Identification of DNS Covert Channel Based on Stacking Method

Peng Yang[1*], Xinxin Wan[1], Guang Shi[1], Hao Qu[2*], Juan Li[3], Lixin Yang[4]

[1] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China.
[2] Heilongjiang Branch of National Computer Network Emergency Response Technical Team/Coordination Center of China Harbin, China.
[3] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 150000, China.
[4] Heilongjiang Preschool Education College Mudanjiang 157000, China.

* Corresponding author. Tel.: 010-82546673, 0459-46675222; email: shiguang@cert.org.cn, 377740292@qq.com
Manuscript submitted July 10, 2020; accepted October 20, 2020.
doi: 10.17706/ijcce.2021.10.2.37-51

Abstract: A covert channel is an information channel which is used by computer process to exfiltrate data through bypassing security policies. The domain name system (DNS) protocol is one of the important ways to implement a covert channel. DNS covert channels are easily used by attackers for malicious purposes. Therefore, an effective detection of the DNS covert channels is significant for computer system and network security. Aiming at the difficulty of the DNS covert channel identification, we propose a DNS covert channel detection method based on stacking model. The stacking model is evaluated in a campus network and the experimental results show that the detection based on the stacking model can detect the DNS covert channels effectively. Besides, it can also identify unknown covert channel traffic. The area under the curve (AUC) of the proposed method, reaching 0.9901, outperforms the existed methods.

Key words: Covert channel, DNS, stacking model.

1. Introduction

The domain name system (DNS) is the infrastructure and the fundamental resource of the modern Internet, the main function of which is mapping domain names to IP addresses. However, the potential vulnerability of the data exfiltration of DNS is ignored by most of users, because the DNS protocol is widely used and most of firewalls do not inspect DNS packets. Therefore, DNS and the DNS protocol are becoming the targets of attackers or abused by cybercriminals. Now, many applications leverage the DNS protocol to implement the stealth tunnels and most DNS covert channels are used in various cyberattacks, including disclosure of sensitive information and enabling stealth tunnels for botnet commands. Although the security devices have the strict access control rules, they usually allow the DNS traffic to pass though, which is the condition of the malicious communication based on the DNS protocol.

Many attackers now use DNS covert channels to attack information systems, steal key information, and damage the integrity and confidentiality of the data. In response to this threat, many different types of DNS tunnel detection methods have been proposed. At present, the detections of DNS covert communication behavior mainly include the analysis of network traffic and the analysis based on domain name strings. The

analysis of network traffic is generally the deep packet inspection of DNS packets to profile the attributions of the covert tunnels. Crotti [1] and Dusi *et al*. [2] proposed the classification methods based on the sequence of packets arrival and the size of packets respectively, which used to detect the data exfiltration hidden in the DNS tunnel, HTTP tunnel and SSH tunnel. Casas *et al*. [3] identified the stealthy DNS communication though calculating the amount of data in the communication and the size of single DNS packet. In the process of analyzing passive DNS traffic data, Marchal *et al*. [4] applied machine learning methods to DNS channel detection, extracted the attributes such as packet length and bytes, and specifically analyzed several tools of the DNS channel. Karasaridis *et al*. [5] studied the malicious behaviors of DNS-related covert channels, calculated the distribution of DNS packet sizes, and identified the packets of DNS covert channels based on the statistical attributes such as the distribution difference and the cross-distribution entropy. Sheridan and Keane [6] set up a covert channel experimental environment to collect the traffic of the active DNS covert channels, extracted fingerprints and implemented matching calculations during the detection phase. Shafieian *et al*. [7] leveraged the ensemble learning techniques which combine the multiple machine learning algorithms to improve the accuracy and robustness of the classifier. These methods are mainly aimed to the covert channels that transmit the large volume of data, which are difficult to detect the DNS covert channels with the small volume. Nussbaum *et al*. [8] and Aiello *et al*. [9] studied on the capability of data exfiltration of the covert channels, analyzing the number of host names related to the domain name, the locations of the domain name, and the history of the domain name. They found that the packet sizes of many malware receive commands and exfiltrate sensitive information through the DNS covert channels are similar to the packet sizes of the benign DNS traffic. Nadler *et al*. [10] focused on the detection of malicious behavior of the low-throughput data leakage based on the DNS protocol, and used the Isolated Forest technology to detect data leakage behavior based on DNS channel. It can successfully identify the high-throughput DNS channels and the low-throughput data leakage behaviors on large-scale data sets.

In terms of the domain name string analysis based methods, combine the payload of the DNS packet and the domain name characteristics of the covert channel to determine whether the DNS request and response packets are abused for the malicious purposes. Farnham [11] used the regular expressions to analyze domain name strings in network traffic, and found the behavior of the covert channels in the network. Based on this method, a practical and commercial covert channel detection system was established and extracted the features of DNS packets that were compared with the preset thresholds. Bilge *et al*. [12] analyzed the characteristics of the benign domain names and the domain names used for DNS covert channels, and calculated the proportion of the longest meaningful string (LMS). The CUSUM algorithm was applied to count the characters distribution of the segments of the domain name. Both of the two methods are to analyze the string characteristics of the domain names used by the covert channels. It is believed that the benign domain names for the practical applications usually consist of the meaningful words. Thus, the mentioned methods mainly detect the DNS covert channels with the special domain name strings, which ignore the DNS covert channels simulating the behavior of the benign domain names.

Based on the above analysis, the current traditional DNS covert channel identification methods have some shortages. Recently, stacking technique in ensemble learning have good performance in the different field, which shows the powerful ability to automatically solve problems. We think that the same type of DNS covert channel has similar behavior patterns in terms of network communication, and the network behavior patterns are difficult to change under the premise of achieving a special purpose. Therefore, we propose a method of DNS covert channel detection based on the stacking model. We analyzed the characteristics of the DNS covert communication traffic and extracted the features of the DNS covert channel packets, which can help to distinguish the traffic of the covert channels from the legitimate traffic.

The experiments were carried out in the real-world environment of the campus network and the stacking model achieved the ideal performance.

The remainder of the paper is arranged as follows: In Section 2, the features of the DNS covert channels and the patterns of communication behaviors are analyzed. The details of the method will be elaborated specifically in Section 3 while the experimental process and results will be presented in Section 4. Finally, we summarize the paper in Section 5.

2. Feature Selection

2.1. DNS Convert Channel

The behavior of the DNS covert channel is generally implemented by two methods: one is to establish a connection with a specific server during the domain name resolution process, which is called a standard DNS covert channel. The other is that the client directly establishes the connection with the server of the covert channel, which is called a non-standard DNS covert channel. For a standard DNS covert channel, an attacker needs a completely controlled DNS resolution server and a registered domain name. The resolution server is must set to the authoritative server of the domain name, then, which can be used as the sever of covert channel. When a client of the covert channel sends a request, which contains a subdomain name under the controlled domain name, to any DNS recursive server. The information hidden in the subdomains is sent to the controlled authoritative server through the standard domain name resolution process of DNS. The fundamental premise of the successfully implementing the non-standard DNS covert channel is that the client of the DNS covert channel can communicate with any DNS server. The attacker binds the tunnel service encapsulated by UDP protocol to port 53 of the server and can directly establishes a connection from the client to the server.

2.2. Features of Deep Packet Inspection

After the analysis of the implementation methods and processes of DNS covert channels, we consider the characteristics of the bidirectional packets of the DNS covert channels, as well as the problems that may exist in the deep packet inspection and the characteristics of network packets parsing processes. We try to construct a feature set to identify the DNS covert channel.

The response packets of the standard DNS covert channel contain the most of response data in the resource records, which makes the sum of the data length of the resource record greatly different with the same statistics in the normal response packets. It also influences the length of the entire packets. Therefore, we take the sum of the lengths of resource records in the answer sections and the sum of the lengths of the total resource records as the classification features. At the same time, we also take the length of the DNS request packet and DNS response packet as classification features. In addition, some DNS covert channels use the uncommon record types (such as TXT record) for the data transmission. Therefore, the number of the unusual record types used in the DNS packets can be checked as a feature.

We also noted the characteristics in the process of the client issuing the domain name requests. It is found that there is a large difference between the QNAME field of the DNS covert channel packets and the normal DNS packets. The fields except QNAME in the DNS question section have the limited data space, according the DNS protocol. Thus, the QNAME field contains most of the data exfiltrated through the covert channel. We calculated the number of the labels of the domain name (the label is the string separated by dots, and for example, the ‘aaaaaaa.com’ has two labels, ‘aaaaaaa’ and ‘com’) and the label length of the second-level domain. Therefore, the number of the labels in the domain name and the length of the second-level domain are used as classification features.

The DNS protocol is not designed to transmit the large volume data, so improving the transmission

efficiency is especially important. In [4], the work presents that an effective way to improve transmission efficiency is to use the binary data in the domain names. Based on the survey of the different DNS covert channel tools, most of the tools adopt the unusual characters. The DNS protocol specifies that the domain name needs to be encoded by the Base32 algorithm. The proportion between the valid amount of information expressed by the binary string itself and the string encoded by Base32 is about 5:3. Therefore, the amount of the binary data carried by the subdomain and the amount of the data stored by the subdomain are two important characteristics. For example, in the experimental environment, the domain name in the communication between DNSCat2 and the command and control server is '3f29016955018bd5b7.malwareserver.com'. It has 3 domain name labels, the length of the secondary domain name label is 13, and the binary data volume of the subdomain name is 9 bytes.

In addition, in order to improve transmission efficiency, it is necessary to occupy all the remaining space in the DNS packet or Raw UDP packet as much as possible. This make the lengths of the packets of the DNS covert channel are different from the benign DNS packets. However, the packets of the Raw UDP tunnel cannot be regarded as the DNS packets, so the length of the UDP payload is additionally considered as a feature.

The benign domain names usually use the words, other meaningful strings or the strings which is easy to remember, while the domain names of the DNS channels are often cluttered and incomprehensible. Therefore, there is a big difference in the character composition between the domain name of covert channel and the benign domain name, that is, the character is selected randomly, which has great randomness, and we use entropy to express the randomness of domain names. Besides, the random string has a high entropy, while normal string has low entropy. Therefore, the entropy of the domain name string is also one of the important factors to identify the DNS covert channels.

When the client or server parses legitimate network packets according the DNS protocol, there is no exception and the resolution is exactly complete. The resolution of the DNS covert channel packet is different. It is more likely that the program will throw up the exceptions and the packets cannot be completely resolved (referring to the absence of data injection). Therefore, the status of the packet resolution and the volume of the injected data should be recorded as classification features. The volume of the injected data is the distance between the end position of the DNS packet payload and the end position of the UDP packet payload.

After the analysis of network packets, we finally extracted in total 16 features of the DNS packets to identify the DNS covert channel traffic, as shown in Table 1.

Table 1. Features of Deep Packet Inspection

No	Name	Comment
1	isExistCname	Whether the response packet has CNAME records.
2	SumOfRDlengthOfResponse	The sum of the lengths of resource records in the answer sections.
3	totalLengthOfRD	The sum of the lengths of the total resource records.
4	NumOfUnusualRecords	The number of the unusual record types.
5	NumDomainLabel	The number of the labels in the domain name.
6	LenOf2LevelDomain	The length of the second-level domain.
7	AmountOfBinDataInSubdomain	The amount of the binary data in the subdomain.
8	AmountOfDataInSubdomain	The amount of the data stored in the subdomain.
9	EntropyOfDomainStr	The entropy of the domain string.
10	isResolutionException	Whether the packet can be successfully parsed.
11	AmountOfInjectedData	The amount of the injected data.
12	LengOfDNSRequest	The length of the DNS request packet.
13	LenOfDNSResponse	The length of the DNS response packet.
14	LenOfUDPPayload	The length of the UDP payload.

Specifically, the isExistCname refers to whether the response packet has CNAME records, which has only two values of 0 or 1. The SumOfRDlengthOfResponse refers to the sum of the lengths of resource records in the answer sections, and we need to count the number and length of resource records only in the answer sections. The totalLengthOfRD refers to the sum of the lengths of the total resource records, unlike the previous one, we need to count the number and length of all resource records in the packet, not just the answer sections. The NumOfUnusualRecords refers to the number of the unusual record types, and the more likely it is that DNS covert channels are related. The NumDomainLabel refers to the number of the labels in the domain name and the LenOf2LevelDomain refers to the length of the second-level domain, these two features reflect the relevance of domain names and the hidden undetected information. The AmountOfBinDataInSubdomain refers to the amount of the binary data in the subdomain and the AmountOfDataInSubdomain refers to the amount of the data stored in the subdomain, these two features reflect the encoding method and the amount of information contained in the domain name, and have a great relationship with the DNS covert channel. The EntropyOfDomainStr refers to the entropy of the domain string, which reflects the randomness of the domain string and the value indicates whether it is a normal domain name or a domain name of covert channel. The isResolutionException refers to whether the packet can be successfully parsed, which has only two values of 0 or 1. The AmountOfInjectedData refers to the amount of the injected data, the LengOfDNSRequest refers to the length of the DNS request packet., the LenOfDNSResponse refers to the length of the DNS response packet, and the LenOfUDPPayload refers to the length of the UDP payload. DNS covert channel needs additional information in the original packet, so some properties of the original packet will be changed. Therefore, the above four characteristics have a greater relationship with the DNS convert channel.

2.3. Features of the Communication Behaviors

2.3.1. Related definitions

In order to clearly describe the features, we make some definitions which is related to the features.

Definition 1 (sourIP, subdomain). If the DNS packet is legit and can be completely resolved, we define this communication session as (sourIP, subdomain), where the sourIP is the source IP address of the DNS request packet or the destination IP address of the DNS response packet, the subdomain is the substring of the domain names which remove the same domain suffix.

Definition 2 (sourIP, destIP). For the invalid DNS packet when the resolution process throws out the exceptions, we define the communication session as (sourIP, destIP), where the sourIP is the source IP address of the DNS request packet, the destIP is the destination IP address of the DNS request packet. If the DNS packet is captured at the gateway, we regard the IP address of the host within the local area network as the sourIP.

2.3.2. Communication features

The process of the DNS communication is continuous. Only the features of the DNS packets cannot meet the requirements of detecting the DNS covert channels. Therefore, we try to extract the features of the communication behavior. We sequentially concatenated the DNS packets which belong to the same communication sessions, and then calculated the statistics of the communication sessions.

We selected the communication features based on the three different aspect observations. (1) The actual situation of capturing the packets. (2) The experimental observations of DNS channel tools. (3) The similarity between the domain names of the DNS channel and the domain names generated by the domain generate algorithm (DGA).

Normally, an application uses the DNS protocol to get the address of the server IP address for the following actions. For instance, DNS queries before issuing HTTP requests. Therefore, DNS requests are

related to other types of requests. However, there are only DNS requests in the DNS channel, which we call the isolated DNS requests. Therefore, the number of requests isolated in DNS requests can distinguish DNS covert channels. In this paper, we consider the number of the isolated DNS requests in the specific time window as an indicator of the DNS channel.

Because the domain name channel mainly depends on a certain server for communication, the client of the DNS channel will send the numerous DNS requests of the specific domain. It results in that the number of host names who have ever sent the requests of the specific domain name is much more than the number of the clients who have sent the requests of the legitimate domain names in the same period. Therefore, the domain name with many host names is likely to be the domain name of the DNS channel [12], and we consider the number of client host names of each name as a feature.

In order to prevent malicious domain names from being detected, domain generation algorithm is usually used to generate a large number of domain names, and only a small number of registered domains are selected. Therefore, most of algorithmically generated domains (AGD) do not exist in the network. Therefore, AGDs are always have many non-existent domain responses (NXDomain) [13]. We notice that the domain name of DNS channel has great randomness in character composition, and has certain regularity in the length of domain name. It is similar to AGDS in string entropy and domain name length. It will also generate a lot of NXDomain responses (such as: Heyoka). Thus, these characteristics can be used for detecting DNS channels.

Table 2. All the Selected Features

Set	Features
S1	1. The total number of the DNS packets sent by the client. 2. The total number of the DNS packets received by the client.
S2	1. The number of the packets which exist CNAME, in the same communication session. 2. The maximum, minimum and mean of each feature of the packets in the same communication session (features shown in Table 1).
S3	1. The number of the isolated DNS request packets. 2. The number of the client host names of each domain name. 3. The number of the NXDomains.

All the features we selected are summarized in Table 2 and we seperated them into 3 sets. S1 describes the volume of the communication traffic between the client and the server. S2 is the set which includes all the features shown in Table 1 and the features of S2 are extracted through the deep packet inspection. In S2, we calculated the maximum, minimum and mean of each feature of the packets in the same session except the isExistCname, whereas we counted the number of the packets which have the CNAME. The characteristics of the communication behaviors of the DNS covert channel are gathered in S3. In total, 45 features are selected to represent a communication session.

3. DNS Convert Channel Detection

After the analysis of the work mechanism of DNS covert channels, we extracted the features of the communication traffic. In this section, we first present the overview of the detection architecture, which is shown in Fig. 1. Then how the modules of the architecture cooperate and together detect the DNS covert channels will be described.

3.1. Overview

The architecture of detecting DNS covert channel consists of three modules, data preprocess module, model training module and prediction module. Here we depict how they cooperate for the detection

respectively.

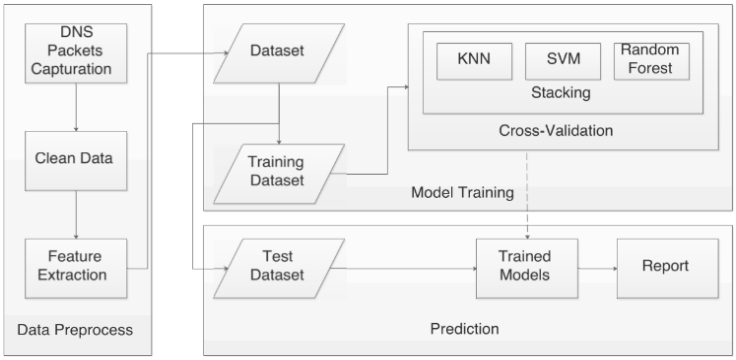

Fig. 1. The Overview of the architecture.

In the beginning, we have to collect the DNS traffic, including the benign traffic and the malicious traffic, for training models and evaluation. In order to simulate the real network environment, we collect the benign DNS traffic from a large Internet Services Provider (ISP). As for the traffic of the covert channels, we ran the DNS covert tools in the simulated environment and captured traffic generated by the tools. We considered that the DNS traffic collected from the ISP may include the suspicious traffic, so data cleaning is necessary. Leveraging the Alexa list of the top 1 million websites, we excluded the domains which ranked after 5000. Then, we transformed the raw data into the tabular data through extracting the features which are selected to represent communication sessions.

The dataset is split into the training dataset and the test dataset for the downstream tasks. We proposed a stacked model consisting of three different machine learning algorithms, the k nearest neighbors (KNN), the linear support vector machine (SVM) and the random forest. We try to capture more different details using the different techniques including the linear classification and the non-linear classification. Then, the prediction of each base model will be ensembled using the stacking technique. The models were trained using the 5-fold Cross Validation.

Finally, the trained model will be evaluated on the test dataset and the experimental results show the extraordinary performance of detecting the DNS covert channel with a low false positive rate.

3.2. Data Preprocess

In order to save the space of storing the raw traffic data, actually, we just logged the necessary fields of each DNS packet such as the timestamp, the IP address and the domain name. We downloaded the Alexa list of the top 1 million website and chose the top 50000 domains as the white list to filter the raw data. It helped us reduced the about 35% amount of traffic which need to be analyzed.

For the traffic of the covert channels, we collected five sorts of the DNS covert channel tools, to simulate the communication situation in the controlled environment.

We removed the domains which have only been queried once in the specific epoch (donate as E) for both the benign traffic and the covert channel traffic. This process ensures all the domain names in the epoch E are queried more than once and the domain names are not occasionally appeared in the observation periods. Next, we concatenated the packets belonging to the same sessions in the epoch E. Based on the communication sessions, the features of each session can be calculated. The new observations of the sessions can be represented as follow,

$$R_i = [a_1, a_2, \ldots, a_n] \tag{1}$$

where R_i is the new observation of the vectorized session and a_i is the value of the selected feature. We

can obtain the dataset $O^{m\times n}$ by transforming all the DNS traffic.

$$O^{m\times n}=\left[R_1,R_2,\ldots,R_m\right] \tag{2}$$

For each attribution in the dataset, the values should be standardized in order to have the same magnitude between the different attributions. Z-Score is a measure of how many standard deviations below or above the mean a raw score is. Taking the column C_i as the example, we can calculate the z-score of elements through the below formula,

$$z_i=\frac{c_i-\bar{c}}{S} \tag{3}$$

where the c_i is the element of the attribution, $\bar{c}$ is the mean value of c_i and S is the standard deviation of c_i. After the Z-Score standardization, we obtained the new data as the input of the models.

3.3. Model Training

In this paper, we proposed a stacking model which ensembles the three different algorithms (KNN, SVM and random forest). We will elaborate the configuration of each base model and how they ensemble together. In order to clearly describe the mechanism of the models, we consider the problem of each model as a binary classification problem where a training set composed of m observations and n attributions.

3.3.1. KNN

The core idea of KNN algorithm is that if most of the k nearest samples in the feature space belong to a certain category, then the sample also belongs to this category and has the characteristics of samples in this category. In the decision-making of classification, the method only determines the category of the sample to be classified according to the category of the nearest one or several samples. The KNN method mainly depends on the limited adjacent samples, rather than on the method of identifying the class field. Therefore, KNN method is more suitable than other methods for the sample set to be divided with more overlapping or overlapping class fields.

The KNN algorithm leverages the features of the k nearest samples to identify the label of unknown samples. Thus, the definition of the distance between two samples is important and, in the experiments, we adopted the Euclidean distance. When a KNN classifier decides the label of a target sample, the strategy is to label the sample with the label which occurs most in the k nearest samples. It has the shortage when the training dataset is unbalanced. In other words, the samples appearing more frequently will dominate the prediction of the target sample. Therefore, we weighted the k nearest samples at the classification to overcome this bias.

$$d\left(x_i,x_j\right)=\sqrt{\sum_{k=1}^{n}\left(x_{i_k},x_{j_k}\right)} \tag{4}$$

$$\hat{y}_t=\arg\max_{c_i}\left\{\sum_{x\in N_k,\hat{y}_x=c_i}e^{\frac{-d\left(x,x_t\right)^2}{2}}\right\},c_i\in\left\{-1,1\right\} \tag{5}$$

where x_i is the vector of a sample, x_{i_k} is an attribution of the sample, $\hat{y}_t$ and $\hat{y}_x$ are the label of the target and the sample respectively, N_k is the set of the k nearest neighbors, and c_i is a label of the label set. Here, after getting the k nearest neighbors, we chose the Gaussian function to weight the distance of the neighbors. For each class in N_k, the sum of the weighted distances, the result of the Gaussian function, needs to be calculated. The label of the target sample is decided by the label of the category having the maximum sum value.

3.3.2. Linear SVM

SVM is a kind of generalized linear classifier which classifies data according to supervised learning. Its basic model is the linear classifier with the largest interval defined in the feature space. The basic idea is to solve the separation hyperplane which can correctly divide the training data set and has the largest geometric interval. Its decision boundary is the maximum margin hyperplane to solve the learning samples.

SVM is a powerful model that can classify samples without error if the sample data is completely linearly separable. It is suitable for solving classification problems with high dimensional feature space and small training set size. We applied the SVM to acquire the "linear knowledge" of the input data. The aim of the SVM is to create a statistical model to predict the label value $\hat{y}_t$ of an element considering only its attribute vector x_t. Given the training samples, the SVM derives, in n-dimension space, an optimal hyperplane that separates the two classes and that will then be used to assess the class of unknown elements. The hyperplane equation is:

$$\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b} = 0 \tag{6}$$

where w is a coefficient vector and b is a scalar offset. The values of the optimal hyperplane parameters (w and b) are found maximizing the distance between the hyperplane and the nearest training observations of the two classes.

3.3.3. Random forest

Actually, the size of the training set of the DNS traffic usually is large and the covert channel identification is difficult. Therefore, the random forest is introduced as the base model to improve the performance and efficiency. The random forest is an ensemble learning algorithm, which belongs to bagging type. By combining multiple weak classifiers, the final result is voted or averaged, which makes the results of the whole model have high accuracy and generalization performance. Like its name implies, random forest consists of a lot of individual decision trees that operate as an ensemble. In the process of classification, each individual tree in the random forest splits out a class prediction and the class with the most votes becomes our model's prediction. The training algorithm of the random forest leverages the bagging technique, if the forest has B trees, we sample n training observations from X, Y, called these X_b, Y_b with replacement and then train a classification tree f_b on X_b, Y_b. The prediction of the random forest can be acquired through majority-voting of the B trees.

3.3.4. Stacking

After all the base models selected, the way how we combine weak learners to produce the more powerful classifier should be discussed. Stacking, that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction base on the different weak models predictions. Stacking is a multi-layer multi model aggregation method. Each layer can include multiple models, and the next layer uses the results of the previous model to learn. Roughly, stacking will mainly try to produce strong model less biased than their components.

For stacking, we chose a two-layer model, and we need to define two things in order to build our stacking model: the base models we want to fit and the meta-model that combines them. The base models we already selected and we decide to learn a neural network as meta-model. Then, the neural network will take as inputs the outputs of our three weak learners and will learn to return final predictions based on them.

The best way to explain the stacking technique is by Fig. 2. The idea of stacking is to divide the training set into N pieces and hold the Nth fold out for validation. For each fold, predictions for each fold is obtained from a fit using the rest of the folds and collected in an out-of-sample predictions matrix. Namely, the meta-model will train on the predictions matrix to obtain the final predictions for all the samples.

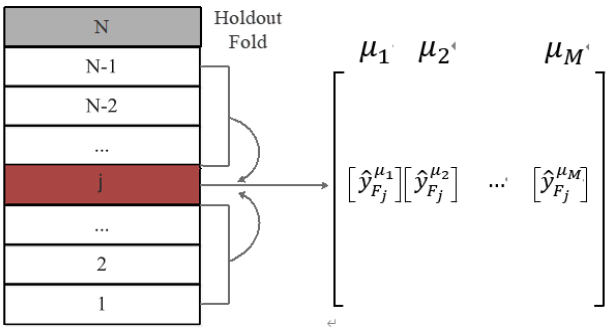

Fig. 2. Stacking the base models on training set.

Specifically, stacking model is to further mine the relationship between data and labels on the basis of the original model, that is, on the basis of the first level model, the model is further trained according to its prediction results to achieve the effect of comprehensive model. In this regard, we divide the data set into training set and test set. For the training set, in order to avoid over fitting, each basic model uses cross validation method to obtain the prediction value of the model for the training set, and then integrates the prediction results of each model as the input training set of the meta model. For the test set, each basic model has predicted the test set during cross validation, and takes the average value, and then integrates the prediction results of each model as the test set of meta model. In this way, the training set is used to train the meta model, and the test set is predicted to get the final result.

4. Experiments and Analysis

4.1. Data Set

The collected data set in the experiments is mixed traffic, including background DNS traffic and DNS hidden channel traffic. The benign DNS traffic was collected from a college campus network. Within 20 days, we acquired 52,632,469 sessions. As we mentioned in section 3.2, the background traffic should be filtered by the white list composed of the top 50k domains in the Alexa list and remove domains which have been quired only once in the epoch E (E=1 hour in experiment). At last, we got the 32,748,103 sessions left.

The traffic of the DNS covert channels was generated by the collected tools, including Indine, Dns2tcp, DNSCat2, DeNiSe and Heyoka, which were frequently abused by cybercriminals. The brief descriptions of these tools are shown in Table 3. Indoine, Dns2tcp and DNSCat2 can implement multiple types of covert communications using different resource record such as NULL, TXT, SRV, MX, CNAME, KEY, etc.

Table 3. The Tools of DNS Convert Channel

Tool	Description
Indoine	This is a piece of software that lets you tunnel IPv4 data through a DNS server. This can be usable in different situations where Internet access is firewalled, but DNS queries are allowed.
Dns2tcp	Dns2tcp is a network tool designed to relay TCP connections through DNS traffic. Encapsulation is done on the TCP level, thus no specific driver is needed (i.e.: TUN/TCP). Dns2tcp client does not need to be run with specific privileges.
DNSCat2	This tool is designed to create an encrypted command-and-control (C&C) channel over the DNS protocol, which is an effective tunnel out of almost every network.
DeNiSe	DeNiSe is a proof of concept for tunneling TCP over DNS.
Heyoka	Heyoka ia a proof of concept of an exfiltration tool which uses spoofed DNS requests to create a bidirectional tunnel. It aims to achieve both performance and stealth.
TCP-over-DNS	A tool leverages encoding data in an address to achieve the communication between client and server.
OzymanDNS	A tool leverages SSH to implement DNS covert channel.

In a controlled network environment, we ran each tool in five hosts and obtained the traffic of the active

status (transmit the information) and the inactive status (do not transmit the information) respectively. We collected the traffic of the covert channels for 480 hours, in total obtaining 12,617,075 sessions. Domains which were queried only once in the epoch E were filtered out. The number of the left sessions is 11,403,570. We held the 30% of the background traffic and the 30% of the covert channels traffic out for testing. The summary of the training set and the testing set is presented in Table 4.

Table 4. Number of Each Type Traffic in Data Set

Type	Training set	Testing set
Indoine	1,468,777	629,476
Dns2tcp	1,374,824	589,210
DNSCat2	1,323,550	567,236
DeNiSe	1,492,795	639,769
Heyoka	1,487,316	637,421
TCP-over-DNS	0	603,213
OzymanDNS	0	589,983
Background	22,923,672	9,824,431
Total	30,070,924	14,080,739

4.2. Evaluation

To evaluate the effectiveness of the DNS covert channel detection, we chose two metrics, the area under the curve of receiver operating characteristic (AUC ROC) and the false positive rate (FPR). The x-axis of the ROC is FPR and the y-axis is the true positive rate (TPR). They are defined as follow respectively:

$$TPR = \frac{TP}{TP + FP} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

where TP is the number of the true positive samples, FP is the number of the false positive samples and TN is the number of the true negative samples.

Traditional DNS covert channel detection methods generally include two aspects of DNS packet analysis and traffic analysis. The typical method of DNS packet analysis is Farnham [11] proposed a DNS detection model that uses regular expressions to match domain names and Karasaridis *et al*. [5] proposed that detect DNS anomalies based on the network traffic statistics. Karasaridis *et al*. calculated the distribution of data message sizes within a certain time window, and estimated a cross-distribution entropy by analyzing and comparing the distribution obtained in the experiment and the standard distribution. The entropy was then used for the detection. Recent years, Shafieian *et al*. [7] believe that traditional DNS covert channel detection methods are not meet the requirement of detecting DNS covert channels. For example, the wide use of Content Delivery Network (CDN) has increased the false positives of traditional detection methods. Therefore, they proposed leverage ensemble learning techniques to detect DNS covert channels. In addition, according to Nadler *et al*. [10] investigation and research, low throughput DNS covert channels are often abused to leak the credit card information, passwords and other private information. Thus, an isolated forest method is proposed for detecting the low throughput and DNS protocol-based data leakage behavior, and the performance of the model is tested on a large size data. For the convenience, we called them Farnham model, Karasaridis model, Shafieian model and Nadler model. We implemented the four models and compared them with our method to verify the advantages of the proposed method in DNS covert channel identification.

We implement the stacking model with scikit-learning library and the parameter setting of the models is

concluded in Table 5.

Table 5. Parameter Setting of Models

Model	Parameters
KNN	k=5
SVM	penalty='12', c=1.0, max_iter=5000
Random Forest	n_estimators=1000
Neural Network	hidden_layer_sizes=(50,25), solver='adam', learning_rate='adaptive', max_iter=5000

We trained the stacking model on the training set and evaluated on the testing set for calculating the metrics. The experimental result is shown in Table 6. It can be seen from Table 6 that the AUC ROC of the stacking model reaches 0.9901, which is a certain improvement compared to the other four methods. The Farnham method extracts features of fully qualified domain names and performs complex regular expression. Only the information of the domain names is considered, so the performance is bad compared to the others. The Karasaridis method uses the statistics of network traffic to identify DNS covert channels. In a time window, the entropy of the size distribution of network packets is calculated. The performance is improved compared to Farnham's method, but the AUC ROC is still lower than our model. In [7], the DNS channel tools used to test the Shafieian method include Iodine, DNSCat2, and Ozyman. The number of trees used in the random forest are small. Therefore, when this method is applied to detect multiple different types of DNS covert channels, the performance is slightly reduced. The Nadler method [10] can effectively detect low-throughput DNS channels on the premise of considering the high-throughput DNS covert channel detection. However, the isolated forest algorithm used in this method is an unsupervised learning algorithm, which needs to establish a baseline on normal DNS traffic. Different network environments need different benchmarks and it results in fluctuations in detection performance. Compared with these two methods, our method is similar to the Shafieian method. However, we used the traffic of the multiple types of DNS channels as training data, and leveraged neural network and stacking technique to combine multiple base models. It has stable detection performance and can effectively identify different DNS covert channels.

Table 6. AUC ROC and PTR of Different Methods

Method	AUC ROC	FPR
Farnham	0.9496	1.68%
Karasaridis	0.9575	1.53%
Shafieian	0.9742	0.99%
Nadler	0.9812	1.01%
Stacking	0.9901	0.54%

In terms of FPR, compared with the other four methods, the Stacking Model has less false alarms. The stacking Model has a more comprehensive identification ability, because of leveraging the depth packet analysis and network traffic characteristics for comprehensive analysis. Compared with the Shafieian method and the Nadler methods, the Stacking Model can combine the different base models to capture more information about DNS channels. Through the stacking technique, the advantages of each base model can be effectively obtained, and the predictions of the base models are used as the input of the neural network learning. It has a stronger ability to identify the traffic of the DNS covert channels, and the false positive rate is lower.

The testing set include two types of DNS covert channels which are not existed in the training set. In the experiment, the stacking model still can successfully detect the unknown traffic (the traffic of TCP-over-DNS and OzymanDNS). Different DNS channel tools have the differences in programming, implementation details, and application scenarios, but the principles are basically similar. Thus, they have similar

characteristics of packets and behavioral features of communication. Therefore, the extracted features can be effectively used for identifying other unknown DNS covert channels.

5. Conclusion

In conclusion, we analyze the communication forms of the common DNS covert channels. The passive DNS data and the typical DNS covert communication channel samples are collected for comprehensive analysis. We studied the characteristics of the DNS covert channel packets and the communication behavior patterns of the tools. We propose a DNS covert channel detection method based on the stacking model which is able to detect the DNS covert channels effectively. It shows the excellent performance on this task by using the stacking technique combined with the base models, which improves the detection accuracy and reduces false positive rate.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Peng Yang and Xinxin Wan conceived of the presented idea. Guang Shi developed the theory and performed the computations. Hao Qu and Juan Li verified the analytical methods. Peng Yang encouraged Lixin Yang to investigate a specific aspect and supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

References

[1] Crotti, M., Dusi, M., Gringoli, F., *et al*. (2007). Detecting HTTP tunnels with statistical mechanisms. *Proceedings of the IEEE International Conference on Communications* (pp. 6162-6168).

[2] Dusi, M., Crotti, M., Gringoli, F., *et al*. (2009). Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. *Computer Network, 53(1)*, 81-97.

[3] Casas, P., Mazel, J., Owezarski, P., & Minetrac. (2011). Mining flows for unsupervised analysis & semi-supervised classification. *Proceedings of the 23rd International Teletraffic Congress. International Teletraffic Congress* (pp. 87-94).

[4] Marchal, S., Francis, J., Wagner, C., *et al*. (2012). DNSSM: A large scale passive DNS security monitering framework. *Network Operations & Management IEEE, 131(5)*, 988-993.

[5] Karasaridis, A., Meier-Hellstem, K., & Hoeflin, D. (2006). NIS04-2: Detection of DNS anomalies using flow data analysis. *Proceedings of the Global Telecommunications Conference* (pp. 1-6).

[6] Sheridan, S., & Keane, A. (2015). Detection of DNS based covert channels. *Proceedings of the 14th European Conference on Cyber Warfare and Security (ECCWS)*.

[7] Shafieian, S., Smith, D., & Zulkernine, M. (2017). Detecting DNS tunneling using ensemble learning. *Proceedings of the International Conference on Network and System Security* (pp. 112-127).

[8] Nussbaum, L., Neyron P., & Richard, O. (2009). On robust covert channels inside DNS. *IFIP Advances in Information & Communication Technology, 297*, 51-62

[9] Aiello, M., Merlo, A., & Papaleo, G. (2013). Performance assessment and analysis of DNS tunneling tools. *Logic Journal of IGPL, 21(4)*, 592-602.

[10] Nadler, A., Aminov, A., & Shabtai, A. (2019). Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computer & Security, 80*, 36-53.

[11] Farnham, G., & Atlasis, A. (2013). Detecting DNS tunneling. *SANS Institute InfoSec Reading Room*, 1-32.

[12] Bilge, L., Kirda, E., Kruegel, C., *et al.* (2011). Exposure: Finding malicious domains using passive DNS

analysis. *Proceedings of the Network and Distributed System Security Symposium*.

[13] Lencun, Y., Bottou, L., & Bengio, Y. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 862*, 2278-2324.

Peng Yang was born in Inner Mongolia in 1982. He received the Ph.D. degrees in computer science from Tianjin University, Tianjin, China, in 2010.

From 2010 to 2020, he has been working in CNCERT, Beijing, China, as a senior engineer. He is the author of more than 10 papers in journals and conferences. His research interests include cyber security and artificial intelligence.

Mr. Yang is currently a master tutor at Beijing Institute of Technology.

Xinxin Wan received her master's degree in pattern recognition and intelligent system in Beijing University of Posts and Telecommunication in 2011. She is currently an information security engineer in CNCERT. Her research interests include information security, data mining, etc.

Guang Shi received the B.S. degree in automation from Zhejiang University, Hangzhou, China, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2012 and 2017, respectively. Currently, he is a network security engineer with National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China. His research interests include neural networks, smart grids, computer networks, and network security.

Hao Qu received the post-graduate degree in software engineering from Heilongjiang University in 2015. He is working at Heilongjiang Branch of National Computer Network Emergency Response Technical Team/Coordination Center of China. His main research interests include network security and information security.

Juan Li graduated from Beijing University of Information Technology in 2013, received the master's degree in computer application technology. Her main research direction is network information security.

Lixin Yang is an associate professor and teaching supervisor of Heilongjiang Preschool Education College. Her scientific research field mainly covers computer teaching methods and strategies, the application of teaching courseware in kindergartens, the cultivation of students' ability in educational technology application, the cultivation of students' ability in courseware design, and the application of information-based teaching in new circumstances.