with attention will capture patterns in request-response pairs, as well as the local order of the packet sequence.

The encoder of Bi-GRU layer consists of a forward GRU and a backward GRU. The forward GRU takes the sequences E as sequential inputs, and at time step t, the t-th embedding vector $\mathbf{e}_t$ of E is sent to the input layer of the forward GRU. The hidden layer combines $\mathbf{e}_t$ and the hidden state $\overrightarrow{h_{t-1}}$ at time step $t-1$ to perform a series of linear operations and $tanh$ activation operations. There are two gates in each GRU, namely an update gate z_t and reset gate r_t. The gate z_t controls how much previous information is retained in the current state and the gate r_t determines how strongly irrelevant sequences are ignored. The forward propagation formulas are computed as follows:

$$z_t = \sigma W_z[\overrightarrow{h_{t-1}}, e_t] \tag{2}$$

$$r_t = \sigma W_r[\overrightarrow{h_{t-1}}, e_t] \tag{3}$$

$$\widetilde{h_t} = tanh(W[r_t \overrightarrow{h_{t-1}}, e_t]) \tag{4}$$

$$\overrightarrow{h_t} = (1 - z_t)\overrightarrow{h_{t-1}} + z_t \widetilde{h_t} \tag{5}$$

The backward GRU takes the sequence in reverse as an input. At time step t, it performs a series of calculations similar to the forward GRU and obtains a hidden state $\overleftarrow{h_t}$, which is spliced with $\overrightarrow{h_t}$ to form the final encoder output h_t, as indicated in Eq. (6). The output of GRU layer is then used as the input for the attention layer.

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \tag{6}$$

The encoder fusion attention will more focus on the hidden state which will be assigned attention weights and generate the distribution of latent variables. The attention layer is utilized to learn a weight α_{ji} for a sequence of hidden state $(h_1, ..., h_n)$ obtained at time step i by the encoder. The context vector $\mathbf{a}_j$ at each time step j, then, computed as a weighted sum of these hidden state h_i:

$$\mathbf{a}_j = \sum_{i=1}^{n} \alpha_{ji} h_i \tag{7}$$

where the weighting factors α_{ji} are calculated as follows

$$\alpha_{ji} = \frac{\exp(\tilde{\alpha}_{ji})}{\sum_{k=1}^{n} \exp(\tilde{\alpha}_{jk})} \tag{8}$$

where $\tilde{\alpha}_{ji}$ is a score function, computed by $\tilde{\alpha}_{ji} = h_j W^t h_i$ and W is a learnable weight matrix. The attention weights need to be normalized by $softmax$ function which ensure that, for each step t, $\sum_{i=1}^{n} a_{ti} = 1$.

3) Variational layer: The VAE [9] is a generative probabilistic model that is approximated by a neural network, forming an autoencoder-like architecture. The encoder neural network $f(x, \phi)$ generate the latent variable z from the input x and decoder neural network $g(z, \theta)$ reconstruct new data by the latent variable z. The random variable $z \sim q_\phi(z|x)$ usually is reparameterized by a deterministic transformation from a standard normal distribution. The variational lowerbound of the marginal likelihood, which can be expressed as:

$$\log p_\theta(x) = D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(z)) + \mathcal{L}_{\text{vae}}(\theta, \phi; x) \tag{9}$$

where $q_\phi(z|x)$ is the approximate posterior and $p_\theta(z)$ is the prior distribution. D_{KL} is the KL divergence of the approximate posterior and the prior. Since the KL divergence

is non-negative, the $\mathcal{L}_{\text{vae}}$ is the variational lowerbound on the marginal likelihood, and can be rewritten as:

$$\mathcal{L}_{\text{vae}}(\theta, \phi; \mathbf{x}) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \tag{10}$$

The first term of Eq. (10) is the reconstruction of x through the posterior distribution $q_\phi(z|x)$ and the likelihood $p_\theta(z)$. The second term of Eq. (10) forces the posterior distribution to be similar to the prior distribution, working as a regularization term.

The VAE is successfully applied in the field of anomaly detection for several advantages. First, latent variables are stochastic variables. In autoencoders, latent variables are defined by deterministic mappings. Since VAE uses the probabilistic encoder for modeling the distribution of the latent variables rather than the latent variable itself, the variability of the latent space can be taken into account from the sampling procedure. This extends the expressive power of the VAE compared to the autoencoder in that even though normal data and anomaly data might share the same mean value, the variability can differ, the anomalous data will have greater variance.

In our model, The encoder network f_ϕ generate the latent variable $\mathbf{z}$ from the input embedding vector E. In details, the outputs of the attention layer are sent into two linear modules to estimate the mean $\mu_\mathbf{z}$ and the log of variance $log\sigma_\mathbf{z}$ of the latent variable. The random latent variable $\mathbf{z} \sim q_\phi(\mathbf{z}|E)$ are reparameterized by sampling from a deterministic transformation as the Eq. (11), where ϵ is from a standard normal distribution. Each latent variable is generated after L times of sampling, where $L = 10$ in our model settings.

$$\mathbf{z} = \sigma_\mathbf{z} * \epsilon + \mu_\mathbf{z} \quad \text{with} \quad \epsilon \sim N(0, 1) \tag{11}$$

4) Reconstruction decoder: The latent variables generated by sampling fed into the decoder which is also a Bi-GRU network. The final outputs of decoder is to reconstruct the sequence $\widehat{X}_p$. Finally, given a sequence X_p the loss function define as:

$$\mathcal{L}(\theta, \phi; X_p) = ||X_p - \widehat{X}_p|| + D_{KL}(q_\phi(\mathbf{z}|X_p)||p_\theta(\mathbf{z})) \tag{12}$$

The first term of Eq. (12) in our model is reconstruction error. The second term is the same as Eq. (10).

D. Anomaly Detection

Generally, deviation-based autoencoder anomaly detection uses a semi-supervised learning method. Our encrypted DNS tunnel detection also uses this method. In the training phase, only data with normal instances is used to train the autoencoder, so that the autoencoder learns the pattern of normal DoH flow sequences. After training, the autoencoder will reconstruct normal data very well. In the inference stage, the appropriate threshold is selected according to the reconstruction error of the training data. The data with high reconstruction error is considered an anomaly, that is, DNS tunnel traffic. Mean Square Error (MSE) is used as the anomaly score, and Algorithm 1 shows the anomaly detection algorithm using reconstruction errors of variational autoencoders.

Algorithm 1 Variational autoencoder anomaly detection

Input: $X = \{x_1, ..., x_n\}$, Normal dataset N, Threshold α
Output: Reconstruction Errors $MSE(X)$

$\phi, \theta \leftarrow$ train a model using the normal dataset N;
for $i = 1$ to n **do**
    $\mu_{z_i}, \sigma_{z_i} = f_\theta(z|x_i)$;
    draw L samples from $z_i \sim N(\mu_{z_i}, \sigma_{z_i})$;
    $\hat{x}_i = g_\phi(z_i)$
end for
$MSE(X) = \frac{1}{n}\sum_{i=1}^{n} ||x_i - \hat{x}_i||$;
if $MSE(X) > \alpha$ **then**
    X is an abnormal DNS tunnel flow;
else
    X is a normal DoH flow;
end if

V. Evaluation

A. Dataset and Data Pre-processing

1) Dataset: To evaluate our detection method, we use CIRA-CIC-DoHBrw-2020 dataset [29] that provide raw pcap file. The dataset contains Non-DoH traffic, normal DoH traffic and malicious DoH traffic. Non-DoH traffic and normal DoH traffic is generated by browsing webs with Mozilla Firefox and Google Chrome. Malicious-DoH traffic is generated by DNS tunnel tools that contain dns2tcp [30], DNSCat2 [31], and Iodine [32]. A DoH proxy client will encrypt every DNS request, send to the public DoH resolvers, and relay DNS response. These public resolvers include AdGuard, Cloudflare, Google DNS, and Quad9. The traffic is captured between the DoH proxy client and the DoH resolver. The malicious tunnel traffic generated by these tools is used to form our abnormal dataset. We also use another dataset [20] to expend our noraml dataset which contains 1,128,904 HTTPS flows, including 33,000 normal DoH flows. Our normal dataset extracts the normal DoH flows in these two datasets. The details of dataset present at Table I.

2) Data Pre-processing: In this paper, we only consider how to detect encryted DNS tunnels on DoH traffic, and do not consider the classification of DoH traffic and Web HTTPS traffic. DoH protocol uses 443 as the destination port, which leads to mixing with other Web HTTPS traffic. But it is easy to filter Web traffic by setting destination port and IP address as filter conditions to obtain the final DoH traffic. For example, the IP address and port of Google's public DNS resolver is 8.8.8.8 and 443 respectively.

TABLE I
Dataset Details

Type	Tools	Flows
Benign DoH	Google	47K
	Firefox	115K
Malicious DoH	dns2tcp	166K
	DNSCat2	30K
	Iodine	45K

In the first step of data preprocessing, we strip out DoH traffic from various HTTPS traffic, Then parse the raw DoH traffic into flows which consist of packets with the same five-tuples (i.e., transport layer protocol, source IP, source port, destination IP, destination port). Finally, we extract flow sequence to form the three-dimensional element (PL, PD, IAT) from flows. The flow sequence is represented as: $[(497, 1, 0.0167), (592, -1, 0.0242), ..., (610, 1, 0.0017)]$, where the second item of element is indicates the direction of the packet (incoming or outgoing). We normalized the input data to enable the model to converge quickly. The minimum frame length of the Ethernet is 64 bytes, and the maximum transmission unit is usually 1500 bytes, so the packet length is normalized in the range of [64, 1500]. Besides, the time data IAT is processed by logarithm. It is worth mentioning that the flow sequence only keep the TLS packets and ignore insignificant packets such as TCP three-way handshake and ACK packets with no payload, but the packets of the TLS handshake are retained. Since the input of the deep neural model is fixed, flow sequence will be truncated if the length exceeds model input length, and conversely filled with zero element.

B. Experimental Settings

1) Comparison Methods: In this paper, we use traditional machine learning methods and deep learning methods as comparison methods. Traditional machine learning methods require manually extract statistical features to train models. The machine learning methods we used are proposed by the paper [29]. It developed and selected 28 statistical features from DoH traffic such as $FlowBytesSent$, $FlowReceivedRate$, $PacketLengthMean$, etc. A detailed list of feature sets can be found in the original paper. In this part of the experiment, Support Vector Machine (SVM), Naive Bayes (NB) and Random Forest (RF) are chosen as comparative baseline ML methods. FS-Net [23] is a popular deep neural network in the field of encrypted traffic classification which utilizes packet length as input, and we also choose it as a comparative DL method. The above methods for comparison all use the training mode of labeled supervised learning. Furthermore, LSTM-AE method and the improved LSTM-VAE [33] are similar to our model, both use the same input and use a semi-supervised training mode, that is, just use normal data to train model.

2) Model Settings: For embedding layer, the embedding vector dimension is set to 128. The dimension of hidden states of each GRU layer is set to 128 in the encoder and decoder. In variational layer, we set $L = 10$ that means take 10 samples on the distribution of each latent variable. Moreover, the dropout rate is 0.1 ratio for Bi-GRU layer to avoid over-fitting, and the Adam optimizer with learning rate 0.001 is used. For all autoencoder-based deep neural models, we randomly select 80% normal data for our training data, and the rest normal data and all abnormal data for testing data. We take 128 as the input sequence length for deep neural model in the comparison experiments. In addition, the 99% quantile of reconstruction error is selected as the threshold in the training results.

TABLE II
COMPARISONS BETWEEN OUR MODEL AND THE OTHER FOUR MODELS

Classifier	Precision	Recall	F1-Score
NB	0.836	0.833	0.832
RF	0.999	0.999	0.999
SVM	0.890	0.885	0.884
FS-Net	0.992	0.995	0.993
ABG-VAE	0.992	0.996	0.994

C. Results and Analysis

1) Comparison experiments: The results of comparison are shown in the Table II. We can see that NB and SVM did not perform well in the experiments. Their precision rates and recall rates are lower than 90%. RF and our model ABG-VAE have a great improvement in performance. Both of them have an accuracy rate of over 99%. The performance of RF is relatively high, whose all metrics have reached 99.9%. However, the RF method requires feature engineering, manually extracting predefined features, and then carefully selecting the features, and also needs to tune the parameters of the model, which is very complicated and time-consuming. The selection of features and parameters has a great impact on the performance of the model. Moreover, the machine learning method based on statistical analysis is to calculate a complete flow within a period of time, and the real-time detection cannot be guaranteed. On the contrary, our model avoids feature engineering under the premise of ensuring performance, and can detect a new incoming flow in real time. Because our model only needs a piece of flow sequence data as input for detection, and the neural networks will automatically learn the representative of features from the raw flow sequences, which is the advantage of the end-to-end learning architecture.

We also conducted comparative experiments on FS-Net which is an end-to-end deep neural network for encrypted traffic classification at the application level. The model only utilizes packet length sequences as its input. We use labeled data to train it in a supervised binary classification. The experimental results are shown in the Table II. Our model has a slight rise in Recall and F1-Score metrics, almost the two model have reached 99% F1-Score. We also analyze the complexity of our model from the number of parameters, training time and testing time. Table III compares the time consumption and parameters between the FS-Net and ABG-VAE in our experimental machine and environment. In addition to the performance advantages, our model has fewer parameters and shorter training and inference time than FS-Net. We can see that compared to FS-Net, our model parameters are reduced by nearly three times, the average

TABLE III
PARAMETERS AND TIME CONSUMPTION OF MODELS

Model	Paremeters	Training time/epoch	Test time
FS-Net	825K	56.58s	4.6s
ABG-VAE	323K	12.24s	1.1s

TABLE IV
COMPARISONS BETWEEN OUR MODEL AND THE OTHER DL MODELS

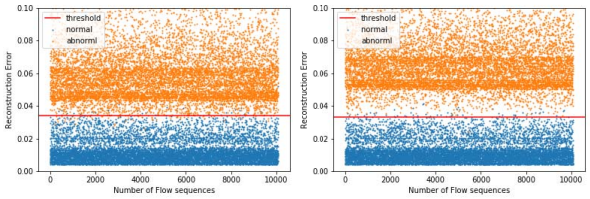
Anomaly	dns2tcp				DNSCat2				Iodine			
Metric	Accuracy	Presion	Recall	F1-Score	Accuracy	Presion	Recall	F1-Score	Accuracy	Presion	Recall	F1-Score
LSTM-AE	0.9361	0.9654	0.9029	0.9331	0.9219	0.9132	0.9323	0.9227	0.9499	0.9302	0.9681	0.9488
LSTM-VAE	0.9666	0.9503	0.9787	0.9643	0.9795	0.9636	0.9955	0.9793	0.9610	0.9581	0.9672	0.9627
ABG-VAE	0.9952	0.9968	0.9939	0.9953	0.9957	0.9968	0.9948	0.9958	0.9879	0.9861	0.9903	0.9882

training time per epoch is reduced by approximately five times, and the test time is reduced by more than four times. Our model has decreased in the number of parameters and training and inference time, making our model have better detection efficiency and real-time performance. Besides, there are not only the three types of DNS tunnel traffic mentioned above in the real world. When encountering unknown abnormal traffic, the performance of the supervised model will decrease. So we performed experiments that only adopted two types of tunnel traffic to train FS-Net, and tested the detection performance of the three tunnel traffic. After many experiments, we found that the precision of the model dropped by 5-10%. Our model only uses normal data for training, and has the ability to effectively detect unknown abnormal traffic.

Similarly, we conducted comparative experiments on two baselines based on autoencoder models. In this set of experiments, we separately tested the detection effects of three types of DNS tunnel traffic. The detailed results is shown in the Table IV. What can be clearly seen is that, our proposed model achieves best accuracy, precision, recall and F1-Score on almost all types of DNS tunnels. Compared to the LSTM-based model, our model adopts the Bi-GRU networks with attention mechanism to model the sequence with the advantage of extracting the whole flow contextual information. The experiment proves the effectiveness of our method for detecting encrypted DNS tunnels.

2) Model analysis: In order to prove the important role of the attention mechanism in our model, we conducted an ablation experiment. We trained two models, one of which removed the attention layer, another kept the original structure. The results, as shown in Fig. 4, indicates a significant difference between the two models. From the reconstruction error scatter plot, it can be seen that the errors of the model with attention are higher than the one without it. What stands out in the figure is that the distribution of abnormal data becomes more concentrated, which facilitates the choice of threshold. Because for anomaly detection problem, it is also very important to determine the appropriate threshold. Therefore, these results suggest that the attention mechanism can automatically learn the features of flow sequences well and is helpful for detecting encrypted DNS tunnels.

Another experiment is about the flow sequences with different input length. Intuitively, the longer the input data sequence is, the more information the model can learn, thereby improving the performance of the model. This assumption is is consistent with the real-world scenario. The normal DNS traffic generated by the user's web browsing is not frequent, so the normal DoH flows are usually short sessions. But for



(a) Without attention. (b) Attention.

Fig. 4. Reconstruction error for flow sequences.

DNS tunnels, the cost of establishing a channel connection is very high. When a tunnel is established, a long connection is maintained for data transmission, which leads to the longer flow sequences. As shown in the Fig. 5, With successive increases in length of the input sequence, the detection performance of the model has a continual growth until the length reaches 128. After that, the model performance has a slight decline. This result may be explained by the fact that most of the normal DoH flow sequences is indeed relatively shorter than the abnormal in the dataset. However, there are some short malicious flows in the dataset, which will cause model performance degradation when the input sequence is longer. The reason is that we paddded zero if the length of the flow sequence is insufficient in data preprocessing phase. This leads to the fact that when the input length is long, the normal flow sequence and some shorter abnormal flow sequences become very similar. When too much padding at the end of the sequence, the neural network cannot learn useful information well.

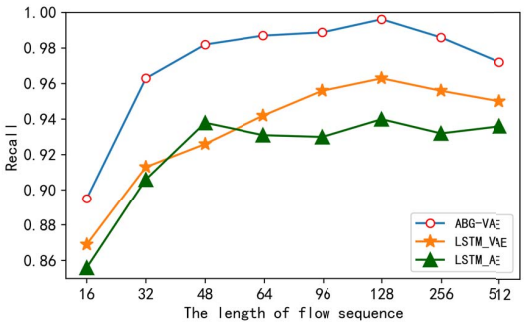


Fig. 5. Effects of different sequence length values on experimental results.

VI. Conclusion

Encrypted DNS traffic has introduced new challenges to network management and security analysis. In this paper, we design an end-to-end deep learning method for encrypted DNS tunnel detection which is based on a variational autoencoder neural network. Extensive experimental results demonstrated that our model achieves better detection performance than the other methods, including traditional machine learning models and baseline deep learning models. Our method does not rely on the analysis of the DNS packet content and the bidirectional GRUs network with attention mechanism can automatically learn the representations of the flow sequence, which means no complicated feature engineering. We also analyzed the influence of the attention mechanism on the model detection results. From the experimental results, it centralizes the distribution of abnormal data, which is helpful for the selection of threshold. Additionally, our research also found that the sequence length has a certain impact on the detection performance.

However, there are still a problems is that various DNS tunnel clients may use obfuscation techniques such as packet padding, making it more difficult to detect malicious traffic. We will study the identification of obfuscated malicious traffic in the future.

References

[1] CloudFlare, "DNS security." [Online], 2019. https://www.cloudflare.com/learning/dns/dns-security/.

[2] Y. Ma, Y. Wu, and J. Ge, *Accountability and Privacy in Network Security*. Springer, 2020.

[3] C. Liu, L. Dai, W. Cui, and T. Lin, "A Byte-level CNN Method to Detect DNS Tunnels," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, 2019.

[4] R. J. Dietrich., "Feederbot—A Bot Using DNS as Carrier for Its CNC." [Online], 2019. https://bit.ly/30ZZ3MN.

[5] C. Mullaney., "Morto worm sets a (dns) record." [Online], 2011. https://symc.ly/2JXKfZS.

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

[7] D. Hjelm, "A new needle and haystack: Detecting DNS over HTTPS usage," *SANS Institute, Information Security Reading Room*, 2019.

[8] Alex.Turing, Genshen Ye, "An Analysis of Godlua Backdoor." 360 Netlab Blog, 2019.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[10] D. A. L. Romaña, "Entropy Based Analysis of DNS Query Traffic in the Campus Network," vol. 6, no. 5, p. 3.

[11] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A Bigram based Real Time DNS Tunnel Detection Approach," *Procedia Computer Science*, vol. 17, pp. 852–860, 2013.

[12] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the DNS protocol," *Computers Security*, vol. 80, no. JAN., pp. 36–53, 2019.

[13] J. Jiang, J. Chen, K. Choo, L. Chao, K. Liu, Y. Min, and Y. Wang, "A Deep Learning Based Online Malicious URL and DNS Detection Scheme," *Springer, Cham*, 2017.

[14] H. Le, Q. Pham, D. Sahoo, and S. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection," 2018.

[15] S. Chen, B. Lang, H. Liu, D. Li, and C. Gao, "DNS covert channel detection method using the LSTM model," *Computers Security*, vol. 104, p. 102095, 2021.

[16] M. Singh, M. Singh, and S. Kaur, "Detecting bot-infected machines using DNS fingerprinting," *Digital Investigation*, vol. 28, no. MAR., pp. 14–33, 2019.

[17] J. Liu, S. Li, Y. Zhang, J. Xiao, and C. Peng, "Detecting DNS Tunnel through Binary-Classification Based on Behavior Features," in *2017 IEEE Trustcom/BigDataSE/ICESS*, 2017.

[18] R. Houser, Z. Li, C. Cotton, and H. Wang, "An investigation on information leakage of DNS over TLS," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pp. 123–137, 2019.

[19] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted dns–¿ privacy? a traffic analysis perspective," *arXiv preprint arXiv:1906.09682*, 2019.

[20] D. Vekshin, K. Hynek, and T. Cejka, "DoH Insight: detecting DNS over HTTPS by machine learning," (Virtual Event Ireland), pp. 1–8, ACM, 2020.

[21] W. Wei, Z. Ming, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017.

[22] X. Liu, J. You, Y. Wu, T. Li, and J. Ge, "Attention-based bidirectional gru networks for efficient https traffic classification," *Information Sciences*, vol. 541, 2020.

[23] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A Flow Sequence Network For Encrypted Traffic Classification," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.

[24] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Computer networks*, vol. 165, no. Dec.24, pp. 106944.1–106944.12, 2019.

[25] "Domain names - implementation and specification." RFC 1035, 1987.

[26] Y. Ma, Y. Wu, J. Li, and J. Ge, "Apcn: A scalable architecture for balancing accountability and privacy in large-scale content-based networks," *Information Sciences*, vol. 527, pp. 511–532, 2020.

[27] Z. Yao, J. Ge, Y. Wu, X. Lin, and Y. Ma, *Encrypted traffic classification based on Gaussian mixture models and Hidden Markov Models*. 2020.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[29] 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, 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)*, 2020.

[30] O. Dembour, "dns2tcp." https://gitlab.com/kalilinux/packages/dns2tcp.

[31] R. Bowes, "dnscat2." https://github.com/iagox86/dnscat2.

[32] B. A. Erik Ekman, "iodine." https://code.kryo.se/iodine.

[33] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder," *IEEE Robotics and Automation Letters*, vol. PP, no. 99, 2017.