# FTPB: A Three-stage DNS Tunnel Detection Method Based on Character Feature Extraction

Kemeng Wu
*Institute of Information Engineering*
*(Chinese Academy of Sciences)*
*School of Cyber Security*
*(Univ. of Chinese Academy of Sciences)*
Beijing, China
wukemeng@iie.ac.cn

Yongzheng Zhang
*Institute of Information Engineering*
*(Chinese Academy of Sciences)*
*School of Cyber Security*
*(Univ. of Chinese Academy of Sciences)*
Beijing, China
zhangyongzheng@iie.ac.cn

Tao Yin
*Institute of Information Engineering*
*(Chinese Academy of Sciences)*
*School of Cyber Security*
*(Univ. of Chinese Academy of Sciences)*
Beijing, China
yintao@iie.ac.cn

***Abstract*—The domain name system(DNS) protocol is one of the most versatile protocols in the world. If a hacker can control the DNS protocol to pass messages and control the host of victim, then no firewall can effectively intercept the DNS protocol. In recent years, the DNS tunnel is one of the most dangerous threats in the field of steganography, which supports a wide range of criminal activities. In order to detect and distinguish different types of DNS tunnels, we present a three-stage DNS tunnel detection method based on character feature extraction. This novel method named FTPB which uses feature extraction to filter out the domain names of the DNS tunnels by feature extraction classifier and then converts them into a high-dimensional vector by term frequency-inverse document frequency(TF-IDF), and reduces the dimension to 2 by principal components analysis(PCA). Ultimately, the data is classified by a binary vector classifier. Our method only needs to extract the character information of the domain names, and can effectively discover different types of DNS tunnels. Compared with traditional detection methods that can only detect DNS tunnels based on content, our research method can also have the ability to detect DNS tunnels based on codebooks, and the outcome of the evaluation substantially prove the efficacy of our method with accuracy and precision are more than 99%.**

***Keywords*—Network security, DNS, DNS tunnel, Character Feature Extraction, Machine learning**

## I. Introduction

DNS is one of the most versatile Internet protocols on the market, but at the beginning of the design, it did not fully consider its security. The firewall does not set rules to intercept data on port 53. This vulnerability is exploited by cybercriminals to design DNS tunnels. The DNS tunnel is not easy to intercept by the firewall, and have fast transmission efficiency in our daily transmissions [1]. In the rapid Internet propagation process, its covertness is also superior to the traditional multimedia-based covert channel. DNS tunnel is a double-edged sword. On the one hand, it poses a great threat to the Internet privacy. First of all, it can illegally bypass the online login process designed by the Internet Service Provider(ISP) and lead to the avoidance of charges. Furthermore, it may lead to the disclosure of a large scale of personal privacy information and official secrets. Thirdly, DNS tunnels may cause national security issue. For example, if terrorists use DNS covert information and communication technology in tactical negotiation, national security will be threatened. On the other hand, if the DNS tunnel is used reasonably, the security of information transmission can also be effectively protected. It can protect the real-time and security of confidential information transmission, and securely transmit business secrets in the financial field. At the same time, through research and detection of the traffic pattern of the DNS tunnel, you can also find potential Trojans and vulnerabilities on the host of user. Therefore, the detection of DNS tunnel has become a hot issue at present, which has highly important research significance.

Although there are many research methods that can be used to detect DNS tunnels, the current detection methods are all used to detect DNS tunnels through machine learning methods. It takes a lot of time in the process of feature extraction, especially based on the behavioral features detection method. The string of the domain name itself contains a wealth of features, which is superior to the detection method based on behavior features, regardless of the computational complexity or timeliness. An excellent DNS tunnel detection model can not only be used as a lightweight algorithm, but also distinguish between different types of DNS tunnel. Therefore, it is of the utmost importance to construct a classifier that can instantaneously identify different types of DNS tunnels.

The main contributions of our paper are as follow:

- We propose a lightweight DNS tunnel detection method based on character feature extraction that only requires domain name information, which can detect DNS tunnel more quickly than recently published methods;
- In addition to the detection of DNS tunnel based on content, our work is capable of detecting DNS tunnel based on codebook.

The rest of the paper can be summarized as follows. In the second section, this paper reviews the relevant work background and establishes the basis of this study. Subsequently, the experimental operation process and specific implementa-

This work has been supported by the National Key Research and Development Program of China under Grant No.U1736218.

978-1-6654-0392-4/20/$31.00 ©2020 IEEE
DOI 10.1109/TrustCom50675.2020.00044

tion methods are elaborated in detail in the third section. In the fourth section, we introduce the source and composition of the data and then set up important metrics that can assess the performance of classification. Finally, the fifth section summarized the whole paper.

## II. Related Work

This section is divided into two parts. In the first part, we introduce the following four types of DNS tunnels: DNS tunnel based on domain name content, DNS tunnel based on codebook, DNS tunnel based on resource record type, DNS tunnel based on TTL. In the second part, we introduce the existing detection methods of DNS tunnel that are divided into machine learning detection methods and non-machine learning detection methods.

### A. Types of Existing DNS Tunnels

*1) DNS Tunnel Based on Domain Name Content:* The encoding side of this DNS tunnel encodes the information to the third-level or fourth-level domain name through Base32, Base64 or Advanced Encryption Standard(AES), and fulfills the decoding at the decoding side so that completing the information transmission. The transmission capacity of such DNS tunnel is the largest. Each request can transfer a maximum of 255 bytes. Below we introduce two different types of DNS tunnels based on content.

IP over DNS Tunnels: These DNS tunnels achieve the purpose of passing hidden information by encapsulating IP packets in DNS requests. DNScat2 [2] is an IPv4-over-DNS tool which use CNAME for writing information written in Java. The Iodine [3] developed in 2010 can use multiple record types: NULL, TXT, SRV, MX, CNAME, A, and can also support a DNS extension EDNS0, which can allow DNS messages to exceed the 512B limit, so more information can be transmitted. Iodine supports Base32 and non-compliant Base64 encoding, which ensures that the transmitted content is not easy to be found.

TCP over DNS Tunnels: By transmitting TCP packets in DNS tunnels, this type of transmission is different from IP over DNS tunnels because its clients do not need to send heartbeat packets to maintain an active connection. OzymanDNS [4] can choose Base32 or Base64 encoding methods, and transmit information through TXT record. DNSscapy [5] can choose two record types CNAME and TXT, it is written in python language, and the encoding method is non-compliant Base64. Weasel [6] communicates over DNS using AAAA queries and answers and encrypt data payloads using AES-128 in CTR mode. This is an improved DNS tunneling tool because it does not use TXT records, which invalidates the method of judging whether it is a DNS tunnel by using TXT records

*2) DNS Tunnel Based on Codebook:* By setting the mapping codebook at the encoding side and decoding side, the DNS tunnel based on codebook fulfills the information transmission. The transmission capacity of this DNS tunnel is relatively small, and the transmission capacity can be increased by designing the size of the codebook. This kind of covert channel has a slow transmission speed and strong covertness, which is very difficult to detect.

PacketWhisper [7] is a DNS tunneling tool based on codebook. It encodes the content through Base64, and then maps these 64 different letters to a specific domain name. It can convert any file type into a list of Fully Qualified Domain Names (FQDNs). Usually, these domain names will have the same second-level domain so that it is easy to identify in the decoding process, and its third-level domain name will be encoded by random numbers or random letters.

*3) DNS Tunnel Based on Resource Record Type:* There are 79 common types of DNS resource records. By encoding different record types for a single request name, the purpose of covert transmission of information can also be achieved.

*4) DNS Tunnel Based on TTL:* The DNS packet contains a 32-bit Time-to-Live (TTL) field for each response record. This field has no specified value, making it an ideal secret carrier. By setting up encoding rules, up to 4 characters of information can be encoded into a TTL field, so as to pass the message out [8]. But due to the transmission rate of this encoding method is too small, its threat is not great.

### B. Existing Detection Methods

*1) Machine Learning Methods:* We divided the previous work on DNS tunnel detection into two categories: machine learning methods and non-machine learning methods. In 2009, Hind [9] first proposed the use of neural networks to extract features to detect DNS tunnels. The paper [10] introduced the detection method by extracting features in DNS payload and using support vector machines for binary classification. The paper [11] extracted 16 character features and behavioral features, and used a random forest algorithm to detect DNS tunnels. Almusawi et al. [12] combined the length feature and the entropy feature of the domain name, and used the kernel SVM algorithm to obtain an average F1 of 0.8. Liu et al. [13] proposed to obtain a 99.96% accuracy rate by extracting behavioral features. Palau et al. [14] explored the viability of 1D-CNN for lexicographical DNS tunnel detection. My previous research on DNS tunnel [15] was to effectively detect and classify DNS tunnels by extracting the characteristics of DNS payload and using logistic regression. The above methods are all detection methods using supervised learning. Another detection method is based on unsupervised detection methods. Nadler et al. [16] proposed to use the Isolation Forest model method, which learned the query and response behavior characteristics of normal domain names. There is also a semi-supervised learning detection method [17] that draws on the idea of anomaly detection. It has used autoencoder algorithm to learn the payload of normal domain name traffic automatically in the training phase, and feeds the mixed traffic into the test phase. If it is the normal domain name traffic, it can be restored, while the DNS Tunnel traffic cannot be restored. However, the above methods can only detect DNS tunnel based on content, other types of DNS tunnel cannot be detected.

*2) Non-machine Learning Methods:* Non-machine learning detection methods can be divided into Payload Analysis Methods and Traffic Analysis Methods. Payload Analysis Methods extracts effective features for DNS query and response packets. These features usually include packet length, entropy value [18]. and domain name distribution [19]–[22], quickly identified normal DNS and DNS tunnel. Traffic Analysis Methods determines whether it was a DNS tunnel by obtaining traffic behavior characteristics. Michael [23] proposed throughput estimates, such as the DNS traffic value for each IP address and the DNS traffic volume for each domain to detect DNS tunnels. Wendy and Anna [24] proposed a flow-based detection method, which counted the number of bytes and the number of streams per stream. However, this method needs to collect a large amount of traffic and could not be detected in real-time. Mohammed et al. [25] proposed a visual DNS tunnel detection method. Since most DNS tunnels used a single host domain name to establish the DNS tunnel, at the same time, the number of its third-level subdomains was numerous and different, and the value of TTL was a single value, so that the DNS tunnel was discovered through the parallel coordinates technique.

## III. Method

FTPB is an acronym of our method which we use the feature extraction, TF-IDF, PCA, and Bagging algorithms. Combining the first letters of these methods is the name of our method. FTPB is divided into three steps, as shown in Fig. 1: coarse-grained classification, vector conversion, fine-grained classification. The input data are as follows: normal domain name, domain name of domain generation algorithm(DGA), the domain name of DNS tunnel based on the content, and the domain name of DNS tunnel based on the codebook. In the first step, we extracted 9 key features of the domain name characters through feature extraction. These features can effectively help us distinguish the DNS tunnel domain name from the normal domain name and DGA domain name. In the second step, the domain name of the DNS tunnel that has been filtered is first mapped to a high-dimensional vector form by TfidfVectorizer, and then reduced to 2 dimensions by PCA. Finally, the two-dimensional word vectors of different categories after dimensionality reduction are classified into different types through machine learning algorithms.

### *A. Coarse-grained Classification*

The first stage of our framework is a coarse-grained feature extraction classifier that determines whether it is the DNS tunnel or other. We assume that the domain names generated by the DNS tunnel, the normal domain name and the domain names generate by the DGA show different character features. The normal domain names are used for the convenience of users to remember. DGA generates a large number of random domain names in a short time to build a stable botnet, and the domain name generated by the DNS tunnel is to transmit encrypted information, so we can extract character features, which can filter out the query name of the DNS tunnel. We will discuss which classification algorithm to use in Section III-C.

**Character Feature**: we extract character feature on the basis of Phoenix [26], so that we can quickly and effectively distinguish the domain name generated by the DNS tunnel from the normal domain name and the domain name generated by DGA.

**CF1:** Longest Meaningful Characters Ratio. This feature represents the ratio of the longest meaningful characters in the first subdomain name on the left to its length. The normal domain name is usually a combination of meaningful names in order to remember easily, the DGA domain name is generated by a random algorithm, and the DNS tunnel needs to encode the transmission information, so the generated subdomain names of the DNS tunnels are likely to be composed of meaningless characters. We usually divide all domain names into at least three characters and match them in the commonly used 10,000 words. The length of the longest word that can be matched is divided by the length of the first subdomain name on the left to get the ratio of meaningful characters.

**CF2:** N-gram Score. This feature can be used to compare the frequency scores of different fields in the first subdomain name on the left to its length. N-gram is a language model of natural language processing, which is widely used in speech recognition research, and also universal in the field of intrusion detection and botnet detection [27]. Taking ”google” as an example, the 1-gram in the n-gram model divides ”google” into 6 fields of g, o, o, g, l, and e. For these 6 letters corresponding to the values in the dictionary to count their occurrences, and 2-gram perform the above statistics on the domain names in the fields of go, oo, og, gl, le, and so on to 5-gram. After the statistics are completed, use n-gram regularized fraction formula:score $= (\sum_i \text{ number } [\text{field}_i])/(|p| - n + 1)$,where $p$ is the number of fields, and number $[\text{field}_i]$ is the occurrences of the i-th field in the dictionary. We use this formula to calculate the regularized score, so we count all the N-gram scores.

**CF3:** Entropy of Subdomain Names. This feature stands for the entropy of the characters in the first subdomain name on the left to its length. Entropy is the average amount of information contained in each received message and can be interpreted as the uncertainty of a character in the subdomain. In a general sense, the formula for calculating the entropy value on the discrete random variable X is $\mathrm{H}(\mathrm{X}) = -\sum_{i=1}^{n} \Pr(x_i) \cdot \log \Pr(x_i)$, Where $\Pr(x_i)$ is the probability of the i-th information symbol (for example, one character) in the series $X$ consisting of the most n symbols.

**CF4:** Numerical Characters Ratio. This feature represents the ratio of the number of numerical characters in the first subdomain name on the left to its length.

**CF5:** Different Alphabetic Characters Ratio. This feature calculates the ratio of the number of different alphabetic characters appearing in the first subdomain name on the left to its length.

**CF6:** Different Numerical Characters Ratio. This feature represents the ratio of the number of different numerical

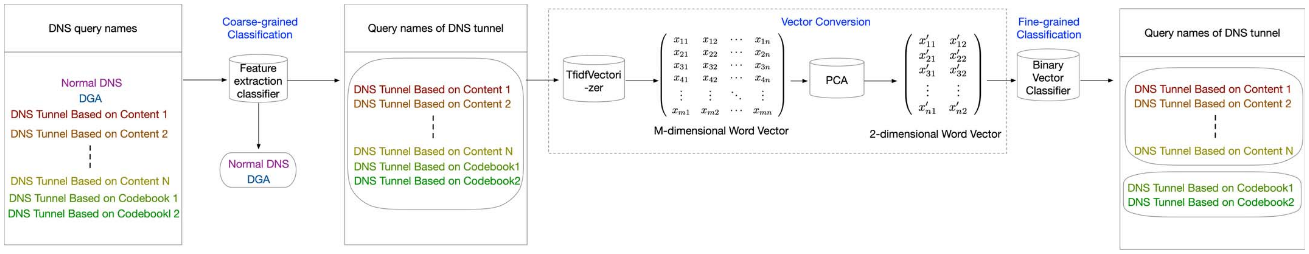
DNS query names
Normal DNS
DGA
DNS Tunnel Based on Content 1
DNS Tunnel Based on Content 2
DNS Tunnel Based on Content N
DNS Tunnel Based on Codebook 1
DNS Tunnel Based on Codebookl 2
Coarse-grained Classification
Feature extraction classifier
Normal DNS
DGA
Query names of DNS tunnel
DNS Tunnel Based on Content 1
DNS Tunnel Based on Content 2
DNS Tunnel Based on Content N
DNS Tunnel Based on Codebook1
DNS Tunnel Based on Codebook2
Vector Conversion
TfidfVectori -zer
M-dimensional Word Vector
PCA
2-dimensional Word Vector
Fine-grained Classification
Binary Vector Classifier
Query names of DNS tunnel
DNS Tunnel Based on Content 1
DNS Tunnel Based on Content 2
DNS Tunnel Based on Content N
DNS Tunnel Based on Codebook1
DNS Tunnel Based on Codebook2

Fig. 1. The main framework of FTPB

characters in the first subdomain name on the left to its length.

**CF7:** Length of Subdomain Names. The function of DNS tunnel is to transmit as much information as possible, especially the DNS tunnel based on the content. Since it hides the information in the first subdomain, its subdomain name will be much longer than the normal domain name and DGA domain name.

**CF8:** Vowel Characters Ratio. Vowel Characters a, e, i, o, u occupy a very important position in linguistics. Whether a word can be pronounced normally depends on whether the vowel syllables exist in the string. By estimating the proportion of vowel characters in the domain name, it can be simplified to measure whether this string follows the phonotactics of linguistics, so as to judge whether it is a domain name generated by a DNS tunnel.

**CF9:** Number of Alphanumeric Swaps. This feature is used to count the number of exchanges of characters and numbers in the first subdomain name on the left. For example, a string bk2by first switches from characters to numbers, and then from numbers to characters, then we count this feature value as 2.

As shown in Table I, we can see the character feature values of different domain types. These features will be put into the training set to obtain the feature extraction model, and then use the this model to classify the testing set. Section III-C will discuss which classification algorithm to use in the coarse-grained classification and fine-grained classification. In this way, most DNS tunnels can be identified.

$$\boldsymbol{x} = \boldsymbol{x}^{\mathbf{0}} \overset{p\left(\boldsymbol{h}|\boldsymbol{x}^{\mathbf{0}}\right)}{\longrightarrow} \boldsymbol{h}^{\mathbf{0}} \overset{p\left(\boldsymbol{x}|\boldsymbol{h}^{\mathbf{0}}\right)}{\longrightarrow} \boldsymbol{x}^{\mathbf{1}} \overset{p\left(\boldsymbol{h}|\boldsymbol{x}^{1}\right)}{\longrightarrow} \boldsymbol{h}^{\mathbf{1}} \tag{3.8}$$

### B. Vector Conversion

After coarse-grained classification, we merely retain the domain names that are recognized as DNS tunnels. At this time, we need to convert the string of the domain name into the form of vectors through TF-IDF, and reduce the dimension to two-dimensional vector through PCA. When the character information of the domain name is converted into a two-dimensional vector, not only can the speed of fine-grained classification in the next stage be improved, but also the different types of DNS tunnels can be visualized.

*1) TF-IDF:* Term Frequency-Inverse Document Frequency (TF-IDF) [28] is a commonly used weighting technique for information retrieval and data mining. It is often used to research keywords in articles and is used by industry for initial text data cleaning. In the process of text classification, the importance of a word increases in proportion to the number of times it appears in the text. If a word appears more frequently in a text and rarely appears in other articles, then we think that this word or phrase has an excellent ability to distinguish between different categories of the articles. Similarly, in the process of DNS tunnel classification, the appearance of specific characters can reflect the encoding characteristics of such DNS tunnels, especially the DNS tunnel based on the codebook. Since it is a fixed codebook, we can discover the high frequency of specific characters. The DNS tunnel based on content makes use of different encoding schemes, so there are specific high-frequency characters that can be operated for recognition.

TF-IDF is divided into two parts. The first part is to count the word frequency. Through n-gram, the frequency of different character combinations is counted and normalized. The formula is as follows:

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

Where $n_{i,j}$ is the number of occurrences of the character or string in the domain names $d_j$, and the denominator is the total number of occurrences of all characters or string in the domain name $d_j$.

The second part is the Inverse Document Frequency (IDF). IDF refers to the frequency of a particular word, which can be defined as the total number of domain names divided by the number of domain names containing the word $t_i$, and then obtained by logarithmic operation. If there are fewer documents containing word $t_i$, the greater the value of IDF, it means that word $t_i$ has a good ability to distinguish between categories. The formula is as follows:

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}| + 1} \tag{2}$$

Where $|D|$ is the total number of domain names in the corpus. $|j : t_i \in d_j|$ represents the number of domain names that contain the word $t_i$. If the word $t_i$ is not in the corpus, it will cause the denominator to be zero, so we generally employ $|\{j : t_i \in d_j\}| + 1$.

When extracting the n-gram frequency, we also need to set the value range. If the frequency of $t_i$ is too large, it may be a first-level domain name, and it does not represent the encoding

TABLE I
SAMPLE OF CHARACTER FEATURE

| Type | CF1 | CF2-1 | CF2-2 | CF2-3 | CF2-4 | CF2-5 | CF3 | CF4 | CF5 | CF6 | CF7 | CF8 | CF9 | DNS Query Name |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alexa | 1.00 | 3389.29 | 155.50 | 14.80 | 1.75 | 0.00 | 2.52 | 0.00 | 0.86 | 0.00 | 7.00 | 0.50 | 0.00 | youtube.com. |
| DGA | 0.27 | 3733.36 | 252.50 | 6.44 | 0.00 | 0.00 | 2.55 | 0.00 | 0.64 | 0.00 | 11.00 | 0.08 | 0.00 | gtuntctblnt.biz. |
| DNS tunnel based on content | 0.14 | 2306.13 | 71.79 | 4.46 | 0.07 | 0.00 | 4.89 | 0.13 | 0.48 | 0.08 | 63.00 | 0.08 | 14.00 | MjU2LGhtY...2.544 19.ns.test.com.[1] |
| DNS tunnel based on codebook | 0.00 | 855.38 | 3.00 | 0.00 | 0.00 | 0.00 | 3.00 | 0.38 | 0.63 | 0.38 | 8.00 | 0.11 | 3.00 | 42zjo3gw.akstat.io. |

[1] The complete content of the domain name is MjU2LGhtYWMtc2hhMi01MTIsaG1hYy1zaGExAAAAGm5vbmUsemxpYkBvcGVuc3N.oLmNvbSx6bGliAAAA Gm5vbmUsemxpYkBvcGVuc3NoLmNvbSx6bGliAAAAAAAAAA.AAAAAAAAAAAAAA.8.f.2.54419.ns.test.com.

nature of this type of DNS tunnel. However, if the frequency of this $t_i$ is too small, the dimension of the converted word vector will be too large, which will increase the calculation time of dimensionality reduction. Therefore, the set frequency range of $t_i$ will choose between 10-75%.

*2) PCA:* Due to the high number of different character combinations in the domain name, the dimension of the word vector after TF-IDF is usually very large, which will greatly increase the classification time and reduce the effect of classification. Therefore, by using the dimensionality reduction method, not only can the key features be retained, but also the classification time can be reduced. This method ensures that the domain name vector generated by the same DNS tunneling tool can have a closer distance in the Euclidean distance. Principal Component Analysis (PCA) is a linear, unsupervised, global dimensionality reduction algorithm. The purpose of this algorithm is to find the principal components of the data, and at the same time, it can use these principal components to represent the original data, so as to achieve the purpose of dimensionality reduction. The dimensionality reduction algorithm of FTPB is described as Algorithm 1:

**Algorithm 1** The dimensionality reduction algorithm of FTPB

```
Input: n-dimensional vector $D = (x_{(1)}, x_{(2)}, \dots, x_{(m)})$.
Output: 2-dimensional vector after dimensionality reduction
    $(D') = (x'_{(1)}, x'_{(2)}, \dots, x'_{(m)})$
Decentralize all word vectors: $x_{(i)} = x_{(i)} - \frac{1}{m}\sum_{j=1}^{m} x_{(j)}$.
Calculate the covariance matrix $XX^T$.
Decompose eigenvalue of $XX^T$.
Extract the feature vector $W = (w_1, w_2)$ corresponding to the largest 2 eigenvalue, Normalize all feature vectors to form a feature vector matrix $W$.
Convert $x_{(i)}$ into a vector of new dimensions $x'_{(i)} = W^T x_{(i)}$
return $D' = (x'_{(1)}, x'_{(2)}, \dots, x'_{(m)})$
```

Figure 2 presents the distribution of ground truth of different types of DNS tunnels after dimensionality reduction. We can discover that the majority of the points after dimensionality reduction can form classification separated from each class, only DNScapy and OzymanDNS will overlap with other classes. Because PackerWhisper is a DNS tunnel based on codebook, the word frequency distribution of its domain name is similar to the normal domain name. The black points surround around the points of PackerWhisper and the positions of their centers are almost the same.

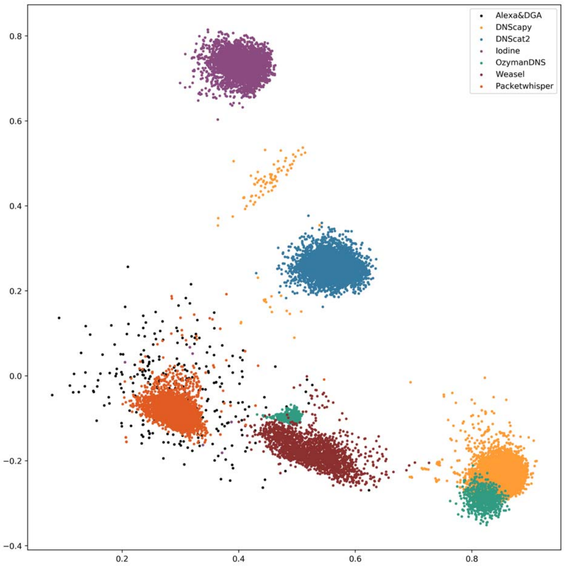

Fig. 2. The distribution of ground truth of different DNS tunnels after PCA

### C. Fine-grained Classification

In the third stage of our framework, we hope to find the best algorithm that can separate DNS tunnel based on content and DNS tunnel based on codebook. We aim to have a classification algorithm that can not only solve coarse-grained classification after feature extraction but also classify different types of DNS tunnels after dimensionality reduction. After comparing different algorithms, we decided to use ensemble learning to solve these two binary classification problems.

Ensemble learning is a machine learning method that puts a series of multiple learning algorithms into effect and proposes a certain rule to integrate various learning results to obtain better predictive performance than a single learning algorithm [29]. Commonly used Ensemble learning algorithms are as follows: bagging, random forest, AdaBoost, and gradient boosting.

Bagging is the most well-known representative of parallel integrated learning methods, and it is an integrated technology proposed by Leo Breiman in 1994 [30]. It is directly based on the bootstrap sampling method. We first randomly take a sample into the sampling set, and then put the sample back into the initial data set, so that the sample may still be selected during the next sampling. In this way, after $m$ times random sampling operation, we get a sampling set containing $m$ samples, some samples in the initial training set appear multiple times in the sampling set, and some never appear. We can choose $T$ sampling sets, and then train a base learner based on each sampling set. This base learner uses a decision tree algorithm, and then combine these base learners to form a strong learner. In this paper, we choose the bagging classifier with the decision tree algorithm J48 based on the C4.5 model as the single classifier to get the best performance.

Random forest is a classifier that contains multiple decision trees, and its output category is determined by the mode of individual trees [30]. The only difference between the random forest and bagging method is that random forest uses randomly selected features when generating decision trees. The reason why the features are randomly selected is that if several features in the training set have strong predictability on the output results, then they are applied by each decision tree, which leads to a correlation between the trees, but they cannot reduce the variance of the model.

The AdaBoost algorithm increases the weight of the sample that cannot be processed by the current classifier in the next iteration and continuously adds a new weak classifier until the termination condition is reached [31]. The final model is a weighted linear combination of weak classifiers, and its weight is relevant to its correct rate. The iterative process uses an exponential loss function.

The gradient boosting algorithm trains the newly added weak classifier according to the negative gradient information of the current model loss function and subsequently combines the trained weak classifier into the existing model in an accumulated form [32]. Compared with AdaBoost which can only use exponential loss function, gradient boosting can use any differentiable loss function.

## IV. Evaluation

In this part, we introduce the composition of the data set and how to evaluate the performance of our model. First, we introduce the composition of the data set. Second, we recommend performance metrics that evaluate our coarse-grained classification and fine-grained classification. And finally, we show our results and the effect of our two classifiers.

### A. Dataset

Our data set is composed of three parts, namely normal domain name, DGA domain name and DNS tunnel domain name. The normal domain name is downloaded by first 100,000 domain names of the Alexa [33] in the latest version. The data of the DGA domain name is the latest data obtained on 20 blacklisted websites through web crawler technology. In the process of crawling data, first of all, we use the requests [34] library to obtain the content of the webpage and read it according to the encoding method of the webpage, and then use different methods to extract the corresponding information in accordance with the different format of the webpage content. Finally, the tldextract [35] library is used to simplify the extracted domain names and extract useful fields. The way to obtain domain names of the DNS tunnels is to establish different types of DNS tunnels and use Wireshark to capture and filter the domain name request. The DNS tunnels include DNS2tcp, DNScapy, DNScat2, OzymanDNS, Iodine, Weasel and PacketWhisper. The first six of them are DNS tunnels based on content, and the last one is a DNS tunnel based on codebook. We extract all the traffic data which contains only the domain name information and save it in the form of csv file. In the coarse-grained classification, we take normal domain names and DGA domain names as negative samples, and DNS tunnel domain names as positive samples, and in the fine-grained classification we label DNS tunnels based on content as positive samples and DNS tunnels based on codebook as negative samples. We randomly split 60% of all data sets as the training set and 40% as the test set. Table II shows the size of data sets of different categories.

TABLE II
THE SAMPLE SIZE OF TRAINING AND TESTING SETS

| Category | Training Set | Testing Set |
|---|---|---|
| Alexa | 59,689 | 39,792 |
| DGA | 9,918 | 6,612 |
| DNScat2 | 8,199 | 5,466 |
| DNScapy | 11,547 | 7,698 |
| Iodine | 8,685 | 5,790 |
| Weasel | 2,839 | 1,892 |
| OzymanDNS | 4,444 | 2,962 |
| PacketWhisper | 9,832 | 6,554 |
| Total | 115,151 | 76,768 |

### B. Performance Metrics

Supervised learning usually evaluates the performance of the algorithm using the four evaluation metrics: accuracy, precision, recall, and F1. First, we need to define four evaluation metrics: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). Accuracy, precision, recall, and F1 formulas are as follows:

$$\text{Accuracy} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \tag{4}$$

$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{5}$$

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{6}$$

$$\text{F1} = \frac{2^{*}\ \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{7}$$

TABLE III
PERFORMANCE COMPARISON OF COARSE-GRAINED CLASSIFICATION

| Metrics | Bagging | RandomForest | GradientBoosting | AdaBoost | KNeighbors | LogisticRegression | GaussianNB |
|---|---|---|---|---|---|---|---|
| Training Time(sec) | 4.927 | 19.354 | 23.369 | 5.117 | 0.291 | 0.895 | 0.052 |
| Testing Time(sec) | 0.132 | 1.270 | 0.143 | 0.491 | 3.803 | 0.003 | 0.021 |
| Accuracy | 0.9931 | 0.9939 | 0.9829 | 0.975 | 0.9275 | 0.9191 | 0.8696 |
| Precision | 0.9916 | 0.9915 | 0.9783 | 0.9634 | 0.8998 | 0.9321 | 0.8183 |
| Recall | 0.991 | 0.9931 | 0.9786 | 0.974 | 0.9196 | 0.8583 | 0.8626 |
| F1 | 0.9913 | 0.9923 | 0.9784 | 0.9687 | 0.9095 | 0.8937 | 0.8399 |

TABLE IV
PERFORMANCE COMPARISON OF FINE-GRAINED CLASSIFICATION

| Metrics | Bagging | RandomForest | GradientBoosting | AdaBoost | KNeighbors | LogisticRegression | GaussianNB |
|---|---|---|---|---|---|---|---|
| Training Time(sec) | 0.308 | 1.758 | 2.468 | 0.707 | 0.049 | 0.066 | 0.006 |
| Testing Time(sec) | 0.019 | 0.224 | 0.037 | 0.128 | 1.008 | 0.000 | 0.002 |
| Accuracy | 0.9995 | 0.9998 | 0.9955 | 0.9883 | 0.9954 | 0.984 | 0.9856 |
| Precision | 0.9989 | 0.9997 | 0.9852 | 0.9566 | 0.9867 | 0.9414 | 0.9566 |
| Recall | 0.9986 | 0.9994 | 0.9944 | 0.9912 | 0.9924 | 0.9875 | 0.9779 |
| F1 | 0.9988 | 0.9995 | 0.9897 | 0.9735 | 0.9895 | 0.9639 | 0.9671 |

In the stage of coarse-grained classification, accuracy is the ratio of true DNS tunnel to the total observations. Precision is the ratio of correctly predicted DNS tunnel to the total positive observations. Recall is the ratio of correctly predicted DNS tunnel to the predicted positive observations. F1 Score is the weighted average of Precision and Recall. In the stage of fine-grained classification, the focus of our detection is the DNS tunnel based on codebook. So our total observations are the total predicted positive observations in the coarse-grained classification, and the goal is to detect DNS tunnel based on codebook accurately.

### C. Result

The experimental environment is deployed in a MacBook with a 2-core Intel Core i5 processor at 2.3GHz with 8G of RAM and running on macOS Catalina. The coding language we use is python, and we use Scikit-learn [36] machine learning library to implement our classifier. In the coarse-grained and fine-grained classification, we both use 7 different algorithms to accomplish classification tasks, and the settings of these algorithms use the default settings. The evaluation process is repeated ten times, and each time the training set and the test set are randomly split, and the average value is taken as our experimental result. Our experimental results of the coarse-grained classification and fine-grained classification are shown in Table III and Table IV.

The experimental results show that both classifiers can obtain relatively high performance while using the bagging algorithm, and the training time and testing time are also shorter. Table III shows the performance comparison of coarse-grained classification, if we only focus on F1, the value of the random forest algorithm is the highest, but its training time and testing time are respectively 3.9 times and 9.6 times of the bagging algorithm, so it is not the optimal algorithm. This shows that the bagging algorithm can be the best algorithm in the feature extraction classifier. In this way, the DNS tunnel can be classified more effectively. In Table IV, we also use 7 classifiers to classify the 2-dimensional vector after PCA. Because of the dimensionality reduction algorithm, our training time and testing time have been greatly reduced. We are comparing bagging algorithm and random forest algorithm, and we can find that the performance difference between the two is not significant, but the training time and testing time of random forest algorithm are 5.7 times and 12 times as long as bagging algorithm, so we still choose to match bagging algorithm as the binary vector classifier.

In order to verify that our method can have better detection capabilities, we reproduced two recently published methods, named Liu [13] and Almusawi [12]. Liu et al. use four kinds of behavioral features including time-interval features, request packet size features, record type features and sub-domain entropy. Almusawi uses kernel SVM for the sake of detecting DNS tunnel and has the ability of multi-label classification. The algorithms used in these two methods are also supervised learning algorithms. In order to demonstrate the effectiveness of FTPB, we used the same dataset in Table II and the same experimental environment. We all select a binary-classification method to classify the domain names marked as normal domain names, domain names of DNS tunnel based on content, and domain names of DNS tunnel based on codebook. The experimental results are shown in the TableV. The results show that these other two methods are inferior to FTPB. In terms of training time and testing time, FTPB has the shortest time in these three methods. Although there is not much difference between FTPB and method of Almusawi in detecting DNS tunnel based on content, FTPB has the best effect in detecting DNS tunnel based on codebook and distinguishing two different types of DNS tunnels. Because method of Liu is a detection method based on behavioral features, it needs to collect the behavioral features of DNS traffic within a period of time, so its training time and testing

TABLE V
COMPARE THE RESULTS WITH OTHER METHODS

| Detection Target | Method | Training Time(sec) | Testing Time(sec) | Presion | Recall | F1 |
|---|---|---|---|---|---|---|
| Alexa VS DNS Tunnel Based on Content | Liu et al. [13] | 161.272 | 188.665 | 0.9322 | 1 | 0.9649 |
| | Almusawi et al. [12] | 11.6171 | 1.9339 | 1 | 0.9927 | 0.9963 |
| | FTPB | **4.042** | **0.1** | 0.9916 | **0.993** | 0.9923 |
| Alexa VS DNS Tunnel Based on Codebook | Liu et al. | 189.523 | 135.892 | 1 | 0.2068 | 0.3427 |
| | Almusawi et al. | 169.226 | 43.563 | 0.8666 | 0.9895 | 0.9240 |
| | FTPB | **3.494** | **0.066** | 0.9885 | **0.9917** | **0.9901** |
| Content VS Codebook | Liu et al. | 75.775 | 73.523 | 0.7198 | 0.2068 | 0.3213 |
| | Almusawi et al. | 23.875 | 8.059 | 0.9959 | 0.991 | 0.9935 |
| | FTPB | **0.937** | **0.037** | **0.999** | **0.9999** | **0.9995** |

time are the longest, and the recall rate of this method is too low to have the ability to detect DNS tunnel based on codebook. This result indicates that FTPB has a better ability to detect DNS tunnel domain names, especially the ability to detect DNS tunnel based on codebook.

## V. CONCLUSION

In this paper, we propose a detection method based on character feature extraction. This method can detect DNS tunnels in the coarse-grained classification stage and identify different types of DNS tunnels in the fine-grained classification stage. It does not require a very high system configuration and only needs to extract the domain name information in the DNS traffic to complete the detection effect with high accuracy, precision, recall and F1.

We believe that this method will have high practical value for the detection of existing DNS tunnels based on content and based on codebook. However, there is no detection capability for DNS tunnel based on resource record type and based on TTL. Future work might consider designing a detection method that can detect all types of DNS tunnels of unknown encoding methods that appear in the future.

## REFERENCES

[1] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. Van Steen, and N. Pohlmann, "On botnets that use dns for command and control," in *2011 seventh european conference on computer network defense*. IEEE, 2011, pp. 9–16.

[2] PIETRASZEK, "dnscat2," 2010. [Online]. Available: http://tadek.pietraszek.oig/projects/DNScat/

[3] E. ANDERSSON, B EKMAN, "iodine," 2011. [Online]. Available: http://code.kiyo.se/iodine/

[4] D. Kaminsky, "Ozymandns," 2004. [Online]. Available: https://room362.com/post/2009/2009310ozymandns-tunneling-ssh-over-dns-html/

[5] P. Bienaime, "Dnscapy," 2012. [Online]. Available: https://code.google.com/archive/p/dnscapy/

[6] "Weasel," 2019. [Online]. Available: https://github.com/facebookincubator/WEASEL#

[7] "Packetwhisper," 2020. [Online]. Available: https://github.com/TryCatchHCF/PacketWhisper

[8] C. Hoffman, D. Johnson, B. Yuan, and P. Lutz, "A covert channel in ttl field of dns packets," 2012.

[9] J. Hind, "Catching dns tunnels with ai," *Proceedings of DefCon*, vol. 17, 2009.

[10] V. T. Do, P. Engelstad, B. Feng, and T. van Do, "Detection of dns tunneling in mobile networks using machine learning," vol. 424, pp. 221–230, 2017.

[11] A. L. Buczak, P. A. Hanke, G. J. Cancro, M. K. Toma, L. A. Watkins, and J. S. Chavis, "Detection of tunnels in pcap data by random forests," in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*. ACM, 2016, p. 16.

[12] A. Almusawi and H. Amintoosi, "Dns tunneling detection method based on multilabel support vector machine," *Security and Communication Networks*, vol. 2018, pp. 1–9, 2018. [Online]. Available: http://dx.doi.org/10.1155/2018/6137098

[13] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang, and C. Peng, "Detecting dns tunnel through binary-classification based on behavior features," in *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 2017, pp. 339–346.

[14] F. Palau, C. Catania, J. Guerra, S. Garcia, and M. Rigaki, "Dns tunneling: A deep learning based lexicographical detection approach," *arXiv preprint arXiv:2006.06122*, 2020.

[15] K. Wu, Y. Zhang, and T. Yin, "Clr: A classification of dns tunnel based on logistic regression," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019.

[16] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the dns protocol," *Computers & Security*, vol. 80, pp. 36–53, 2019.

[17] K. Wu, Y. Zhang, and T. Yin, "Tdae: Autoencoder-based automatic feature learning method for the detection of dns tunnel," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[18] S. Jaworski, "Using splunk to detect dns tunneling," *SANS Institute InfoSec Reading Room*, 2016.

[19] K. Born and D. Gustafson, "Ngviz: detecting dns tunnels through n-gram visualization and quantitative analysis," *arXiv preprint arXiv:1004.4359*, 2010.

[20] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis." in *Ndss*, 2011, pp. 1–17.

[21] S. Lockington, "Detecting the bad from the good," 2012. [Online]. Available: http://scottfromsecurity.com/blog/2012/01/16/detecting-the-bad-from-the-good/

[22] A. Das, M.-Y. Shen, M. Shashanka, and J. Wang, "Detection of exfiltration and tunneling over dns," *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 737–742, 2017. [Online]. Available: http://dx.doi.org/10.1109/ICMLA.2017.00-71

[23] M. Himbeault, "A novel approach to detecting covert dns tunnels using throughput estimation," Thesis, 2014.

[24] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, "Flow-based detection of dns tunnels," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2013, pp. 124–135.

[25] Y. F. Mohammed and D. R. Thompson, "Visualization of dns tunneling attacks using parallel coordinates technique," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2019, pp. 89–101.

[26] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: Dga-based botnet tracking and intelligence," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 192–211.

[27] C. Wressnegger, G. Schwenk, D. Arp, and K. Rieck, "A close look on n-grams in intrusion detection: anomaly detection vs. classification," in

*Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, 2013, pp. 67–76.

[28] J. Ramos *et al.*, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the first instructional conference on machine learning*, vol. 242. New Jersey, USA, 2003, pp. 133–142.

[29] L. Rokach, “Ensemble-based classifiers,” *Artificial intelligence review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[30] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] G. Rätsch, T. Onoda, and K.-R. Müller, “Soft margins for adaboost,” *Machine learning*, vol. 42, no. 3, pp. 287–320, 2001.

[32] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.

[33] “Alexa web information company,” 2019. [Online]. Available: https://www.alexa.com/topsites

[34] K. Reitz, “requests: Python http for humans,” 2020. [Online]. Available: http://python-requests.org

[35] J. Kurkowski, “Python domain extraction library tldextract.” [Online]. Available: https://github.com/john-kurkowski/tldextract

[36] “Scikit-learn: Machine learning in python.” [Online]. Available: http://scikit-learn.org/stable/