Hindawi
Security and Communication Networks
Volume 2021, Article ID 5587837, 11 pages
https://doi.org/10.1155/2021/5587837

Research Article

Obfuscated Tor Traffic Identification Based on Sliding Window

Wenliang Xu and Futai Zou

School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Correspondence should be addressed to Futai Zou; zoufutai@sjtu.edu.cn

Received 22 February 2021; Revised 5 May 2021; Accepted 31 May 2021; Published 15 June 2021

Academic Editor: Mamoun Alazab

Tor is an anonymous communication network used to hide the identities of both parties in communication. Apart from those who want to browse the web anonymously using Tor for a benign purpose, criminals can use Tor for criminal activities. It is recognized that Tor is easily intercepted by the censorship mechanism, so it uses a series of obfuscation mechanisms to avoid censorship, such as Meek, Format-Transforming Encryption (FTE), and Obfs4. In order to detect Tor traffic, we collect three kinds of obfuscated Tor traffic and then use a sliding window to extract 12 features from the stream according to the five-tuple, including the packet length, packet arrival time interval, and the proportion of the number of bytes sent and received. And finally, we use XGBoost, Random Forest, and other machine learning algorithms to identify obfuscated Tor traffic and its types. Our work provides a feasible method for countering obfuscated Tor network, which can identify the three kinds of obfuscated Tor traffic and achieve about 99% precision rate and recall rate.

1. Introduction

With the rapid development of Internet technology and the explosive growth of information data, users pay more and more attention to their personal privacy information. Although the commonly used HTTPS protocol can ensure that the visitor's communication data is not eavesdropped on by a third party, it cannot hide his identity information. Therefore, anonymous communication technology is proposed to protect the communication data and help the user conceal his IP address and other pieces of private information. Anonymous communication technology has developed from the Mix [1] technology to the commonly used Tor (The second-generation Onion Router) [2], I2P (Invisible Internet Project), Freenet [3], and so on. Besides, some blockchain-based anonymous communication techniques have been proposed in recent years.

Tor is the most popular anonymous communication technology among these technologies. It is a type of overlay network that uses software to create layers of network abstraction that can be used to run multiple separate, discrete virtualized network layers on top of the physical network. Volunteers provide the onion routers running on Tor from various countries and regions around the world. When a user wants to connect to the Tor network, he randomly selects three onion routes via the onion directory servers, handshake in turn with them to establish a communication circuit, and can access the Internet anonymously through this communication circuit.

In addition to providing anonymous access to the Internet, the Tor network also provides hidden services, the so-called darknet. Much illegal content that is not allowed to appear on the surface web (i.e., the Internet) floods the darknet, including drug dealing, gun dealing, and gambling websites. In addition, the first ransomware Curve-Tor-Bitcoin Locker used the Tor network to hide its traffic, and action appeared in mid-2014 [4]. In other words, some people use Tor to protect their privacy and achieve anonymous access to the website. In contrast, other people use it to hide their illegal purposes and commit criminal activities. Therefore, it is necessary to identify and detect Tor traffic.

In the Tor network, the information of the onion routers is stored in directory servers, including lists of active onion routers, their IP addresses, their locations, and current public keys. This information is public, which everyone can access, so Tor is very easy to block by closing the connection of onion routers' IP addresses or port 9001, which the Tor commonly uses. In order to improve the availability and anonymity of the network, Tor introduces many methods to bypass censorship, including Tor Bridge mechanisms, Meek-based obfuscation [5], Format-Transforming Encryption-(FTE-) based obfuscation [6], and Obfs4-based obfuscation [7]. These obfuscation methods bring difficulties for detecting and identifying Tor traffic.

Since the darknet can be used to commit criminal activities, we should be capable of blocking the Tor connection. This capability requires us to propose approaches to identify Tor traffic because detecting Tor traffic is a prerequisite for blocking it. What is more, the various obfuscation mechanisms introduced by Tor require us to detect the obfuscated Tor traffic. Existing researches only identify the traffic obfuscated by one of the obfuscation mechanisms. Therefore, we propose an approach based on a sliding window to identify different kinds of obfuscated Tor traffic. In other words, we will distinguish Meek-based, FTE-based, and Obfs4-based Tor traffic from normal traffic. Besides, prior works do not make any obfuscated Tor traffic dataset publicly available for the research community to use and build upon. In summary, this paper makes the following contributions:

(1) Utilizing a sliding time window to split TCP flows instead of extracting features from a single overall flow, which helps to reduce the number of data packets required for detection and effectively improve the real-time detection.

(2) Conducting detailed experiments and analysis to show the effectiveness of the features extracted from the sliding window.

(3) Establishing a general multiclassification model with universal features to detect three kinds of obfuscated Tor traffic, which achieves about 99% precision rate and recall rate.

(4) A large amount of three kinds of obfuscated Tor traffic which has been collected and published on the web (https://github.com/QQQQing/Obfuscated-Tor-Traffic).

The rest of the paper is organized as follows. Section 2 discusses related work in Tor traffic identification. Section 3 introduces the features used in the model and shows their effectiveness. Section 4 elaborates on the experiment, including the models, the dataset, and the metrics to evaluate the models' performance. Section 5 discusses the computational complexity and the limitations of our method. And finally, Section 6 concludes the paper.

2. Related Work

At present, traffic detection commonly uses three methods: port-based method, DPI (Deep Packet Inspection)-based method, and flow feature-based method. However, the first two techniques are not suitable to detect obfuscated Tor traffic for the following two reasons: (1) the commonly used port of onion routing nodes changes from 9001 to 443, which is the same port used by SSL/TLS protocols; (2) the Tor traffic is encrypted, so there is no special keyword in the packets. Therefore, a flow feature-based method is proper to solve such a problem. A flow feature-based method is always combined with deep learning algorithms and machine learning algorithms, widely used in the Intrusion Detection System (IDS). For example, Swarna Priya et al. [8] used deep neural networks to develop effective and efficient IDS in the IoMT (the Internet of Medical Things) environment. Khan et al. [9] present a framework based on decision trees that effectively detect P2P botnets.

There are numerous researches on nonobfuscated Tor traffic identification based on time-related and packet-related features from the traffic flows [10–19]. In Table 1, we analyze different approaches for nonobfuscated Tor traffic identification. Some of them trained a supervised machine learning model [10–16], such as Random Forest and SVM (support vector machine), and others introduced an unsupervised machine learning model [15]. Some researchers also generated fingerprints for SSL/TLS flows to distinguish Tor traffic from normal traffic [16–18]. Particularly in [16], He et al. compared the means of fingerprint matching and ML-based classification and pointed out the advantages and disadvantages of both. Zhioua [19] proposed an HMM-based approach to identify the Tor traffic.

Since more and more methods have been developed to detect nonobfuscated Tor traffic, Tor utilizes obfuscation to improve its reliability and anonymity. These obfuscation methods each have variant mechanisms to protect Tor traffic from detection. As mentioned above, Meek-based, FTE-based, and Obfs4-based obfuscation are three officially supported plugins. Next, we will briefly introduce their working mechanism and corresponding detecting method proposed by existing research. In Table 2, we analyze different approaches for obfuscated Tor traffic identification.

2.1. Meek-Based Obfuscation. The key to Meek-based obfuscation lies in the use of domain front technology, which uses different domain names at different communication layers [5] and tunnels to avoid censorship. There are three entities involved in the communication: Tor client with Meek plugin, fronted server with an allowed domain name provided by the cloud service provider, and Tor server with Meek plugin. When a client wants to connect to the Tor network, it encapsulates the Tor request into a TLS layer with the domain name of the fronted server in the header and then sends the request to the fronted server. After the fronted server received the packet, it unpacks the internal request and sends it to the Tor server. Since the server is not allowed to push data to the client actively, the client needs to continuously poll the fronted server to check whether the Tor server sends data back and finally obtains the response content. In short, Meek-based obfuscation avoid censorship by using a cloud server with an allowed domain name to forward the requests to the Tor, and as a result, the traffic just seems like ordinary cloud service traffic.

The polling mechanism results in a large number of shorter packets appearing in the communication process,

Table 1: Analysis of nonobfuscated Tor traffic identification techniques.

Reference	Feature selection	Type of algorithm/ method	Dataset	Evaluation metrics
Lashkari et al. [10]	Time-related features	Random Forest; C4.5; KNN	Self-collected, called Tor-nonTor (ISCXTor2016)	Precision; recall
Hodo et al. [11]	Time-related features using correlation-based feature selection	Artificial neural network; support vector machine	The same as [8]	Accuracy; precision; false positive rate
Almubayed et al. [12]	Features generated by NetMate (http://f00l.de/netmate/)	Naïve Bayes; C4.5; Random Forest; support vector machine	Self-collected	Precision; FP rate
Mayank and Singh [13]	Statistics calculated by NetAI (http://caia.swin.edu.au/urp/dstc/netai)	Random Forest; J4.8; AdaBoost	Self-collected	TP rate; FP rate; ROC Area
Cuzzocrea et al. [14]	Features calculated by ISCXFlowMeter (the tools implemented by [8])	J4.8; BayesNet; jRip; OneR; RepTREE	Self-collected	TP rate; FP rate; precision; recall; F-measure; MCC; ROC Area; PRC area
Rao et al. [16]	Packet level and flow level features	Gravitational clustering	Self-collected	Rand statistic; Jaccard coefficient; FM; averaged accuracy
He et al. [16]	TLS/SSL-related features; packet size-related features	TLS fingerprint-based method; packet size distribution-based method	CAIDA Equinix Chicago (http://www.caida.org/data/passive/passive_2010_dataset.xml)	TP rate; false positive rate
Barker et al. [17]	TLS/SSL-related features	Just logical judgment	Self-collected	-
Bai et al. [18]	Traffic fingerprints; characteristic strings	AC-BM algorithm	Self-collected	Recognition rate; misrecognition rate
Zhioua [19]	Interpacket times	Hidden Markov models	Self-collected	Precision; F-measure

Table 2: Analysis of obfuscated Tor traffic identification techniques.

Reference	Feature selection	Type of algorithm/method	Target	Evaluation metrics
Yao et al. [20]	Interpacket time distribution; packet size distribution	The Mixture of Gaussians-based Hidden Markov Model	Meek-based	Accuracy; precision; recall; F1-Score
He et al. [21]	TLS/SSL-related features; time-related features; packet size-related features	Support vector machine	Meek-based	Accuracy; recall
Gao [22]	Time-related features; packet size-related features	Support vector machine; randomness detection	Obfs4-based	Precision
Zhai [23]	Time-related features; packet size-related features	Naïve Bayes; C4.5; BayesNet	FTE-based	Precision; recall; F1-Score

which is an obvious characteristic. According to this feature, Yao et al. [20] proposed a method based on a Mixture of Gaussians-based Hidden Markov Model (MGHMM), which characterize the interpacket time distribution and the packet size density distribution of flows. He et al. [21] summarize the connection characteristics based on the polling mechanism, including the static and dynamic features of the flows, and then apply SVM to identify Meek-based Tor traffic.

2.2. Obfs4-Based Obfuscation. Obfs4 is the latest plugin of Obfs (the four letters in obfuscation) proxy, which uses encryption algorithms to disguise Tor traffic as ordinary encrypted traffic such as the SSL/TLS protocol. The main practice is to use ECC (Elliptic Curve Cryptography) to encrypt the data and randomly fill the payload, which changes the size of the packet and conceals the packet length-related features in the flows. After random packet length padding, only the recipient with the key can derive the correct packet length value and then reassemble the packet correctly. Obfs4 strengthens its random characteristics, so Gao [22] used a randomness test to identify Obfs4-based Tor traffic and achieved good results.

2.3. FTE-Based Obfuscation. The core idea of FTE-based obfuscation is to use regular expressions to replace the bytes appearing in Tor traffic. For example, a regular expression of HTTP protocol keywords can be used to disguise Tor traffic as HTTP traffic to deceive the DPI system. However, the characteristics of flows have not changed significantly. Therefore, Zhai [23] used features of flows to characterize the traffic and used a machine learning algorithm to identify FTE-based Tor traffic.

3. Feature Engineering

3.1. Analysis of Traffic Characteristics. Traffic identification technology based on flow features usually uses inner-packet interval and packet size features because different protocols are likely to have a unique distribution of inner-packet interval and packet size, which can well characterize the protocol. The difference among three types of obfuscation Tor traffic and normal traffic on the above two types of features is caused by the following:

(1) The Meek client polls the fronted server for the response, so there will be a large number of shorter packets in Meek-based Tor traffic.

(2) There are limited onion routers in the Tor network, so each relay will load a large number of requests, resulting in a relatively larger inner-packet interval.

(3) Each packet needs to be encrypted $(n+1)$ times before being sent out, then decrypted by n relays in the circuit selected by the client, and finally decrypted by the target server. The parameter n refers to the number of relays. This encryption and decryption process will cause the inner-packet interval of Tor traffic to be generally larger than that of normal traffic.

In order to verify whether the obfuscated Tor traffic is different from normal traffic, we conduct the following experiment. Firstly, we extract flows identified by the five-tuple (source IP address, source port number, destination IP address, destination port number, and protocol) from each traffic file and filter out the flows of TCP protocol. Then for each packet in a flow, we calculate the size of it and the time interval from the previous packet, which is called the inner-packet interval. At last, the inner-packet interval and packet size of all flows in each kind of traffic are summarized, and a curve of CDF (Cumulative Distribution Function) is drawn. The definition of CDF is $F(x) = P(X \leq x)$. It means that, for a function $f(x)$ and a specific value x_0 of an independent variable, the value of $F(x_0)$ is the sum of the probability of occurrence of all values less than or equal to x_0.

We take the inner-packet interval as an example to illustrate how to draw a CDF curve. The inner-packet interval can be considered continuous, so we should map it to a certain interval to discretize the values. We can set the time interval as follows:

$$[0, 0.1), [0.1, 0.2), \ldots, [10, 10.1], \ldots \tag{1}$$

Then, we put the time interval into the corresponding interval and record the number of inner-packet intervals in every interval as

$$x_{[0,0.1)}, x_{[0.1,0.2)}, \ldots, x_{[10,10.1]}, \ldots \tag{2}$$

In the actual experiment, we make the interval shorter, and the curve of the CDF will have higher precision.

The inner-packet interval CDF curve and packet size CDF curve of each traffic type are shown in Figures 1 and 2. From these eight CDF curves, we can observe the following characteristics:

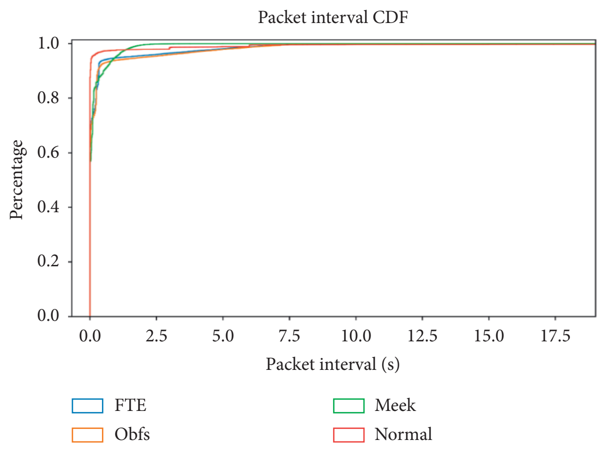

Figure 1: CDF curve of inner-packet interval. We calculate the CDF values of the inner-packet interval of FTE-based, Obfs4-based, Meek-based, and normal traffic and then show them as curves.

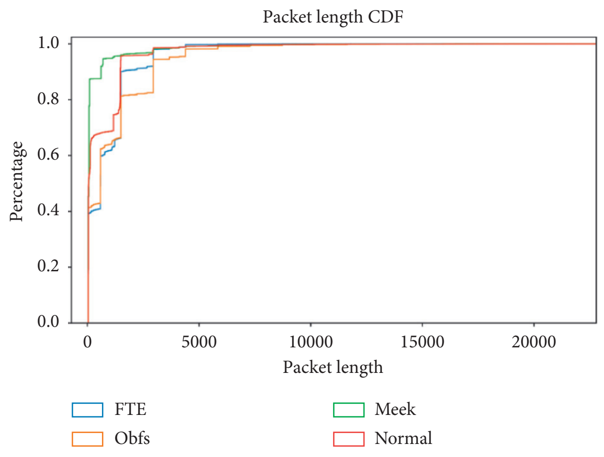

Figure 2: CDF curve of packet size. We calculate the CDF values of packet size of FTE-based, Obfs4-based, Meek-based, and normal traffic and then show them as curves.

(1) In terms of packet size, numerous small-size packets appeared in Meek-based Tor traffic, accounting for about 90%. And about 65% of small-size packets occur in normal traffic, while Obfs4-based Tor traffic and FTE-based Tor traffic have the least proportion of small-size packets, only 40%.

(2) In terms of the inner-packet interval, the inner-packet interval of normal traffic is generally shorter. About 95% of inner-packet intervals are less than 0.2 seconds, and the number of short-time intervals in Meek-based Tor traffic is slightly less. Obfs4-based Tor traffic and FTE-based Tor traffic have the least short-time-interval packets.

In other words, we can distinguish Meek-based Tor traffic easily from the other two kinds of obfuscated Tor traffic. Observing Obfs4-based Tor traffic and FTE-based Tor traffic, we find that their inner-packet interval distribution is similar, but there are certain differences in packet size distribution. We will discuss how our method can distinguish these two types of traffic in Section 3.2.

3.2. Analysis of Feature Extraction with Sliding Window. This section will analyze whether the feature of packet size extracted in the sliding window can better distinguish Obfs4-based Tor traffic from FTE-based Tor traffic.

At present, many traffic identification technologies treat a single flow as a whole and extract features from it. Here, a flow is defined as all packets identified by the same five-tuple. However, this approach may cause an obvious problem: it is hard to achieve real-time detection. As long as we treat a flow as a whole, we may meet with the following dilemma:

(1) If we extract features before a flow ends, it will lead to inaccurate feature extraction
(2) If we extract features after a flow ends, it will lead to high latency to detect the flow

Therefore, we try to split the flow with a sliding window and treat the packets in the sliding window as a whole to extract features. To show the effectiveness of extracting features with a sliding window, we split the packets of Obfs4-based Tor traffic and FTE-based Tor traffic with a sliding window. Here, we temporarily set the size of the sliding window as 100 packets and its sliding distance as ten packets. After splitting the flow, we will obtain a large number of packet sets, and each set consists of 100 packets. Next, we randomly select 1000 packet sets from the whole sets of Obfs4-based Tor traffic and FTE-based Tor traffic and then extract vectors of packet size from each packet set. Finally, we handle these vectors as time series and use DTW [24] (Dynamic Time Warping) algorithm to calculate the distance between these series. The advantage of using the DTW algorithm to calculate the distance between time series is that each element of two target series does not have to correspond to each other, and the distance between two series can be measured more accurately.

Specifically, we mark the sets of packet size vectors of Obfs4-based Tor traffic as $X = x_1, x_2, \ldots, x_{1000}$ and those sets of FTE-based Tor traffic as $Y = y_1, y_2, \ldots, y_{1000}$, where x_i and y_j, $0 \leq i,\ j \leq 1000$, respectively, represent the vectors of packet size of two types of Tor traffic, whose length equals 100. Next, we calculate the distances between every two vectors of X and X, Y and Y, and X and Y using the DTW algorithm, and we will get three matrices of size 1000 × 1000. For instance, the matrix M_{XY}, which consists of the distances between every two vectors of X and Y, can be calculated as

$$M_{XY} = \begin{bmatrix} \mathrm{DTW}(x_1, y_1) & \cdots & \mathrm{DTW}(x_1, y_{1000}) \\ \vdots & \ddots & \vdots \\ \mathrm{DTW}(x_{1000}, y_1) & \cdots & \mathrm{DTW}(x_{1000}, y_{1000}) \end{bmatrix}. \tag{3}$$

Similarly, we can calculate the matrices M_{XX} and M_{YY}. In order to show the difference of vector distances between different kinds of Tor traffic more intuitively, we use a heatmap to visualize the matrices. The value of each pixel of the heatmap ranges from 0 to 200,000, and each pixel denotes the DTW distance between two vectors. Therefore, the brighter the pixel is, the further distance it represents. Figures 3(a) and 3(b) have relatively darker colors than Figure 3(c). It means that the distance between vectors of the same type of Tor traffic is smaller than that of different types of Tor traffic. In short, by splitting the flows into sets of packets, the packet size feature, which extracts from each set of packets, can reflect the difference between different types of obfuscated Tor traffic, especially, Obfs4-based Tor traffic and FTE-based Tor traffic.

3.3. Feature Extraction. According to the analysis in Sections 3.1 and 3.2, it is effective to identify the obfuscated Tor traffic with the inner-packet interval-related and packet size-related features. Next, we will introduce how these features are extracted.

The process of feature extraction is shown in Figure 4, and its steps are as follows:

(1) Extract flows from the traffic file (e.g., pcap files) according to the five-tuple (source IP address, source port, destination IP address, destination port, and protocol), and record the direction, length, and occurrence time of each packet in the flows.
(2) Split the flows into sets of packets with a sliding window whose window size and sliding distance, respectively, equal n packets and m packets, and each set consists of n packets.
(3) Extract 12 features listed in Table 3 from each packet set, and especially, the inner-packet interval is the difference between the occurrence time of two adjacent data packets.

4. Experiments

4.1. Dataset. The existing public dataset of Tor traffic is the Tor-nonTor (ISCXTor2016) dataset [5]. It is provided by the University of New Brunswick. They defined a set of tasks to generate a representative dataset of actual traffic and gave detailed labels. But this dataset only contains nonobfuscated Tor traffic, so it cannot be applied in our experiments.

Thus, we collected normal traffic, Meek-based Tor traffic, Obfs4-based Tor traffic, and FTE-based Tor traffic as our experimental dataset. The collection architecture is shown in Figure 5. We deployed Meek-based, Obfs4-based, and FTE-based Tor clients on six cloud servers, two for each type. Then we wrote scripts to access the Internet to generate network traffic automatically. We use TShark (a command-line tool for the network analysis tool Wireshark) to capture traffic, store it as a network traffic file in pcap format, and centrally upload it to the data center server every day. What is more, we recorded the IP addresses associated with different kinds of obfuscated Tor traffic.

We collected traffic twice and used them as the training set and the test set. The time and the websites visited of the two collections are different.

Next, we process the data according to the feature extraction method described in Section 3. Here, one piece of

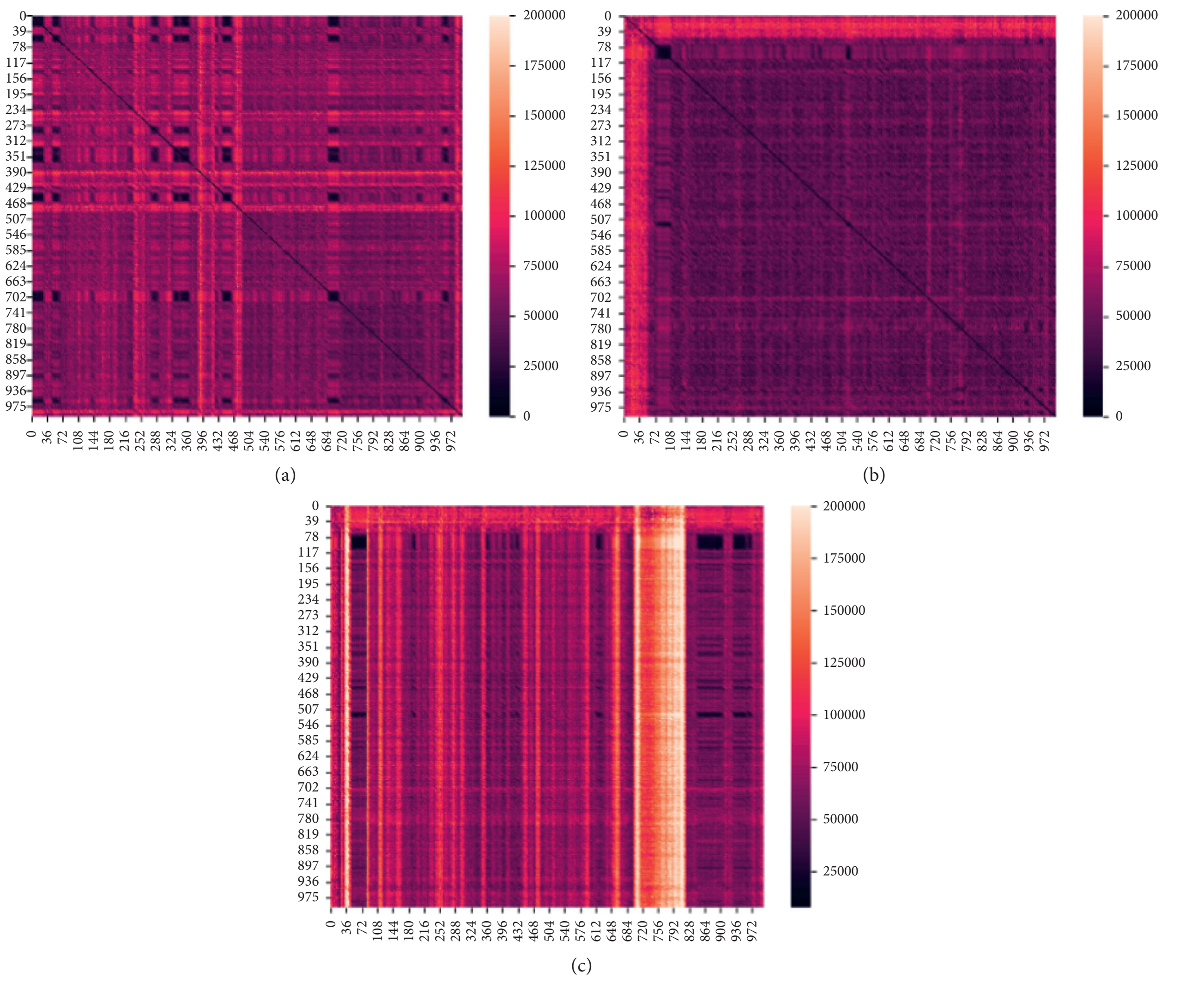

Figure 3: Heatmaps of DTW distance matrix. (a) The heatmap of matrix M_{XX} which is calculated between every two vectors of X and X, (b) the heatmap of matrix M_{YY}, and (c) the heatmap of the matrix M_{XY}. Here, X is the sets of packet size vectors of Obfs4-based Tor traffic, and Y is those sets of FTE-based Tor traffic.

data refers to 12 features and its corresponding label (i.e., Meek-based Tor, Obfs4-based Tor, FTE-based Tor, or normal traffic) extracted from a packet set with 100 packets as the unit.

The summary of different types of traffic in the two parts of the dataset is shown in Table 4. It should be noted that the experiments in this paper are all based on this dataset.

4.2. Evaluation Metrics. In this experiment, we set the following metrics to indicate whether the model can accurately predict samples and to reflect whether the model can perform well on the training set, validation set, and test set.

4.2.1. Confusion Matrix. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class [25]. In the confusion matrix, we can get the TP (instances which are actually true and predicted to be true), FP (instances which are actually false but predicted to be true), TN (instances which are actually false and predicted to be false), and FN (instances which are actually true but predicted to be false) for each category in the confusion matrix. These values are used for calculating other metrics such as precision and recall. For a four-category problem, the confusion matrix and TP, FP, TN, and FN values of category 2 are shown in Table 5. Briefly, let the confusion matrix be M, the total number of categories be n, and a target category be i; we can calculate TP, FP, TN, and FN as equations (4)–(7):

$$\mathrm{TP} = M[i,i], \tag{4}$$

$$\mathrm{FP} = \sum_{j=1}^{n} M[j,i] - \mathrm{TP}, \tag{5}$$

$$\mathrm{FN} = \sum_{j=1}^{n} M[i,j] - \mathrm{TP}, \tag{6}$$

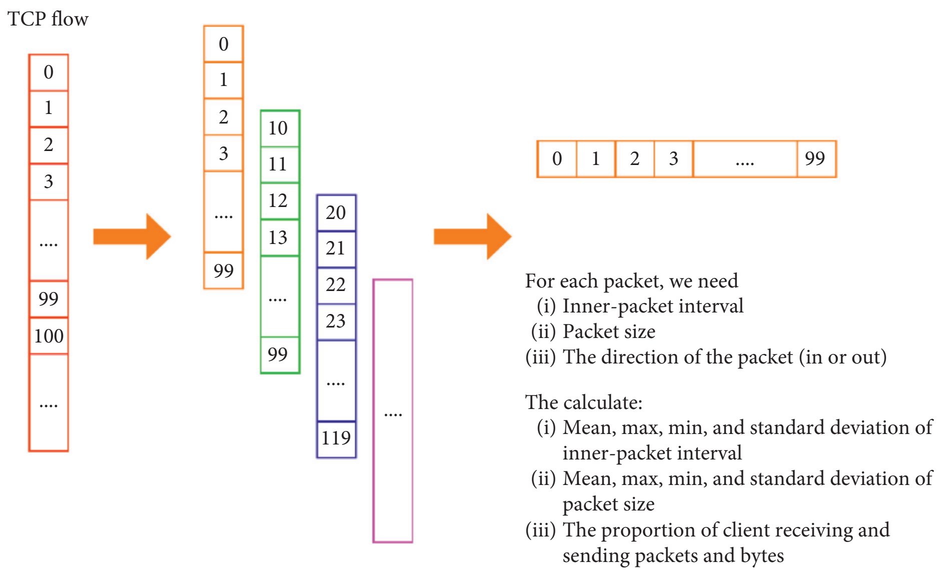
TCP flow
0
1
2
3
....
99
100
....
10
11
12
13
20
21
22
23
119
For each packet, we need
(i) Inner-packet interval
(ii) Packet size
(iii) The direction of the packet (in or out)
The calculate:
(i) Mean, max, min, and standard deviation of inner-packet interval
(ii) Mean, max, min, and standard deviation of packet size
(iii) The proportion of client receiving and sending packets and bytes

Figure 4: The process of feature extraction. First, we split the TCP flow into sets of packets. Then for each packet in each packet set, we extract its inner-packet interval, packet size, and direction. At last, we calculate 12 features of each packet set.

Table 3: Features extracted. We extracted 12 features from packets split by the sliding window.

No.	Name of feature	Description of feature
1	packet_size_max	The max value of packet size
2	packet_size_min	The min value of packet size
3	packet_size_mean	The mean value of packet size
4	packet_size_std	The standard deviation of packet size
5	packet_interval_max	The max value of inner-packet interval
6	packet_interval_min	The min value of inner-packet interval
7	packet_interval_mean	The mean value of inner-packet interval
8	packet_interval_std	The standard deviation of inner-packet interval
9	rate_bytes_send	The proportion of client sending bytes
10	rate_bytes_receive	The proportion of client receiving bytes
11	rate_packets_send	The proportion of client sending packets
12	rate_packets_receive	The proportion of client receiving packets

$$\mathrm{TN} = \sum_{k=1}^{n} \sum_{j=1}^{n} M[k, j] - \mathrm{TP} - \mathrm{FN} - \mathrm{FP}. \quad (7)$$

4.2.2. Precision. The fraction of correctly predicted positive instances among all predicted positive instances is shown in equation (8):

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \quad (8)$$

4.2.3. Recall. The fraction of correctly predicted positive instances among all actually positive instances is :

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \quad (9)$$

4.2.4. Accuracy. The fraction of correctly predicted instances among all instances is

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \quad (10)$$

4.2.5. F1-Score. Harmonic average of precision rate and recall rate is

$$F1 - \mathrm{Score} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \quad (11)$$

Among the four metrics, when the categories are unbalanced, the accuracy rate cannot well reflect the situation of the model. In contrast, the precision rate, recall rate, and F1-Score can reflect whether the model is effective even when the number of positive instances and the number of negative examples differ greatly. Therefore, we are more inclined to use precision rate, recall rate, and F1-Score as evaluation indicators.

4.3. Algorithm Selection. In recent years, machine learning and deep learning have played an extremely important role

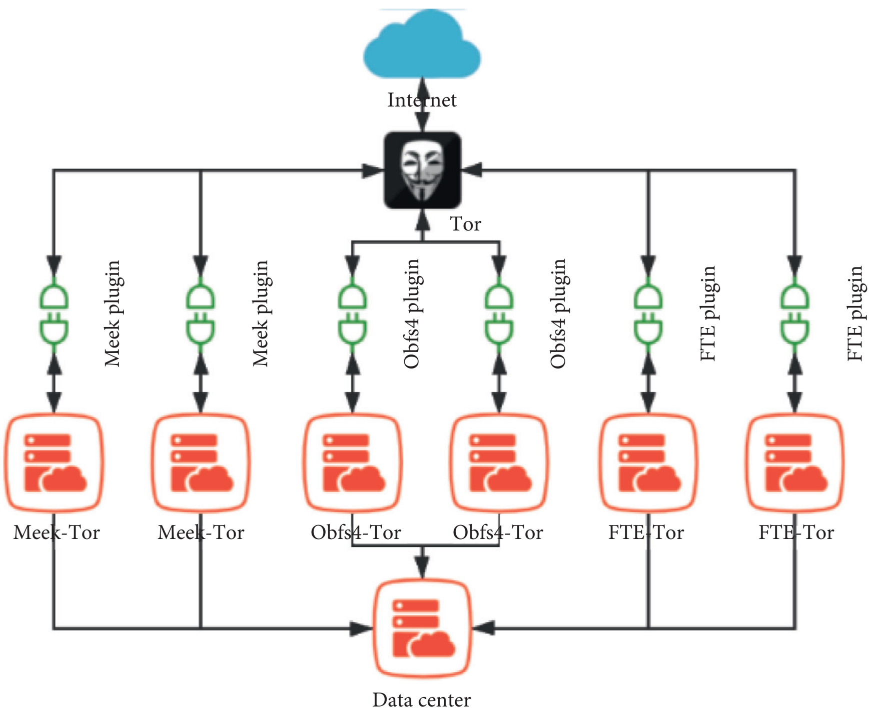

Figure 5: Architecture of traffic collection. We used six servers to collect obfuscated Tor traffic. We deployed Meek-based, Obfs4-based, and FTE-based Tor clients on these servers, two for each type. In addition, we uploaded data to a data center each day.

Table 4: Summary of the dataset. We have collected a huge amount of obfuscated Tor traffic. We show the IP addresses related to each kind of traffic and the number of pieces of data.

Type	Size (GB)	Related IP	Training and validation set	Test set
Meek	7.2	117.18.232.200, . . .	1,000,786	768,874
Obfs4	12.2	190.2.145.7, ...	480,758	414,871
FTE	12.5	188.138.75.101, ...	457,556	628,915

Table 5: An example for a 4-class confusion matrix.

		Predicted			
		Class 1	Class 2	Class 3	Class 4
Actual	Class 1	TN	FP	TN	TN
	Class 2	FN	TP	FN	FN
	Class 3	TN	FP	TN	TN
	Class 4	TN	FP	TN	TN

in many fields, such as face recognition [26] and intrusion detection [27]. They are effective for both classification problems and regression problems. Identifying three kinds of obfuscated Tor traffic is a multiclassification problem. Compared with deep learning, which requires more powerful hardware, machine learning is faster in the training phase and actual detection. Since the labels in this experiment are relatively easy to obtain, we can use a supervised machine learning algorithm to solve this problem.

In summary, our problem is a supervised and multiclassification problem. We choose seven machine learning algorithms, (1) XGBoost [28]; (2) GBDT [29]; (3) Random Forest [30]; (4) CART decision tree algorithm; (5) KNN (K-Nearest Neighbor) algorithm; (6) logistic regression; (7) Support vector machine [31], and compare the seven algorithms with each other.

4.4. Experiment for the Parameters of Sliding Window. The window size and sliding distance of a sliding window have effects on the result of the models. These effects include model precision rate, feature extraction time, and model prediction speed. We take different values for window size n and sliding distance m, and we use Random Forest algorithm to establish a model for testing.

In this experiment, we set the range of sliding distance m from 5 to 50 and the range of window size n from 50 to 500. The experimental results are shown in Table 6.

The higher the precision rate, the shorter the feature extraction time, and the shorter the model prediction time mean, the better the parameter effect. Therefore, the best parameters of the sliding window are a window size of 100 and a sliding distance of 10. It is worth noting that there are only minor differences between different parameters.

4.5. Experiment Results and Analysis. We use the seven machine learning algorithms selected in Section 4.3 to train the model with a training set and then use the validation set

Table 6: Result of the experiment for the parameters of the sliding window. Each item in the table is expressed in the form of precision rate/feature extraction time/model prediction time. We comprehensively consider three indicators (the importance of precision rate, feature extraction time, and model prediction time decrease in order) of each pair (n, m).

	$n = 50$	$n = 100$	$n = 200$	$n = 500$
$m = 5$	0.9984/1112/12	0.9991/1348/12	0.9990/1996/9.2	0.9991/3691/8.5
$m = 10$	0.9989/560/4.9	**0.9993/698/5.0**	0.9992/994/4.1	0.9993/1843/4.4
$m = 15$	0.9987/370/4.0	0.9991/470/2.9	0.9992/473.2.9	0.9993/1257/2.7
$m = 20$	0.9988/281/2.3	0.9990/350/2.0	0.9991/504/1.9	0.9992/939/2.0
$m = 50$	0.9984/113/0.8	0.9991/143/0.8	0.9990/206/0.7	0.9991/386/0.8

to verify and test set to test them. We record the result as a confusion matrix and calculate the precision rate, recall rate, and F1-Score. The results of the seven models that perform on the validation set and test set are shown in Tables 7 and 8, respectively.

From these two tables, we can conclude that the features extracted by the sliding window are so effective that the models such as XGBoost and Random Forest achieve high precision and recall rates. What is more, we can find that the four models tree-based algorithms, including XGBoost, GBDT, Random Forest, and CART decision tree, all have good performance. Even the worst-performing CART decision tree model also has a precision rate and recall rate of more than 90%. The XGBoost and Random Forest models have about 99% precision and recall rate in half of the categories. In addition, the KNN algorithm also has good performance. In contrast, the two linear classifiers, logistic regression and support vector machine, perform poorly in this task, with a minimum precision rate of 61.72% and a recall rate of 49.23%. All in all, the nonlinear classifier has good performance in identifying obfuscated Tor traffic.

Besides, almost every model has better detection effectiveness on Meek-based obfuscated traffic than on Obfs4-based and FTE-based Tor traffic. It can be attributed to the polling operation of the Meek plugin, which will introduce a lot of short packets in communication and be a significant characteristic. The effectiveness of Obfs4-based and FTE-based Tor traffic is a bit lower because of the similarity of the packet size and interval-related characteristics of these two kinds of Tor traffic. However, we distinguish them from each other by using the sliding window to split the packets, leading to a high precision rate for detecting Obfs4-based and FTE-based Tor traffic. The results of our experiments meet the analysis of the traffic characteristics in Sections 3.1 and 3.2.

4.6. Comparison. In Section 2, we mentioned four methods [20–23] that proposed detection methods for Meek-based Tor traffic, FTE-based Tor traffic, and Obfs4-based Tor traffic, respectively. In this section, we compare their methods with ours. The comparison result is shown in Table 9.

Our method has a high precision rate while being able to identify three types of obfuscated Tor traffic. Although the precision rate of detecting Meek-based Tor traffic and Obfs4-based Tor traffic is slightly inferior to the method that can only detect one kind of obfuscated Tor traffic, the gap is within 2%. The gap is because our model is a four-category model. If the problem is reduced to a two-category problem, our model will perform better. We try to implement another experiment with only Meek-based Tor traffic and normal traffic. In other words, we create a two-category model using the Random Forest algorithm. We train the model, evaluate it, and test it with the same dataset as the four-category model. Finally, we achieve a higher precision rate of 99.95% on the validation set and 99.92% on the testing set, which is almost the same as the precision achieved in [20].

5. Discussion

5.1. Computational Complexity. Since the model using the Random Forest algorithm achieves the best performance in the testing set, we take it as an example to calculate the computational complexity. As mentioned in [32], the Random Forest algorithm has an average time complexity $T(k)$ of $O(M\ dk\ \ \log k)$, where M indicates the number of trees, d indicates the number of features, and k indicates the number of samples. The time complexity of our model also depends on the window size and sliding distance of the sliding window, which, respectively, equal m and n. Thus, we can calculate the time complexity $T(k)$ as equation (12):

$$T(k) = O\left(Md\frac{kn}{m}\log\frac{kn}{m}\right). \tag{12}$$

Let $q = kn/m$; then we can rewrite equations (12) as equation (13):

$$T(k) = O(Mdq\ \ \log q). \tag{13}$$

In equation (13), q indicates the actual number of the samples in our experiments.

5.2. Limitation Analysis.

(1) Our method performs well on our own dataset, but we did not test them on other datasets because we cannot find open datasets of obfuscated Tor traffic. We expect to test on these datasets when they are accessible. And we believe that our algorithm is universal, and the sliding window mechanism and feature extraction mechanism are still applicable to other datasets.

(2) More obfuscation mechanisms may be introduced in the future, or some obfuscation mechanisms will upgrade. It would be a new challenge for the

Table 7: Evaluation results of models.

	Precision				Recall				F1-Score			
	Meek	Obfs4	FTE	Normal	Meek	Obfs4	FTE	Normal	Meek	Obfs4	FTE	Normal
XGBoost	**0.9991**	0.9908	0.9954	0.9992	0.9997	0.9961	0.9899	0.9986	0.9994	0.9934	0.9926	0.9989
GBDT	0.9936	0.9781	0.9877	0.9966	0.9986	0.9901	0.9756	0.9911	0.9961	0.9841	0.9816	0.9938
RF	0.9991	**0.9943**	**0.9972**	**0.9995**	**0.9997**	**0.9975**	**0.9937**	**0.9989**	**0.9994**	**0.9959**	**0.9954**	**0.9992**
CART	0.9707	0.9700	0.9234	0.9673	0.9761	0.9606	0.9632	0.9467	0.9734	0.9653	0.9429	0.9569
KNN	0.9947	0.9827	0.9883	0.9967	0.9978	0.9924	0.9833	0.991	0.9962	0.9875	0.9858	0.9938
Logistic	0.9344	0.8612	0.7794	0.8424	0.9855	0.7976	0.7290	0.8515	0.9593	0.8282	0.7534	0.8469
SVM	0.9959	0.8673	0.9087	0.8522	0.9314	0.7949	0.7959	0.9938	0.9626	0.8295	0.8486	0.9176

Table 8: Test results of models.

	Precision				Recall				F1-Score			
	Meek	Obfs4	FTE	Normal	Meek	Obfs4	FTE	Normal	Meek	Obfs4	FTE	Normal
XGBoost	0.9886	0.9741	0.9839	**0.9976**	0.9988	**0.9963**	**0.9840**	0.9814	0.9937	**0.9851**	0.9839	0.9894
GBDT	0.9758	0.9604	0.9795	0.9850	0.9944	0.9895	0.9650	0.9704	0.9850	0.9747	0.9722	0.9776
RF	**0.9951**	**0.9761**	**0.9886**	0.9960	**0.9989**	0.9942	0.9821	**0.9866**	**0.9970**	0.9851	**0.9853**	**0.9913**
CART	0.9218	0.9586	0.9011	0.9209	0.9568	0.9536	0.9094	0.8925	0.9390	0.9561	0.9052	0.9065
KNN	0.9882	0.9308	0.9826	0.9904	0.9912	0.9887	0.9707	0.9605	0.9897	0.9589	0.9766	0.9752
Logistic	0.8951	0.7729	0.6172	0.6620	0.9390	0.8238	0.4923	0.7164	0.9165	0.7975	0.5477	0.6881
SVM	0.9884	0.672	0.898	0.7447	0.8544	0.7946	0.6745	0.9473	0.9165	0.7282	0.7704	0.8339

Table 9: Comparison with other methods. If an item in the table is "—," it means that this method cannot detect such kind of obfuscated Tor traffic.

Method	Precision for Meek-based Tor traffic	Precision for FTE-based tor traffic	Precision for Obfs4-based Tor traffic
Ours	99.51%	**98.86%**	97.60%
Yao et al. [20]	**99.98%**	—	—
He et al. [21]	99.5%	—	—
Gao et al. [22]	—	94.8%	—
Zhai et al. [23]	—	—	**98.8%**

detection. We think that the sliding window mechanism is beneficial to highlight traffic features, and we can adjust the feature extraction mechanism to introduce such SSL/TLS-related features to alleviate these problems from new obfuscation mechanisms.

6. Conclusions

Tor has been used to hide the traffic of criminal activities on the darknet, so it is important to identify Tor traffic and block it when needed. Besides, Tor has introduced multiple kinds of obfuscation techniques to avoid traditional detection. However, existing detection methods only deal with one of the obfuscation techniques. Their detection granularity is in the unit of whole flow, which will lead to a relatively lower detection efficiency. Therefore, we proposed a sliding window-based method for detecting three kinds of obfuscated Tor traffic. We conducted a detailed analysis of the characteristics of different obfuscation mechanisms and analyzed the effectiveness of the extracted features, which were confirmed in the final experimental results and analysis.

We use the dataset collected by ourselves to conduct identification experiments. The test results' high precision and recall rate show that the 12 features extracted by the sliding window can effectively identify three kinds of obfuscated Tor traffic. And as a comparison to other methods which can identify only one type of obfuscated Tor traffic, the performance of our models is outstanding, even though we perform a multiclassification task. In summary, the features extracted through the sliding window and the models we use can effectively identify the three kinds of obfuscated Tor traffic.

Data Availability

The dataset of multiple kinds of obfuscated Tor traffic including FTE-based Tor traffic, Meek-based Tor traffic, and Obfs4-based Tor traffic is publicly available at https://github.com/QQQQing/Obfuscated-Tor-Traffic.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (no. 2018YFB0803503) and the National Natural Science Foundation of China (no. 61831007).

References

[1] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[2] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, “Anonymous connections and onion routing,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.

[3] I. Clarke, O. Sandberge, B. Wiley, and T. W. Hong, “Freenet: a distributed anonymous information storage and retrieval system,” in *Designing Privacy Enhancing Technologies-*Springer, Berlin, Germany, 2001.

[4] Security Alliance, “Know your ransomware: CTB-locker,” 2017, https://www.secalliance.com/blog/ransomware-ctb-locker.

[5] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, “Blocking-resistant communication through domain fronting,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 46–64, 2015.

[6] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Protocol misidentification made easy with format-transforming encryption,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, Berlin, Germany, November 2013.

[7] Yawning, “Obfs4,” 2014, https://github.com/Yawning/obfs4.

[8] R. M. Swarna Priya, P. K. R. Maddikunta, M. Parimala et al., “An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture,” *Computer Communications*, vol. 160, pp. 139–149, 2020.

[9] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz, and M. Alazab, “An adaptive multi-layer botnet detection technique using machine learning classifiers,” *Applied Sciences*, vol. 9, no. 11, p. 2375, 2019.

[10] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of tor traffic using time based features,” in *Proceedings of the ICISS*, Mumbai, India, December 2017.

[11] E. Hodo, X. Bellekens, E. Lorkyase, A. Hamilton, C. Tachtatzis, and R. Atkinson, “Machine learning approach for detection of nontor traffic,” in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, Reggio Calabria, Italy, August 2017.

[12] A. Almubayed, A. Hadi, A. Hadi, and J. Atoum, “A model for detecting tor encrypted traffic using supervised machine learning,” *International Journal of Computer Network and Information Security*, vol. 7, no. 7, pp. 10–23, 2015.

[13] P. Mayank and A. K. Singh, “Tor traffic identification,” in *Proceedings of the 2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 85–91, Nagpur, India, November 2017.

[14] A. Cuzzocrea, F. Martinelli, F. Mercaldo, and G. Vercelli, “Tor traffic analysis and detection via machine learning techniques,” in *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data) IEEE*, Boston, MA, USA, December 2017.

[15] Z. Rao, W. Niu, X. Zhang, and H. Li, “Tor anonymous traffic identification based on gravitational clustering,” *Peer-to-Peer Networking and Applications*, vol. 11, no. 3, pp. 592–601, 2018.

[16] G. He, M. Yang, J. Luo, and L. Zhang, “Online identification of Tor anonymous communication traffic,” *Ruanjian Xuebao/Journal of Software*, vol. 24, no. 3, pp. 540–556, 2013.

[17] J. Barker, P. Hannay, and P. Szewczyk, “Using traffic analysis to identify the second generation onion router,” in *Proceedings of the 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing*, IEEE, Melbourne, Australia, October 2011.

[18] X. Bai, Y. Zhang, and X. Niu, “Traffic identification of tor and web-mix,” in *Proceedings of the 2008 Eighth International Conference on Intelligent Systems Design and Applications*, IEEE, Kaohsiung, Taiwan, November 2008.

[19] S. Zhioua, “Tor traffic analysis using hidden markov models,” *Security & Communication Networks*, vol. 6, no. 9, pp. 1075–1086, 2013.

[20] Z. Yao, J. Ge, Y. Wu et al., “Meek-based tor traffic identification with hidden markov model,” in *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), EEE*, Exeter, UK, June 2018.

[21] Y. He, X. Li, M. Chen, and W. Wang, “A Tor anonymous communication identification method based on cloud traffic obfuscation,” *Journal of Engineering Science and Technology*, vol. 49, no. 2, pp. 121–132, 2017.

[22] R. Gao, “*Obfs4 Anonymous Network Traffic Identification Research*,” M.S. thesis, Beijing Jiaotong University, Beijing, China, 2018.

[23] Y. Zhai, “Research on Tor anonymous traffic identification technology based on FTE,” M. S. thesis, Tianjin University, Tianjin, China, 2018.

[24] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” *KDD Workshop*, vol. 10, no. 16, 1994.

[25] D. M. W. Powers, “Evaluation: from precision, recall and *F*-measure to ROC, informedness, markedness and correlation,” 2020, http://arxiv.org/abs/2010.16061.

[26] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, “Deep face recognition: a survey,” in *Proceedings of the 2018 31st SIB-GRAPI conference on graphics, patterns and images (SIB-GRAPI)*, IEEE, Paraná, Brazil, November 2018.

[27] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, “Intrusion detection by machine learning: a review,” *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994–12000, 2009.

[28] T. Chen and G. Carlos, “Xgboost: a scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, San Francisco, CA, USA, August 2016.

[29] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

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

[31] P. H. Chen, C. J. Lin, and B. Schölkopf, “A tutorial on ν-support vector machines,” *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005.

[32] A. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2017.