IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 3, 2021, accepted April 19, 2021, date of publication June 2, 2021, date of current version June 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3085500

Application Behavior Identification in DNS Tunnels Based on Spatial-Temporal Information

HUIWEN BAI[1], WEIWEI LIU[1], (Member, IEEE), GUANGJIE LIU[2], YUEWEI DAI[2], AND SHUHUA HUANG[1], (Graduate Student Member, IEEE)

[1]School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China
[2]School of Electrics and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China

Corresponding author: Guangjie Liu (gjieliu@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant U1836104 and Grant 61702235, and in part by the Fundamental Research Funds for the Central Universities under Grant 30918012204.

ABSTRACT Due to the capability of passing through heavily censored networks or gateway equipped with the traffic-monitoring module, DNS tunnel has been the dominant covert communication technique for command and control between the victim and the attacker in network attack events. Although the discovery of DNS tunnel has been intensively studied, the internal application behavior identification for DNS tunnels still remains a challenging problem. The fine-gained identification can help to reveal more behavior information wrapped in DNS tunnels. In this study, we investigate the spatial-temporal information from the raw packets to identify the internal application behaviors in DNS tunnels. Multi-dimensional features on packet length and timing for DNS tunnels with different internal application behaviors are incorporated with a machine-learning algorithm to identify the internal application behaviors in DNS tunnels. We consider 4 common types of application behaviors in our research, including browsing webpages, emailing, downloading data, and controlling the remote servers. The experimental results show that the proposed scheme can achieve higher identification accuracy with a much lower packet consuming rate when compared with the state-of-the-art internal protocol identification scheme. The experiment results depict that our proposed scheme is better in terms of F-score, which can reach 99% with only 100 packets.

INDEX TERMS DNS tunnel, internal application behavior identification, spatial-temporal features, machine-learning algorithm.

I. INTRODUCTION

Domain Name System (DNS) is the infrastructure of the Internet. As a distributed database that maps domain names and IP addresses to each other, it enables users to access Internet more easily. For this reason, network cybersecurity and management equipment in boundaries usually have to permit DNS traffic. Unfortunately, it has made DNS protocol become the most suitable network traffic covers to build covert tunnels for hiding the network contents and behaviors. Recent years have witnessed the increasing use of stealthily prevalent DNS tunnels in malicious cyber activities, which can be used for command and control (C&C) channel and data exfiltration. In addition, DNS tunnels can be also employed as a manner to circumvent censorship from the enterprise or the states. The viability and performance of DNS-based covert channels was firstly discussed in ref [1]. It shows that tunneled TCP connections can achieve high throughput and low latency. Then, the botnet Feederbot [2] was exposed, the reverse engineering results show that it tunnels C&C in DNS traffic for the end-to-end communication between bots and the bot master. Thus, DNS-based C&C communication has posed a huge threat for network security.

In addition, DNS-based data exfiltration has also threatened the data security of Internet users. In ref [3], it is pointed out that DNS-tunnels have been leveraged by cyber criminals in cases involving the theft of millions of accounts. In recent FireEye report [4], a new variant of the point of sale (POS) malware family NewPosThings was exposed, which is called MULTIGRAIN. Its source code consists of some modules that slightly modified from NewPosThings. This variant is highly targeted and digitally signed, which can steal payment

The associate editor coordinating the review of this manuscript and approving it for publication was Varun Gupta.

card data via DNS tunnel. The function of DNS-based data exfiltration is newly added in this malware family. This new variant, once executed on the target host, will collect sensitive information including user information and card data, encode them with a custom Base32 encoding algorithm, and then makes a DNS query with this information to a hardcoded domain. The attacker will be thus notified with the encoded data. The decoded data, is always sufficient to attempt card fraud in most scenarios. Besides, some other POS malware families such as BernhardPOS and FrameworkPOS also use DNS tunnels for data exfiltration.

With the increasing threats caused by the abuse of DNS tunnels, the prevention of malicious network activities is a challenging work. Because the packet payload is usually encrypted in the tunnel for bypassed the inspection at network boundaries, it is very important for the administrator of the network to figure out what happens in the tunnel and whether the traffic should be allowed. For this reason, a fine-grained identification of user application behaviors hiding in DNS tunnels is necessary for the network management and the cybersecurity system.

There are several methods to detect and identify tunnel-like DNS traffic including character-based ones and statistics-based ones. Character-based methods rely on the distribution of characters in the domain name to detect the tunneling traffic. Statistics-based methods rely on information that can be obtained from packet header (e.g. bytes transmitted, packets inter-arrival times, and so on). They rely on packet header high-level features which makes them an available option to deal with encrypted payloads. These methods usually employ machine learning techniques to perform model training and abnormal DNS traffic detection. To make out what happened in DNS tunnels, there are some studies to identify the internal protocol in the DNS tunnel. It is first proposed to identify the protocol within DNS tunnels in ref [5]. And in ref [6], they employ three features (namely, IP packet length, DNS Query Name Entropy, and DNS Query Name Length) to identify the protocol within DNS tunnels. In ref [7], DNS tunneling traffic is detected and they also classify the type of DNS tunneling traffic, including FTP-DNS tunneling, HTTP-DNS tunneling, HTTPS-DNS tunneling, and POP3-DNS tunneling. However, only identifying the internal protocol is not fine-grained enough to help for revealing the he nature of the tunnel, and the performance of the identification is not expected. The malicious use of DNS tunnel is still on increasing [8], [9]. The application behavior identification(APBI) problem in DNS tunnels, remains a significant challenge, which can help to dig out more detail information wrapped in DNS tunnels. It is definitely benefit for network management or cyber security.

However, to the best of our knowledge, there is still an absence of APBI in DNS tunnels. In this paper, we proposed a scheme by constructing the rapid identification model of user application behavior hiding in DNS tunneling traffic. We focus on the spatial-temporal analysis of the DNS tunneling traffic, i.e. all features are based on the length of DNS query or answer domain and packet timestamp information. According to the consumption requirement from the practical implementation consideration, the spatial-temporal features are hoped to be extracted from the packet sequence with as short as possible. Therefore, we extract the feature from the flow fragments which are obtained from the complete DNS flow. The selected features are examined based on several commonly used machine learning algorithms, namely Decision Tree, Random Forest, and Bayes Net. A comparison of performance results for the three machine learning algorithms is given. For the data set captured from the current network environment, the Random Forest provide the best performance, with the F-score of 99%. Compared to the state-of-the-art scheme, the comparison results denote that our proposed scheme can achieve higher accuracy on the identification of internal user application behaviors in DNS tunnels. The main research contributions of this paper are as bellow.

1. We propose to classify the DNS tuneling traffic in fine-gained, namely identifying the internal application behaviors hiding in the DNS tunnels. We summarize the 4 common kinds of user application behaviors in DNS tunnels and propose an identification scheme to identify the DNS-tunneled behaviors based on the machine-learning methods.

2. We design a series of spatial-temporal features extracted from the fragmented packets and we employ the information gain ratio metrics to evaluate and select the features for classifying application behaviors hiding in the DNS tunnels. The identification method is applied to the fragment consisting of part of the DNS flow packets.

3. We develop several experiments on the collected actual network DNS tunneling traffic to evaluate the proposed scheme. In addition, we obtain the optimized classifier type and fragment size for identify different application behavior types hiding in DNS tunnels. The comparison experiments can show that the proposed feature set can obtain the superior performance compared to the state-of-the-art.

The remainder of this paper is organized as follows. In the next section, we summarize the related work on the detection of abnormal DNS traffic and internal protocol identification in tunnels. In Section 3, the main behaviors within DNS tunnels and their characteristics are presented. Moreover, the possible available spatial-temporal features are summarized and refined based on the information gain ratio metrics evaluation. Then we introduce the collecting dataset in our research in Section 4. Section 5 is devoted to evaluate the performance of the proposed scheme and the designed feature set. In Section 6, the conclusion is drawn and future work is discussed.

II. RELATED WORK

The detection of covert channels using the DNS has been studied for the past decade and many studies have been published. Some works rely on the character rules in the DNS packet to detect abnormal traffic. Born *et al.* [10] empirically show that normal DNS question names follow Zipf's law,

such as an English-like distribution, while they as tunnels show a much flatter distribution since tunneled data is often compressed or encrypted and then encoded for transmission. Born *et al.* [11] then develop an n-gram visualization called NgViz so that operators can use their spatial reasoning to quickly identify DNS tunnels from normal use. Qi *et al.* [12] calculate the bigram frequency to form a character frequency table to score DNS packets and detect the tunneling packets. Xu *et al.* [13] compute the byte distribution in normal and tunneling traces, and the Jensen-Shannon (JS) Divergence D_{JS} is used to quantify the difference between two probability distributions.

Preston [14] explores the application of supervised machine learning for analyzing DNS traffic. Das *et al.* [15] develop machine learning models to detect information exfiltration from compromised machines and the establishment of C&C servers via tunneling. Ahmed *et al.* [16] develop and evaluates a real-time machine-learning mechanism for detecting exfiltration and tunneling of data over DNS. They implement the scheme on live 10 Gbps traffic streams from the network borders of the two organizations. To avoid the manual feature extraction, deep learning methods are employed to detect DNS tunnels. Lai *et al.* [17] propose a feature free detection method using deep neural network to train DNS packets. Liu *et al.* [18] propose to use CNN architecture to detect DNS tunnels. Zhang *et al.* [19] use DNS query payloads as the predictive variable and use word embedding to process the input. Several common neural networks are employed as the classifier models. The one-model detection decision maker and the multi-model detection decision maker are built to make final decision. Campbell [20] *et al.* propose an unsupervised learning approach, namely Self-Organizing Maps, to explore the tunneling behaviors in DNS traffic. They perform evaluations on eight different combinations of datasets. Results show that the approach demonstrates a robust ability to separate benign and tunneling behaviors across a variety of training schemes.

To improve the detection rate of DNS tunnels, several methods based on flows instead of packet detection are proposed. Ellens *et al.* [21] combines flow information with statistical methods for anomaly detection. They employ and tailor threshold method, brodsky-darkhovsky method, and distribution-based method to implement five flow-based detectors for DNS tunnels. Aiello *et al.* [22] demonstrated that DNS tunnels produced by standard applications according to the signatures of Feederbot, Iodine, and DNS2TCP, host or network intrusion detection systems can detect the them. To detect previously unseen DNS tunnels without signatures, Liu *et al.* [23] propose an effective and applicable DNS tunnel detection mechanism which can analyze the DNS traffic of Recursive DNS (RDNS). They not only focus on the character distribution but also time frequency, DNS record types and DNS query length. Buczak *et al.* [24] proposed to detect DNS tunnels using Random Forest method with a mix of features designed to focus on evaluating single and multiple DNS packets. Before training the classifier, they employ the information gain and information gain ratio to select the discriminative features. Generating discriminative input features is a key requirement for achieving highly accurate classifiers. Davis *et al.* [25] propose automated feature engineering to derive a suite of additional features from a given set of basic features. Wu *et al.* [26] propose an automatic extraction feature method (TDAE) based on Autoencoder that can operate on a large amount of unlabeled data and quickly identify the traffic generated by the DNS tunneling tools. Most of the above methods are for the domains, TXT resource records (RRs) and other less common types of RRs. A/AAAA RR-based DNS tunnels have been ignored since they were newly developed. Luo *et al.* [27] extract features from the domains and 4 types of RRs from the amount of information and the content of information. After feature extraction, features are divided into 5 groups: one group is extracted from domains and applied to detect the potential exfiltrating traffic; the other 4 groups are extracted from the 4 types of RRs and used to detect the potential infiltrating traffic.

The analysis of internal behavior of DNS tunneling traffic is still scarce. Homem *et al.* [5] focus their attention on the protocol identification within DNS tunnels. The study was limited to the identification of two network protocols: HTTP and FTP. Two similarity values computed by Mean Differences are used to judge the protocol types. The accuracy of the protocol identification is about 75%. Homem *et al.* [6] also employed a combination of traffic features and classifiers to identify the protocol within DNS tunnels. Three features were extracted from DNS tunnel traffic: IP packet length, DNS Query Name Entropy, and DNS Query Name Length. With these features, k-Nearest Neighbors, Decision Trees, and Multinomial Neural Networks are used as classifier to identify the internal protocols. Almusawi *et al.* [7] propose a multi-label support vector machine in order to detect and classify the DNS tunneling. The proposed method was evaluated using a benchmark dataset that contains contained numerous DNS queries and was compared with a multi-label Bayesian classifier based on the number of corrected classified DNS tunneling instances. They classify the DNS tunnels as FTP-DNS tunneling, HTTP-DNS tunneling, HTTPS-DNS tunneling, and POP3-DNS tunneling.

In addition, there are some works to explore what was encapsulated in the TCP based tunnels. Bernaille *et al.* [28] propose the early recognition of encrypted applications with use of first few packets of an SSL connection. Alshammari and Zincir-Heywood [29] propose to identify Skype and SSH traffic by using classifier with C4.5 Adaboost and so on and the protocols (Shell, SCP, SFTP, etc.) were also distinguished within the SSH tunnel. Mujtaba *et al.* [30] propose to use packet stream statistics including packet size distribution to differentiate and identify applications tunneled in HTTP and SSL. Dusi *et al.* [31] propose to use a clustering algorithm based on Gaussian mixture models to application identification in SSL tunnel. Montieri *et al.* [32], [33] try to analyze whether the anonymity network being observed can be classified and, in affirmative case, whether the traffic

TABLE 1. Description of common application behaviors hiding in DNS tunnels.

Behavior type number	Description of internal user behaviors of DNS tunnels.
Type 1	Control the remote servers.
Type 2	Download data.
Type 3	Email.
Type 4	Browse the webpages.

type and application running hidden within them could be inferred. They consider four classifiers: two based on the Bayesian approach and other two based on decision trees. The obtained results show that anonymity networks can be easily discerned, and the traffic type and the service running within it can be accurately inferred as well (by a judicious use of the appropriate classifier). Bovenzi *et al.* [34] design a big data enabled hierarchical (BDeH) framework implementing double parallelism, i.e. that integrates the advantages of model parallelism given by hierarchical traffic classification [32], with those originated by data parallelism, provided by big data technologies.

To the best of our knowledge, for DNS traffic, it has not been proven whether statistics features obtained from the side channel can classify internal application behaviors and there is an absence of identification of internal user application behaviors hiding in DNS tunnels. To address this problem, we propose a method of spatio-temporal feature analysis to identify the application behaviors within DNS tunnels.

III. SPATIAL-TEMPORAL FEATURE ANALYSIS

In this section, we first describe our study object, namely the internal user application behaviors of DNS tunnels. Then we introduce the features for identifying different internal user behaviors in DNS tunnels, including feature analysis and feature selection scheme. At last, we introduce the selected spatial-temporal features extracted from network traffic flow fragments.

A. INTERNAL USER APPLICATION BEHAVIOR DESCRIPTION

Generally, a user employs the Internet to get information such as news, emails or other online readings exhibiting as structured documents, online audio/videos or livestreams, downloading files located at HTTP/FTP servers or cloud storage providers, file-sharing peers in a P2P network, and so on. He/she also employs the Internet to send information such as mails, online blogs or notes, uploading or synchronizing files, casting livestream, and so on. There are also some interactive communications in the form of text/audio/video messaging between humans, and remote controlling operations such as SSH and RDP, and other online cooperation activities such as collaborative documenting, online games, and so on.

With the consideration of the common use of covert tunnels, in this research, we summarized 4 types of network application behaviors as the first study of behavior identification within DNS tunnels. The description of application behaviors is shown in Table. 1.

B. DNS TUNNELING FLOW FRAGMENTING

The number of packets in an DNS flow is not fixed. It is determined by factors such as the size of the current transmission object, MTU, and MSS. Flow fragmentation is to divide each flow into several equal-length fragments, and each fragment is used as a sample. Assuming that each DNS tunneling flow contains only one type of behavior and the behavior distribution of each flow is uniform, namely, any continuous number of packets in the flow can reflect some characteristics of these behaviors. Fragmenting the data, on one hand, is to make the packet data amount of each sample consistent. On the other hand, it can reduce the computation overhead and improve the classification efficiency so as to achieve real-time identification requirements. The specific fragmenting rules are described as follows:

Assuming there are n packets in a flow excepting handshake packets, the flow can be denoted as $\{p_1, p_2, \ldots, p_n\}$. Each flow is divided according to the fragment length $intvl$:

If $n \leqslant intvl$, the flow is directly split to a fragment.

If $n > intvl$, the flow is split to $m = \lfloor \frac{n}{intvl} \rfloor$ fragments. They can be denoted as:

$$Flow = \{frag_1, frag_2, \ldots, frag_m\}, \tag{1}$$

where $frag_i = p_{intvl*i}, p_{intvl*i+1}, \ldots, p_{intvl*(i+1)-1}$.

The size of the fragment will determine the amount of information in the fragmented sample. Fragment with too small size (such as 1 packet for 1 fragment) will affect the accuracy of classification. Increasing the fragmenting size can increase classifying performance, however, it can also increase the computational cost of the classifier. To observe the impact of different segment sizes on the classification results, we will conduct comparison experiments under different fragment sizes in Section. V.

C. DNS TUNNELING TRAFFIC FEATURE ANALYSIS

Packet size is always an important information for analyzing network traffic. Different application behaviors generally show unique character patterns in the packet length sequence. In DNS tunnels, the packet size cannot be larger than a fixed number which is determined by the DNS server parameters, the DNS types and so on. Therefore, the transferred data generally conforms to a certain pattern depending on the application behaviors. Packet size, as one of the most important metadata in the observed side channel, is taken into our consideration to identify the type of behavior types in DNS tunnels. Since the user data is embedded in the DNS request and answer fields, we extract the query name and answer data of DNS packets for analysis. The advantage is that the application behavior inside the tunnel can be analyzed more accurately. After a simple analysis, we found that there are some differences of length features among traffic with different user application behaviors. The growth of the length of the data downloading behavior flows is most regular. On the contrary, there is basically no fixed rule about the growth of the length of webpage browsing behavior flows. And at the start of the SMTP stream, the growth of the length is

more regular, but it was not in a rule later. Although the length sequence has certain rules, it is difficult to distinguish the internal behavior accurately through intuitive observation. Therefore, we rely on machine learning method to classify the internal application behavior of DNS tunnels. By summarizing the existing relate work [33], we can obtain several static length features, namely: minimum, maximum, mean, median absolute deviation, standard deviation, variance, skew, kurtosis, percentiles.

In this research, besides of the above characteristics, we also analyze the content size distribution (CSD). The content of a DNS packet represents the query name or the answer. We consider $P = \{p_1, p_2, \ldots, p_i\}$ as the packet flow for $i = 1, 2, \ldots, n$, and consider $L = \{l_1, l_2, \ldots, l_i\}$ as the lengths of query names or answers of DNS packets. The packets with different user application behaviors in DNS tunnels will be put into 30 bins according to the size of query name or answer. Therefore, each bin will cover several bytes. We consider $B = \{b_1, b_2, \ldots, b_j\}$ as the bins for $j = 1, 2, \ldots, m$. Which bins the packets will be put in is determined as:

$$b_j = \frac{l_i}{w}, \tag{2}$$

where w is the number of bytes covered by each bin. The number of packets falling in a particular bin form the length distribution for the application behaviors.

In addition, the total size of the data send and received by the client, the proportion of big packets and small packets, and the number of changes in packet length were also analyzed here. These statistical features reflect the inherent characteristics of different application behaviors within DNS tunnels.

Timing is another important attribute in traffic analysis. The inter-arrival time (IAT) of network flow packets has its features with different internal user application behaviors in DNS tunnels. IAT may be in the vicinity of a fixed value for some behaviors but others may be not fixed at all. The IAT distribution of data downloading behaviors has the most obvious features than webpage browsing behaviors and it is relatively fixed. By summarizing the existing relate work [33], we can obtain several IAT features, namely: minimum, maximum, mean, median absolute deviation, standard deviation, variance, skew, kurtosis, percentiles.

In this research, IAT between packets in the same direction and between query and corresponding response packet are employed. Besides of above features, another feature were used, namely duration time of the flow fragment.

Character information, including frequency and semantics, is commonly used in DNS tunnel detection. In this paper, we try to use information entropy to describe character characteristics. The concept of entropy is first introduced in thermodynamics to describe the second law of thermodynamics. Boltzmann studied the relationship between thermodynamic entropy and the number of microstates, and gave the formula:

$$s = k \log W. \tag{3}$$

TABLE 2. Summary of the selected statistic features.

Feature name	Feature description
contentSize_corr	Statistic value, including distribution, minimum, maximum, mean, variance, skew, kurtosis.
pack_ratio	Proportion of different types of packets, such as large packets: small packets.
IAT_corr	Statistic value, including duration, minimum, maximum, mean, variance, skew, kurtosis.
contentEntropy	Static values of content entropy, including, mean, variance.

The entropy of information is defined in terms of the entropy of thermodynamics. They are not the same thing but have some connection. Entropy is a measure of the uncertainty of random variables in information theory. The entropy (H) of a discrete random variable (X) is defined as:

$$H = -\sum_{i}^{m} p_i log(p_i). \tag{4}$$

There has a fixed number m of possible events $A_1, \ldots, A_m$ whose probabilities of occurrence $p_1, \ldots, p_m$ are known. Two things within data processing result in a high value of entropy. Firstly, data compression, as the bits needed for data representation should be minimized. Secondly, data encryption, as any predictable behavior available in the source data has to be removed. Both processing steps end with a data stream with equal probabilities for each event/symbol.

We summarize the selected statistic features in Table. 2.

D. FEATURE SELECTION METHOD

Feature selection is an important issue in the model generation for ML algorithms. On one hand, too many features can cause overfitting [37], and on the other hand, irrelevant features can have a negative effect on the training process, such as the increasing of training samples [38]. Meanwhile, the irrelevant features will also cause unnecessary computation expense. Assuming we have a feature set $X = (x_1, x_2, \ldots, x_n)$, the feature selection can be denoted as select a subset X_op from X according to the evaluation criteria. There are several approaches to select features [39]–[42], and we chose a typical approach, namely information gain (IG) and information gain ratio (IGR). Assuming there is a sample set D, for feature A, its information gain can be denoted as:

$$g(D, A) = H(D) - H(D|A), \tag{5}$$

where $H(D)$ is empirical entropy of sample set D. The empirical entropy can be denoted as:

$$H(D) = -\sum_{i=1}^{k} \frac{|c_k|}{|D|} \log \frac{|c_k|}{|D|}, \tag{6}$$

where $|C_k|$ is the sample number of class k, $|D|$ is the total number of samples.

In Eq.(5), $H(D|A)$ is the conditional entropy of the sample set D given feature A. It can be denoted as:

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i)$$

$$= -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{m} \frac{|D_{ik}|}{|D|} \log \frac{|D_{ik}|}{|D|}, \quad (7)$$

where $|D_i|$ is the number of samples of the i^{th} value of feature A, $|D_{ik}|$ is the number of samples that belong to subset C_k in D_i.

IG can be used to determine the importance of features for predicting an output label. Since the information gain tends to be characterized by more values, more values may not mean more information. Therefore, the information gain ratio adds a penalty parameter on the basis of the information gain to make up for the defect of the IG. The formula of IGR is denoted as:

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}, \quad (8)$$

where $H_A(D)$ is the reciprocal of the penalty parameter. It can be denoted as:

$$H_{A(D)} = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|}, \quad (9)$$

where $|D|$ is the total number of samples and $|D_i|$ is the number of samples of the i^{th} value of feature A.

E. FEATURE EXTRACTION AND SELECTION

We combine the discovered features of DNS tunneling traffic and the possible traffic behaviors of different application categories for further analysis. Finally, 62 features were present. The IG and IGR of each feature of the sample data are calculated by the feature selection method offered in Section 3.2.4. The results are shown in Table. 3.

Histogram of IG and IGR is shown in Fig. 1.

We select the features that IG value or IGR value is bigger than the average value. The IG average value is 0.68387 and the IGR average value is 0.13538. Finally, the features: {1, 2, 3, 9, 15, 16, 19, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 47, 48, 49, 50, 57, 58, 59, 60, 61, 62} are selected by IG and IGR methods. And these 37 features are finally employed to identify the behaviors within DNS tunnels.

IV. DATASET COLLECTION

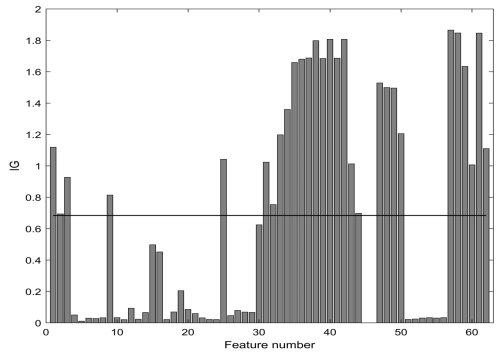
In this section, we first describe the experimental environment used for data collection. Then we introduce the process of collecting data sets.

A. EXPERIMENTAL ENVIRONMENT ESTABLISHMENT

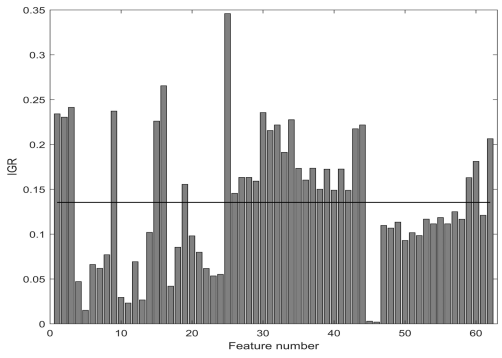
To the best of our knowledge, there are no well-known DNS tunnel network traffic captures available. To this end, we employ two commonly used tools, namely Iodine and tcp2dns, to build up two DNS tunnels for our research, and the network topology is shown in Fig. 2. The first tunnel was built between an Amazon EC2 cloud server located in Dublin, Ohio, USA, and the computer which has access to the Internet via the campus network in Nanjing University of Science and Technology (NJUST). The second tunnel was

TABLE 3. IG and IGR of the extracted features.

No.	feature	feature description	IG	IGR
1-30	CSD	The distribution of the size of DNS content.	0.2444	0.1266
31	min size of s2c	The minimum of bytes sent by server to client.	1.0239	0.2156
32	min size of c2s	The minimum of bytes sent by client to server.	0.7537	0.2218
33	max size of s2c	The maximum of bytes sent by server to client.	1.1980	0.1913
34	max size of c2s	The maximum of bytes sent by client to server.	1.3599	0.2276
35	average of s2c	The average of bytes sent by server to client.	1.6591	0.1734
36	average of c2s	The average of bytes sent by client to server.	1.6803	0.1604
37	variance of s2c	The variance of bytes sent by server to client.	1.6876	0.1736
38	variance of c2s	The variance of bytes sent by client to server.	1.7980	0.1501
39	skewness of s2c	The skewness of bytes sent by server to client.	1.6842	0.1725
40	skewness of c2s	The skewness of bytes sent by client to server.	1.8069	0.1490
41	kurtosis of s2c	The kurtosis of bytes sent by server to client.	1.6865	0.1726
42	kurtosis of c2s	The kurtosis of bytes sent by client to server.	1.8069	0.1489
43	big: small in s2c	The proportion of big packets and small ones sent by server to client.	1.0130	0.2175
44	big : small in c2s	The proportion of big packets and small ones sent by client to server.	0.6982	0.2218
45	min IAT in s2c	The minimum of IAT in packets sent by server to client.	0.0001	0.0030
46	min IAT in c2s	The minimum of IAT in packets sent by client to server.	0.0001	0.0020
47	max IAT in s2c	The maximum of IAT in packets sent by server to client.	1.5285	0.1097
48	max IAT in c2s	The maximum of IAT in packets sent by client to server.	1.4993	0.1068
49	average IAT in s2c	The average of IAT in packets sent by server to client.	1.4956	0.1135
50	average IAT in c2s	The average of IAT in packets sent by client to server.	1.2055	0.0930
51	variance IAT in s2c	The variance of IAT in packets sent by server to client.	0.0220	0.1017
52	variance IAT in c2s	The variance of IAT in packets sent by client to server.	0.0244	0.0985
53	skewness IAT in s2c	The skewness of IAT in packets sent by server to client.	0.0306	0.1186
54	skewness IAT in c2s	The skewness of IAT in packets sent by client to server.	0.0330	0.1115
55	kurtosis IAT in s2c	The kurtosis of IAT in packets sent by server to client.	0.0306	0.1186
56	kurtosis IAT in c2s	The kurtosis of IAT in packets sent by client to server.	0.0330	0.1115
57	duration in s2c	Duration time of the first packet to the last packet in the fragment.	1.8651	0.1250
58	duration in c2s	Duration time of the first packet to the last packet in the fragment.	1.8469	0.1167
59	average entropy of s2c	The average of the entropy of packets sent by server to client.	1.6341	0.1630
60	variance entropy of s2c	The variance of the entropy of packets sent by server to client.	1.0066	0.1812
61	average entropy of c2s	The average of the entropy of packets sent by client to server.	1.8465	0.1211
62	variance entropy of c2s	The variance of the entropy of packets sent by client to server.	1.1099	0.2064



(a) IG of the designed features.



(b) IGR of the designed features.

FIGURE 1. IG and IGR of the designed features.

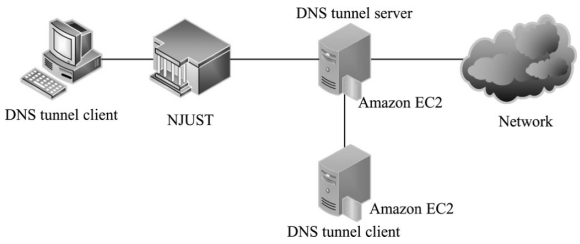


FIGURE 2. Network topology of experiment environment.

built between the Amazon EC2 cloud server and another Amazon EC2 server.

The principle of the two applications is similar, but there are some differences in the implementation. Iodine can forward all the IP packets while tcp2dns only deal with packets with the TCP protocol. The upstream data is GZIP compressed and encoded. The upstream encoding is auto detected by system, and several encoding methods (e.g., Base32, Base64, and so on) can be selected. There is a maximum of 256 bytes in each DNS request in which domain name can contain up to 63 bytes. There are several DNS request types, with the NULL and PRIVATE types expected to provide the largest bandwidth for downstream. The PRIVATE type uses value 65399 in the private-use range. Other available types are TXT, SRV, MX, CNAME and A (returning CNAME). Normally, the most suitable request type is automatically detected and used. The DNS response for

TABLE 4. Simulation of common user behaviors in DNS tunnels.

No.	User behaviors description	Simulated behaviors
1	Remote server controlling	Control remote server by employing SSH protocol.
2	Data downloading	Download data while limiting data upload.
3	Emailing	Send and receive text, pictures, or file attachments from time to time.
4	Webpage Browsing	Browse graphic websites, use search engines.

non-NULL/PRIVATE queries can be encoded with the same set of codecs as upstream data. The DNS tunneling based user-data fragments tunneled were encapsulated in the name field behind a header, prepended as a sub-domain of the tunneling server. The header preceding the tunneled data contains the data including the user id, the codecs, the fragment size, the fragment number, the sequence number, whether compression is used and a Cache Miss Counter. In this paper, NULL and TXT type were set for DNS tunnels. With the particular Resource Record, the encapsulated data was held in the Query/Answer name field. The DNS-response fragment size was normally auto-probed to get maximum bandwidth. Here they were selected because of their popularity, ease of use and availability of documentation.

B. DATA COLLECTION

The simulated behaviors corresponding to various behaviors are shown in Table. 4. To simulate webpages browsing behaviors, more than 200 websites with different types (including shopping, news, education, finance, government, corporate websites, travel, libraries, etc.) were visited by the browser. To simulate online video watching behaviors, we watch online videos through different video sites. To simulate remote server controlling behaviors, different remote controlling commands were employed to control the remote computer. To simulate data downloading, we. download files located at HTTP/FTP servers or cloud storage providers and "scp" command is also employed. To simulate emailing behaviors, we send and receive email through several mailboxes.

The training set samples in the experiment contain four user behaviors that are shown in Table. 1. The amount of the collected data for each behavior category is shown in Table 5. To capture the traffic, the software, Wireshark was employed. And the result of the trace file of traffic was saved into some pcap files. We simulate four types of commonly user behaviors, and all the network behaviors were finished in a DNS tunnels individually. As shown in the Table. 5, there are 1212 DNS flows, 4.98GB traffic data in total in the dataset.

In our research, the C++ program is designed to process each packet in the captured DNS flow, and spatial-temporal features are then extracted from the flow fragments. The feature data are finally saved in files with form of comma-separated format which can be directly converted into csv format files for further use. In the csv file,

TABLE 5. Dataset of DNS tunnel.

Internal behavior description	Number of DNS flows	Size of DNS traffic (MB)
Browsing webpages	198	1395
Control remote server.	587	1256
Email.	227	1443
Download data.	200	1008
Total	1212	5102

each row represents an instance for traffic with a kind of protocol in DNS tunnels, and the labels in the top of the columns are the attributes of the traffic. In addition, we will make our collected dataset publicly available online. As the dataset is too big to upload to Internet, we have processed the dataset and extracted the length, timestamp and direction of each packet. The online address is: https://github.com/Kevin0726/DNSTunnel/.

V. EXPERIMENTS AND EVALUATIONS

In this section, to evaluate the performance of the scheme for identifying application user behaviors in DNS tunnels, we did several experiments with the dataset introduced in Section 4. At first, we introduce the evaluation metrics for the experiment result.

A. EVALUATION METRICS

To evaluate the identification effectiveness of the proposed scheme, the following terms are used for determining the quality of the behavior identification model:

True Positive (TP): the number of behavior A samples correctly identified to the behavior A.

True Negative (TN): the number of samples that are not type A identified as other types.

False Positive (FP): the number of behavior A samples wrongly identified to the other types.

False Negative (FN): the number of samples that are not type A wrongly identified as behavior type A.

Based on the aforementioned terms, the following most commonly used evaluation metrics are considered.

Accuracy: It estimates the ratio of the correctly recognized network traffic flows to the entire test dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{10}$$

Precision: It estimates the ratio of the correctly identified network traffic flows to the total number of samples classified to network class. It is denoted as

$$Precision = \frac{TP}{TP + FP}. \tag{11}$$

Recall: It estimates the ratio of the correctly identified network traffic flows to the number of all network traffic flows. It is also called True Positive Rate (TPR). It is denoted as

$$Recall = \frac{TP}{TP + FN}. \tag{12}$$

TABLE 6. The number of samples for each fragment size.

Fragment size(packets)	10	40	70	100	130	160	200
Type 1 (samples)	201319	50277	28705	20075	15428	12524	10008
Type 2 (samples)	224903	56170	32056	22412	17224	13980	11164
Type 3 (samples)	266648	66573	37979	26555	20405	16559	13222
Type 4 (samples)	254740	63627	36325	25403	19520	15846	12658
Total size	947610	236647	135065	94445	72577	58909	47052

$F_{1-Score}$: It is the harmonic mean of Precision and Recall. It is denoted as

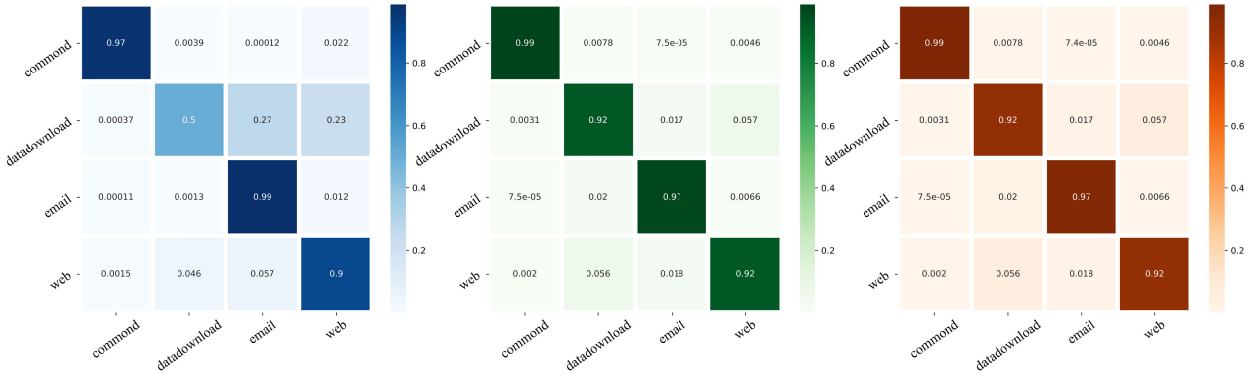
$$F_{1-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{13}$$

B. EXPERIMENT RESULTS

To evaluate the performance of the scheme for identifying the application behaviors in DNS tunnels, we select three different machine learning algorithms. Common machine learning classification algorithms include decision trees, random forests, support vector machines, Bayesian classification, and neural networks. The support vector machine (SVM) learning strategy is to find a hyperplane in the feature space. This hyperplane separates two classes and maximizes the interval between the two classes [43], which is more suitable for binary classification scenario. The accuracy of neural networks depends on the huge amount of data, and multi-layer neural networks are expensive, and they are not suitable for the multi-class experiments in this paper.

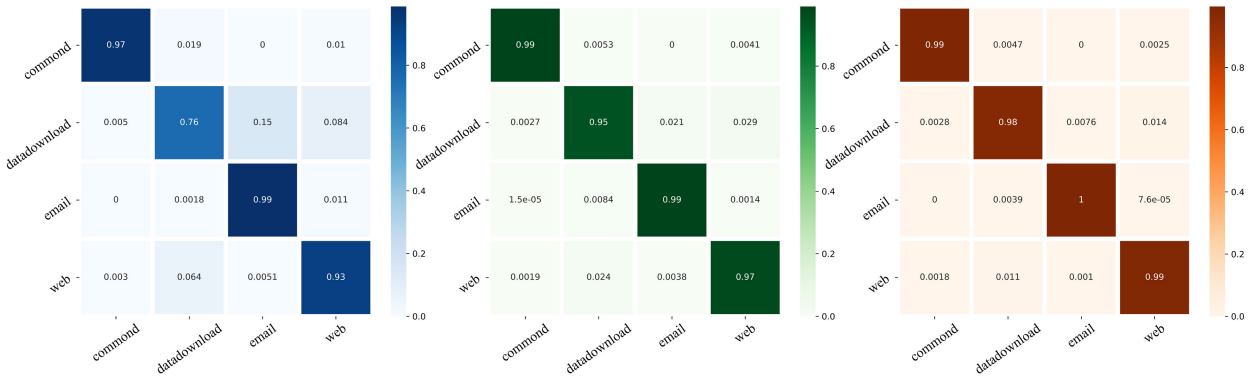
In addition, the fragment size is another important impactor in our research. Seven types of fragments size are selected and they are 10, 40, 70, 100, 130, 160, and 200. The number of samples corresponding to each fragment size is shown in Table 6.

We employ the three types of algorithms to train the models on the seven types of data sets respectively. And we use ten-fold cross-validation to examine the classification effect of each model. The performance details for each case is shown in the confusion heatmaps in Fig. 3-9. Each figure contains the confusion matrix heatmap of three algorithms in a certain fragment size. For each heatmap, the darker is the color on the diagonal, the higher is the *TP* value. Non-diagonal elements in each line represent the *FN* value, and the non-diagonal elements in each column represent *FP* value. From these figures, the comparison results are obvious that Bayes Net gets the worst performance and Random Forest obtains the best performance. In addition, as the fragment size increasing, the identification performance of each algorithm is also increasing. In the following subsections, we compare the classification performance for different algorithms and different fragment sizes. In details, we also analyze the impact of different algorithms and segment sizes for different application behavior types.



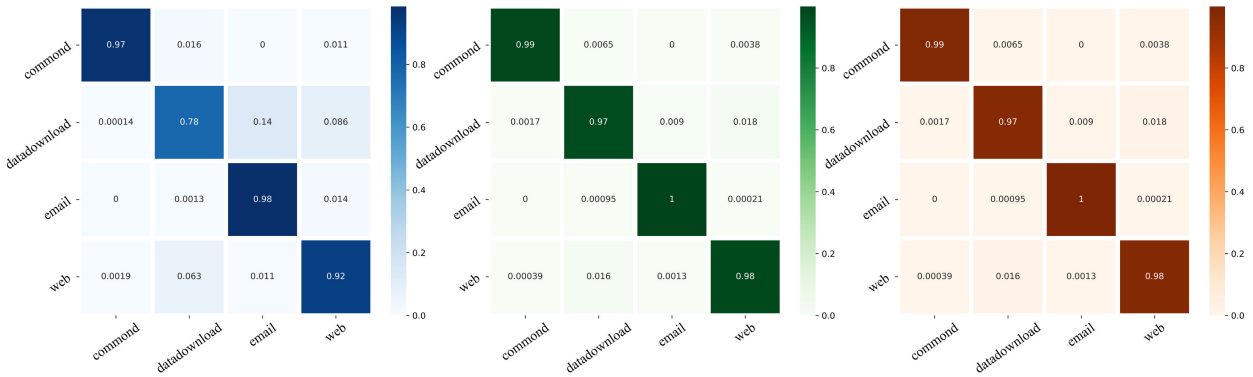
(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 3. Confusion matrix heat map with fragment size 10 with three algorithms.



(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 4. Confusion matrix heat map with fragment size 40 with three algorithms.

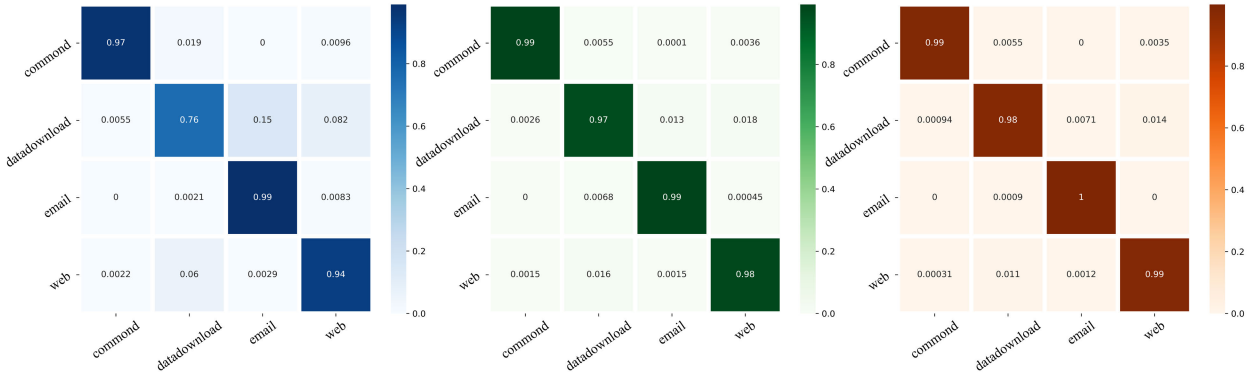


(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 5. Confusion matrix heat map with fragment size 70 with three algorithms.

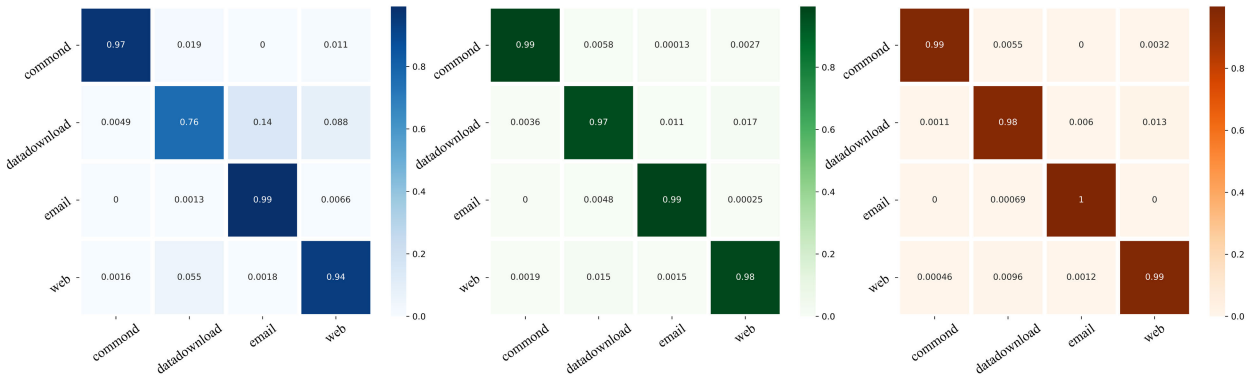
C. COMPARISON RESULTS BETWEEN THREE MACHINE LEARNING ALGORITHMS

This section compares the classification effect of Decision Tree, Random Forest, and Bayes Net on the same sample-set. The sample set is the training set collected in Section 4.2. Bayes Net is a general term for a class of classification methods based on the Bayesian formula. The simplest is Naive Bayes, which calculates the types of points to be judged based on conditional probability, and is a relatively easy to understand model. But it requires that each feature must be kept



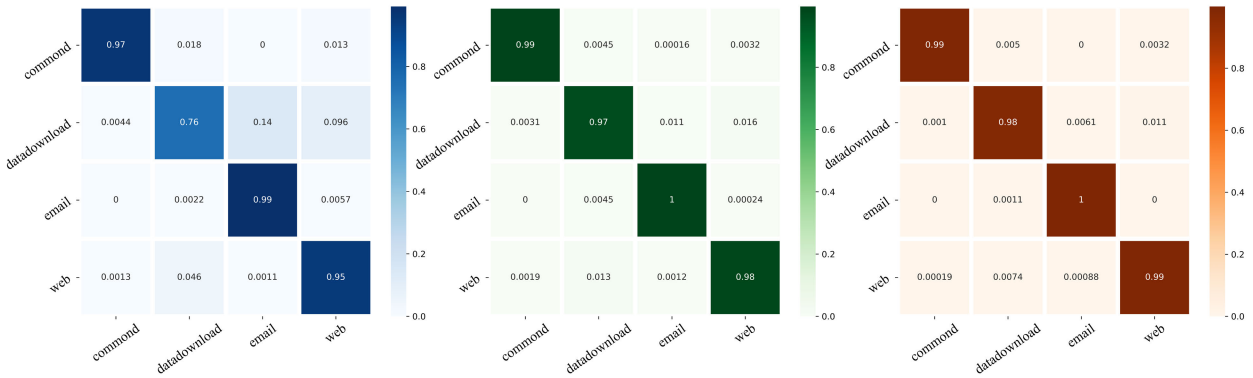
(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 6. Confusion matrix heat map with fragment size 100 with three algorithms.



(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 7. Confusion matrix heat map with fragment size 130 with three algorithms.

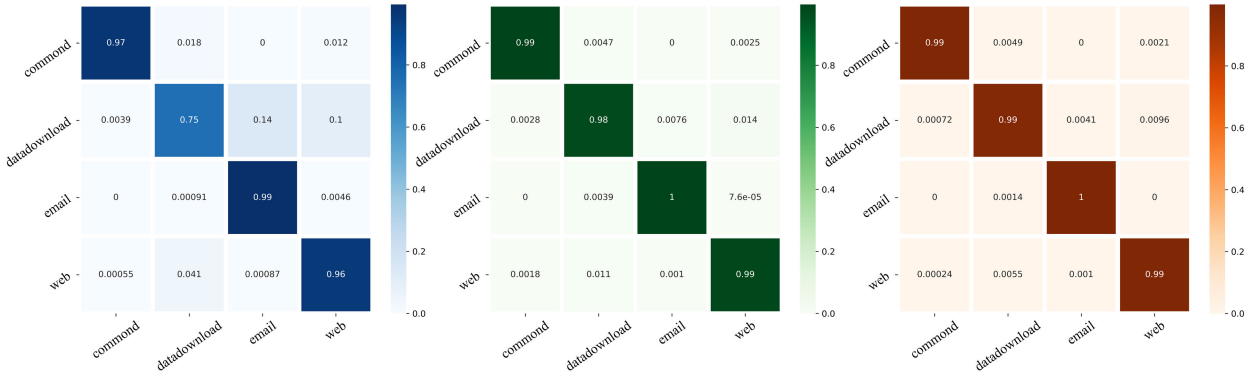


(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 8. Confusion matrix heat map with fragment size 160 with three algorithms.

relatively independent. Such a condition is almost impossible to achieve in the actual classification scenario. The Bayes Net is more commonly used in Bayesian classification. It is an uncertainty processing model that simulates causality in human reasoning and can efficiently process large-scale data.

Decision Tree is also one of the most commonly used classifiers. Its training process is the process of building a binary tree. Each non-leaf node in the tree represents a decision on a certain attribute of the sample, and each subsequent branch of the node corresponds to a possible value of the attribute.



(a) Bayes Net. (b) Decision Tree. (c) Random Forest.

FIGURE 9. Confusion matrix heat map with fragment size 200 with three algorithms.

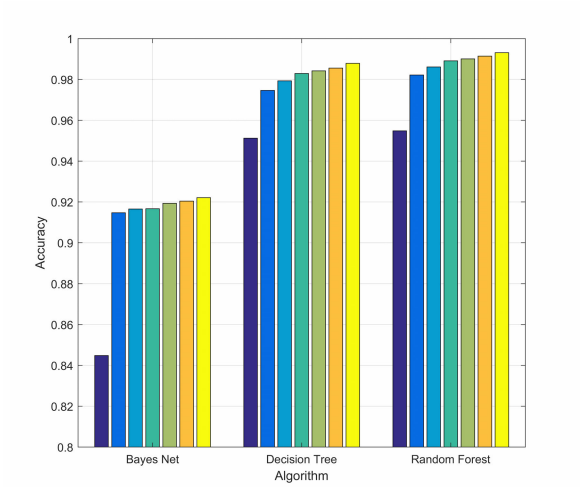


FIGURE 10. F-score comparison of identification with the three machine learning algorithms, Decision Tree, Bayes Net, and Random Forest.

The leaf node is the classification to which the sample belongs. The classification starts from the root node of the binary tree, moves to the corresponding branch according to the attribute value corresponding to the instance, and then repeats this process on the node of the new branch. The model building process of the decision tree does not rely on domain knowledge, the algorithm is simple and clear, and it can process a large amount of data in a short time. Random Forest is an integrated algorithm, which belongs to the type of Bagging (Bootstrap Aggregating, Bootstrap Aggregation). Random Forest is to use multiple decision tree classifiers, and when generating each tree, randomly select a few features from the existing features for modelling. By voting or averaging all classification results, the overall model results have higher accuracy and generalization performance. The corresponding evaluation values (including Precision, Recall, and $F_{1-Score}$) are shown in Table. 7.

To facilitate the comparative analysis of the identification effects of the three machine learning algorithms, we put the accuracy of the three machine learning algorithms into Fig. 10.

From Fig. 3-9, Random Forest obtains the best accuracy among the three machine learning algorithms in all the fragment size cases. The identification accuracy of the Bayes Net is relatively the worst. From Tab. 7, the F-score value of Bayes Net in cases of behavior type 2, 3, and 4 is obviously lower than the other two algorithms, and only type 1 is basically flat. The reason is that the character of the 4th type behavior is more discriminative compared to the other types. The remote command and control behavior is a type of interactive behavior. In terms of time features, its characteristics depend on the user's operation habits, including operation speed, etc. And in terms of length characteristics, it differs from web browsing and data downloading in that there are seldom large and continuous payloads of data packets. However, for behavior type 2 and 4, the performance is relatively lower than the others in each algorithm. The reason is that the behavior data downloading and web browsing is similar in some characters. Both of them may have large and continuous packets from server to client. And in temporal characters, they may complete a conversation as soon as possible. Therefore, they may have the similar static feature values. For all the cases, both Decision Tree and Random Forest obtain the performance with higher F-score than Bayes Net. The identification results of Decision Tree and Random Forests is similar, and their F-score values are above 97% for all the behavior types with fragment size value more than 100.

D. IMPACT OF FRAGMENT SIZE FOR IDENTIFICATION PERFORMANCE

A comparison of the accuracy values under each fragment size is shown in Fig. 11. With the increase of fragment size, the accuracy of different identification models is becoming higher. The identification performance of the model with the fragment size of 10 is obviously inferior to other fragment size. The Bayes Net, Decision Tree, and Random Forest models with fragment size 10 obtains the accuracy of 84.5%, 95.1%, and 95.5%, respectively. When the fragment size is increasing to 40, the accuracy is increasing a lot. The Bayes Net, Decision Tree, and Random Forest models with

TABLE 7. Evaluation metrics of scheme for identifying user application behaviors in DNS tunnels.

Fragment size	scheme	Evaluation metrics	Type 1	Type 2	Type 3	Type 4
10	Bayes Net	precision	0.998	0.897	0.778	0.796
		recall	0.974	0.504	0.986	0.895
		f-score	0.986	0.646	0.870	0.843
	Decision Tree	precision	0.994	0.908	0.969	0.938
		recall	0.987	0.923	0.974	0.924
		f-score	0.991	0.915	0.971	0.931
	Random Forest	precision	0.994	0.913	0.972	0.942
		recall	0.988	0.927	0.976	0.930
		f-score	0.991	0.920	0.974	0.936
40	Bayes Net	precision	0.990	0.893	0.884	0.908
		recall	0.971	0.764	0.987	0.928
		f-score	0.980	0.823	0.933	0.918
	Decision Tree	precision	0.995	0.958	0.979	0.969
		recall	0.991	0.947	0.990	0.970
		f-score	0.993	0.952	0.984	0.970
	Random Forest	precision	0.998	0.972	0.984	0.977
		recall	0.991	0.960	0.998	0.978
		f-score	0.995	0.966	0.991	0.977
70	Bayes Net	precision	0.997	0.900	0.886	0.903
		recall	0.973	0.776	0.985	0.925
		f-score	0.985	0.833	0.933	0.914
	Decision Tree	precision	0.993	0.964	0.985	0.976
		recall	0.989	0.958	0.992	0.976
		f-score	0.991	0.961	0.989	0.976
	Random Forest	precision	0.998	0.975	0.991	0.981
		recall	0.990	0.972	0.999	0.983
		f-score	0.994	0.974	0.995	0.982
100	Bayes Net	precision	0.991	0.897	0.883	0.913
		recall	0.971	0.761	0.990	0.935
		f-score	0.981	0.823	0.933	0.924
	Decision Tree	precision	0.995	0.968	0.988	0.981
		recall	0.991	0.967	0.993	0.981
		f-score	0.993	0.968	0.990	0.981
	Random Forest	precision	0.999	0.981	0.993	0.985
		recall	0.991	0.978	0.999	0.987
		f-score	0.995	0.979	0.996	0.986
130	Bayes Net	precision	0.992	0.904	0.889	0.910
		recall	0.970	0.763	0.992	0.941
		f-score	0.981	0.827	0.938	0.925
	Decision Tree	precision	0.994	0.972	0.989	0.982
		recall	0.991	0.968	0.995	0.981
		f-score	0.992	0.970	0.992	0.982
	Random Forest	precision	0.998	0.983	0.994	0.986
		recall	0.991	0.980	0.999	0.989
		f-score	0.995	0.981	0.997	0.987
160	Bayes Net	precision	0.993	0.914	0.891	0.904
		recall	0.968	0.758	0.992	0.951
		f-score	0.981	0.828	0.939	0.927
	Decision Tree	precision	0.994	0.976	0.990	0.983
		recall	0.992	0.970	0.995	0.984
		f-score	0.993	0.973	0.992	0.984
	Random Forest	precision	0.999	0.986	0.994	0.988
		recall	0.992	0.982	0.999	0.992
		f-score	0.995	0.984	0.996	0.990
200	Bayes Net	precision	0.995	0.922	0.891	0.903
		recall	0.971	0.752	0.994	0.958
		f-score	0.983	0.829	0.940	0.929
	Decision Tree	precision	0.995	0.979	0.993	0.986
		recall	0.993	0.976	0.996	0.987
		f-score	0.994	0.977	0.994	0.986
	Random Forest	precision	0.999	0.988	0.996	0.990
		recall	0.993	0.986	0.999	0.993
		f-score	0.996	0.987	0.987	0.992

fragment size 10 obtains the accuracy of 91.6%, 97.5%, and 98.2%, respectively. The accuracy increases 7.1%, 2.4%, and 2.7%, for the three algorithms respectively. And when the fragment size continues to increase, the accuracy is increasing slowly. When the fragment size increase from 40 to 200, the accuracy increases about 1%.

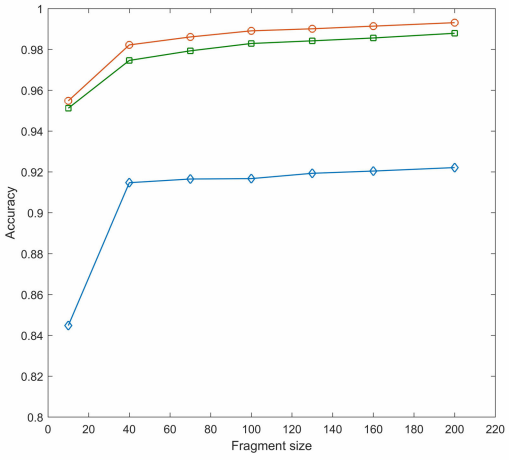


FIGURE 11. Impact of the identification performance using different fragment size.

As the fragment size increasing, less samples can be used for training and testing. And in the current network environment, the larger the size is, the more packets need to be cached, namely the more hardware resources and time resources are consumed. Therefore, on the premise that we have obtained a suitable fragment size, we give up continuing to find the larger fragment size. On the other hand, the use of smaller fragment size also helps to improve the speed and reduce the consumption of hardware resources in the detection in the real network environment.

E. COMPARISON EXPERIMENTS WITH THE STATE-OF-THE-ART METHOD

In our scheme, generating discriminative features is a key requirement for achieving high performance. Therefore, we design the spatial-temporal feature set to improve the resolution of different behaviors in DNS tunnel through analysis, such as heartbeat packet ratio and length distribution feature. To prove the effective of the proposed feature set, we conduct a comparison experiment with features mentioned in [7] and [32]. As in these two works, they also study the problem of hierarchical identification of tunneling traffic. For feature set 1 from ref [7], the feature set consisting of the request length, IP packet sender length, IP packet response length, encoded DNS query name length, request application layer entropy, IP packet entropy, and query name entropy, is extracted. For feature set 2 from [32], we extract length and timing static features. And the feature set 3 represents our designed feature set. For each feature set, nine experiments are developed in our research. We select three fragment size including 40, 100, 200, respectively and three machine learning algorithms including Bayes Net, Decision Tree, and Random Forest, respectively. And the comparison detail description is shown in Table. 8, Table. 9, and Table. 10.

According to the comparison experiments, our proposed feature set is more discriminative to identify behaviors

TABLE 8. Comparison experiment details with Bayes Net for identifying the user application behaviors in DNS tunnels.

Fragment size	Feature set	Evaluation metrics	Type 1	Type 2	Type 3	Type 4
40	feature set 1	precision	0.974	0.801	0.836	0.570
		recall	0.707	0.396	0.879	0.917
		f-score	0.820	0.530	0.857	0.703
	feature set 2	precision	0.979	0.735	0.900	0.915
		recall	0.971	0.838	0.936	0.768
		f-score	0.975	0.783	0.917	0.835
	feature set 3	precision	0.990	0.893	0.884	0.908
		recall	0.971	0.764	0.987	0.928
		f-score	0.980	0.823	0.933	0.918
100	feature set 1	precision	0.585	0.831	0.899	0.744
		recall	0.886	0.523	0.876	0.704
		f-score	0.705	0.642	0.887	0.724
	feature set 2	precision	0.992	0.792	0.859	0.924
		recall	0.972	0.764	0.962	0.853
		f-score	0.982	0.778	0.908	0.887
	feature set 3	precision	0.991	0.897	0.883	0.913
		recall	0.971	0.761	0.990	0.935
		f-score	0.981	0.823	0.933	0.924
200	feature set 1	precision	0.634	0.865	0.888	0.887
		recall	0.964	0.628	0.899	0.725
		f-score	0.765	0.728	0.893	0.798
	feature set 2	precision	0.997	0.791	0.850	0.958
		recall	0.970	0.762	0.988	0.846
		f-score	0.983	0.776	0.914	0.899
	feature set 3	precision	0.995	0.922	0.891	0.903
		recall	0.971	0.752	0.994	0.958
		f-score	0.983	0.829	0.940	0.929

TABLE 9. Comparison experiment details with Decision Tree for identifying the user application behaviors in DNS tunnels.

Fragment size	Feature set	Evaluation metrics	Type 1	Type 2	Type 3	Type 4
40	feature set 1	precision	0.989	0.849	0.856	0.628
		recall	0.726	0.472	0.950	0.933
		f-score	0.837	0.606	0.901	0.750
	feature set 2	precision	0.976	0.935	0.963	0.947
		recall	0.967	0.921	0.978	0.951
		f-score	0.971	0.928	0.970	0.949
	feature set 3	precision	0.995	0.958	0.979	0.969
		recall	0.991	0.947	0.990	0.970
		f-score	0.993	0.952	0.984	0.970
100	feature set 1	precision	0.991	0.869	0.932	0.679
		recall	0.753	0.644	0.955	0.945
		f-score	0.856	0.740	0.943	0.791
	feature set 2	precision	0.982	0.946	0.975	0.962
		recall	0.969	0.949	0.980	0.966
		f-score	0.975	0.948	0.977	0.964
	feature set 3	precision	0.995	0.968	0.988	0.981
		recall	0.991	0.967	0.993	0.981
		f-score	0.993	0.968	0.990	0.981
200	feature set 1	precision	0.988	0.728	0.944	0.899
		recall	0.797	0.920	0.957	0.814
		f-score	0.882	0.813	0.951	0.854
	feature set 2	precision	0.978	0.953	0.978	0.970
		recall	0.973	0.951	0.985	0.968
		f-score	0.976	0.952	0.981	0.969
	feature set 3	precision	0.995	0.979	0.993	0.986
		recall	0.993	0.976	0.996	0.987
		f-score	0.994	0.977	0.994	0.986

in DNS tunnels than the other two data sets. For both precision and recall, our proposed feature set is superior. Feature set 1 has the worst performance since the features employed in [7] are not distinguishability enough to distinguish the different behaviors in the DNS tunnels. And the features they utilized is mainly the classification of DNS

TABLE 10. Comparison experiment details with Random Forest for identifying the user application behaviors in DNS tunnels.

Fragment size	Feature set	Evaluation metrics	Type 1	Type 2	Type 3	Type 4
40	feature set 1	precision	0.989	0.845	0.858	0.629
		recall	0.726	0.474	0.952	0.933
		f-score	0.837	0.608	0.903	0.752
	feature set 2	precision	0.990	0.947	0.978	0.959
		recall	0.989	0.932	0.996	0.954
		f-score	0.990	0.939	0.987	0.957
	feature set 3	precision	0.998	0.972	0.984	0.977
		recall	0.991	0.960	0.998	0.978
		f-score	0.995	0.966	0.991	0.977
100	feature set 1	precision	0.990	0.878	0.934	0.681
		recall	0.753	0.645	0.964	0.947
		f-score	0.856	0.744	0.949	0.792
	feature set 2	precision	0.994	0.965	0.986	0.975
		recall	0.990	0.959	0.994	0.975
		f-score	0.992	0.962	0.990	0.975
	feature set 3	precision	0.999	0.981	0.993	0.985
		recall	0.991	0.978	0.999	0.987
		f-score	0.995	0.979	0.996	0.986
200	feature set 1	precision	0.988	0.738	0.947	0.904
		recall	0.799	0.921	0.976	0.815
		f-score	0.883	0.819	0.961	0.857
	feature set 2	precision	0.993	0.970	0.989	0.975
		recall	0.989	0.963	0.994	0.978
		f-score	0.991	0.966	0.992	0.976
	feature set 3	precision	0.999	0.988	0.996	0.990
		recall	0.993	0.986	0.999	0.993
		f-score	0.996	0.987	0.987	0.992

tunneling traffic and normal traffic. The performance using feature set 2 is also somewhat inferior to that with use of our proposed feature set. As we add several discriminative features that designed for identifying behaviors in DNS tunnels, e.g., CSD and packet ratio. The experiment can depict that the designed features can help to improve the identification performance. For all the data sets, identification performance improves as the fragment size increasing. For the same data set, the performance of the Random Forest method is better than that of the Decision Tree method. And Bayes Net method has the worst performance. The results are consistent with the previous experiments.

VI. CONCLUSION AND FUTURE WORK

Current research has paid little attention to the internal APBI of DNS tunnels. To this end, we propose a APBI scheme based on carefully selected spatial-temporal statistical features. These features are further used in the APBI problem. Plenty of experiments are performed and the superior classifier is drawn based on the comparison. For the Decision Tree classifier, the impact of fragment size is as parameter to develop several experiments to obtain the suitable one. In addition, three commonly used machine learning classifiers including Decision Tree, Random Forest, and Bayes Net, are employed to identify the application behaviors in DNS tunnels to prove the feasibility of the scheme, and find a superior machine learning classifier.

In this research, we only consider four types of internal user behaviors. In future work, more details about types of remote control and data exfiltration which directly refer to cybersecurity will be taken into consideration. More intensive

classification between network traffic with similar spatial-temporal pattern should be further studied. In addition, we will try the deep-learning-based methods to extract maybe richer features automatically in the future work.

REFERENCES

[1] Z. Wang, "Combating malicious DNS tunnel," 2016, *arXiv:1605.01401*. [Online]. Available: http://arxiv.org/abs/1605.01401
[2] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. V. Steen, and N. Pohlmann, "On botnets that use DNS for command and control," in *Proc. 7th Eur. Conf. Comput. Netw. Defense*, Sep. 2011, pp. 9–16.
[3] E. Skoudis. *The Six Most Dangerous New Attack Techniques and What's Coming Next*. Accessed: 2012. [Online]. Available: https://blogs.sans.org/pentesting/files/2012/03/RSA-2012-EXP-108-Skoudis-Ullrich.pdf
[4] C. Lynch, C. Teodorescu, and D. Andonov. (2019). *Multigrain-Point of Sale Attackers Make an Unhealthy Addition to the Pantry*. [Online]. Available: https://www.fireeye.com/blog/threat-research/2016/04/multigrain pointo.html
[5] I. Homem, P. Papapetrou, and S. Dosis, "Entropy-based prediction of network protocols in the forensic analysis of DNS tunnels," 2017, *arXiv:1709.06363*. [Online]. Available: http://arxiv.org/abs/1709.06363
[6] I. Homem and P. Papapetrou, "Harnessing predictive models for assisting network forensic investigations of DNS tunnels," in *Proc. InADFSL Conf. Digit. Forensics, Secur. Law*, 2017, pp. 1–17.
[7] A. Almusawi and H. Amintoosi, "DNS tunneling detection method based on multilabel support vector machine," *Secur. Commun. Netw.*, vol. 2018, pp. 1–9, Jan. 2018.
[8] David, "The role of DNS in BotNet command and control (C&C)," Open DNS Inc., Secur., San Francisco, CA, USA, White Paper, 2011, pp. 1–24.
[9] I. Valenzuela, "Game changer: Identifying and defending against data exfiltration attempts," *SANS Cyber Defense Summit*, pp. 1–42, 2015.
[10] K. Born and D. Gustafson, "Detecting DNS tunnels using character frequency analysis," 2010, *arXiv:1004.4358*. [Online]. Available: http://arxiv.org/abs/1004.4358
[11] K. Born and D. Gustafson, "NgViz: Detecting DNS tunnels through n-gram visualization and quantitative analysis," in *Proc. 6th Annu. Workshop Cyber Secur. Inf. Intell. Res. (CSIIRW)*, 2010, pp. 1–4.
[12] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A bigram based real time DNS tunnel detection approach," *Procedia Comput. Sci.*, vol. 17, pp. 852–860, Jan. 2013.
[13] K. Xu, P. Butler, S. Saha, and D. Yao, "DNS for massive-scale command and control," *IEEE Trans. Depend. Sec. Comput.*, vol. 10, no. 3, pp. 143–153, May 2013.
[14] R. Preston, "DNS tunneling detection with supervised learning," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Nov. 2019, pp. 1–6.
[15] A. Das, M.-Y. Shen, M. Shashanka, and J. Wang, "Detection of exfiltration and tunneling over DNS," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 737–742.
[16] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, and V. Sivaraman, "Real-time detection of DNS exfiltration and tunneling from enterprise networks," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 649–653.
[17] C.-M. Lai, B.-C. Huang, S.-Y. Huang, C.-H. Mao, and H.-M. Lee, "Detection of DNS tunneling by feature-free mechanism," in *Proc. IEEE Conf. Dependable Secure Comput. (DSC)*, Dec. 2018, pp. 1–2.
[18] C. Liu, L. Dai, W. Cui, and T. Lin, "A byte-level CNN method to detect DNS tunnels," in *Proc. IEEE 38th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Oct. 2019, pp. 1–8.
[19] J. Zhang, L. Yang, S. Yu, and J. Ma, "A DNS tunneling detection method based on deep learning models to prevent data exfiltration," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2019, pp. 520–535.
[20] A. J. Campbell and N. Zincir-Heywood, "Exploring tunneling behaviours in malicious domains with self-organizing maps," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2020, pp. 1419–1426.
[21] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, "Flow-based detection of DNS tunnels," in *Proc. Int. Conf. Auto. Infrastruct., Manage., Secur., Emerg. Manage. Mech. Future Internet (IFIP WG)*. Berlin, Germany: Springer, 2013, pp. 124–135.
[22] M. Aiello, M. Mongelli, and G. Papaleo, "DNS tunneling detection through statistical fingerprints of protocol messages and machine learning," *Int. J. Commun. Syst.*, vol. 28, no. 14, pp. 1987–2002, Sep. 2015.
[23] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang, and C. Peng, "Detecting DNS tunnel through binary-classification based on behavior features," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 339–346.
[24] 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 *Proc. 11th Annu. Cyber Inf. Secur. Res. Conf.*, Apr. 2016, pp. 1–4.
[25] J. J. Davis and E. Foo, "Automated feature engineering for HTTP tunnel detection," *Comput. Secur.*, vol. 59, pp. 166–185, Jun. 2016.
[26] K. Wu, Y. Zhang, and T. Yin, "TDAE: Autoencoder-based automatic feature learning method for the detection of DNS tunnel," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.
[27] M. Luo, Q. Wang, Y. Yao, X. Wang, P. Yang, and Z. Jiang, "Towards comprehensive detection of DNS tunnels," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–7.
[28] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *Proc. Int. Conf. Passive Act. Netw. Meas.* Berlin, Germany: Springer, 2007, pp. 165–175.
[29] R. Alshammari and A. N. Zincir-Heywood, "Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?" *Comput. Netw.*, vol. 55, no. 6, pp. 1326–1350, Apr. 2011.
[30] G. Mujtaba and D. J. Parish, "A statistical framework for identification of tunnelled applications using machine learning," *Int. Arab J. Inf. Technol.*, vol. 12, no. 6A, pp. 785–790, 2015.
[31] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Detection of encrypted tunnels across network boundaries," in *Proc. IEEE Int. Conf. Commun.*, May 2008, pp. 1738–1744.
[32] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescapé, "A dive into the dark Web: Hierarchical traffic classification of anonymity tools," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1043–1054, Mar. 2019.
[33] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity services tor, I2P, JonDonym: Classifying in the dark (Web)," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 3, pp. 662–675, May 2020.
[34] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A big data-enabled hierarchical framework for traffic classification," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2608–2619, Oct. 2020.
[35] Z. Yang, Y. Hongzhi, L. Lingzi, H. Cheng, and Z. Tao, "Detecting DNS tunnels using session behavior and random forest method," in *Proc. IEEE 5th Int. Conf. Data Sci. Cyberspace (DSC)*, Jul. 2020, pp. 45–52.
[36] O. Santos, *Network Security With NetFlow and IPFIX: Big Data Analytics for Information Security*. Indianapolis, IN, USA: Cisco Press, 2015.
[37] E. Keogh and A. Mueen, "Curse of dimensionality," *Ind. Eng. Chem.*, vol. 29, no. 1, pp. 48–53, 2009.
[38] P. Langley and W. Iba, "Average-case analysis of a nearest neighbor algorithm," in *Proc. IJCAI*, vol. 93, 1993, p. 889.
[39] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artif. Intell.*, vol. 151, nos. 1–2, pp. 155–176, Dec. 2003.
[40] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, Jan. 2018.
[41] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature selection for high-dimensional data," *Prog. Artif. Intell.*, vol. 5, no. 2, pp. 65–75, 2016.
[42] X. Liu, L. Wang, J. Zhang, J. Yin, and H. Liu, "Global and local structure preservation for feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1083–1095, Jun. 2014.
[43] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

HUIWEN BAI received the B.S. degree in automation from the Nanjing University of Science and Technology, Nanjing, China, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include deep learning and network security.

WEIWEI LIU (Member, IEEE) received the B.S. degree in automation and the Ph.D. degree in control science and engineering from the Nanjing University of Science and Technology, Nanjing, in 2010 and 2015, respectively. From 2014 to 2015, he was a Visiting Scholar with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He is currently an Associate Professor with the School of Automation, Nanjing University of Science and Technology. He has published over 30 articles in these areas, including the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and *Annals of Telecommunications*. His research interests include multimedia signal processing and network traffic analysis. He is an active Reviewer of several journals, including *Digital Signal Processing* and *Security and Communication Networks*.

GUANGJIE LIU received the B.S. degree in electrical and computer engineering and the Ph.D. degree in control science and engineering from the Nanjing University of Science and Technology, Nanjing, in 2002 and 2007, respectively. He is currently an Associate Professor with the School of Automation, Nanjing University of Science and Technology. His research interests include multimedia systems and deep learning.

YUEWEI DAI received the B.S. and M.S. degrees in system engineering from the East China Institute of Technology, Nanjing, China, in 1984 and 1987, respectively, and the Ph.D. degree in control science and engineering from the Nanjing University of Science and Technology, in 2002. He is currently a Professor with the School of Automation, Nanjing University of Science and Technology. His research interests include multimedia security, system engineering theory, and network security.

SHUHUA HUANG (Graduate Student Member, IEEE) received the B.S. degree in automation from the Nanjing University of Science and Technology, Nanjing, China, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include detection of wireless covert communication and network security.