TMT-RF: Tunnel Mixed Traffic Classification Based on Random Forest

Panpan Zhao[1,2], Gaopeng Gou[1,2], Chang Liu[1,2], Yangyang Guan[1,2], Mingxin Cui[1,2], and Gang Xiong[1,2(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhaopanpan,gougaopeng,liuchang,guanyangyang,cuimingxin,xionggang}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Bejing, China

Abstract. With the explosive growth of the use of tunnels, network anomaly detection and security management are facing huge challenges, of which the first and an important step is tunnel traffic classification. Previous research on the classification of encrypted traffic is mainly based on machine learning methods using statistical features and deep learning methods using packet arrival time and packet length sequence. However, these works mainly focus on the identification of single application traffic. In a real scenario where a single user uses a tunnel, the traffic within a time may contain multiple applications. Due to the tunnel traffic has the same five-tuple, we can't get the start and end times of each application. Compared with encrypted application traffic classification, it is more difficult to identify applications in tunnels. In this paper, firstly we propose a TMT-RF framework to identify two mixed applications in IPSec tunnels. Then we introduce the first use of NoiseSplit module to split the traffic and then use a CombineBurst module for the second split. Finally, we collected four mixed traffic data sets of three types to evaluate our proposed method. Experimental results demonstrate that TMT-RF not only achieves a splitting accuracy of 93% in positive-time separation applications, but also outperforms other state-of-the-art methods on the data sets for zero-time separation applications and negative-time separation applications.

Keywords: Traffic classification · IPSec tunnel · Machine learning

1 Introduction

With the development of the Internet, the volume of encrypted network traffic is growing rapidly. Due to the encapsulation and encryption characteristics of tunnel traffic, it occupies a large part of the encrypted traffic [18]. More and more users use tunnel technology to protect the security of communication [4,12]. While the tunnel technology guarantees communication security, it also brings challenges to network management. According to Gartner's 2020 report

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 398, pp. 418–437, 2021.
https://doi.org/10.1007/978-3-030-90019-9_21

[26], 70% of malicious network services bypass firewalls and intrusion detection systems through encryption and tunneling technology. Therefore, tunnel traffic identification is important in the network security and network management domain [1,2,5,15,16,24].

Prior studies have proposed some methods of encrypted traffic identification which is mainly divided into methods based on machine learning [8,14,17] and deep learning [11,28]. The machine learning method mainly extracts the following traffic features from the encrypted traffic, such as the first n packet length sequence with direction [17], packet length statistical characteristics [8] and statistical characteristics of packet arrival time interval [14]. Deep learning methods such as convolutional auto-encoding and convolutional neural networks are used for encrypted traffic classification [11].

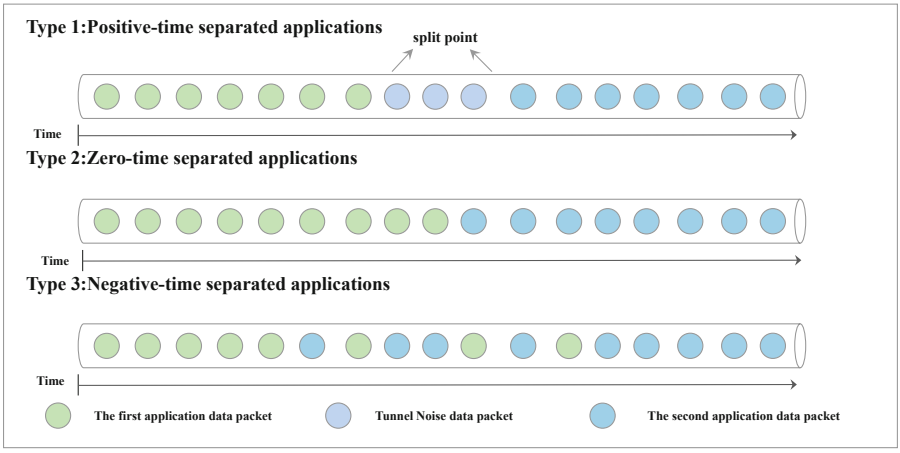

Fig. 1. Three types of mixed traffic in a tunnel

Mixed encrypted application traffic can be identified by flow classification because each application traffic has a different five-tuple (source IP, destination IP, source port, destination port, IP version). However, it is a challenging task to identify tunnel traffic accurately and efficiently. Firstly, the application traffic is encapsulated and encrypted in tunnels, and great improvements have been made to the original feature details [21]. Moreover, the application traffic in a tunnel has the same five-tuple [9], it is difficult to find the start and end time of the application. The encrypted traffic classification methods cannot be directly used to classify tunnel traffic. Secondly, there is a great difference in the mixture of tunnel traffic. The outcome of the classification is influenced by the form and overlap rate of the mixture.

Recently, researchers have proposed some methods for the classification of mixed websites on Tor which first split the mixed flow into a single flow, and then use the existing encrypted traffic classification model to classify the single

flow. Splitting the mixed stream is mainly to find the split point. These methods propose to use data packet arrival time interval [29] and packet direction [6] characteristics, and feed these characteristics into KNN, Hidden Markov Model to identify split points [30]. The recognition result of the split points are affected by the mixed format and mixed rate of the applications.

In this paper, we propose a traffic classification method in a scenario where a single user using lots of different applications in an IPSec tunnel, relaxing the single application assumption. Figure 1 illustrates the terminology and scenario of this paper. For the first type of traffic, we try to find the split point and then classify the application traffic. We no longer seek to find the split point for the second and third forms of traffic but we split the mixed flow into many segments according to certain laws to classify the application traffic. To develop a successful tunnel traffic classification method, we make the following novel contributions:

- We innovatively propose a **framework** for identifying mixed traffic in IPSec tunnels, which can realize the classification of mixed traffic in a single user using many different applications scenarios. The framework specifically includes two split modules and a classification module. This is the first study to achieve the classification of mixed applications traffic in IPSec tunnels.
- A **module** to identify positive-time separated applications (Type 1). We propose a splitting algorithm based on the combination of packet length threshold and classifier. The results show that the splitting accuracy reaches 93%. Compared with processing without this module, our classification accuracy has been improved by 30%.
- A **module** to identify applications with zero-time separated (Type 2) and negative-time separated (Type 3). We propose a classification algorithm which uses a splitting algorithm to split the network data stream into several segments, and then predicts each segment, ignoring the impact of overlapping parts on application identification. Experiments show that the module achieves the best performance on overlapping data sets and is better than the most advanced methods (Sectioning).
- A **playback** method for labeling of mixed traffic in IPSec tunnels. Using this method, three types of mixed traffic in the tunnel are generated: positive-time separation applications traffic, zero-time separation applications traffic, and negative-time separation applications traffic.

The remaining of the paper are organized as follows. Section 2 reviews the related works. Then, Sect. 3 presents the two proposed algorithm models in detail. Section 4 describes the data collection process. Section 5 displays the experimental result. Finally, the paper is concluded in Sect. 6.

2 Related Work

Many researchers have proposed a wealth of encrypted traffic analysis methods, such as, based on machine learning and deep learning. Recent researchers have

also proposed many methods for identifying mixed traffic on Tor. Next, we will introduce these three parts in detail.

2.1 Machine-Learning-Based Methods

In 2014, based on the burst feature and Haar feature, Shi et al. [22] used Bayesian networks to achieve 96.7% of dynamic pages and 97.8% of static pages. Faiz et al. [14] employed a set of Flow Spatio-Temporal Features (FSTF) to six well-known classifiers and the boosting technique consistently performed the best on all the given datasets using the FSTF. The results show that the classification accuracy of VoIP traffic is above 98.5%.

Meng et al.[17] selected 8 consecutive data packets after the tcp connection is established as the feature, such as packet length, packet time interval and packet direction, to describe network traffic using five algorithms: svm, c4.5, k-means, bayes and EM. They achieved an accuracy of 94.5% and described three types of traffic including http over ssh, ftp over ssh, scp and sftp using k-means. Alice et al. [8] used svm and gmm algorithms to select the first n directional packet length sequences of each ssh flow, and the accuracy of the two classification methods was 71.5% for seven types of traffic. Ding et al. [7] used 20-dimensional packet length and packet time statistical characteristics to describe network traffic using the c4.5 algorithm, which achieved an accuracy rate of 95.3%. The described http traffic includes http, smtp over http, and p2p over http.

2.2 Deep-Learning-Based Methods

He et al. [13] converted the payload of the data packet into a grayscale image and then classified it through a convolutional neural network. The classification accuracy of the five applications reached 98.5%. Vu et al. [27] developed a novel time-series feature extraction technique. The proposed method consists of two main steps. First, A feature engineering technique to extract significant attributes of the encrypted network traffic behavior through analyzing the time series of receiving packets. In the second step, a deep learning technique is developed to exploit the advantage of time series data samples in providing a strong representation of the encrypted network applications. Guo et al. [11] preprocessed the network traffic of six applications into conversation pictures, and then used convolutional auto-encoding and convolutional neural network algorithms. The accuracy rate of the algorithm classification based on cnn was 92.92%.

Zhou et al. [32] classified vpn and non-vpn traffic by using entropy estimation, and then used convolutional neural networks for further classification. The input of the convolutional neural network was packet length, packet time interval, and packet direction. Ali et al. [20] presented an end-to-end traffic classification method, using a multilayer perceptron and a recurrent neural network algorithm. Lu et al. [27] proposed a method of using time series to classify encrypted traffic. The first step was to extract time-series information, and the second step was to use the LSTM algorithm for classification.

2.3 Mixed Traffic Classification Methods

Gu et al. [10] first proposed a study on the classification of mixed traffic on Tor. They proposed to use the thinking time between two web pages to identify the first-page using fine-grained features and the second page using coarse-grained features. Wang et al. [29] presented a mixed website traffic identification framework on Tor. The framework used the splitting decision algorithm to judge whether the time threshold split was successful, and used the splitting finding algorithm to further split the traffic that failed to pre-split. This method has a poor splitting effect on overlapping traffic.

Cui et al. [6] proposed a splitting method of continuous websites based on the hidden markov model and classification algorithm of overlapping websites based on Sectioning. The algorithm based on the hidden markov model is too complicated, and the state transition probability within each website needs to be calculated every time. There are certain problems with the use of Sectioning algorithms. One of these is that there is a bad classification result for the mixed situation of short and long streams.

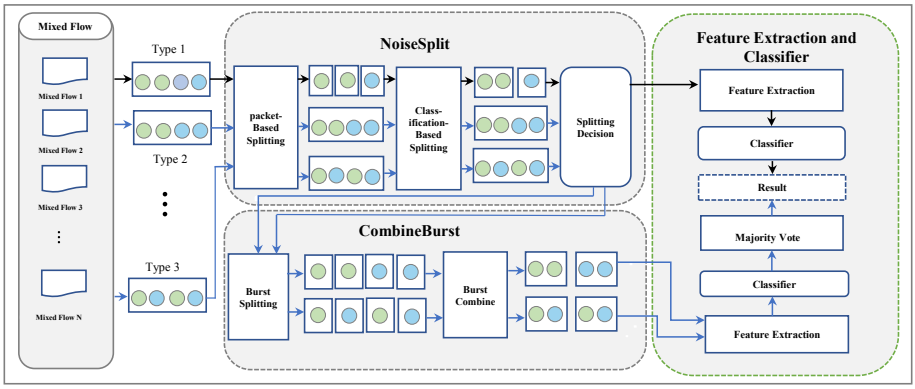

Fig. 2. The TMT-RF framework

3 Methodology

In this section, Fig. 2 presents the TMT-RF framework for the tunnel traffic classification. The upper middle part of Fig. 2 demonstrates the process in which type 1 traffic is split through NoiseSplit module. The lower middle part of Fig. 2 shows the CombineBurst module's traffic splitting for type 2 and type 3. The right part of Fig. 2 demonstrates the process in which the features are acquired through the feature extraction and classification results are acquired through the classifier.

The tunnel mentioned in this paper specifically refers to the IPSec tunnel. Internet Protocol Security (IPSec) is a secure network protocol suite that authenticates and encrypts the packets of data to provide secure encrypted communication between two computers over an Internet protocol network. IPSec mainly includes Authentication Headers (AH), Encapsulating Security Payloads (ESP) and Internet Security Association and Key Management Protocol (ISAKMP).

The traffic types (type 1, type 2, type 3) appearing in Fig. 2 are the same as those in Fig. 1. The three types of traffic represent three scenarios for using applications in IPSec tunnels. The type 1 traffic is generated in a scenario where there is a period of time between the first application and the second. The type 2 traffic is generated when there is no time gap between the first application and the second application. When there is overlap between the first application and the second application, the type 3 traffic is generated.

3.1 NoiseSplit

There are three steps in NoiseSplit module, as shown in Fig. 2. The positive time of the two applications is accompanied by tunnel noise which is the heartbeat packets in IPSec tunnels. Therefore, the split of mixed traffic is based on a rule that if multiple consecutive packets are identified as tunnel noise, mixed tunnel traffic will be split up into two segments. Since the packet length of tunnel noise traffic is extremely distinguishable, to make it efficient and fast, this module uses the packet length threshold method for preliminary splitting.

Based on the split of the packet length threshold, some application packets below the threshold may also be identified as tunnel noise, resulting in multiple cuts. We use machine learning technology to identify the noise filtered out in the first part and decide whether to split. After packet-based splitting and classifier-based splitting produce multiple different results, split decision selects different processing for different results. Three types of results will be generated from packet-based splitting, including that mixed traffic is split correctly, single application traffic is split, and multi-applications mixed traffic is not split.

Packet-Based Splitting. The first type of mixed traffic can be split by tunnel noise. As shown in Fig. 3, it can be seen that the packet length distribution of tunnel noise is quite different from the applied packet length distribution. The packet length of tunnel noise is distributed in a small value range. Therefore, the tunnel noise can be identified through the packet length feature, and then the mixed traffic can be split. L_{split} is the packet length threshold for identifying tunnel noise. Our choice of L_{split} seeks to maximize the chance of tunnel noise identification accuracy. If the value of L_{split} is inappropriate, it may cause multiple splits and non-splits of the mixed flow. We hope that any source of error is impossible. Therefore, an appropriate value must be selected. We will show how to choose L_{split} and how it affects the accuracy of splitting in the experiment.

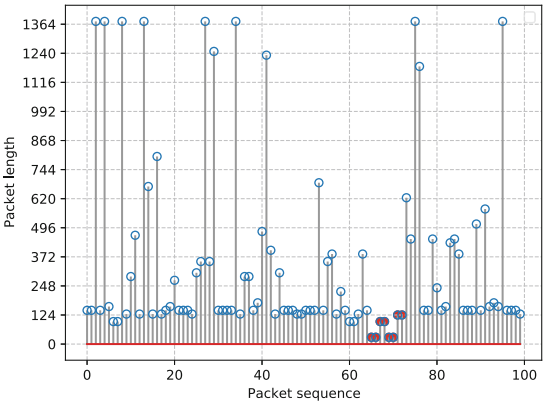

Fig. 3. Packet length distribution of positive-time separation applications

Classification-Based Splitting. The split method based on the packet length threshold may have three results. The first result is that mixed traffic is split correctly. This result does not need to be further processed and can be directly inputted into the classifier for classification. The second result is that single application traffic is split. This happens when the packet length threshold is inappropriate. There is a high probability that mixed traffic will be split into multiple parts. The classification-based method is to reduce the occurrence of this result. The third result is that multi-applications mixed traffic is not split. This happens when there is overlap between two applications. The classifier-based approach does not try to solve this problem, and the failure of the split will be further processed in the CombineBurst Module.

The idea of splitting based on the classifier is aimed to not only further improve the accuracy of tunnel noise identification but also greatly enhance the accuracy of splitting through the classifier. Random forest is an ensemble learning algorithm that is composed of decision trees [19]. It is an extended variant of bagging. The randomness of random forest is mainly reflected in the random training samples of each tree, and the selection of attributes is random. Random forest is simple and efficient, with low overhead.

For the split based on the classifier, we choose the random forest classifier, using the 54-dimensional statistical features of the packet-length sequence. The accuracy of split is measured by the following formula.

$$diff(P,Q) = \begin{cases} 1 & Q - R \leq P \leq Q + R \\ 0 & else \end{cases} \tag{1}$$

where P is the predicted value, Q is the actual value, R is the error range.

$$SA = \frac{1}{N} \sum_{i=1}^{N} diff(P_{predict}(i), P_{true}(i)) \tag{2}$$

where P_{true}(i) is the packet sequence number of the real split point of the i-th sample, $P_{predict}$(i) is the packet sequence number of the predict split point of the i-th sample, N is the total number of samples.

Splitting Decision. After the first two splits, there may be the three results mentioned above. It is necessary to identify different results for further processing. In the splitting decision, we use the knn classifier, which is an supervised learning algorithm [23]. KNN obtains k nearest neighbors based on the distance measurement method of formula (3). In knn classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

$$L(x_i, x_j) = (\sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}|^2)^{\frac{1}{2}} \tag{3}$$

The splitting decision takes single application traffic and multi-applications traffic as input, and uses knn for binary classification. If the input is multi-applications mixed traffic, the prediction is true. Otherwise, the output is false. If the output result is true, it needs to be further processed by the combineBurst module and then be sent to the classifier. If the output result is false, it can be directly sent to the classifier for classification.

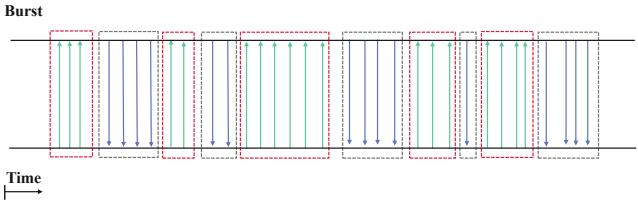

Fig. 4. Visualisation of burst.

3.2 CombineBurst

The NoiseSplit module can split the first type of mixed traffic well, but it cannot split the second and third types of mixed traffic. The combineBurst module is used to solve these types of traffic. There is overlap in the third type of mixed traffic. The overlap rate is the ratio of the total number of packets in the overlapping part to the total number of packets in the entire data stream, which is expressed as follows.

$$OR(T) = \frac{T_{overlap}}{T} \tag{4}$$

where $T_{overlap}$ is the overlapping part of the traffic, and T is the entire traffic. The overlapping part does not help the identification of applications. The previous

idea is to spend time finding the split point, and then inputs the classifier to classify. The combineBurst module no longer spends most of the time searching for split points, but splits overlapping traffic into lots of segments and makes predictions for each segment.

A burst is the group of all network packets occurring together that satisfiesthe condition that the most recent packets with the same direction, the burst direction, of the previous packet [25,31]. Data packets from the client to the server are marked with a positive sign. Data packets from the server to the client are marked with a negative sign. This is visually depicted in the traffic burstification section of Fig. 4, where we can see the burst is separated by the packet direction.

Burst Splitting. The granularity of the data packet is too fine, while the granularity of the data stream is too coarse. The data stream can be split up into bursts according to specific tasks. A burst is composed of several consecutive data packets in the same direction. Overlapping traffic is split up into several parts according to burst. The length of overlapping streams of different applications is different, and the number of bursts generated may also be different. A burst may contain one, two, or multiple data packets.

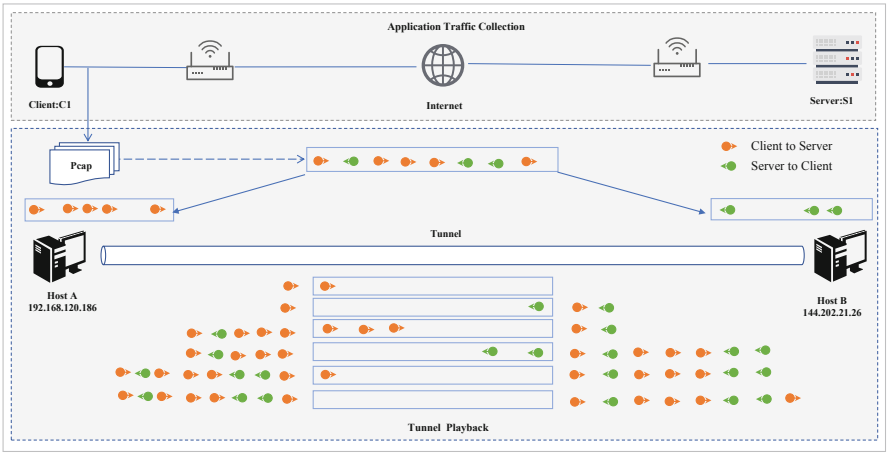

Fig. 5. Play back application data in a tunnel

Burst Combine. After overlapping traffic is split according to bursts, multiple bursts will be generated. If we simply split according to burst and then identify, there are the following two problems:

– Limited representation ability. The data packets contained in the burst are too few, and the ability to describe the application is limited.

– Increased delay time. If we split according to burst, a lot of data will be generated. Each burst needs to be classified, which will increase the delay time of the classifier.

To solve the above two problems caused by burst splitting, we propose to combine several continuous bursts into a segment to split. If a segment contains too many bursts, it will reduce the delay time. Meanwhile, the segment may contain more than two application data packets, it also will cause a drop in split accuracy. If a segment contains too few bursts, it will not reduce the delay time and increase the burden of the classifier. Therefore, we hope to choose a suitable value $N_{combine}$ that satisfies the following conditions as much as possible:

– Each segment should contain as few applications as possible. Ideally, each segment contains only a single application.
– Each segment should be as long as possible to contain as much information as possible for the corresponding application, while reducing the burden of the classifier.

It is easy to see that the above two conditions are contradictory: the shorter the segment, the less information it contains. The longer the segment, the easier it is to include multiple applications. We will show how to choose $N_{combine}$ and how it affects the accuracy of classification in the experiment.

3.3 Feature Extraction and Classifier

Feature Extraction. Feature extraction involves extracting 54 statistical features from each segment (using the features in Appscanner). For each segment, three series of packets are considered, including incoming packets only, outgoing packets only, and two-way traffic. Two-way traffic consists of incoming and outgoing packets. For each series (3 in total), we calculate the following values, including minimum, maximum, average, median absolute deviation, standard deviation, variance, skewness, kurtosis, percentile (from 10% to 90%), and the number of elements in the series (18 in total) [25].

Classifier. We use the Random Forest model as the classifier. The specific introduction of the random forest model is in the classifier-based module. The difference between the random forest model and the previous one is: the random forest model based on the classifier module is mainly used to identify whether it is tunnel noise through binary classifications, and the random forest model of this module is mainly multi-classification to identify the applications.

Majority Vote. KNN obtains the classification result through majority voting of k nearest neighbors. Majority voting is also used in our framework. The classifier will output the results for multiple segments of a stream, and through majority voting, to determine which applications constitute the data stream.

4 Data Collection

In this section, we introduce the data collection framework, mainly including application traffic collection and tunnel playback. Next, we will introduce each part in detail.

4.1 Application Traffic Collection

Since mobile traffic involves user private data, public mobile traffic data sets are currently not available. Therefore, existing work uses self-collected data sets to validate the proposed method. Firstly, we use black domain software to turn off irrelevant applications in the background (Black domain is a free super practical optimization tool for android rogue apps. It does not require root permission to perfectly prevent app from starting, running in the background, and waking up in the background).

Then the adb-monkey [3] is used to randomly click on the app, using tcpdump to capture application traffic. We collected traffic of 30 applications in seven types. These seven types are mainly instant messaging applications, file transfer applications, email client applications, email web applications, media applications, social applications, and VoIP applications. These data will be used in tunnel playback module.

4.2 Tunnel Playback

Existing tunnel traffic collection methods have some shortcomings and poor scalability. Firstly, the existing tunnel traffic collection methods are difficult to solve the labeling problem of tunnel mixed traffic. Secondly, the traffic in the tunnel has the same five-tuple, so it is difficult to filter out the background traffic. There is a lot of background traffic in the application traffic, which will make it difficult for the application to get the accurate ground truth. Besides, when collecting mobile applications traffic, it is necessary to obtain the root permission of the mobile device, resulting in poor scalability.

To overcome the above shortcomings, we propose a method to generate tunnel traffic using tunnel playback. Tunnel playback is to simulate the client and server of the application at both ends of the host that establishes the tunnel to send packets. Figure 5 details the process of tunnel playback. During playback, the wait-stop mechanism and the timeout retransmission mechanism are used to ensure orderly packet sending and reduce packet loss.

The tunnel playback process is as follows. Firstly, a tunnel is established between host A and host B. Host A simulates the client of the application, and host B simulates the server of the application. Then, host A reads the data packets in the c2s (client to server) direction of the pcap file in order and sends them to host B through the tunnel. Next, host B reads the pcap file in order, and when it receives the previous data packet in the c2s direction, it sends data packets in the s2c (server to client) direction to host A. The two hosts read and send packets in sequence. Finally, the packets in pcap are completely sent to the tunnel. During tunnel playback, we use tcpdump to capture tunnel traffic on the host A.

The generation of tunnel mixed traffic depends on the application traffic collected by the application traffic collection module. In the experiment, we collected single-application label traffic and multi-applications label traffic in IPSec tunnels. For single-label applications, we collected 7 types of tunnel traffic for 30 applications through tunnel playback, and each class contains 100 instances. For the Multi-applications label dataset, we collected four three types of tunnel traffic. Through pcap splitting, random (different applications traffic) and orderly (same applications traffic) mixing, pcap merging, tunnel playback and other operations, three types of mixed traffic in the tunnel are generated. We generated three types of mixed traffic using 30 applications traffic collected by the application traffic collection module.

- **Positive-time separation applications traffic**. From 30 applications, we arbitrarily select two applications to mix, and collect mixed traffic containing 60 classes, each class includes 100 instances.
- **Zero-time separation applications traffic**. This data set contains 60 classes, and each class contains 100 instances.
- **Negative-time separation applications traffic**. The application traffic with negative time separation includes two types of data sets, 5% and 10%. Each type of data set includes 60 classes, and each class includes 100 instances.

5 Experimental Evaluation

In this section, we first introduce the experiment setup. Then, we analyzed the impact of three factors (data packet length threshold, measurement range and number of burst combinations) on the classification results. Finally we analyzed the classification results of traffic in three scenarios, and the result of the comparison experiment.

5.1 Experiment Setup

Comparison Methods. Some of the most advanced methods as a comparison method are summarized as follows.

- Sectioning (Packet-based) [6] uses data packets to split the mixed stream evenly into several sections, predicts each section, and gets the classification result.
- Sectioning (Time-based) uses time information to split the mixed stream evenly into several sections, and each section is predicted to obtain the classification result.

The classifier used by the sectioning algorithm is knn. In the comparative experiment, we find that random forest is better than knn for classification, so we choose random forest.

Setting of NoiseSplit Module. The selection of L_{split} will be introduced in the experiments. The noise classifier takes 54-dimensional statistical features of the packet-length sequence as input, and the classifier uses the random forest model. Splitting decision uses the knn model.

Setting of CombineBurst Module. The selection of $N_{combine}$ will be introduced in the experiments. The classifier takes 54-dimensional statistical features of the packet-length sequence as input, and the classifier uses a random forest model.

Cross-Validation. Our first step is to split each application instance into the training and test sets under 10-fold cross-validation. 10% of the instances are in the test set and the rest are in the training set. We replay the mixed 90% of the collected application instances into the tunnel to generate training sets, and replay the mixed 10% of the traffic into the tunnel to generate test sets.

Assessment Criteria. We analyze the accuracy of the split point. Formula 2 specifically introduces the measurement index of the result of finding the split point-splitting accuracy. For the classification of applications, we use the following metrics to measure.

- Precision: Precision is calculated using Eq. (5).

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

- Recall: Recall is calculated according to Eq. (6).

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

- F1-score: F1 is calculated according to Eq. (7), and it's the harmonic mean of precision and recall.

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

- Accuracy: Accuracy is calculated according to Eq. (8).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{8}$$

  where TP refers to the number of true positives, FP refers to the number of false positives, FN refers to the number of false negatives, and TN refers to the number of true negatives.

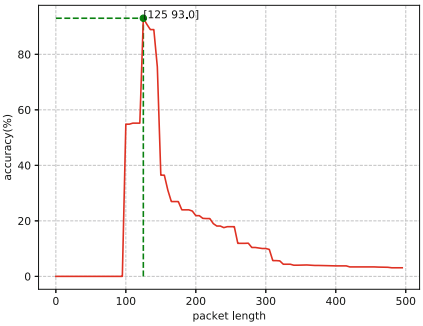

Fig. 6. The accuracy of the split varies with the packet length threshold (L_{split})

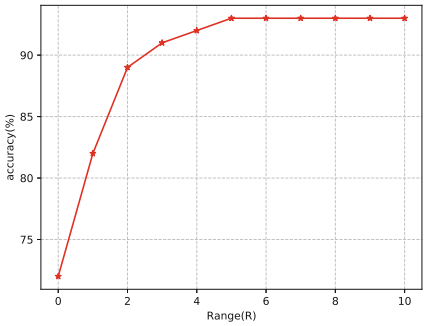

Fig. 7. Prediction accuracy of split with varying measure range (R)

5.2 Impact of Packet Length Threshold

In order to analyze the impact of packet length threshold on splitting accuracy, we conducted experiments to split mixed traffic by packet length threshold. In the experiment, L_{split} takes a value from 5 to 500.

Figure 6 shows the accuracy that results from correctly splitting mixed flow, when using different packet length thresholds. When the packet length threshold is 100 or 145, the classification result immediately changes greatly. If the packet length is in the range of 100–145, the accuracy is greater than 50%. When the packet length is 125, the splitting accuracy reaches the highest value, which is 93%. The results show that when the packet length threshold is between 100 and 145, it helps to improve the accuracy of the split.

It can be seen that the packet length threshold has a great influence on the accuracy of the split. As the packet length threshold increases, the accuracy of splitting first increases and then presents a downward trend. If L_{split} is selected too small, it will be lower than the noise packet length, and it is difficult to identify tunnel noise, which reduces the accuracy of the split. When L_{split} is selected too large, the packet length higher than the tunnel noise will filter out some applications traffic and reduce the accuracy of the split.

5.3 Impact of Measure Range

To analyze the changes of splitting accuracy under different measurement ranges, we conducted an experiment, and the results of the experiment are shown in Fig. 7.

The R in formula (1) means the R packets within the error range of the split point are all correct split points. As shown in Fig. 7, the prediction accuracy of split has varying measure ranges. With the increase of R from 0 to 10, the splitting accuracy of the mixed stream has been increased from 71% to 93%.

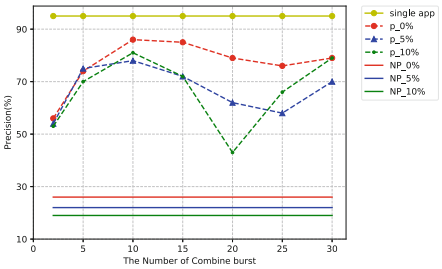

Fig. 8. Prediction accuracy of the first app with varying number of combine burst ($N_{combine}$) and overlap %

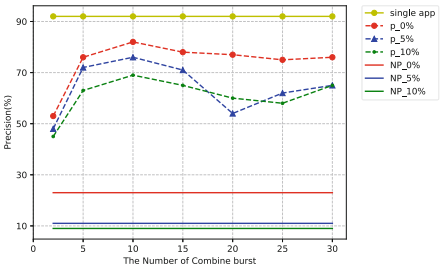

Fig. 9. Prediction accuracy of the second app with varying number of combine burst ($N_{combine}$) and overlap %

When the value of R reaches 5, the accuracy of classification no longer increases. The result shows that the split deviation of the split point is maintained within 5 packets before and after.

Figure 7 shows that with the increase of R, the accuracy of recognition just started to increase and then stabilized. When the R setting is small, the accuracy will be very low. Therefore, the R setting should be as large as possible within the error tolerance range in the experiment.

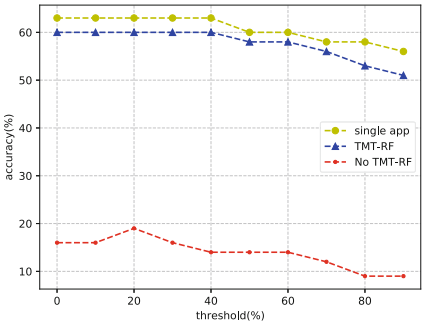

Fig. 10. Prediction accuracy of first app with varying possibility threshold

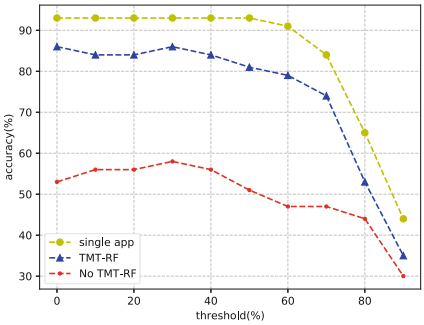

Fig. 11. Prediction accuracy of second app with varying possibility threshold

5.4 Impact of the Number of Burst Combinations

We conducted an experiment to explore the effect of the number of combined bursts on the classification results. The experimental results are shown in Figs. 8 and 9. In the results, P represents the classification result processed by the TMT-RF framework, and NP represents the classification result not processed by the TMT-RF framework.

Figures 8 and 9 show the precision of the first and the second application classification respectively. Under different overlap rates, the classification results

of the applications vary greatly. Figures 8 and 9 present that the classification precision of mixed traffic has increased by 50% after processing by the CombineBurst module. As the combined burst value is different, the precision of the classification also changes. Experiments show that if the number of bursts combined is 10 or 15, it will help improve the classification results.

The experimental results show that the overall trend of rising first and then falling. Too small $N_{combine}$ selection may result in too little applications information included, which makes it difficult to identify effectively, and reduces the precision of identification. Too large selection of $N_{combine}$ may result in the inclusion of multiple applications information and reduce the precision of recognition.

5.5 Applications Classification of Positive-Time Separation

Through the above analysis of the impact of the packet length threshold and measurement range on the splitting accuracy, in the experiment, we set L_{split} as 125 and R as 6. As shown in Figs. 10 and 11, after processing by the TMT-RF framework, the recognition result of the first application in the mixed traffic is very close to the recognition result of a single application.

Compared with those not processed by the TMT-RF framework, the classification result is increased by 30%. Compared with the second application, the first application is closer to the classification result of a single application. Overall, it can be seen that the mixed flow has a greater impact on the second application than the first. It indicates that the first few packets have a greater impact on the accuracy of tunnel traffic identification.

Table 1. The precision of the applications under different overlap rates

R(%)	0 overlp					5 overlap					10 overlap				
P(%)	0	20	40	60	80	0	20	40	60	80	0	20	40	60	80
First	0.87	0.87	0.85	0.87	0.84	0.83	0.75	0.61	0.83	0.71	0.78	0.78	0.70	0.76	0.71
Second	0.84	0.84	0.84	0.74	0.71	0.76	0.76	0.65	0.58	0.56	0.69	0.69	0.67	0.62	0.57

Table 2. Experimental result on precision, recall and F1 (The Best result are in bold)

Algorithm	0% overlap						5% overlap						10% overlap					
	First APP			Second APP			First APP			Second APP			First APP			Second APP		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
CombineBurst	**0.87**	**0.71**	**0.76**	**0.74**	**0.45**	**0.56**	**0.83**	**0.69**	**0.73**	**0.58**	**0.40**	**0.47**	**0.76**	**0.69**	**0.72**	**0.62**	0.17	**0.27**
Sectioning(Packet-based)	0.83	0.18	0.22	0.45	0.33	0.34	0.61	0.11	0.13	**0.58**	0.29	0.38	0.59	0.09	0.11	0.52	0.17	0.22
Sectioning(Time-based)	0.78	0.45	0.51	0.64	0.25	0.26	0.76	0.26	0.28	0.47	0.29	0.27	0.72	0.26	0.30	0.32	**0.23**	0.21

5.6 Applications Classification of Zero-Time Separation

By analyzing the impact of the number of burst Combinations on application classification, we set $N_{combine}$ to 10. In this scenario, there is no overlap between the two applications, and all three methods have achieved good results. In Table 1, R represents the overlap rate of application traffic, and P represents the probability threshold. Table 1 shows that the classification results of our method do not change much, under different output probability thresholds. The classification result of the first application is better than the classification result of the second application

Table 2 shows that the CombineBurst method shows a very good performance than other methods on this data set. The CombineBurst gets 87% precision and 76% f1 value. Overall, combineBurst has achieved good performance, because it uses information between data streams in the same direction. The results demonstrate that CombineBurst method is very helpful to improve the identification precision of tunnel traffic.

5.7 Applications Classification of Negative-Time Separation

We set $N_{combine}$ to 10 as shown above. As shown in Table 1, there are classification results with different overlap rates in this scenario. Table 1 shows that overlap rate has a great impact on application identification, the higher the overlap rate, the greater the impact. Table 1 also shows that the output threshold also have a certain impact on the accuracy of recognition.

As shown in Table 2, as the overlap rate increases, the gap between the CombineBurst method and other methods gradually increases. Compared with the most advanced methods, CombineBurst shows the best performance at different overlap rates. The precision of the combineBurst method is slightly greater than the other two methods, and the recall rate and f1 value are much larger than these two comparison methods, mainly due to the combinationBurst method using the direction information between packets. With the increase of the overlap rate, the combineBurst method has the least decrease in precision, recall, and f1 value, showing strong robustness in the comparison method.

6 Conclusion

In this paper, we propose a classification framework for mixed traffic in a single user usage scenario in IPSec tunnels. In this scenario, each flow is classified according to the application visited by the user in the flow. To achieve the identification of three types of traffic in IPSec tunnels, we propose the NoiseSplit module to segment and classify the first type of traffic, and the CombineBurst module handles the second and third types of traffic. We collected four mixed traffic data sets of three types to evaluate our proposed method. Experimental results show that TMT-RF not only achieves the best performance of 93% on the data set for positive-time separation applications, but also outperforms

other state-of-the-art methods on the data sets for zero-time separation applications and negative-time separation applications. Future research needs to further explore the classification of applications used in multi-person scenarios in IPSec tunnels.

Acknowledgement. This work is supported by The National Key Research and Development Program of China (No. 2020YFE0200500, No. 2018YFB1800200 and No.2020YFB1006100) and Key research and Development Program for Guangdong Province under grant No. 2019B010137003.

References

1. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile encrypted traffic classification using deep learning. In: 2018 Network Traffic Measurement and Analysis Conference (TMA), pp. 1–8. IEEE (2018)
2. Aceto, G., Ciuonzo, D., Montieri, A., Pescapè, A.: Mimetic: mobile encrypted traffic classification using multimodal deep learning. Comput. Netw. **165**, 106944 (2019)
3. Alam, M.S., Vuong, S.T.: Random forest classification for detecting android malware. In: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 663–669. IEEE (2013)
4. Booth III, E.H., Lingafelt, C.S., Nguyen, P.T., Temoshenko, L., Wang, X.: System and method to monitor and determine if an active ipsec tunnel has become disabled. US Patent 6,668,282, 23 Dec 2003
5. Cao, Z., Xiong, G., Zhao, Y., Li, Z., Guo, L.: A survey on encrypted traffic classification. In: Batten, L., Li, G., Niu, W., Warren, M. (eds.) ATIS 2014. CCIS, vol. 490, pp. 73–81. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45670-5_8
6. Cui, W., Chen, T., Fields, C., Chen, J., Sierra, A., Chan-Tin, E.: Revisiting assumptions for website fingerprinting attacks. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 328–339 (2019)
7. Ding, Y.j., Cai, W.d.: A method for http-tunnel detection based on statistical features of traffic. In: 2011 IEEE 3rd International Conference on Communication Software and Networks, pp. 247–250. IEEE (2011)
8. Dusi, M., Este, A., Gringoli, F., Salgarelli, L.: Identifying the traffic of ssh-encrypted applications (2014)
9. Freier, A., Karlton, P., Kocher, P.: The secure sockets layer (SSL) protocol version 3.0. Tech. rep., RFC 6101 **11** (2011)
10. Gu, X., Yang, M., Luo, J.: A novel website fingerprinting attack against multi-tab browsing behavior. In: 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 234–239. IEEE (2015)
11. Guo, L., Wu, Q., Liu, S., Duan, M., Li, H., Sun, J.: Deep learning-based real-time VPN encrypted traffic identification methods. J. Real Time Image Process. **17**(1), 103–114 (2020)
12. Hamed, H., Al-Shaer, E., Marrero, W.: Modeling and verification of IPSEC and VPN security policies. In: 13th IEEE International Conference on Network Protocols (ICNP 2005), pp. 10-pp. IEEE (2005)
13. He, L., Shi, Y.: Identification of SSH applications based on convolutional neural network. In: Proceedings of the 2018 International Conference on Internet and e-Business, pp. 198–201 (2018)

14. Islam, F.U., Liu, G., Liu, W.: Identifying voip traffic in VPN tunnel via flow spatio-temporal features. Math. Biosci. Eng. **17**(5), 4747–4772 (2020)
15. Korczyński, M., Duda, A.: Markov chain fingerprinting to classify encrypted traffic. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 781–789. IEEE (2014)
16. Lotfollahi, M., Siavoshani, M.J., Zade, R.S.H., Saberian, M.: Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Comput. **24**(3), 1999–2012 (2020)
17. Meng, J., Wang, L., Xiong, G., Yao, Y.: Study on SSH application classification based on machine learning. Comput. Res. Dev. **2** (2012)
18. MontazeriShatoori, M., Davidson, L., Kaur, G., Lashkari, A.H.: Detection of doh tunnels using time-series classification of encrypted traffic. In: 2020 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 63–70. IEEE (2020)
19. Pal, M.: Random forest classifier for remote sensing classification. Int. J. Remote Sens. **26**(1), 217–222 (2005)
20. Parchekani, A., Naghadeh, S.N., Shah-Mansouri, V.: Classification of traffic using neural networks by rejecting: a novel approach in classifying vpn traffic. arXiv preprint arXiv:2001.03665 (2020)
21. Pfeiffer, M., Girlich, F., Rossberg, M., Schaefer, G.: Vector packet encapsulation: the case for a scalable ipsec encryption protocol. In: Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1–10 (2020)
22. Shi, Y., Biswas, S.: Website fingerprinting using traffic analysis of dynamic webpages. In: 2014 IEEE Global Communications Conference, pp. 557–563. IEEE (2014)
23. Su, M.Y.: Using clustering to improve the knn-based classifiers for online anomaly network traffic identification. J. Netw. Comput. Appl. **34**(2), 722–730 (2011)
24. Sun, G., Liang, L., Chen, T., Xiao, F., Lang, F.: Network traffic classification based on transfer learning. Comput. Electr. Eng. **69**, 920–927 (2018)
25. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 439–454. IEEE (2016)
26. Top, G.: Strategic technology trends for 2020 (2020)
27. Vu, L., Thuy, H.V., Nguyen, Q.U., Ngoc, T.N., Nguyen, D.N., Hoang, D.T., Dutkiewicz, E.: Time series analysis for encrypted traffic classification: a deep learning approach. In: 2018 18th International Symposium on Communications and Information Technologies (ISCIT), pp. 121–126. IEEE (2018)
28. Wang, P., Li, S., Ye, F., Wang, Z., Zhang, M.: Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan. In: ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE (2020)
29. Wang, T., Goldberg, I.: On realistically attacking tor with website fingerprinting. Proc. Priv. Enhancing Technol. **2016**(4), 21–36 (2016)
30. Xu, Y., Wang, T., Li, Q., Gong, Q., Chen, Y., Jiang, Y.: A multi-tab website fingerprinting attack. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 327–341 (2018)

31. Yan, F., Xu, M., Qiao, T., Wu, T., Yang, X., Zheng, N., Choo, K.K.R.: Identifying wechat red packets and fund transfers via analyzing encrypted network traffic. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 1426–1432. IEEE (2018)
32. Zhou, K., Wang, W., Wu, C., Hu, T.: Practical evaluation of encrypted traffic classification based on a combined method of entropy estimation and neural networks. ETRI J. **42**(3), 311–323 (2020)