A Byte-level CNN Method to Detect DNS Tunnels

Chang Liu, Liang Dai, Wenjing Cui and Tao Lin
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
{liuchang2,dailiang,cuiwenjing,lintao}@iie.ac.cn

***Abstract*—DNS tunnel is a technology used to bypass firewalls for data exfiltration. It takes advantage of the characteristic of firewalls that always allow DNS traffic pass. Most of the studies have realized the detection of DNS tunnels from the perspective of traffic and payload. In recent years, due to the popularity of machine learning and data mining methods, researchers began trying to use them to detect DNS tunnels. However, these detection methods require a lot of professional knowledge to extract and construct feature set, and the detection results are often unsatisfactory. In order to solve the problem of manual feature extraction limitation and improve the detection accuracy of DNS tunnels, we propose a deep learning method, called Byte-level CNN, to detect the DNS tunnels in this paper. Furthermore, we propose to represent DNS packets with bytes for the first time, as the converted computable data to the CNN model. Compared with traditional machine learning methods, our byte-level CNN method can learn full information in the whole DNS packets, especially the sequential and structural information, besides common statistical information, and therefore achieves good performance results. The simulation showed that the detection accuracy and recall rate are significantly improved, and the complex manual feature extraction process is avoided as well.**

***Index Terms*—DNS tunnels detection, byte-level CNN, data exfiltration, DNS packet**

I. Introduction

DNS is a very important protocol in our network world. It maps a long string of IP address that are not suitable for remembering to a readable domain name consisting of characters [6]. On the one hand, any communication from local host to Internet depends on DNS service, except static IP communication. For this reason, restricting DNS communication may lead to the failure of legal remote service connection. Therefore, it is necessary to be cautious to block DNS traffic. On the other hand, because DNS protocol is not designed for data transmission, people usually ignore that it may be used as a covert communication tunnel for data exfiltration and other malicious activities. From the attacker's perspective, these two points make DNS a good candidate for being a tunnel for data leakage [7]. Up to now, there are many alternative DNS tunneling tools used to build DNS tunnels to bypass paid Wi-Fi for free access to websites. In addition, attackers can also carry out some malicious activities through DNS tunneling tools, such as command and control channel between compromised host and C&C server, file transmission and so on. In addition to DNS tunneling tools, there are some types of malware such as Feederbot [8] and Porto [9] which can perform malicious activities.

978-1-7281-1025-7/19/$31.00 ©2019 IEEE

Recently, the problem of DNS tunnels has become more serious. More and more security experts and companies have begun to make research on the detection of DNS tunnels, which generally can be divided into two categories: detection based on load analysis and detection based on traffic analysis [8]. Load analysis mainly extracts features such as domain name length, character entropy and total byte number of the load of a DNS packet, while traffic analysis mainly extracts features such as DNS traffic per domain name, number of subdomain names per domain name and number of NXDomain responses for a specific time. For load analysis, long domain name, large character entropy and large total byte number usually indicate the existence of DNS tunnels. And for traffic analysis, DNS tunnels often generate more DNS traffic per domain, more subdomain names per domain name and more NXDomain responses than normal DNS service. However, these methods are based on rules which are predefined. The DNS tunnels can only be determined when these predefined rules are satisfied, which come from a very time-consuming and complex generation process.

With the popularity of machine learning, researchers of cyber security try to use machine learning methods to detect DNS tunnels. Machine learning methods can replace rule-based methods by automatically generating rules based on training data. The key is feature extraction and model building. Almusawi et al. [12] proposed using multi-label SVM to detect HTTP-DNS tunnels, HTTPS-DNS tunnels, FTP-DNS tunnels and POP3-DNS tunnels, while Homem et al. [14] compares the performance on detecting these four type tunnels of four methods, which are decision tree, k-neighborhood, SVM and neural network model respectively. Nadler et al. [13] proposed using an isolated forest anomaly detection model to detect high throughput DNS tunnel and low throughput malware. However, in the process of building models with manual feature extraction, feature engineering is a very important stage. The quality of the constructed feature set from original DNS packets or DNS traffic directly determines the performance of model. Unfortunately, the extraction of manual feature set is often time-consuming and inadequate because of the lack of knowledge in related fields.

In order to solve the above problem, we choose CNN (short for convolutional neural network), a deep learning method that can realize automatic feature extraction, to construct

the detection model of DNS tunnel. This method can learn all field information, especially the sequential and structural information contained in the original data. Besides, CNN has better detection performance than the traditional machine learning methods with manual feature extraction process, such as higher accuracy and lower false positive rate. The key problem of using deep learning method to detect DNS tunnels is to get a suitable representation of the raw data. There has been already some research on using a character-level CNN on text classification and detection of malicious URLs [20][21][22]. They represent the raw data by converting characters to computational vectors. But if we use the same method to represent the DNS data, we can only use the domain name or subdomain name to detect because of their textual nature. In fact, other information in DNS packets can also be used as indicators of DNS tunnels. For example, in terms of the body of DNS packets, query types like TXT, SRV and NULL rarely appear in normal DNS traffic, but often appear in the traffic generated by DNS tunnels. Moreover, the answers field of response packets from DNS tunnels is also encoded and encrypted, which show more random than that of response packets from normal DNS traffic. In terms of the header of DNS packets, DNS response packets from normal traffic usually contain multiple resource records, and the number of answered resource records, authorized resource records and additional resource records is often not unique. While the header of DNS packet which is generated by DNS tunnels usually contains unique answered resource record, no authorized resource records and no additional resource records.

In order to fully utilize these information, we propose using bytes to represent the application layer data of DNS packets. More specifically, we firstly use One-Hot method to encode these bytes and then use an embedding layer in the deep neural network to learn embedding vectors to reduce the dimension of them, so as to solve the problem of matrix sparsity. Our main contributions are as follows.

- The deep learning method CNN is used to detect DNS tunnels for the first time, which can achieve considerably good detection effect at the basis of automatic feature extraction.
- The DNS packets are represent with bytes as the computable data for the CNN model for the first time. This method can make full use of the overall information inside DNS packets.
- The proposed method can also learn sequential and structural information in one DNS packet which is incapable of the traditional machine learning methods.
- We compare the performance of traditional machine learning methods such as SVM, logistic regression, neural network with that of our byte-level CNN method based on the load of DNS packets by simulation experiment.

The rest of the paper is organized as follows. We describe about DNS tunnels detection and introduce the formulation of this problem in Section II. In Section III, we introduce our byte-level CNN method. In Section IV, we evaluate our byte-level CNN method and compare it with traditional machine learning methods. We present the related work in Section V and conclude this paper in Section VI.

II. DNS Tunnels Detection

A. Data Exfiltration Based on DNS Protocol

Due to the rules of the firewall, the host located in the private network may not be able to access the public network directly, while the DNS service is an exception. Any communication from local host to the Internet, except static IP communication, depends on DNS service. Restricting DNS traffic may lead to a failure of connection to legal remote service , so the firewall often allow DNS traffic to pass. Hence, the attackers can use DNS protocol for data exfiltration, such as bypassing paid Wi-Fi to get free access to the Internet, transferring commands between C&C server and compromised host of botnet, stealing sensitive information of companies and so on.

Some DNS tunneling tools or malware can be used to build DNS tunnels, so as to achieve the purpose of using DNS protocol to communicate. Take an example of bypassing paid Wi-Fi to illustrate the DNS tunnel tools. Firstly, the DNS tunnel tool is installed and started on both sides of communication (internal host and external server). During the process of starting the tunnel, both sides of communication agree on the encoding method of data (such as Base32, Base64, etc.). Secondly, the client tunnel process of the internal host should encode the resource request according to the pre-agreed encoding method and then HTTP contents are sent to the public network server as the data of the sub-domain name part of the query name in the DNS request packet. Thirdly, after receiving the DNS request, the DNS tunnel process of the public network server decodes the sub-domain name using the pre-agreed encoding method, restores it to a complete network resource request such as HTTP request, and requests network resources through the proxy. Finally, after acquiring the resource, which is encoded by the agreed encoding method and returned to the internal host in the way of DNS response (the encoding resource is in the Answers field of DNS response), the internal host decodes the DNS response in the same way as it requires the HTTP content. So far, a communication process using DNS between the internal host and the public network server has been completed.

B. Issues of Existing Detection Methods

Traditional DNS tunnels detection methods using machine learning can only extract limited features, such as packet length, domain name entropy, the volume of unique queries, the number of subdomain names per domain name, and query types. In fact, besides domain name entropy, the domain name structure of DNS tunnel data is obviously different from that of normal DNS service data. However, traditional methods do not use this kind of structure information.

Character-level CNN method to detect malicious URL can be used to detect DNS tunnels based on subdomain name or domain name to detect. This method can only make use of the information of domain name in the packet load, while

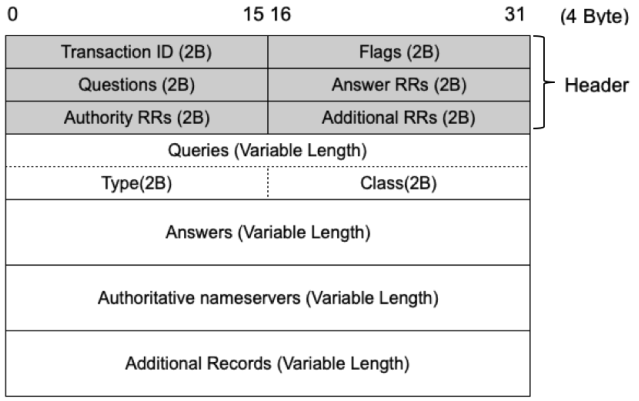

Fig. 1. The structure of DNS packet

can not make use of information of other fields and their structure and sequence information in a DNS packet. For example, the *Queries* field in a DNS packet includes query *types* and *class*, and a DNS response packet includes *Answer RRs*, *Authority RRs*, *Additional RRs*, and *Answers* field, etc. In the actual DNS tunnel simulation, we find that DNS tunnel data often includes only one answer record, and there are no authorization records and additional records, while the normal DNS service produces include more types and numbers of resource records. These information are not utilized by the character-level CNN method based on domain name or subdomain name. The structure of DNS packet is as Fig. 1.

In order to utilize the information that cannot be utilized by the above mentioned methods, we propose a byte-level CNN detection method based on the whole DNS packet. In text data fields such as *Queries* and *Answers*, the model effect of byte-level CNN almost equivalent to that of character-level CNN because they use the same field information of no matter the byte or the character. In addition, the length of the fields in the header of DNS packets are multiples of the byte length, which provides an easier and powerful support for this method. the reason is that one byte or two bytes combination can represent a state of a field of DNS packet, for instance the value 0003 (hexadecimal) of Answer RRs indicates that the DNS response packet has three resource records.

C. Problem Setting

The problem in our paper is mainly to detect DNS tunnels at a basis of DNS packets, i.e. to judge whether a DNS packet comes from DNS tunnels. To achieve this goal, we formulate the problem as a binary classification task. A set consisting of N DNS packets can be represented as $\{(\mathbf{d_1}, y_1), \ldots, (\mathbf{d_N}, y_N)\}$, in which $\mathbf{d_n}, n = 1, 2, \ldots, N$ represents a DNS packet and $y_n \in \{0, 1\}$ marks the DNS packet, 1 indicating that the packet is from a DNS tunnel, while 0 indicating that the packet is generated by a normal DNS service. In the process of formulating DNS tunnel detection problem, we should find an appropriate feature representation of input data firstly, i.e. $\mathbf{d_n} \rightarrow \mathbf{x_n}, and\ \mathbf{x_n} \in \mathbb{R}^L$ is an L-dimensional feature vector representing a DNS packet. Next we need to learn a predictive function $f : \mathbb{R}^L \rightarrow \mathbb{R}$, which predicts the class assignment of a DNS packet instance. The prediction of the function can be expressed as

$$\widehat{y_t} = sign(f(\mathbf{x_t})) \tag{1}$$

, the goal of which is to learn a function that can minimize the total number of classification errors $\sum_{t=1}^{T} \mathbb{I}_{\widehat{y_t} \neq y_t}$ on the whole data set. This is usually realized by minimizing a loss function, and many types of loss functions can be used. For our method, the function is a convolutional neural network, which is CNN for short.

III. Byte-level CNN Method

We use CNN to detect DNS packets. CNN has achieved great success in the field of image processing and computer vision, and has achieved remarkable results in the field of natural language processing in recent years. It can learn useful structural and sequential information from text represented by character vector, word vector or original data. For DNS packets, before building the CNN model, we have to convert the original incomputable data to the computable data. This step we call feature representation, after which CNN model is able to be realized. We describe about our byte-level CNN method in detail as follows.

A. Feature Representation

As shown in the Fig. 1, a DNS packet consists of a sequence of bytes and each byte has 256 values ($0 \sim 255$). Each DNS packet has different length due to different load. In order to facilitate CNN training and detection, we convert each DNS packet into a vector represented by byte of the same length, and the vector length is L. If the DNS packet is longer than L, it is directly truncated to L. If it is less than L, it is filled to L length with $\langle pad \rangle$ byte, the value of which is set to 256. Since each DNS packet can be presented with these 257 values of byte, we propose a byte-based feature representation method. We use One-Hot method to encode each byte into a 257-dimesional vector. For the byte vector whose value is i, we set the value of its i^{th} dimension to 1, while values of other dimensions to 0. As you can see in Fig. 2, the first byte with value 187 of the DNS packet is encoded to a 257-dimensional byte vector using one-hot method. Its 187^{th} dimension is set to 1 and other dimensions are set to 0.

However, One-Hot encoding method has the problem of matrix sparsity. To solve this problem, we introduce an embedding layer to realize dimensionality reduction of byte vectors before constructing CNN network. From a mathematical point of view, our goal is to obtain the matrix representation of DNS packets that is $\mathbf{d} \rightarrow \mathbf{x} \in \mathbb{R}^{L \times k}$, in which $\mathbf{d}$ is the original DNS packet and $\mathbf{x}$ is its matrix representation. The output DNS packet matrix $\mathbf{x}$ consists of k-dimensional byte vectors. L is the length of DNS packet matrix, i.e. the total number of bytes in a packet. L and k are all hyper parameters of the model. K-dimensional byte vector is an embedding vector of byte that makes up the DNS packet, which is extracted

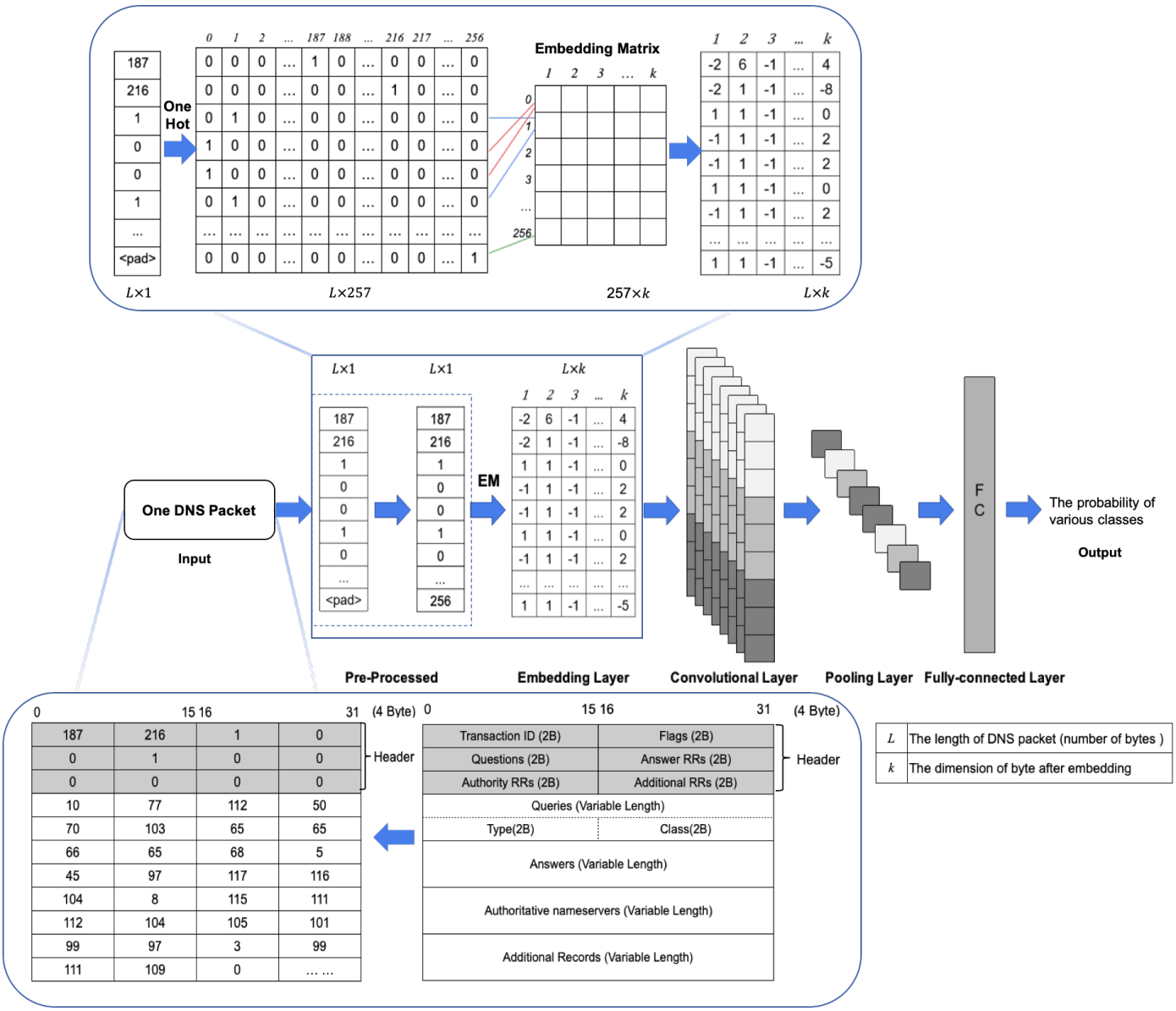

Fig. 2. The process of classification for a DNS packet

by an embedding matrix, $EM \in \mathbb{R}^{257\times k}$. This embedding matrix is randomly initialized and learned through the training process of the model. A DNS packet matrix can be expressed as $\mathbf{x_{1:L}} = x_1 \parallel x_2 \parallel \cdots \parallel x_L$, in which $\parallel$ is a byte connector and $x_1, x_2, \ldots, x_L$ are byte vectors. The length of packet byte sequence L is usually fixed, with filling strategy for DNS packets shorter than L and truncation strategy for DNS packets longer than L.

As shown in Fig.2, we assume L is 300, k is 64 and the number of byte values is 257. The DNS packet with length of 300 is first represented by bytes as a form of integer with a matrix of $\mathbf{d} \in \mathbb{R}^{300\times 1}$. Next, each byte is encoded by One-Hot encoding method as $\mathbf{d}' \in \mathbb{R}^{300\times 257}$. The byte encoded by One-Hot has 257 dimensions. Finally, an embedding matrix $EM \in \mathbb{R}^{257\times 64}$ is used to transform the DNS packet encoded by One-Hot to the representation $\mathbf{x} \in \mathbb{R}^{300\times 64}$.

B. CNN Model

The matrices after One-Hot encoding and embedding can be used to start standard CNN training and testing. The convolutional layer of CNN convolves DNS packet matrix. We use $*$ to represent the convolution operator. A one-dimensional convolution operation is realized by a convolution kernel with length of h, which can perform the convolution on the byte sequence fragment of length h and be expressed as $\mathbf{W} \in \mathbb{R}^{k\times h}$. After the convolution operation, the nonlinear activation function f is usually used to generate a new feature

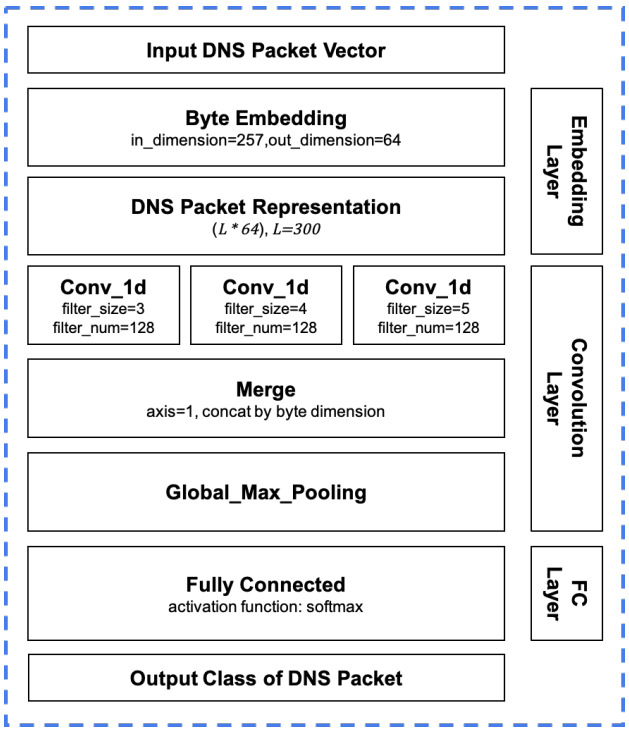

Fig. 3. The model configuration of CNN

map c_i. The convolution formula is as equation (2).

$$c_i = f(\mathbf{W} * \mathbf{x_{i:i+h-1}} + b_i) \tag{2}$$

In the above formula b_i represents bias. The new features generated by convolution operation on all byte sequence fragments of input DNS packet vector e.g. $x_1 \parallel x_2 \parallel x_3, x_2 \parallel x_3 \parallel x_4, \ldots, x_{L-2} \parallel x_{L-1} \parallel x_L$ when the length of convolution kernel is 3, can be expressed as $\boldsymbol{c} = [c_1, c_2, \ldots, c_{L-h+1}]$. Afterwards, we use a pooling layer to downsample these new features. The pooling layer can select the important features which can characterize the data while reducing the dimensions. There are two methods of pooling: Max Pooling and Average Pooling.

The convolutional layer of CNN contains different kind of convolution kernels of different lengths, and each kind of convolution kernel contains multiple convolution kernels of the same length. These convolution kernels can extract different local features of byte sequence fragments of different lengths. In CNN, a convolutional layer and its subsequent pooling layer constitute a block, and CNN is composed of multiple such blocks stacked. The pooled feature of the last block is flattened and then entered into the fully connected layer for classification. The entire network can be back propagated by minimizing the loss function to achieve the goal of training the model. The following Fig. 2 shows the process of classification for a DNS packet.

When building a CNN network which is used to detect DNS tunnel packets, we select $h = 3, 4, 5$ as the length of the convolution kernel, with 128 convolution kernels for each length. The dimension of vector that byte is embedded in is $k = 64$. The dimension of each DNS packet is fixed and set $L = 300$. On the model configuration, in convolutional layer, we use three convolution branches of convolution kernel length $h = 3, 4, 5$, each of which consist of convolution kernels with a specific length. After the convolution and nonlinear activation, the mapped features of three convolution branches are merged, and the global max pooling is selected as the pooling method to map the most important feature. At last, the new mapped features after pooling are passed to Softmax activation function of the fully connected layer for final classification, so as to determine whether the DNS packet is from DNS traffic tunnel. The follow Fig. 3 shows the configuration of the CNN model.

IV. Evalution

Our experimental work is mainly divided into three stages: data collection, data pre-processing and experiment evaluation. Details are as follows.

A. Data Collection

There are many tools for building DNS tunnels, such as Iodine, dns2tcp, dnscat2, OzymanDNS and ReverseDNShell mentioned [1][2][3][4][5]. They have different encoding methods, use different query types and achieve different functions. The query types, encoding methods and query domain length can be set by manual parameter configuration. For example, iodine uses various query types like SRV and NULL which are seldom used by other DNS tunnel tools. Dns2tcp usually uses Base64 encoding method to encode data which is to be transmitted. In the data collection phase, we simulate the use of different tunnel tools to generate DNS tunnel traffic for the purpose of both bypassing the paid Wi-Fi to access to the Internet and controlling the infected hosts. Details of the traffic implementation are shown in Table 1. In addition, we use the DNS Grind tool to query the top 1 million domain names to generate normal DNS traffic. In order to prevent data imbalance in training CNN model, we set the ratio of normal DNS packets and DNS tunnel packets used for training as 1:1, totaling about 800,000 DNS packets. At the same time, in order to ensure that the detection results are independent of the training process of the model, the detection data we use are not used for training, totaling about 200,000 DNS packets. To maintain data consistency, we only use request packets, although response packets can also be a good candidate. We use the method of monitoring the network interface backstage and saving the captured DNS data as pcap file to obtain normal DNS data set and DNS tunnel data set. The pcap file is the most primitive form of the data set, which can be used for training and testing model after the subsequent data pre-processing stage.

B. Data Pre-processing

Since the captured DNS packets is saved in the form of pcap file, we first get the application layer data of DNS request from pcap files and represent each DNS request packet as a DNS request packet vector composed of bytes in an integer form.

TABLE I
THE DETAILS OF DNS TRAFFIC SIMULATION

DNS Tunneling Tools	Encoding Method	Query Type	Number for Train	Number for Test
Iodine	Base32 Base64 Base128 Raw	A,MX, CNAME, TXT,SRV, NULL	400000	100000
Dns2tcp	Base64	TXT		
Dnscat2	Base32	A,AAAA, CNAME, MX,TXT		
OzymanDNS	Base32	TXT,A		
ReverseDNShell	Base64	TXT,A		
Normal		A,NS,MX, TXT,PTR, AAAA	400000	100000

[a]The last line illustrates the composition of normal DNS data.

TABLE II
THE COMPOSITION OF FEATURE SET

Feature Categories	Feature Number	Features
Data Length	1	DNS Request Packet Length
Domain Entropy	1	Domain Entropy
Query Type	9	Is "A" Type
		Is "AAAA" Type
		Is "NS" Type
		Is "MX" Type
		Is "CNAME" Type
		Is "TXT" Type
		Is "PTR" Type
		Is "SRV" Type
		Is "NULL" Type

Pcap file is composed of a pcap file header which is long about 24 bytes long and a series of captured packets, where each packet is composed of a packet header which is about 16 bytes long and MAC data frame. We write a python script to get the application layer data of each DNS packet from the pcap file.

After obtaining each DNS request packet from the pcap file and representing it as a vector expressed by bytes in an integer form, it is necessary to truncate and pad the DNS request packet to ensure the alignment of input data. Then they are encoded with One-Hot method, and entered into the constructed CNN network for training, testing and evaluating.

C. Experiment Evaluation

a) Machine Learning Methods: Traditional machine learning methods require manually extracted feature set to train models. Since DNS tunnels usually increase the length of the payload of DNS packets for the purpose of data exchange and data encoding technology usually makes DNS domain names more random, DNS packet length and DNS domain name entropy can be used as important features to identify DNS tunnels. Furthermore, the length of information that common types such as A, AAAA, PTR can carry often cannot meet the needs of data transmission by DNS tunnel tools. Therefore, DNS query type can also be used as an important feature to distinguish DNS tunnel. Before testing our proposed method, we first use Logistic Regression, SVM and Neural Network machine learning methods to train and test on the DNS data set mentioned above based on the eleven features of DNS request packet length, DNS query name entropy and DNS query type. Table 2 shows the composition of feature set.

TABLE III
PERFORMANCE COMPARISON OF FOUR MODELS

Models	Accuracy(%)	Presision(%)	Recall(%)	F1-Score
SVM	86.46	86.69	86.16	0.8643
Logistic Regression	95.80	92.30	99.95	0.9597
Neural Network	98.99	98.31	99.69	0.9899
Byte-level CNN	99.98	1.00	99.96	0.9998

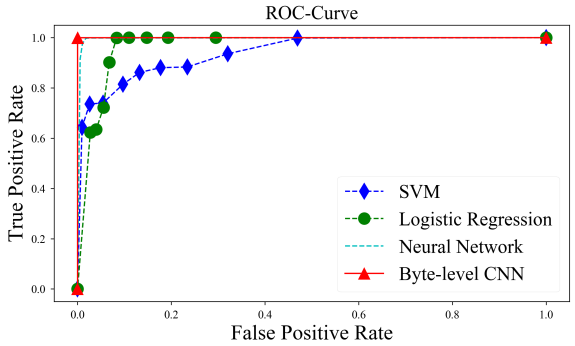

Fig. 4. ROC curves of four models

b) Experiment Results: Table 3 compares the results of testing using our byte-level CNN model with those of testing using three traditional machine learning methods. As can be seen from Table 3, the performance of SVM method is the worst, and its accuracy, precision and recall rate are relatively low, only about 86%. The recall rate of logistic regression method is relatively high, as much as 99.95%, but the accuracy rate is relatively low, indicating that logistic regression model misclassify more normal DNS data as DNS tunnel data. The neural network model can achieve better precision and recall rate and its accuracy reach 98.99%. Finally, our proposed byte-level CNN model has a higher accuracy, precision and recall rate than the traditional machine learning method. With a high recall rate, since its precision is close to 100%, and its F1-score is extremely high, in fact the highest among the methods we show. This shows that our method mistakes fewer normal DNS packets for DNS tunnel packets than simple neural network. As far as the rare misclassified data, we find that most of the domain names are relatively long in length, poor in semantics and chaotic in structure. While for a small number of undetected data, we find that the subdomain names used for payload are very short, and are generally from polling in DNS tunnels. Fig. 4 shows the ROC curves of these four methods. The solid line represents the byte-level CNN model proposed by us, and it can be seen from the figure that it shows an obvious right-angle shape, while the ROC curve of SVM model and logistic regression model show poor performance

and not present obvious trend of right-angle. The performance of neural network model is also good, but it is inferior to the byte-level CNN method we propose.

V. Related Work

In recent ten years, many security experts have been studying DNS tunnels and its detection methods. With regard to the detection of DNS tunnels, Farnham et al. [6] concluded that DNS tunnels can be analyzed from two aspects: load analysis and traffic analysis. The content of load analysis mainly includes the size of DNS packet, subdomain domain name entropy, query type, specific signature, the proportion of numeric characters in domain name, the proportion of the length of the longest semantic substring, the number of special characters and other statistical characteristics. Traffic analysis mainly includes analyzing DNS traffic of each IP address, DNS traffic of each domain name, number of subdomain names of each domain name, number of NXDomain responses and existence of isolated DNS requests, etc. Kara et al. [16] proprosed to analyze the resource record activities of domain names and build DNS zone profiles to detect payload distribution channels.

With the popularity of machine learning methods, many researchers have begun to use them on DNS tunnels detection and their related research are based on load features or traffic features. Homem et al. [14] proposed three manually extracted features which are IP packet length, DNS domain name length and DNS domain name entropy. The detection results of four types of DNS tunnels, namely, HTTP-DNS, HTTPS-DNS, POP3-DNS and FTP-DNS using four machine learning methods (kNN, decision tree, SVM and neural network) were compared. The experimental results show that neural network can obtain the most ideal detection effect that the detection accuracy can achieve 95%. They also proposed a MeanDiff method to estimate the application layer protocols carried by DNS tunnels in [17]. Almusawi et al. [12] put forward to extracting six features for each connection: the length of DNS request packet, the length of IP request packet, the length of IP response packet, the entropy of application layer of request packet, the entropy of IP packet and the entropy of domain name. At first, k-means clustering method was used for data discretization, and then a kernel SVM method was used for multi-label classification of four different types of DNS tunnels, namely, HTTP-DNS tunneling, HTTPS-DNS tunneling, POP3-DNS tunneling and FTP-DNS tunneling, achieving an average accuracy rate of 75.9% and a recall rate of over 80%. Nalder et al. [13] proposed to extract six features within a sliding window on each domain name, which are character entropy, IP-hostname exchange resource records type distribution, unique query ratio, unique query volume, query length average and longest meaningful word over domain length average. The isolated forest method was used to train on normal DNS dataset and detect anomalies on test dataset. This method can solve the problem of unbalanced proportions of normal and abnormal data, and can detect malicious and low throughput data exfiltration over the DNS protocol which may last for months as well. Cambiaso et al. [15] proposed an innovative algorithm which combined Principal Component Analysis and Mutual Information to profile DNS tunnels. Other machine learning methods on DNS tunneling detection are as in [10] [11] [18] [19]. These traditional machine learning methods are based on the manually extracted feature sets (mainly statistical features) to get the computable data of the model. The main problems of manual feature extraction are as follows: 1) the process of feature extraction is time-consuming and labor-consuming, and the experts with professional knowledge are limited. Meanwhile, the quality of feature set is greatly affected by human factors; 2) the manual feature extraction of traditional machine learning methods mainly extracts the statistical features of data. Therefore, for data with spatial and temporal structure, its structural and sequential information cannot be fully utilized.

The popular deep learning method appearing recently is an improvement to the traditional machine learning method. As long as an appropriate representation of original data into vectors is found, deep learning can achieve automatic feature extraction, meanwhile retaining the time and space structure of data, and making full use of the information contained in it. Deep learning method was mainly used in the field of image processing at the beginning, and has achieved good results in the field of text classification. Zhang et al. [20] put forward a character-level CNN method for text classifications. Jiang et al. [21] first used character-level CNN to detect malicious URLs in the field of security. Le et al. [22] also used character-level and word-level CNN to detect malicious URLs, and achieved 99.29% detection accuracy. The idea of using character-level CNN to detect malicious URLs can also be used to detect DNS tunnels based on DNS subdomain names. However, this method still has the disadvantage of not making full use of all the DNS information. In fact, the DNS load also has query type, the number of resource records and the length information which can be used to detect tunnels. Therefore, our proposed byte-level CNN method which is based on the entire DNS packet and represented by byte to detect DNS tunnels can not only realize automatic feature extraction and learn structural and sequential information, but utilize all of the load information as well.

VI. Conclusion

In this paper, we propose a byte-level CNN method to detect DNS tunnels. Compared with the existing traditional machine learning methods, our method can not only automatically extract feature set, but learn sequential and structural information of data as well. Compared with character-level CNN method which is based on the subdomain name or domain name to detect DNS tunnels, our method can make full use of the information of the whole DNS packet. From the above experimental results, we can see that our byte-level CNN method has achieved very good results, achieving 99.98% detection accuracy, 99.96% recall rate and nearly 100% precision.

Even the above good results, because DNS tunnel data in our data set mainly comes from DNS tunnel tools, our method may not detect DNS tunnel data generated by some malware using DNS protocol to attack. The reason is that some malware will take measures to limit the length of domain names in order to avoid being detected. For example, a malware called FrameworkPOS successfully stole much credit card account information in 2014 [13]. Some malware may use clear text to transfer without using encoding technology to reduce the randomness of domain names in DNS payload. In the future, research into these types of malware that use the DNS protocol for data transmission should be carried out. Futhermore, the method we propose at present is based on DNS payload. In the future research, we can also consider adopting the deep learning algorithms with excellent performance in the field of artificial intelligence from the perspective of DNS traffic, such as: LSTM, GAN (Generative Adversarial Networks [23]), etc. Moreover, the method from the traffic perspective is often more suitable for the application in the real network environment.

ACKNOWLEDGMENT

This research is supported by the National Science Foundation of China (No. 61572497, 61772279).

REFERENCES

[1] B. Andersson, iodine, https://code.kryo.se/iodine.
[2] O. Dembour, dns2tcp, https://tools.kali.org/maintaining-access/dns2tcp.
[3] R. Bowes, dnscat2, https://github.com/iagox86/dnscat2.
[4] D. Kaminsky, OzymanDNS, https://dankaminsky.com/?s=OzymanDNS.
[5] D. Borges, ReverseDNShell, http://www.github.com/ahhh/Reverse_DNS_Shell.
[6] G. Farnham, "Detecting dns tunneling," SANS Institute, 2013.
[7] S. Bromberger, "Dns as a covert channel within protected networks," National Electronic Sector Cyber Security Organization (NESCO), 2011.
[8] C. J. Dietrich, C. Rossow, F. C. Freiling, et al., "On botnets that use dns for command and control," in 7th European Conference on Computer Network Defense (EC2ND), pp. 9-16, September 2011.
[9] C. Mullaney, "Morto worm sets a (dns) record," Symantec Official Blog, 2011.
[10] M. Aiello, M. Mongelli, and G. Papaleo, "Basic classifiers for DNS tunneling detection," in 2013 IEEE Symposium on Computers and Communications (ISCC), pp. 880-885, IEEE, July 2013.
[11] V. T. Do, P. Engelstad, B. Feng, and T. van Do, "Detection of DNS tunneling in mobile networks using machine learning," in International Conference on Information Science and Applications (ICISA), vol. 424, pp. 221-230, Springer, Singapore, 2017.
[12] A. Almusawi and H. Amintoosi, "DNS Tunneling detection method based on multilabel support vector machine," Security and Communication Networks, 2018.
[13] A. Nadler, A. Aminov, and A. Shabtai. "Detection of malicious and low throughput data exfiltration over the DNS protocol," Computers & Security, vol. 80, pp. 36-53, January 2019.
[14] I. Homem and P. Papapetrou, "Harnessing predictive models for assisting network forensic investigations of dns tunnels," in ADFSL Conference on Digital Forensics, Security and Law, Daytona Beach, 2017.
[15] E. Cambiaso, M. Aiello, M.mongelli and G. Papaleo, "Feature transformation and Mutual Information for DNS tunneling analysis," 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), IEEE, 2016.
[16] A. M. Kara, H. Binsalleech, M. Mannan, A. Youssef and M. Debbabi, "Detection of malicious payload distribution channels in DNS," in 2014 IEEE International Conference on Communications (ICC), pp.853-858, June 2014.
[17] I. Homem, P. Papapetrou and S. Dosis, "Entropy-based prediction of network protocols in the forensic analysis of dns tunnels," in World Congress on Internet Security (WorldCIS), London, IEEE, 2016.
[18] A. L. Buczak, P. A. Hanke, G. J. Cancro, M. K. Toma, L. A. Watkins, and J. S. Chavis, "Detection of tunnels in PCAP data by random forests," in Proceedings of the 11th Annual Cyber and Information Security Research Conference, ACM, 2016.
[19] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang and C. Peng, "Detecting DNS Tunnel through Binary-Classification Based on Behavior Features," in 2017 IEEE Trustcom/BigDataSE/ICESS, IEEE, August 2017.
[20] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," In Advances in Neural Information Processing Systems (NIPS), pp. 649–657, 2015.
[21] J. Jiang, J. Chen, K. R. Choo, C. Liu, K. Liu, M. Liu and Y. Wang, "A deep learning based online malicious URL and DNS detection scheme," in International Conference on Security and Privacy in Communication Systems, pp. 438-448, Springer, Cham, 2017.
[22] H. Le, Q. Pham, D. Sahoo and S. C.H. Hoi, "URLnet: Learning a URL representation with deep learning for malicious URL detection," arXiv preprint arXiv:1802.03162, 2018.
[23] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," arXiv preprint arXiv:1607.02533, 2016.