Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

DNS covert channel detection method using the LSTM model

Shaojie Chen[a,*], Bo Lang[a], Hongyu Liu[a], Duokun Li[a], Chuan Gao[b]

[a] State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[b] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

ARTICLE INFO

Article history:
Received 20 May 2020
Accepted 21 October 2020
Available online 28 January 2021

Keywords:
DNS
Covert channel detection
FQDN
LSTM
Grouped filtering

ABSTRACT

DNS is a kind of basic network protocol that is rarely blocked by firewalls; therefore, it is used to build covert channels. Malicious DNS covert channels play an important role in data exfiltration and botnets and do great harm to the network environment. To detect DNS covert channels, researchers extract multiple features from different perspectives of DNS traffic. At present, many detection methods using machine learning are based on manual features, which usually include complex data preprocessing and feature extraction. Additionally, these methods seriously rely on expert knowledge, and some potential features are hard to discover. To address these problems, we propose a DNS covert channel detection method using the LSTM model, which does not rely on feature engineering. First, we use the FQDNs of DNS packets as the input and implement an end-to-end detection approach using LSTM. Then, we filter the detection results of the LSTM model with the grouped filtering method to further reduce the false positive rate. Using the packets from the Internet and the packets generated by running different DNS covert channel tools, we construct our datasets, in which generalization test datasets are included in addition to the FQDN and the DNS packet datasets for model training. Our method achieves an accuracy rate of 99.38% on the test dataset and a recall rate of 98.52% on the generalization test dataset, which are better than the state-of-the-art methods. This method is also tested in a real network environment and has detected multiple malicious DNS covert channel events.

1. Introduction

The domain name system (DNS) is a decentralized system that provides a worldwide service. The DNS protocol is mainly used to build the mapping relationship between domain names and Internet Protocols (IPs) and to convert readily memorized domain names to numerical IP addresses; it is a kind of basic network protocol that is rarely blocked by firewalls. Therefore, attackers often use the DNS to build covert channels; they encapsulate the covert data in DNS packets and thereby transmit the covert data through DNS resolving. In recent years, a variety of methods have been developed to build DNS covert channels, which can be divided into two categories: IP connection-based covert channels and query name-based covert channels. The IP connection-based covert channels are directly established connections through IP addresses between a client and a server, which uses the port of 53 usually occupied by the DNS, and does not need to perform in

* Corresponding author.
E-mail addresses: chenshaojie@buaa.edu.cn (S. Chen), langbo@buaa.edu.cn (B. Lang), liuhongyu@buaa.edu.cn (H. Liu), liduokun@buaa.edu.cn (D. Li), gc@cert.org.cn (C. Gao).
https://doi.org/10.1016/j.cose.2020.102095
0167-4048/© 2020 Elsevier Ltd. All rights reserved.

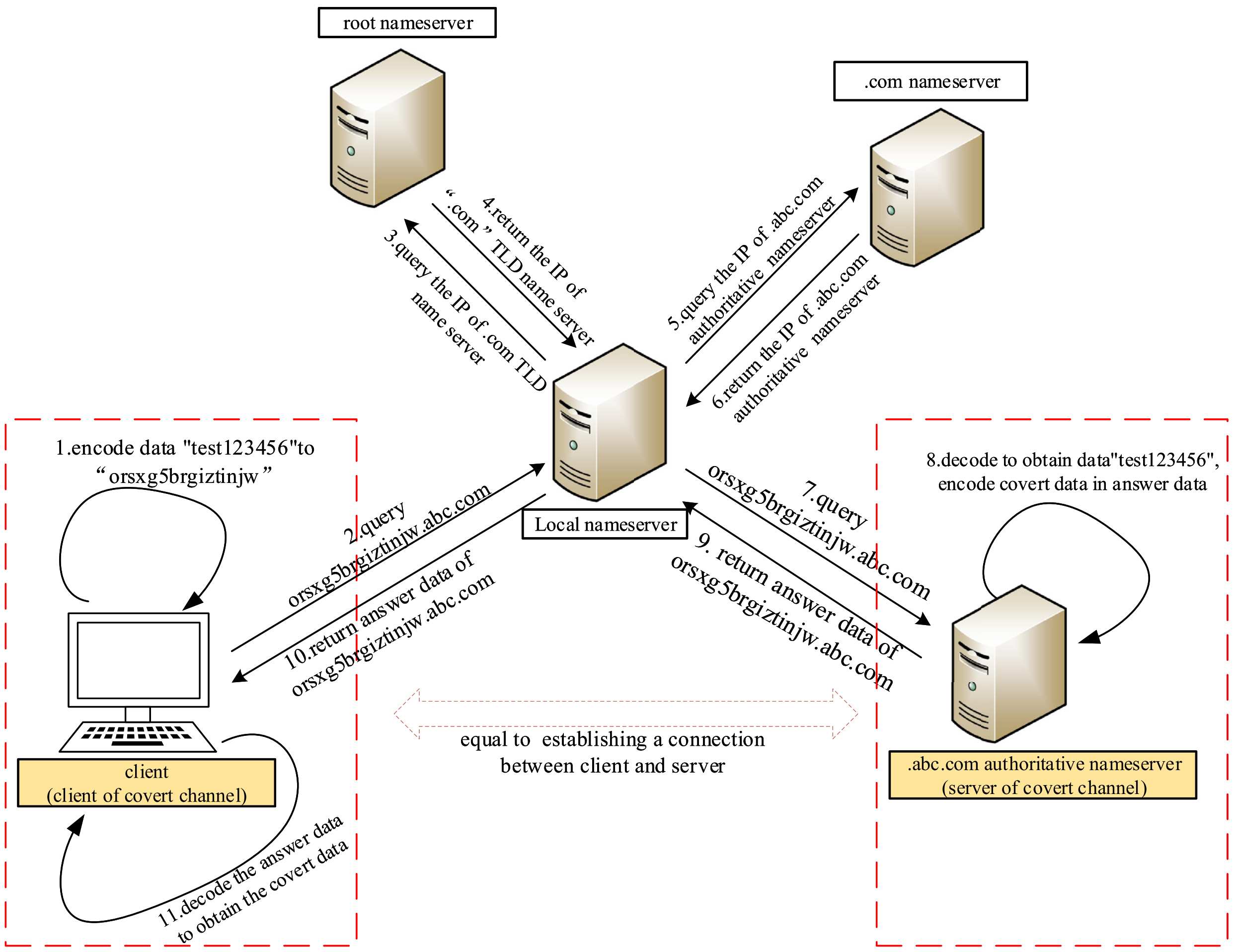
root nameserver
.com nameserver
3.query the IP of .com TLD name server
4.return the IP of ".com" TLD name server
5.query the IP of .abc.com authoritative nameserver
6.return the IP of .abc.com authoritative nameserver
Local nameserver
1.encode data "test123456"to "orsxg5brgiztinjw"
2.query orsxg5brgiztinjw.abc.com
10.return answer data of orsxg5brgiztinjw.abc.com
7.query orsxg5brgiztinjw.abc.com
9. return answer data of orsxg5brgiztinjw.abc.com
8.decode to obtain data"test123456", encode covert data in answer data
client (client of covert channel)
.abc.com authoritative nameserver (server of covert channel)
equal to establishing a connection between client and server
11.decode the answer data to obtain the covert data

Fig. 1 – Schematic diagram of the query name-based DNS covert channel.

accordance with the constraints of the request for comments (RFC) of the DNS (Mockapetris, 1987). Arbitrary data with raw user datagram protocol (UDP) packets could be transmitted on the channels, and a large amount of malformed DNS packets might appear. The query name-based covert channels include a client and a server. The server acts as the authoritative nameserver for a domain, and the client establishes a connection with the server by querying the fully qualified domain names (FQDNs) of the domain. The "FQDN" in our paper only represents the FQDNs that were queried by clients. The principle of this kind of covert channel is shown in Fig. 1. The authoritative nameserver for the domain "abc.com" is the server of the covert channel. The client converts the data to a string by specific encoding formats such as Base16, Base32, and Base64. For instance, the client encodes "test123456" with Base32 and obtains the string "orsxg5brgiztinjw". Then, he/she concatenates the encoded string and the domain ("abc.com") to form an FQDN "orsxg5brgiztinjw.abc.com" and queries the FQDN from the local name server. The request is finally transmitted to the authoritative name- server for the domain "abc.com". This server decodes the subdomain part to obtain the original data "test123456". Then, the response data are encoded by the server and sent to the client as the resource record (RR) data in the response packet. In this way, the client establishes a covert channel with the server.

A variety of tools for constructing DNS covert channels have appeared, such as Iodine (Ekman, 2014), Dnscat2 (Ron, 2019), Cobaltstrike (Strategic Cyber LLC, 2019), and FrameworkPOS (Rascagneres, 2016). The covert channels could achieve a relatively high communication efficiency. Leijenhorst et al. (van Leijenhorst et al., 2008) experimentally proved that the DNS covert channel could reach a speed of 110 KB/s with a delay of only 150 ms. In actual networks, there are legal uses and malicious uses of DNS covert channels. In terms of legal uses, Pan et al. (Pan et al., 2018; Pan et al., 2019) used the DNS covert channel to encrypt DNS requests and then built a private DNS service to prevent the IP addresses of clients from being obtained by the recursive nameservers. McAfee LLC. uses the domain "avts.mcafee.com" for its global threat intelligence (GTI) file reputation system (McAfee LLC., 2019). The client queries the FQDN, which encodes the file information in the subdomain part of "avts.mcafee.com", and the server uses different returned IP addresses to represent the attribute of the file. There are other similar domains that are used by legal DNS covert channels, such as ”sophosxl.net” (Sophos Ltd, 2018) and ”e5.sk” (ESET, spol.s r.o, 2019). Malicious DNS covert channels generally include two categories, i.e., DNS exfiltration and DNS tunneling (Steadman and Scott-Hayward, 2018). DNS exfiltration is the basic form of DNS covert channels. The client encapsulates the data in DNS packets, and the authoritative nameservers restore the data. For DNS tunneling, attackers encapsulate another protocol in DNS traffic and establish communications through DNS resolving. For instance, some cyber criminals stole 56 million of debit and credit card numbers from the customers of *Home Depot* using DNS exfiltration (Rascagneres, 2016) in 2014. In 2018, attackers used the domain ”dns.0nlinecc.com” and realized the control of other hosts based on the *PlugX* remote control program and DNS tunneling (360 Active Defense, 2018). Malicious DNS covert channels have caused great harm to the network environment.

In recent years, the detection of DNS covert channels has attracted the attention of researchers, and various detection methods have appeared. Individuals have extracted features and detected DNS covert channels based on the differences between normal DNS traffic and DNS covert channel traffic, such as the time intervals between packets, FQDNs and resource record data. From the perspective of the detection data unit, the methods can be divided into two categories: single packet-based methods and packet set-based methods. For the single packet-based methods, researchers work on single DNS requests or response packets or request/response packet pairs and they generally focus on payloads (Qi et al., 2013; Ahmed et al., 2019; Zhang et al., 2019; Das et al., 2017) (such as FQDNs or resource record data of the response packets) or other contents (Liu et al., 2019a; Engelstad et al., 2017; Buczak et al., 2016). For example, FQDN-based methods (Qi et al., 2013; Ahmed et al., 2019; Zhang et al., 2019) only extract features from FQDNs; RR-data-based detection methods (Das et al., 2017) mainly target specific types of resource record data of DNS response packets, such as the TXT type. The DNS packet set-based methods (Das et al., 2017; Born and Gustafson, 2010b, a; Aiello et al., 2013; Aiello et al., 2014; Aiello et al., 2015; Cambiaso et al., 2016; Aiello et al., 2016; Nadler et al., 2019; Binsalleeh et al., 2014; Kara et al., 2014; Liu et al., 2017; Karasaridis et al., 2006; Ellens et al., 2013) usually focus on all DNS packets passing through a node in the network. Using this method, the packets are divided into sets based on the contents of the packets, such as the time, the domain of the FQDNs, and the request/response IP. Then, multidimensional features are extracted from each set and determinations are made using machine learning models. In addition to detecting the existence of covert channels, some studies have further detected the protocol types used in the covert channel (Homem et al., 2017; Homem and Papapetrou, 2017). In this paper, we only focus on the detection of DNS covert channels in traffic.

Among the aforementioned two detection methods, single packet-based methods are more real-time and have low computational complexity, while packet set-based methods can sufficiently use session information and supress the interference of the lack of information in single packets. However, DNS packet set-based methods generally have high operation complexity and require a large amount of memory resources. Another problem is that the packets in the same set might vary greatly, which makes it hard to extract effective features. Additionally, these methods might extract the time features of the packets. As the time intervals of the packets in the simulated environment and the real network are significantly different, effective time features cannot be easily obtained from the datasets. For the single packet-based methods, those that use RR data only focus on a few types of resource records, which makes it difficult to achieve universal detection effects, as the DNS covert channel could use a variety of resource records. FQDN-based methods only inspect FQDNs of each DNS packet. Additionally, the FQDNs are robust to environments, which is good for achieving the high generalization ability of the detection methods, and DNS covert channel packets usually show characteristics of randomness on the FQDNs, which are not affected by the type of resource records. Therefore, we believe that FQDNs are effective and efficient inspection points and study FQDN-based detection methods. Since both DNS exfiltration and DNS tunneling have similar characteristics on the FQDNs, this kind of method can detect the two kinds of covert channels.

Due to the limitations of feature engineering, researchers have begun to pay more attention to automatic feature extraction. Deep learning models can automatically extract features from data and have become a hot topic in machine learning. In recent years, long short-term memory (LSTM) models have been widely used in speech processing (Graves et al., 2013), text processing (Senior et al., 2015), machine translation (Bahdanau et al., 2014), etc. Convolutional neural network (CNN) models have been widely used in computer vision (Krizhevsky et al., 2012) and sentence classification (Kim, 2014), etc. Both LSTM and CNN have demonstrated good performance in their application scenarios. However, in the area of malicious network traffic detection, deep learning methods have just begun to appear. For instance, LSTM models and CNN models have been used for payload detection in network traffic (Liu et al., 2019b). Aiello et al. (Aiello et al., 2014; Aiello et al., 2015) used deep neural networks (DNNs) to detect covert channels, but the networks were only used for classification rather than feature extraction. Liu et al. (Liu et al., 2019a) directly used deep learning models to detect byte sequences. Zhang et al. (Zhang et al., 2019) adopted different deep learning models to detect DNS covert channels by FQDNs; however, they do not pay much attention to the generalization capability.

In our paper, we consider the requested FQDNs of DNS packets as payloads. We train LSTM and CNN models to inspect the payloads to determine whether there contains DNS covert channel traffic. According to our experimental results, we find that the LSTM model performs better than the CNN model.

There are some content delivery networks (CDNs) and specific DDOS attacks (Griffioen and Doerr, 2019) that use FQDNs similar to the FQDNs of DNS covert channels. These samples may lead to false alarms by using only FQDN inspection. In addition, some legal DNS covert channel traffic may also be detected by the model. Therefore, to further reduce the false positive rate, by combining the idea of packet set detection methods, we propose grouping and whitelist methods to further filter the results given by the LSTM model.

To summarize, our method has the following contributions:

1. The LSTM model is introduced to the detection of DNS covert channels using FQDNs, which do not rely on feature engineering and could achieve end-to-end real-time detection. We also analyze and compare the performance of the LSTM model and the CNN model. The LSTM model performs better and achieves an accuracy of 99.38%. Because the network environment is extremely complex and the DNS covert channel traffic may vary greatly, we especially pay more attention to the generalization capability of our model. We construct generalization verification and test datasets during the model training process. The LSTM model achieves a recall rate of 98.5%, which is much higher than other methods.
2. According to the characteristics of the LSTM model and FQDN, we designed multiple data preprocessing methods. The preprocessing methods could retain the original information as much as possible without grouping the large amount of DNS traffic. The simple and efficient processing methods are good for achieving effective end-to-end real-time detection.
3. We design a packet grouped filtering method based on domains, and a whitelist filtering method to filter the suspected DNS covert channel packets detected by the LSTM model. The filtering methods effectively remove the non-covert channel packets and legal covert channel packets and significantly reduce the false positive rate of our method.

The remainder of the paper is organized as follows. Section 2 introduces the related work of DNS covert channel detection. Section 3 introduces the motivation and method of the detection method based on the LSTM model. Section 4 shows the construction of the datasets and the experiments of the model on the datasets and then compares our method with the state-of-the-art method. Section 5 summarizes the paper.

2. Related work

In recent years, researchers have performed many studies on DNS covert channel detection. The methods can be divided into rule-based detection methods and machine learning detection methods. In rule-based detection methods, researchers manually analyze the DNS covert channel traffic and then obtain firewall rules to monitor the DNS covert channels. For example, Sheridan et al. (Sheridan and Keane, 2015) used Iodine to construct a DNS covert channel, performed an experimental analysis on the obtained traffic, and finally obtained some *Snort* rules for the Iodine DNS covert channel traffic.

Alternatively, machine learning detection methods can be divided into two categories according to the detection goals: covert detection and covert protocol classification. Covert detection distinguishes DNS covert channels from normal traffic, which can be further divided into two kinds from the perspective of determination units, namely, single packet-based detection methods and packet set-based detection methods. Covert protocol classification detects the protocol types used in the covert channel. For instance, Homem et al. (Homem et al., 2017; Homem and Papapetrou, 2017) assumed that covert channel traffic has already been detected by the methods from Born et al. (Born and Gustafson, 2010b) and Qi et al. (Qi et al., 2013), and based on this assumption, the method of (Homem et al., 2017) classifies the internal protocol type by the entropy value of the FQDNs. The method of (Homem and Papapetrou, 2017) extracts the IP packet length, the length of the FQDN and the entropy of the FQDN as features and classifies the internal protocol type through machine learning models.

In terms of the datasets, there is currently no suitable public dataset for DNS covert channel detection. Researchers usually have to construct datasets by themselves. Most of them utilize DNS traffic in actual networks or the FQDNs in the Alexa top ranking (Alexa Web Information Company. Topsites, 2020) data as the normal DNS traffic set and use traffic from DNS covert channel tools (Iodine, Dnscat2, etc.) as the abnormal traffic set.

2.1. *Single packet-based detection methods*

Single packet-based detection methods detect DNS covert channels by the content of single DNS packets or request/response pairs. Common detection contents are the payloads of packets, such as the FQDNs or RR data, in the response packets. In addition, there are studies that use other contents, such as IPs, packet lengths, and the byte sequences of DNS packets.

FQDN-based detection methods assume that in a DNS covert channel, the data sent by the client could usually only be encoded in the subdomain part of the FQDNs. Born et al. (Born and Gustafson, 2010b, a) and Qi et al. (Qi et al., 2013) believed that the character distribution of normal DNS FQDNs generally conforms to the distribution of natural language, while the character distribution of covert channel FQDNs tends to conform to a uniform distribution. Therefore, these investigators segmented the FQDNs into words by the n-gram method and analyzed the word frequency to make judgments. Among these methods, Qi et al. (Qi et al., 2013) used each single FQDN as a determination unit and calculated a score for each FQDN based on the bigram character frequency. Then, they used a threshold method to make determination. However, Born et al. (Born and Gustafson, 2010b, a) used FQDN sets as determination units. Ahmed et al. (Ahmed et al., 2019) extracted 8-dimensional features of the FQDN, such as the count of characters, count of the uppercase characters, number of labels, average label length, and entropy. These authors constructed a dataset of benign FQDNs to train an isolation forest model to detect the covert channel FQDNs. Zhang et al. (Zhang et al., 2019) used various deep learning models, such as DNN, CNN and recurrent neural network (RNN) models, to inspect the FQDNs of the DNS packets.

RR-data-based detection methods mainly detect covert channels by the resource record data of the DNS response packets. Das et al. (Das et al., 2017) detected DNS exfiltration and DNS tunneling separately. For DNS tunneling detection, these researchers only focused on the resource record data of the TXT type, from which 10-dimensional features (such as the ratio of uppercase alphabets, ratio of lowercase alphabets, number of digits, number of dots in string, and entropy) were extracted. These coworkers first clustered the labeled data and obtain several clusters, then classified new samples into these known clusters and finally made determinations by the categories of the known clusters.

In addition to the two types of methods above, researchers have proposed methods that do not or not only use payloads as determination contents. Liu et al. (Liu et al., 2019a) used the one-hot method to encode the byte sequences of DNS packets and used the CNN model to judge whether or not the packets were from covert channels. Engelstad et al. (Engelstad et al., 2017) mainly aimed to detect DNS tunneling in mobile networks and used the pairs of request and response packets as the determination unit. These investigators extracted some features (such as the time, source IP, destination IP, length-up, length-down, and information of packets) from each pair of packets and used one-class support vector machine (SVM) and K-means methods to make determinations. Buczak et al. (Buczak et al., 2016) also used the pairs of request and response packets as data objects. They extracted 59 dimensional features, such as the ratio of the unique characters of the FQDNs, the mean and variance of the request packet sizes, the response packet sizes, and the packet time intervals, some of which are extracted from sliding windows. These researchers made determinations by using a random forest model.

2.2. *Packet set-based detection methods*

Packet set-based detection methods usually group the DNS packets into sets, extract features on each set, and then make determinations with machine learning methods. The grouping conditions mainly include the time slices, domains of FQDNs, request/response IPs, network flows, etc.

The grouping methods by time slices usually group the total DNS packets that pass through a node, i.e., a local DNS nameserver into different sets according to time slices or sliding windows, and then extract features and make determinations. The purpose of these methods is to detect packet sets containing DNS covert channels. Aiello et al. (Aiello et al., 2013) aimed to measure DNS tunneling over a working period of the local server. These investigators used the packet sets composed of n pairs of request/response packets as determination units and extracted 12 dimensional features from the packet sets, including the mean, variance, skewness and kurtosis of the request packet size, response packet size and time interval. Then, Aiello and coworkers used the Bayes classifier to make determinations. The parameter n could be adjusted to meet the different real-time needs. Aiello et al. (Aiello et al., 2014; Aiello et al., 2015) proposed a new approach for DNS tunneling detection that contains two levels of classifiers. The first level classifier is similar to the method in (Aiello et al., 2013) and also uses three other methods (k-nearest neighbor (KNN), neural network, SVM algorithms) in addition to the Bayes classifier to classify packet sets in a short period of time. Then, the second level classifier uses ensemble learning methods, such as boosting or bagging, to classify the packet sets in a longer period of time based on the results from the first level classifier. Cambiaso et al. (Cambiaso et al., 2016) and Aiello et al. (Aiello et al., 2016) believed that normal DNS traffic has greater mutual information between each packet than DNS covert channel traffic. They grouped the DNS packets into sets with the sliding window size of n pairs of request/response packets, extracted the same features as Aiello et al. (Aiello et al., 2013) from each set, and reduced the feature dimensionality to 2 with principal component analysis (PCA). Then, these authors calculated the mutual information between the features of consecutive r sets and finally used the threshold method to detect DNS covert channels.

The grouping methods with domains divide all acquired DNS packets according to the domains of the FQDNs. Since the server of a covert channel is the authorization nameserver of a domain (or subdomain), such grouping methods could group the packets of one server into the same set. Nadler et al. (Nadler et al., 2019) mainly detected low-throughput DNS exfiltration. They use sliding windows with a size of λ minutes and extracted 6-dimensional features, such as the character entropy, unique query ratio, non-IP query-type rate, and query length average. Then, these researchers trained an isolation forest model using the features of normal DNS traffic to detect DNS exfiltration. To better detect low-throughput exfiltration, they combined the result of n windows in a longer inspection window. Binsalleeh et al. (Binsalleeh et al., 2014) used the domain of FQDNs as the determination units. First, they analyzed the corresponding relationship of the subdomains and TXT-type resource record data with the graph method and set up the query and response patterns of each domain. Then, these investigators used the query and response patterns to make determinations. Based on passive DNS records, Kara et al. (Kara et al., 2014) determined each DNS zone. A DNS zone is a subdomain that has *NS* resource record, which means there are authoritative nameservers of the subdomain, and the nameserver of a DNS zone gives the answer of the FQDNs under the zone. For each DNS zone, these coworkers used the distributions of different resource record types to make determinations.

To characterize the session information of DNS covert channels, some researchers further used not only domain but also IPs as the basis of grouping. These methods could group the traffic based on the client and the server of a DNS covert channel at the same time. Born et al. (Born and Gustafson, 2010b, a) detected DNS covert channels by n-gram character frequency analysis of the FQDNs. They calculated the unique n-gram character rank frequencies of each FQDN set and then compared the frequencies with the fingerprints calculated from legitimate traffic to make determinations. In paper (Born and Gustafson, 2010b), these authors proved that a small amount of data, such as 100 FQDNs, could effectively separate normal and tunneled traffic. Then, in paper (Born and Gustafson, 2010a), the traffic was split into files by the IP address and domain. The FQDNs in each file are used as a determination unit. The DNS exfiltration detection method proposed by Das et al. (Das et al., 2017) grouped the FQDNs of DNS packets based on the same request IP and the same domain. These researchers concatenated the subdomain parts of the FQDNs in one packet set into a string and extracted 8 dimensional features from the concatenated string, such as the entropy, length of the string, ratio of the number of uppercase alphabets, and number of unique subdomains. Then, Das and coworkers used a logistic regression model to make determinations. Liu et al. (Liu et al., 2017) grouped the pairs of request/response packets according to a window size of *n* pairs and guaranteed that the packets of each set have the same request IP, response IP and domain of FQDNs. They extracted 18-dimensional features of the grouped pairs, which could be divided into four categories: time interval, packet size, subdomain entropy and record type. Then, these investigators used binary classification models, such as decision trees, SVMs and logistic regression models, to detect the covert channels.

Some studies determine the DNS flows, which means a set of packets having a set of common properties, such as source IP address, destination IP address, source port, and destination port. Karasaridis et al. (Karasaridis et al., 2006) set up a tunneling attack detector (TUNAD) to detect DNS tunneling anomalies in near real time, which uses hourly flow data as the determination units. They separated the DNS data into requests, responses, and unknown query types. Then, these researchers analyzed the distribution of the packet length of the three query types and used the frequency of large packets for each type to make determinations. Ellens et al. (Ellens et al., 2013) extracted 5 features from the raw flow data or time-binned flow data and set up 5 flow-based detectors using the threshold method, Brodsky-Darkhovsky method and distribution-based method.

3. Method

3.1. Motivation

For the learning problems of sequence data, the RNN model could memorize the previous states and apply them to the output calculations. Compared with the structure of traditional neural networks whose nodes in the same layer are not connected, RNN models can utilize the context information of the sequence data. The LSTM model (Hochreiter and Schmidhuber, 1997) is based on the RNN model and adopts the LSTM unit to enhance the ability of the traditional RNN model. It is considered an effective method for some sequence detection problems (Greff et al., 2016). In recent years, the LSTM model has been widely used in speech (Graves et al., 2013), text (Senior et al., 2015), machine translation (Bahdanau et al., 2014), etc. In terms of network traffic detection, the LSTM model has also been used for payload detection (Liu et al., 2019b).

The FQDNs of the normal DNS packets could be regarded as a semantical sequence, while the FQDNs of DNS covert channel packets contain encoded covert data and are more similar to random sequences. In view of the good performance of the LSTM model in text classification, we use the LSTM model to classify the FQDNs of the DNS packets to detect DNS covert channels. In addition, because the LSTM model can accept the input of streaming data, it only needs simple data preprocessing and can achieve end-to-end real-time detection.

To improve the performance of the model, it is necessary to preprocess the FQDNs before inputting them into the LSTM model. Only the subdomain parts of the FQDNs can be used to encode covert data, so we only inspect the subdomain part of the FQDNs. In addition, the character cases and positions of the separator "." of the subdomain parts are mainly different between the FQDNs generated by different covert channel tools. Therefore, we also performed other preprocessing, such as converting the characters of subdomain parts into lowercase and removing the separator ".".

Legal DNS covert channel packets are essentially covert channel packets in real network traffic. Therefore, we regard them as covert channel samples when constructing our datasets. In addition, the FQDNs that belong to some cloud server or are used in DDOS attacks will also be detected by the LSTM model. For this circumstance, we design a grouped and whitelist filtering method based on the existing knowledge to filter the non-covert and legal DNS covert channel packets.

3.2. Model structure

We propose a DNS covert channel detection method using the LSTM model, as shown in Fig. 2. Our model includes three parts: data preprocessing, model determination and grouped filtering. The data preprocessing part extracts the FQDNs from the DNS packets and processes them into the strings needed. The details of the data preprocessing method will be introduced in Section 3.3.

In the determination model part, the processed strings are input into the LSTM model to make determinations. Our LSTM model contains one hidden layer. Because the FQDNs are generally short, the first 128 characters can largely guarantee that it contains the original information. Therefore, the number of hidden states of our LSTM model is set to 128. The state number corresponds to the length of the network, and the number of layers corresponds to the depth of the network. The FQDNs are first preprocessed utilizing the methods in Section 3.3 and then are converted into word vectors that will be input into the LSTM model. After the embedding layer of the model, the word vectors are input into the LSTM units one by one. Each LSTM unit accepts the current input and the state from its last LSTM unit and propagates the calculation result to its next LSTM unit. The state of the last LSTM unit is input into the softmax layer to obtain the prediction. The grouped filtering part uses the grouped filtering method and the whitelist filtering method, which is constructed based on the existing knowledge, to further filter the covert channel packets that were determined by the LSTM model. The aim of the filtering methods is to remove false positive packets. The details of the grouped filtering will be introduced in Section 3.4.

3.3. Data preprocessing

An FQDN can be defined as "hostname.subdomain.2LD.TLD", where the top-level domain (TLD) (Wikipedia, 2020) is managed by the root nameserver and the 2LD is usually registered by a company or a group below a certain TLD. Therefore, all FQDNs that have the same 2LD.TLD generally belong to the same company or group. The 2LD.TLDs referred to as domains. In addition, some TLDs named country code TLDs (ccTLDs) often have subdomains such as "com.cn", "net.uk", "gov.au", and "edu.co". Such 2LD.TLDs are not registered by a company or a group and are also regarded as TLDs. In this circumstance, these 2LD.TLDs are considered TLDs, and the 3rd level label

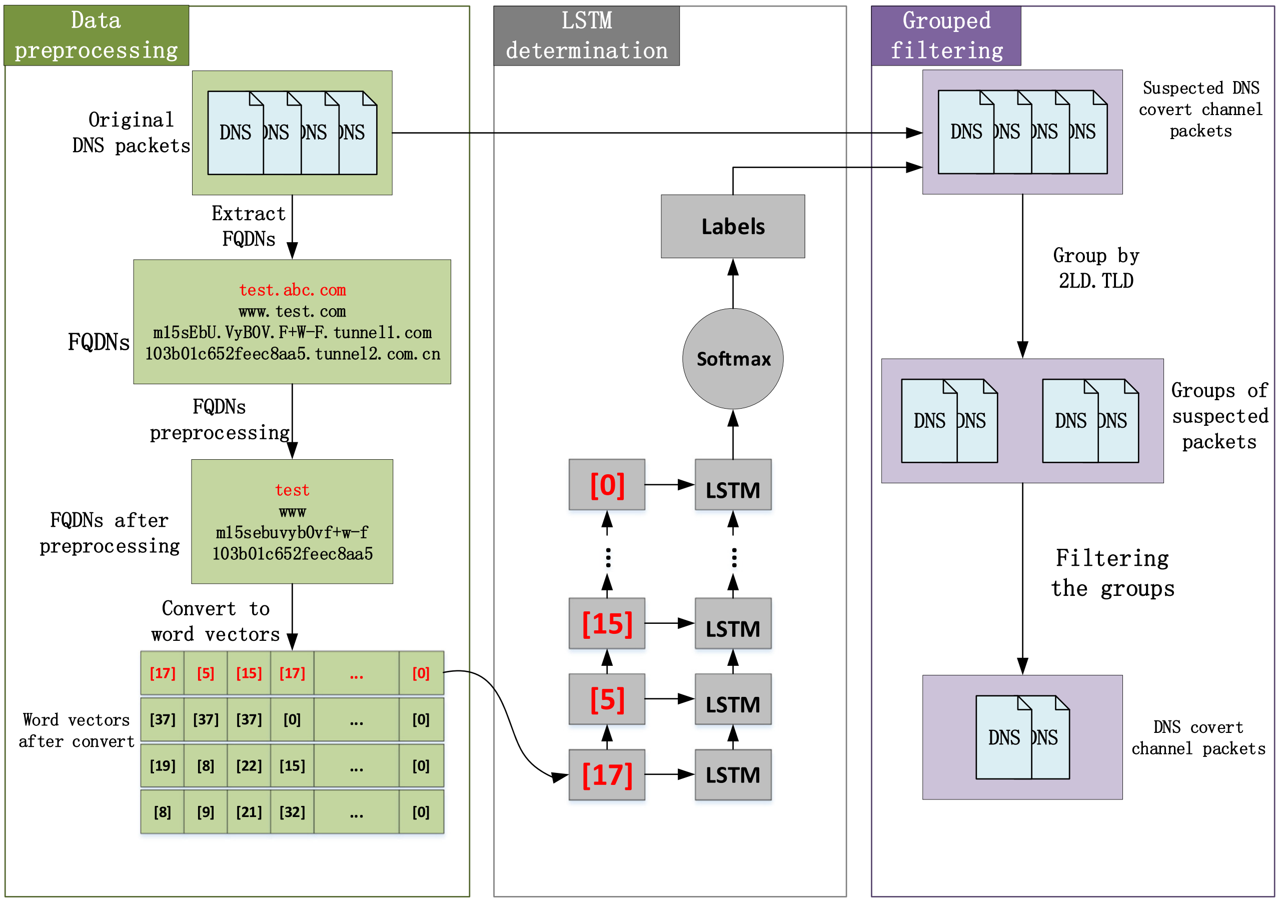

Fig. 2 – Structure of the DNS covert channel detection method.

of the FQDN is considered 2LD. The 2LD.TLDs that can be regarded as TLDs are listed in (Gavin, 2014). The remainder of the FQDNs except 2LD.TLD is ”hostname.subdomain”. Here, we call it the subdomain part, which can contain multilevel labels or can be empty. When preprocessing the FQDNs, the invisible characters in the FQDNs are represented by ”/” + hexadecimal, such as ”/8b” or ”/ac”.

According to the composition of FQDNs and the characteristics of the LSTM model, we design the following data preprocessing method, which contains four parts:

(1) **Remove the 2LD.TLDs from the FQDNs**. For the FQDN in a DNS packets, the 2LD.TLD only indicates which domain the FQDN belongs to. The convert channel cannot use 2LD.TLD to encode covert data. There is generally no useful information in this part. Removing this part of the FQDNs could reduce the interference of uncorrelated information and improve the quality of the original data. In addition, the duplications of 2LD.TLDs in some FQDNs may lead to the problem of a unification pattern. Thus, removing the 2LD.TLDs during the training phase could improve the generalization ability of the LSTM model.

(2) **Convert all uppercase characters to lowercase characters**. In real DNS traffic, the FQDNs of DNS packets are case-insensitive during resolution. There are many encoding methods (such as Base16, Base32, and Base64) in the covert channel FQDNs. The Base64 encoding method is case-sensitive. However, after the conversion to lowercase characters, the Base64-encoding FQDNs still retain their random-string-like characteristics. Because different covert channel tools use different character cases of FQDNs, converting characters into lowercase country code TLD can avoid unnecessary interference with the generalization ability of the LSTM model.

(3) **Remove the separators ”.” out of the FQNDs**. The maximum length of each label in the FQDN is 63. When encoding long covert data whose length is larger than 63 into the FQDNs, the encoded string needs to be separated by ”.” to form a legal FQDN. In addition, the separator ”.” is generally not used to encode data. Moreover, different covert channel tools have different separation methods. Therefore, removing the separator ”.” could better restore the original data contained in the covert channel FQDNs. For normal DNS FQDNs, removing the separator ”.” usually does not significantly change their characteristics.

(4) **Unify the length of strings**. After the aforementioned three steps, we obtain strings transformed from the FQDNs. We regard the strings as character sequences

Algorithm 1 Grouped filtering for false positive packets

```
Input: packets for all suspected DNS covert channel packets determined by the LSTM model;
    ts1 for threshold of number of packets containing the same FQDN;
    ts2 for threshold of number of FQDNs;
    ts3 for threshold of ratio of FQDNs queried by multiple IPs;
    whitelist for white 2LD.TLDs;
Output: Covert channel packets;
 1: groups = group_by_2LD_TLD (packets)
 2: for all group in groups do
 3:     sub_groups = group_by_FQDN (group)
 4:     // Step 1 Remove the duplicate queried FQDNs in each group.
 5:     for all sub_group in sub_groups do
 6:         if sub_group.pk_count () ts1) then
 7:             sub_groups.delete(sub_group)
 8:         end if
 9:     end for
10:     // Step 2 Remove the group which contains a small number of FQDNs.
11:     if sub_groups.count() ts2 then
12:         groups.delete(group)
13:     end if
14:     // Step 3 Remove the group in which the FQDN is queried by multiple IPs.
15:     count_more_than_1_IP = 0
16:     for all sub_group in sub_groups do
17:         if sub_group.has_more_than_1_IP () then
18:             count_more_than_1_IP += 1
19:         end if
20:     end for
21:     if count_more_than_1_IP / sub_groups.count() ts3 then
22:         groups.delete(group)
23:     end if
24:     // Step 4 Filter by using a whitelist.
25:     if group.2LD_TLD in wihte_list then
26:         groups.delete(group)
27:     end if
28: end for
29: return groups
```

and convert them into sequences of word vectors. The word vector converted from each character is one dimensional. For instance, the character ”t” is converted to word vector ”[17] (Zhang et al., 2019)” in Fig. 2. To ensure the consistency of sequence lengths, we unify the sequence length. According to the RFC specification of the DNS, the total length of an FQDN is less than 253. The FQDNs of normal DNS packets are usually shorter than those of covert channels. To retain more original information, the sequence length is fixed to 128 characters. For word vector sequences that are less than 128 in length, we pad the sequences using the fixed word vector of ”[0]”. For word vector sequences that are greater than 128 in length, we retain the first 128 vectors in the sequence.

3.4. *Grouped filtering method*

The grouped filtering method targets the DNS traffic, which is determined as the DNS covert channel by the LSTM model. This method aims to reduce the false positive rate of our model. Currently, we perform grouped filtering based on the traffic of one day. According to Kara et al. (Kara et al., 2014), the best grouping method should use DNS zones as the basis of grouping, in which all traffic of one covert channel server could be grouped together as a determination unit. However, compared with passive DNS data, in streaming DNS traffic, it is harder to find DNS zones. Therefore, our grouping method is performed according to 2LD.TLD. In this way, the DNS traffic is divided into *n* groups, and the false positive traffic is filtered based on thresholds. The algorithm of our grouped filtering method is shown in Algorithm 1.

Algorithm 1 mainly contains the following four steps:

(1) **Remove the duplicate queried FQDNs in each group.** The client of a covert channel encodes covert data into the subdomain part of an FQDN and transmits the data by querying the FQDN. Under normal conditions, each covert channel FQDN is unlikely to be queried repeatedly. In particular, to avoid the local DNS nameserver from catching the request packets, the FQDNs should not be duplicated. Therefore, FQDNs with multiple queries are generally not DNS covert channel FQDNs. Hence, we remove the FQDNs for which the

number of request and response packets are greater than *ts*1. According to our experience with real traffic analysis, we set *ts*1 to 1000.

(2) **Remove the group that contains a small number of FQDNs.** A DNS covert channel is used to establish communications between a client and a server; therefore, it usually does not send only a few packets. Our LSTM model might determine some normal FQDNs as covert channel FQDNs. Generally, these FQDNs could be scattered into multiple 2LD.TLDs, and we could remove these scattered FQDNs using this method. The threshold of the number of FQDNs in one group is set as *ts*2. If the number of FQDNs in one group is less than *ts*2, then the group will not be considered to contain DNS covert channel packets. According to our experience with real traffic analysis, *ts*2 is set to 5.

(3) **Remove the group in which the FQDN is queried by multiple IPs.** The client of a covert channel encodes the covert data into the subdomain part of the FQDNs for transmitting the data to the server. Hence, the covert channel FQDN cannot be queried by multiple IP addresses in a short period of time. Therefore, if the ratio of FQDNs queried by multiple IP addresses is higher than *ts*3, then the packets in this group are not considered DNS covert channel packets. According to our experience, *ts*3 is set to 50%.

(4) **Filter by using a whitelist**. We collected the 2LD.TLDs of the FQDNs of cloud servers and legal DNS covert channels to construct a whitelist, and the whitelist is used to filter the results obtained from the last step.

4. Experiments

4.1. *Datasets*

We obtain DNS covert channel packets by running 8 DNS covert channel tools: Iodine (Ekman, 2014), Dnscat2 (Ron, 2019), Dns2tcp (Dembour and Collignon, 2014), DNShell v1.7 (Dan, 2015), Ozymandns (Dan, 2004), Cobaltstrike (Strategic Cyber LLC, 2019), DNSExfiltrator (Arno0x0x. DNSExfiltrator, 2018), and DET (Qasim, 2018). We use these tools to simulate file transfer behaviors with DNS covert channels in an independent network environment and use Wireshark software to capture DNS traffic. The simulation environment mainly includes covert channel clients, local DNS nameservers, the root nameserver of ".com", and the covert channel server. Since all the FQDNs we obtained use the TLD ".com", the root nameserver is omitted. The two nameservers use the Centos 7.5 operating system. Due to the different requirements of running environments of the tools, the operating system of the covert channel client and server is changed accordingly. The client uses Ubuntu 16.04 or Windows 7, and the server uses Centos 7.5 or Windows 7. Samples of each software are collected with the two transmission directions: client to server and server to client. For Iodine, Dnscat2, Dns2tcp and Cobaltstrike, samples are collected separately by different resource record types. In addition, samples from Iodine are also collected separately using Base32, Base64 and Base128 encoding methods which are all encoding methods Iodine supports.

The benign samples of the non-covert channel mainly consist of DNS packets from real DNS traffic. We construct two kinds of datasets: the dataset of FQDNs and the dataset of the DNS packets. For the attack samples of each software, half of them are used to construct the dataset of FQDNs, and the other half are used to construct the dataset of the DNS packets.

4.1.1. *Dataset of FQDNs*

(1) **FQDN samples**

To ensure the diversity of the benign samples as much as possible, we collect all FQDNs that have resolved IPs. We group them according to their 2LD.TLDs and remove the groups of FQDNs containing long and random subdomains, such as suspected covert channel FQDNs and cloud server FQDNs. In addition, the 2LD.TLDs used for the IPv4 reverse resolution, such as ”in-addr.arpa”, usually have a large amount of distinct FQDNs, and the FQDNs have high similarity. To ensure the balance of benign samples, we only use subsets of these FQDNs and finally obtain a benign sample set of 3,000,000 FQDNs.

The sample set of covert channel FQDNs mainly contains two parts: FQDNs generated by covert channel tools and covet channel FQDNs from real traffic. We mix the FQDNs generated by the five tools, i.e., Iodine, Dnscat2, Dns2tcp, DNShell v1.7 and Ozymandns, to obtain a simulated attack set of 1,000,000 FQDNs. In addition, we find some FQDNs of the DNS covert channels in real traffic, including legal and malicious channels. We take out some FQDNs for the generalization test, and the remaining FQDNs are mixed to form an actual attack FQDN set. Here, both legal and malicious covert channel FQDNs are considered as covert channel samples when constructing our datasets.

(2) **Dataset Composition**

We randomly draw FQDN samples from the benign set, the simulated attack set and the actual attack set by 360,000, 240,000, and 120,000, respectively. After mixing the FQDNs, we randomly selected 300,000 samples as the training set, 120,000 samples as the validation set, and the other 300,000 samples as the test set.

In addition, we construct one generalization validation set and four generalization test sets from the remaining 3 tools and other samples. The generalization validation set contains 50,000 samples, and the four generalization test sets separately contain 40,000, 1,400,000, 40,000, and 10,000 FQDNs, which are from DNSExfiltrator, ExfiltrationAttackFQDNs (Ahmed et al., 2019), DET, and the domain ”sophosxl.net”, respectively.

The FQDNs in the first generalization test set are similar to some FQDNs in the training set. The FQDNs in the second and third groups are random but quite shorter, which may be harder to detect. The fourth channel consists of covert channel FQDNs from actual traffic. The ”sophosxl.net” is a legal covert channel domain that is used by Sophos LTD. (Sophos Ltd, 2018) We believe that the FQDNs of that domain should be detected by our model and then filtered by the

Table 1 – Compositions of the dataset of FQDNs.

Datasets	Name	Size	Composition
Training set of FQDNs	FQDN_Train	Total: 300K Attack: 150K Benign: 150K	FQDNs of 5 tools(100K), FQDNs of actual covert channels (50K), FQDNs of non-covert channel (150K)
Validation set of FQDNs	FQDN_Val	Total: 120K Attack: 60K Benign: 60K	FQDNs of 5 tools (40K) FQDNs of actual covert channels (20K), FQDNs of non-covert channel(60K)
Test set of FQDNs	FQDN_Test	Total: 300K Attack: 150K Benign: 150K	FQDNs of 5 tools (100K) FQDNs of actual covert channels (50K), FQDNs of non-covert channel (150K)
Generalization validation set	Gener_Val	Attack: 150K	FQDNs of Cobaltstrike (150K)
Generalization test set1	Gener_Test1	Attack: 40K	FQDNs of DNSExfiltrator (40K)
Generalization test set2	Gener_Test2	Attack: 1,400K	FQDNs of Exfiltration-AttackFQDNs (Ahmed et al., 2019) (1,400K)
Generalization test set3	Gener_Test3	Attack: 40K	FQDNs of DET (40K)
Generalization test set4	Gener_Test4	Attack: 10K	FQDNs of ”Sophosxl.net“ (10K)

Table 2 – Compositions of dataset of DNS packets.

Datasets	Name	Size	Composition
Test set of DNS packets	Packet_Test	Total: 126M Attack: 2M Benign: 124M	Benign: packets of the query FQNDs of 2LD.TLD in Alexa Top 10K Attack: packets of the 5 tools

whitelist, while Ahmed et al. (Ahmed et al., 2019) only used them as benign FQDNs to train the model.

The compositions of the dataset of FQDNs are shown in Table 1.

4.1.2. *Dataset of DNS packets*

The dataset of the DNS packets is used to test the complete detection effects of our method, including the LSTM model and the grouped filtering algorithm. We collect the DNS traffic of one day and filter them with the 2LD.TLD in Alexa Top 10K to construct the benign packet set. We then removed the FQDNs of the 2LD.TLDs of the legal DNS covert channels, such as ”mcafee.com”, and finally obtain 124,000,000 DNS packets that are used as the benign DNS packets.

We mix all of the traffic generated by Iodine, Dnscat2, Dns2tcp, DNShell v1.7 and Ozymandns and obtain 200,000 attack DNS packets. The details of the dataset are shown in Table 2.

4.2. *Indicators*

For the dataset of FQDNs, the positive and negative sample sizes are equal. For the dataset of the DNS packets, the positive and negative sample sizes are quite different. We consider using different indicators for the two datasets. For the dataset of FQDNs, we use the following indicators to evaluate the effect of the model.

Accuracy (*A*)**:** The ratio of the correctly classified samples to the total samples, which represents the confidence of the classification.

$$A = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

Precision (*P*)**:** The ratio of the true positives (*TPs*) to the sum of the *TPs* and the false positives (*FPs*), which represents the confidence of the classification.

$$P = \frac{TP}{TP + FP} \tag{2}$$

Recall (*R*)**:** The ratio of the *TPs* to the sum of the *TPs* and the false negatives (*FNs*), which represents the completeness of the classification.

$$R = \frac{TP}{TP + FN} \tag{3}$$

F-Measure (*F*)**:** The harmonic mean of the precision and recall, which represents the synthesis of the performance of the classification.

$$R = \frac{2 * P * R}{P + R} \tag{4}$$

For the dataset of the DNS packets, we consider using two indicators: recall and false positive rate, which is defined as the following to evaluate the effects of the model.

False-positive-rate (*FPR*)**:** The ratio of the false positives (*FPs*) to the sum of the *FPs* and the true negatives (*TNs*), which represents the probability of making mistakes.

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

Where *TP* represents the number of samples correctly classified as positive, *FN* represents the number of samples incorrectly classified as negative, *FP* represents the number of samples incorrectly classified as positive, and *TN* represents the true negatives that are the correctly classified negative samples.

4.3. Experimental settings

We conduct 3 groups of experiments in total.

(1) **Experiment 1** The experiment on the effects of the detection models. The experiment compares the effects of our LSTM model and other models that have good performance in single packet-based detection. The CNN is a model that has been widely used in malicious network traffic detection. For example, Liu et al. (Liu et al., 2019b) used a CNN in payload-based attack detection. Liu et al. (Liu et al., 2019a) and Zhang et al. (Zhang et al., 2019) used a CNN in single packet-based covert channel detection and obtained good results with respect to their test set; however, they did not pay additional attention to the generalization capability. Qi et al. (Qi et al., 2013) made determinations only by threshold. Ahmed et al. (Ahmed et al., 2019) utilized white FQDNs to train the isolation forest model and achieved a recall rate of more than 95% with a false positive rate of less than 2%, which realized the best performance at present.
Therefore, we compare our LSTM model with the CNN model and the Ahmed method (Ahmed et al., 2019) on the dataset of the FQDNs.
(2) **Experiment 2** The experiment on the effects of the data preprocessing methods.
The experiment compares the effects of the LSTM model before converting the characters into lowercase, after converting characters to lowercase, and after removing the separators ”.”.
(3) **Experiment 3** The experiment on the effects of the grouped filtering methods.
The experiment compares the effects before grouped filtering, after grouped filtering (excluding whitelist filtering), and after whitelist filtering on the dataset of the DNS packets.

4.4. Results

(1) **Experiments on the detection models**
We conduct three experiments on the dataset of the FQDNs, which are the LSTM model, CNN model and Ahmed method (Ahmed et al., 2019). The LSTM model iterates 20 epochs with batch size of 128, and obtains adopts *AdamOptimizer* as the optimizer.
In the experiment of the CNN model, the preprocessed FQDNs are converted into images according to the ASCII values of the characters and scaled to 128 dimensions according to the image scaling method. The CNN model contains 3 convolutional layers, 2 max-pooling layers, and 1 softmax layer. The CNN model iterates 20 epochs with batch size of 128, and adoptsobtains *AdagradOptimizer* as the optimizer.
Ahmed et al. (Ahmed et al., 2019) only used benign FQDNs to train the isolation forest model, and we could not obtain the source code and dataset that they used. Therefore, we need to reproduce the model and dataset as they mentioned. Because the training datasets that they used are quite different from ours (such as the different uses of ”sophosxl.net”), to better ensure the effect of their isolation forest model, we especially constructed a training dataset. According to the dataset construction method of the paper, we used the FQDNs of the 2LD.TLDs in the Alexa top 10K and the FQDNs of ”sophosxl.net” from real traffic to construct the benign dataset, which consists of 800,000 FQDNs. The benign dataset is used to train the isolation forest model presented in the paper. In our experiment, the Ahmed method (Ahmed et al., 2019) uses the best performance parameters as n_estimators=20, max_samples=25, and contamination=0.1.
The experimental results of each method are shown in Table 3.
From the comparison results of the three models, the generalization test result of the CNN is poor, and the result of the LSTM model is generally better than that of the CNN model and Ahmed method. From the effects on the four generalization test datasets, we find that all three methods perform well on Gener_Test1

Table 3 – Experimental results of the LSTM model, CNN model and Ahmed method (Ahmed et al., 2019).

Datasets	Indicators	LSTM model	CNN model	Ahmed method
FQDN_Test	Accuracy	99.39%	88.51%	90.60%
	Recall	99.62%	83.79%	83.67%
	Precision	99.15%	92.52%	97.14%
	F1	99.39%	87.94%	89.90%
Gener_Val	Recall	98.89%	92.16%	95.96%
Gener_Test1	Recall	100%	100%	100%
Gener_Test2	Recall	98.52%	5.13%	97.86%
Gener_Test3	Recall	99.32%	3.85%	99.52%
Gener_Test4	Recall	99.99%	100%	99.89%

Table 4 – Effect comparisons of each preprocessing method.

Indicators	No converting into lowercase	Converting into lowercase	Converting into lowercase and remove separators “.”
Accuracy on FQDN_Train	98.51%	99.79%	99.54%
Accuracy on FQDN_Val	98.49%	99.76%	99.50%
Recall on Gener_val	97.85%	96.85%	98.89%

Table 5 – Effect comparisons of the grouped filtering method.

Indicators	LSTM	LSTM + rouped filtering	LSTM + grouped filter-ing + whitelist filtering
False positive rate on test set	0.1797%	0.0285%	0.0032%

Table 6 – FQDN samples of the DNS covert channel detected in real traffic.

Type	2LD.TLD	FQDN samples
Malicious	0nlinecc.com	mllakfobmgfcfjihnemjlokdjiinhcghfmebdgclbabfakpoodnimnlckhjmibh.gglfaefdkcpbeajpnoc nhmmlbkgjliaifhkgpfeejdocdbianpbognlmalfk kjp.iehjgofdeidnccbham.dns.0nlinecc.com
Suspected	micorsoff.com	KIKNJCPCKDJLLMQLPDOINNMCLHKMJBIGHLGQGFFKEPDECJBOQCPHOMNB MGLLKQJ.FJKIPHEGJFOEDDICNBCQG PLOQNFMKMPLEKJJOIDHIGNFCEHDM.ns.micorsoff.com
	micrrsoft.net	onhdqipcphpbcpikejooheqjkpdfmkfqofhlbbkgdmmbfhomhcqhknddmifoodh.jqojedkmpffok hqqfjlcblgfmobhhqmjccilnednihoqdjjcp.ns. micrrsoft.net
	1xd9.com	6nbtunlbpv2b77onjowh7apiub.o2rq47laaglh7jmkqivca8musq.58.1xd9.com
	85h5.com	4n4b9c6rgtjon8r3nj6pa5958v.cwko47laa7sxjrhlr9vgvac48q.58.85h5.com
	jp6m.com	3eaoxtk2kfe4troa23luikq5wa.nagms7laa3pl9dits23nheveea.58.jp6m.com
Legal	mcafee.com	a-0.19-a7000001.d030081.16a8.ae1.3ea5.210.0.sj3nqhkpbl14tmm9lbjttr6vw6.avts.mcafee.com
	sophosxl.net	3.1o18sr00n75p8674p378q015r8ono2275059841rns3n29pnsp7s97224q0o062.56np0r741pn373 93648s275srp92rn3r86p31411p426rp3638303sn r0q8rqpn.r7174pqq28r7pn28843s976883qr63 45nsr568qos970o9576p7r980s.793r6q08581r9 9n5.i.00.s.sophosxl.net
	e5.sk	z64levapqyvebjuxt4olsrtyc4bqeaqbaeaq.a.e.e5.sk e2jcuaacaakmyricah5sp2gsaqaaa473e7unnhs4pbg5yricafpouai3aaaaan5.zq3hacaaoaafqai hzwmdxswrkbu5w6iik6kiovtrov52ogbi.a.j.e5.sk

Table 7 – FQDN samples of false positive.

2LD.TLD	FQDN samples
cloudfront.net	af5ae754c619f16c5933635460f6ad5db.profile.dfw50-c1 .cloudfront.net a4bbf78b123415d2ba7d1a62cb0a54e4a.profile.dfw53-c1 .cloudfront.net
qcloudcjgj.com	59d98278d7f0347f4a42e5e027ac7e51.qcloudcjgj.com dc4b3c5656c7a064a88403a11a510d52.qcloudcjgj.com
alicloudwaf.com	4khm6eer8drcylmpjukkvjhu99xqnnps.alicloudwaf.com zzygx0cgba1pvbdlvslse7fqwfjwlwnm.alicloudwaf.com

and Gener_Test4, in which the FQDNs are long and random. CNN model performs poorly on Gener_Test2 and Gener_Test3, which consist of FQDNs that are random but quite shorter. Therefore, the LSTM model and Ahmed method could better ensure the generalization ability than the CNN model, and the LSTM model performs the best.

(2) **Experiments on data preprocessing method**
The experiment is performed on the dataset of FQDNs. To prove the effectiveness of our preprocessing method, the effects of three data preprocessing methods are compared: no converting characters into lowercase, converting characters into lowercase, and removing separators".". In the experiment, we use the FQDN_Train set for training and the FQDN_Val set and the Gener_Val set to determine the effects. Each model is saved every 1,000 iterations, and the accuracy rate on the FQDN_Val set and the recall rate on the Gener_Val set are calculated. For each preprocessing method, the best model is selected, which reaches a higher accuracy rate on the FQDN_Val set and a higher recall rate on the Gener_Val set at the same time.
Table 4 shows the results of the best model of each preprocessing method.
The experimental results of the three preprocessing modes show that, with no converting characters into lowercase, the accuracy could only reach 98.51%; with converting characters into lowercase, the accuracy rates could reach up to 99.79%, while the generalization ability had a high recall rate of 96.85%. When further removing the separators ".", although the accuracy rates are slightly lower, the recall rate is significantly higher. Hence, the preprocessing method of converting characters into lowercase and removing the separators "." as a whole is effective.

(3) **Experiment on the grouped filtering method**
The grouped filtering method aims at decreasing the false positive rate of the model. The experiment is performed on the dataset of the DNS packets. We compare the false positive rate of the LSTM model with

the three filtering modes: no filtering, after grouped filtering without whitelist filtering, and after grouped filtering with whitelist filtering. Here, when testing, our whitelist only includes the five 2LD.TLDs: ”baidu.com”, ”cloudfront.net”, ”gstatic.com”, ”rackcdn.com”, and ”office.com”.

The experimental results are shown in Table 5. The false positive rates of the three filtering modes are compared. From the results, our grouped filtering method significantly reduces the false positive rate of the model.

4.5. *Results of real traffic detection*

In actual traffic detection, we have detected malicious DNS covert channel traffic using ”0nlinecc.com” (360 Active Defense, 2018) and suspected DNS covert channel traffic using the following 2LD.TLDs: ”micorsoff.com” and ”micrrsoft.net”. In addition, we also find some FQDNs of ”1xd9.com”, ”85h5.com”, ”jp6m.com”, etc. These covert channels found by our method are reviewed by experts. In addition, some legal DNS covert channel traffic is found, such as covert channels based on ”mcafee.com” (using ”avts.mcafee.com” and ”avqs.mcafee.com”), ”sophosxl.net”, and ”e5.sk”. These covert channels are mentioned in Section 1. Some FQDN samples of these 2LD.TLDs are shown in Table 6.

There are some false positive samples of the LSTM model, which mainly include the FQDNs of some fixed 2LD.TLDs. To determine the real purpose of these FQDNs, we use multiple methods to obtain information of these 2LD.TLDs, such as searching from Virustotal (Virus Total, 2012), looking for official websites, and referring to other studies. We add these 2LD.TLDs and 2LD.TLDs used by legal covert channels to our whitelist. There are some 2LD.TLD samples of false positive: ”cloudfront.net” (Amazon Web Services Inc. Amazon CloudFront, 2020) (CDN of Amazon Web Services), ”qcloudcjgj.com” (Tencent Cloud, 2019) (CNAME use of Tencent Cloud), ”alicloudwaf.com” (Alibaba Cloud, 2019) (CNAME use of Alibaba Cloud), etc. Some FQDN samples of these FQDNs are shown in Table 7.

In the process of running our methods on real traffic, we tentatively employ the idea of active learning to further improve the performance of the LSTM model. Malicious and legal DNS covert channel traffic FQDNs are selected and labeled and then are added to the dataset of the LSTM model.

5. Conclusions

In this paper, a DNS covert channel detection method using the LSTM model is proposed. The method makes determinations by inspecting the FQDNs of DNS packets and introduces grouped filtering algorithms to further reduce the false positive rate. Our method contains the simple yet effective preprocessing of original FQDNs without feature engineering and supports the end-to-end detection of FQNDs. The grouped and whitelist filtering methods could directly utilize the existing knowledge to filter the detection results of the LSTM model. We construct an independent network environment and use multiple DNS covert channel tools to construct the datasets. In addition, we pay much more attention to enhancing the generalization ability of our method and specifically construct multiple generalization test datasets. Our method reaches an accuracy rate of 99.38% and recall rates over 98.5% on several generalization validation sets. Additionally, our method detects multiple events of malicious DNS covert channels in real traffic.

For future work, we would like to optimize the model structure and the loss function to further improve its generalization ability. In addition, active learning methods can be used to find additional typical benign and malicious FQDNs from actual networks to optimize the composition of our datasets.

Declaration of Competing Interest

The authors declare that no conflict of interest exits in the submission of this manuscript, and manuscript is approved by all authors for publication. And the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

REFERENCES

360 Active Defense., 2018. https://bbs.kafan.cn/thread-2131704-1-1.html.

Ahmed J, Gharakheili HH, Raza Q, Russell C, Sivaraman V. Real-time detection of DNS exfiltration and tunneling from enterprise networks. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE; 2019. p. 649–53.

Aiello M, Mongelli M, Cambiaso E, Papaleo G. Profiling DNS tunneling attacks with PCA and mutual information. Log. J. IGPL 2016;24:957–70.

Aiello M, Mongelli M, Papaleo G. Basic classifiers for DNS tunneling detection. In: 2013 IEEE Symposium on Computers and Communications (ISCC). IEEE; 2013. p. 000880–5.

Aiello M, Mongelli M, Papaleo G. Supervised learning approaches with majority voting for DNS tunneling detection. In: International Joint Conference SOCO’14-CISIS’14-ICEUTE’14. Springer; 2014. p. 463–72.

Aiello M, Mongelli M, Papaleo G. DNS tunneling detection through statistical fingerprints of protocol messages and machine learning. Int. J. Commun. Syst. 2015;28:1987–2002.

Alexa Web Information Company. Topsites, 2020. https://www.alexa.com/topsites.

Alibaba Cloud., 2019. https://www.alibabacloud.com/help/zh/doc-detail/45267.htm.

Amazon Web Services Inc., Amazon CloudFront, 2020. https://aws.amazon.com/cloudfront/.

Arno0x0x., DNSExfiltrator, 2018. https://github.com/Arno0x/DNSExfiltrator.

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

Binsalleeh H, Kara AM, Youssef A, Debbabi M. Characterization of covert channels in DNS. In: 2014 6th International Conference on New Technologies, Mobility and Security (NTMS). IEEE; 2014. p. 1–5.

Born K, Gustafson D. NgViz: detecting DNS tunnels through n-gram visualization and quantitative analysis. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research; 2010a. p. 1–4.

Born, K., Gustafson, D. Detecting dns tunnels using character frequency analysis. arXiv preprint arXiv:1004.4358 2010b.

Buczak AL, Hanke PA, Cancro GJ, Toma MK, Watkins LA, Chavis JS. Detection of tunnels in PCAP data by random forests. In: Proceedings of the 11th Annual Cyber and Information Security Research Conference; 2016. p. 1–4.

Cambiaso E, Aiello M, Mongelli M, Papaleo G. Feature transformation and Mutual Information for DNS tunneling analysis. In: 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE; 2016. p. 957–9.

Dan B., DNShell v1.7, 2015. https://github.com/ahhh/Reverse_DNS_Shell.

Dan K. Ozymandns; 2004. https://dankaminsky.com/2004/07/29/51/.

Das A, Shen MY, Shashanka M, Wang J. Detection of Exfiltration and Tunneling over DNS. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE; 2017. p. 737–42.

Dembour, O., Collignon, N. Dns2tcp tool, 2014. http://www.hsc.fr/ressources/outils/dns2tcp/.

Ekman E. Iodine; 2014. https://code.kryo.se/iodine/.

Ellens W, Z˙ uraniewski P, Sperotto A, Schotanus H, Mandjes M, Meeuwissen E. Flow-based detection of DNS tunnels. In: IFIP International Conference on Autonomous Infrastructure, Management and Security. Springer; 2013. p. 124–35.

Engelstad P, Feng B, van Do T. Detection of DNS tunneling in mobile networks using machine learning. In: International Conference on Information Science and Applications. Springer; 2017. p. 221–30.

ESET, spol.s r.o. Ports and addresses required to use your ESET product with a third-party firewall, 2019. https://support.eset.com/kb332/.

Gavin, M. Second-level-domains, 2014. https://github.com/gavingmiller/second-level-domains/blob/master/SLDs.csv.

Graves A, Jaitly N, Mohamed AR. Hybrid speech recognition with deep bidirectional LSTM. In: 2013 IEEE workshop on automatic speech recognition and understanding. IEEE; 2013. p. 273–8.

Greff K, Srivastava RK, Koutn´ık J, Steunebrink BR, Schmidhuber J. LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. 2016;28:2222–32.

Griffioen H, Doerr C. Taxonomy and adversarial strategies of random subdomain attacks. In: 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS). IEEE; 2019. p. 1–5.

Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9:1735–80.

Homem, I., Papapetrou, P. Harnessing predictive models for assisting network forensic investigations of DNS tunnels 2017.

Homem, I., Papapetrou, P., Dosis, S. Entropy-based prediction of network protocols in the forensic analysis of DNS tunnels. arXiv preprint arXiv:1709.06363 2017.

Kara AM, Binsalleeh H, Mannan M, Youssef A, Debbabi M. Detection of malicious payload distribution channels in DNS. In: 2014 IEEE International Conference on Communications (ICC). IEEE; 2014. p. 853–8.

Karasaridis A, Meier-Hellstern K, Hoein D. In: Global Telecommunications Conference, 2006. GLOBECOM’06. Detection of DNS anomalies using flow data analysis; 2006.

Kim, Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 2014.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. NIPS (Vol. 25). Curran Associates Inc.

Liu C, Dai L, Cui W, Lin T. A byte-level CNN method to detect DNS tunnels. In: 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). IEEE; 2019a. p. 1–8.

Liu H, Lang B, Liu M, Yan H. CNN and RNN based payload classification methods for attack detection. Knowl.-Based Syst. 2019b;163:332–41.

Liu J, Li S, Zhang Y, Xiao J, Chang P, Peng C. Detecting DNS tunnel through binary-classification based on behavior features. In: 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE; 2017. p. 339–46.

McAfee LLC. FAQs for Global Threat Intelligence File Reputation, 2019. https://kc.mcafee.com/corporate/index?page=content&id=KB53735.

Mockapetris, P.V. RFC1034: domain names-concepts and facilities, 1987.

Nadler A, Aminov A, Shabtai A. Detection of malicious and low throughput data exfiltration over the DNS protocol. Comput. Secur. 2019;80:36–53.

Pan L, Chen J, Hu A, Yuchi X. EPT: EDNS privacy tunnel for DNS. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer; 2019. p. 50–62.

Pan L, Yuchi X, Wang J, Hu A. A public key based EDNS privacy tunnel for DNS. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (Trust- Com/BigDataSE). IEEE; 2018. p. 1722–4.

Qasim, R. DET (extensible) data exfiltration toolkit, 2018. https://github.com/qasimraz/DET.

Qi C, Chen X, Xu C, Shi J, Liu P. A bigram based real time DNS tunnel detection approach. Proc. Comput. Sci. 2013;17:852–60.

Rascagneres P., New FrameworkPOS variant exfiltrates data via DNS requests 2016. https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests.

Ron B., Dnscat2, 2019. https://github.com/iagox86/dnscat2.

Senior A, Sak H, Shafran I. Context dependent phone models for LSTM RNN acoustic modelling. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2015. p. 4585–9.

Sheridan S, Keane A. Detection of dns based covert channels. In: European Conference on Cyber Warfare and Security. Academic Conferences International Limited; 2015. p. 267.

Sophos Ltd. Information on the Sophos Extensible List, 2018. https://community.sophos.com/kb/117936.

Steadman J, Scott-Hayward S. DNSxD: detecting data exfiltration over DNS. In: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE; 2018. p. 1–6.

Strategic Cyber LLC. Cobalt strike, 2019. https://www.cobaltstrike.com/.

Tencent Cloud., 2019. https://cloud.tencent.com/document/product/627/18633.

van Leijenhorst, T., Chin, K.W., Lowe, D. On the viability and performance of DNS tunneling 2008.

Virus Total. Virustotal-free online virus, malware and url scanner, 2012. https://www.virustotal.com.

Wikipedia. Top-level domain, 2020. https://en.wikipedia.org/wiki/Top-level_domain.

Zhang J, Yang L, Yu S, Ma J. A DNS tunneling detection method based on deep learning models to prevent data exfiltration. In: International Conference on Network and System Security. Springer; 2019. p. 520–35.

Shaojie Chen received the bachelor’s degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2016. And he is now a doctoral student of the School of Computer Science and Engineering, Beihang University. His research interests include big data analysis, deep learning and network intrusion detection.

Bo Lang received the Ph.D. degree from Beihang University (BUAA), Beijing, China, 2004. She has been a visiting scholar at Argonne

National Lab/University of Chicago for one year. She is a Professor in School of Computer Science and Engineering, Beihang University (BUAA). Her current research interests include big data analytics and information security. As a principle investigator or core member, she has engaged in many national research projects founded by the National Natural Science Foundation, National High Technology Research and Development (863) Program, etc.

Hongyu Liu received his bachelor's degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2015. And he is now a doctoral student of the School of Computer Science and Engineering, Beihang University. His research interests include machine learning, deep learning, intrusion detection and network security.

Duokun Li received the bachelor's degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2018. And he is now a postgraduate student of the School of Computer Science and Engineering, Beihang University. His research interests include machine learning, network intrusion detection and malicious traffic detection.

Chuan Gao, Engineer, he is current a researcher in National Computer Network Emergency Response Technical Team/ Coordination Center of China(CNCERT/CC). He received his BS in Computer Science from Northwestern Polytechnical University in 2013, and MS in Computer Science from Beihang University in 2017. His research interests mainly include computer networks, cyber security, Reverse engineering, Artificial Intelligence and so on.