A Distance-based Method for Building an Encrypted Malware Traffic Identification Framework

JIAYONG LIU¹, ZHIYI TIAN¹, RONGFENG ZHENG², LIANG LIU¹
¹College of Cybersecurity Sichuan University, Chengdu, Sichuan, P.R.China
²College of Electronics and Information Engineering Sichuan University, Chengdu, Sichuan, P.R.China
Corresponding author: Zhiyi Tian (e-mail: ZhiyiTian@foxmail.com).

This work was supported in part by National Natural Science Foundation of China under Grant 61872255.

ABSTRACT The popularity of encryption method brings a great challenge to malware traffic identification. Traditional classes defined by expert experience are usually classified based on the host behaviours of malware, such as banking malware or ransomware, which are often irrelevant to its communication traffic behaviours. It leads to the fact that the boundaries of traffic feature dataset of different malware classes are fuzzy, and make these traditional classes unhelpful for classification based on traffic features. Meanwhile, traditional machine learning-based encrypted malware traffic identification methods, such as using the multi-classification supervised learning model, are inefficient both in model training and detection, and its detection accuracy cannot meet the demand. In this paper, we propose a distance-based method which utilizes unsupervised learning algorithm Gaussian Mixture Model (GMM) and Ordering Points To Identify the Clustering Structure (OPTICS) to calculate the Distance between malwares, and make use of the Distance to define the new malware class called FClass. Then, a set of models are trained by XGBoost algorithm to build an identification framework based on the FClass. The performance of the proposed method has been evaluated by comparing it with the other four methods. The results show that the proposed distance-based method is more efficient and accurate.

INDEX TERMS Malware traffic identification, Encryption traffic, Unsupervised learning, Supervised learning

I. INTRODUCTION

The identification of encrypted malware traffic has been a research hotspot since the 1990s. In its 2018 cybersecurity report, Cisco noted that its analysis of 400,000 malwares found that as of October 2017, about 70 percent of them communicated by using encryption methods [1]. Common encryption methods, such as SSL/TLS protocol, mainly encrypt the payload of network traffic. Thus traditional detection methods like Deep Packet Inspection and machine learning method based on plaintext features [2]–[4] are no longer applicable. At present, the main method to detect encrypted malware traffic is supervised learning method which is based on those features that are not affected by encryption, for example, statistical features extracted from packets [5], features from logs like DNS logs [6], and features extracted from the encrypted protocols [7].

The above researches have good performances in encrypted malware traffic detection. However, there is a growing need to identify the specific type of malware traffic, in the face of increasingly variable behaviours of malware traffic. If the malicious traffic generated by which type of malware could be identified, the security system can take targeted measures to reduce the loss more quickly. The identification of malware traffic requires not only discovering malware traffic but also identifying the traffic generated by which type of malware. It turns a binary problem into a multi-classification problem.

With its rapid development, malware evolved into different classes, such as banking malware [8], phishing malware [9], DDoS malware [10]. However, most of these classes are classified according to their host malicious behaviours, such as encrypting or stealing specific files of victims, and such classification methods are often irrelevant to their traffic features. The traffic features between different classes of malware can be similar or completely different. It poses a significant challenge to the problem, that the boundaries of
traffic feature dataset among the existing classes are relatively fuzzy. This challenge is exacerbated by the fact that more and more malwares encrypt their communication traffic.

Meanwhile, it is not hard to find that the performance of the multi-classification model is worse than that of the binary classification model, under the same condition [11]. The class imbalance among malware traffic also brings enormous difficulties to the training of the multi-classification model. Compared with binary classification models, multi-classification models are slower in training, less accurate in the identification classification, and inflexible in using feature sets and datasets.

In this paper, we propose a distance-based method for identifying the encrypted malware traffic. It combines supervised learning with unsupervised learning by utilizing unsupervised learning algorithms to help to calculate the Distance designed by this paper. This Distance can be used to measure similarities between malwares and help to build a flexible supervised learning identification framework. To improve the accuracy and efficiency of identification, a set of binary models are trained to build the framework. After that, the framework will identify each class of malware in turn, then identifying the specific malware in each class.

Specifically, our method consists of the following 3 main steps: (1) Traffic clustering. Because of the class imbalance problem, we use GMM [12] algorithm to extract the Center Samples of every kind of malware, and then employ OPTICS [13] algorithm to cluster Center Samples. (2) Building framework. We use the Distance between malwares to define the malware class, \textit{FClass}. And the identification framework is built based on the \textit{FClass}. (3) Training models. A set of binary supervised learning models are trained to constitute the identification framework, and the framework is used to identify the encrypted malware traffic. The main contributions of this paper are listed as follows:

1) We propose a distance-based method which could establish a framework for identifying multiple kinds of encrypted malware traffic. The proposed method combines supervised learning with unsupervised learning, can build a more efficiency and accuracy framework according to the dataset.

2) We designed a Distance to measure the similarity between malware traffic datasets on complex datasets, and it could be used to define the new malware class \textit{FClass}. The \textit{FClass} is constructed based on the extracted features, which reflect the correlation and difference between malwares.

3) Complete experiments are implemented based on these datasets, namely CICIDS2017 [14], malware-traffic-analysis [15], stratosphere IPS [16]. All these datasets are frequently used in research to evaluate the performance of traffic identification. We also use them to evaluate the performance of the proposed method by comparing to two other common methods, multi-classification model, binary models with 1-layer structure, and two advanced method CluClas [17], MalClassifier [18].

The remainder of this paper is as follow: In section 2, we discuss related work. Section 3 discusses the problems that this paper attempts to solve. In section 4, we introduce the methods proposed in this paper. In section 5 we present the experimental results and comparative analysis. Finally, we summarized the paper, and the future work is provided.

II. RELATED WORKS

A. MALWARE ANALYSIS-BASED METHOD

The traditional malware classification is mostly based on the host behaviour of malware [19]–[21], for example, API-call sequences or created processes. On the basis of these features, the classification methods are constantly updated.

S. Huda et al. [22] proposed a semi-supervised approach to detect the unknown attack on cyber-physical systems (CPS) by using features from host behaviour. In this method, the data without labels and with labels are clustered by Global K-means with cosine similarity at the same time, and the distance between the labeled data and the clustering center is calculated to update the model and detect unknown attacks. This method provides a good way to combine supervised learning with unsupervised learning.

J. Kim et al. [23] proposed a static analysis method called tDCGAN to solve the problem of zero-day malware detection. This method generates similar fake malware raw code data by comparing real malware data to achieve the purpose of detecting zero-day malware. In the malware identification based on supervised learning, it is often encounter that the trained model cannot well complete the identification task due to the lack of samples. tDCGAN can solve this problem to a certain extent.

However, the models trained by such features can only detect malware after capturing the malware or it runs on the victim host, which is too late that the malware has already caused damage. Moreover, the cost of acquiring such data is much higher than that of network traffic data.

Traffic-based malware detection does not encounter these problems because it is able to detect malware at its entire life cycle and more sensitive during the infection of malware.

B. TRAFFIC-BASED METHOD

There has been many researches on the detection of malware based on traffic features. There are two main research directions, one is how to improve the classification ability of the model; the other is using a set of models to form a framework to solve a complex classification problem. For the first direction of research, the most important thing is to find highly distinct features which can be used to train better models. Blake et al. [11] took the unencrypted handshake content of TLS protocol, byte distribution and sequence of packet lengths as features. The results show that these features have good discrimination ability. The features extracted from transaction, session, flow and conversation windows have been proved experimentally to apply to the detection of unknown traffic [24]. In our research, we select some features from these studies for evaluating our method.

Many studies have shown that more features are not better, and it is necessary to choose suitable features for training.
models [24], [25]. Many studies aim to design feature selection methods to improve detection efficiency [26], [27]. M. Shafig et al. [28] used the feature selection method of weighted mutual information metric and area under the ROC curve to improve the performance of supervised learning on class imbalance dataset. Feature selection can also improve the performance of classifiers in large data volumes [29].

J. Stiborek et al. [30] proposed a vocabulary-based method to remove the randomization employed by malware authors and projects samples into a low-dimensional feature space. This method defines a Distance to measure the similarity between features and clustering the original data. The constructed feature set performs well in the malware classification.

J. Wang et al. [31] proposed a clustering algorithm SE to determine the attacks phases of malicious traffic. The algorithm uses seed to control the similarity between clusters, and the optimal clustering effect can be achieved by adjusting seed. However, the clustering algorithm alone cannot determine the identity of the attacker, so the attacks can only be divided into different phases. Inspired by this method, similar parameters are designed in our paper to make better use of clustering results and construct an optimal identification framework.

When the classification task is complex, it is difficult to get a good classification result by using only one model or one kind of machine learning algorithm. Many studies are using a framework that combines multiple methods to identify traffic. There are some studies that only use supervised learning to build traffic identification framework. W. Shbair et al. [32] proposed a hierarchical structure that only uses the Random Forest algorithm to identify HTTPS services. This method first determines who the service provider is and then what service the provider provides. It can identify the specific service used in HTTPS. For example, it can tell whether the specific service being used under the google.com domain is Google Maps or Google Drive with high accuracy.

Compared with the unsupervised model, the supervised model has better performance in the classification of network traffic. However, it also has some obvious defects, such as the vulnerability to abnormal samples and the high acquisition cost of labeled samples. The combination of supervised learning and unsupervised learning can effectively solve the problem of using one method alone.

Yi Li et al. [33] proposed a framework combining supervised and unsupervised learning to detect domain generation algorithm(DGA) domains. This method first uses Decision Tree-J48 to construct a classifier to discover DGA domain and then uses DBSCAN to cluster DGA domain discovered. R. Ashfaq et al. [34] proposed a semi-supervised learning approach. They first train a Neural Network with random weights classifier by using L hidden nodes. And then the fuzziness of unlabeled samples is obtained to further updated the classifier.

A. Fahad et al. [17] proposed a method called CluClas by using the k-means algorithm to remove noise and select the centroid of each cluster as representative training instances. And then using representative training instances to build Hidden Markov Model(HMM) to identify network traffic. This method is effective in dealing with background noise. The idea is to select the most suitable samples through unsupervised learning to train supervised model. This method is also a very useful attempt to combine supervised learning with unsupervised learning.

B. Alahmadi et al. [18] proposed a malware classification system called MalClassifier. This system first encode the PCAP files to a textual sequence, then measure the sequence by a fuzzy value similarity which combines a variety of similarity measurement methods. The most distinctive sequence will be selected as profiles for a malware family. Finally, the flow value similarity of these profiles from different malware family are used to train a supervised machine learning model. The core idea of this method is similar to our method, both of which hope to improve the classification accuracy by measuring the similarity of malware traffic samples. The difference is that MalClassifier uses similarity to train the model, while our method uses similarity to build the identification framework.

Combining supervised learning with unsupervised learning is a very effective way to improve the performance of identification, especially the way using unsupervised learning to improve the performance of supervised learning. It is not difficult to find from previous studies that this combination method can effectively play the advantages of the two kinds of algorithms and achieve the better identification effect. Therefore, our paper also adopts such a train of thought.

III. PROBLEM STATEMENT

A. MALWARE TRAFFIC CLASSES BASED ON EXPERT EXPERIENCE

The most natural idea of building an identification framework is to classify the malware traffic into several large classes based on expert experience, then identify each class separately, and subdivide malwares in the large class. The quality of the expert experience directly determines the performance of the framework. However, there are many challenges in the real-world environment.

There is no uniform standard to determine the class of malware [35]. A malware even has a different names in different detection engines [36]. Thus, it is difficult to judge a malware belongs to which malware class previously.

According to their communication behaviours, malwares could be roughly divided into two types by expert experience. (1) Infecting type. This kind of type belongs to the malware which mainly carry out infection activities. After the completion of the infection, it will perform further malicious activities. There are two popular ways to infection. The first one is based on the Exploit kit (EK) on a web page, such as RIG. The other is to use the phishing email to infect, such as Emotet [37]. (2) Connecting Command and Control(C2) server type. After the intrusion, the malware will communicate with C2 to obtain new instructions or steal...
sensitive information. Traffic of this type is generated mainly by malware hidden in normal traffic.

However, in the real-world environment, the boundaries between malware behaviours are less clear. It is possible for malware to generate both types of traffic. For example, Zeus malware can steal data and inject data into a websites HTML at the same time [8]. Meanwhile, infecting traffic from the same malware may also have both EK-based behaviours and email-based behaviours. This makes it difficult to categorize malwares into fixed classes, especially when malware encrypted its traffic.

Furthermore, even malwares can be classified into fixed classes by expert experience. The mismatch between the basis of classification based on expert experience and the features used for identification model could also lead to a great challenge. For example, the expert experience will associate malware with its signature, or some similar host behaviours, such as what function API is called [38]. But these malware classes may have little correlation with their traffic features.

Figure 1 shows a comparison of the feature values mean and variance of four kinds of malwares. Features in the figure are used in this paper, which is numerical statistical features, and will be explained in detail later. These two indexes can demonstrate the relationship between feature sets in some degree. Expert experience believes that there is a connection and similarity between Emotet and Hancitor and Trickbot [38], [39]. They propagate primarily through phishing mail, and there are similarities in their behaviours. Hence, based on expert experience, they can be grouped together. Moreover, we set traffic of RIG as a contrasting dataset. However, after comparison, it is found that some of these features have similarities, while others show completely different behaviours. According to experts experience, malwares that should behave similarly do not show obvious similarity in the mean and variance of their features. We can observe from figure 1 that Hancitor and RIG have more lines in the same position, and the blank position is closer. Hence, Malware Hancitor behaves more like a RIG than two other kinds of malware that experts have linked to it. Our experiments later in this paper also proved this conclusion. It may because the features we use do not reflect the similarities that expert experience believes. However, it is difficult to design the features that exactly match the expert experience. Therefore, we need to classify malware according to the extracted features, rather than design features based on expert experience. It needs to measure the similarity between malware traffic features in the complex dataset.

Our research is dedicated to designing a Distance to measure the similarity between malwares and define the $F_{Class}$ containing multiple kinds of malware, in which the malwares have similar communication behaviours. According to the $F_{Class}$, we could built an efficient identification framework and use more appropriate features to identify them.

### B. CLASS IMBALANCE

Another problem of using a supervised learning method to identify malware traffic is the class imbalance. The class imbalance has two meanings, one is data imbalance, and the other is feature imbalance. Malware traffic in the daily network communication accounted for a tiny proportion. Furthermore, the dataset size of traffic from different kinds of malwares also varies greatly. It leads to a problem that if the original dataset is directly used for training, the model would be biased towards those classes with large data volume, and cannot identify the classes with small data volume.

It is found in figure 1 that some features are relatively similar, whereas some features are entirely dissimilar. It indicates that different malware traffic cannot be identified using the same feature set. If the similar features can be used to detect the malwares of similar behaviour, and then use the dissimilar features to subdivide these malwares further, the identification accuracy can be effectively improved.

In our research, we use a set of binary models rather than one multi-classification model. When training each model, the proportion of positive and negative samples will be adjusted according to the number of each kind of malware samples, which is more flexible than the multi-classification model. Meanwhile, different models could be trained by different features sets. Then these models are used to comprehensively identify encrypted malware traffic to achieve unbiased traffic identification effect.

### IV. DESIGN

#### A. OVERVIEW OF PROPOSED METHOD

In this paper, we propose a distance-based method to solve the multi-classification problem of encrypted malware traffic identification. The proposed method can measure the similarity between malwares and form an adaptive identification framework, according to the distribution of traffic feature
dataset. The process of this method is demonstrated in the following algorithm 1.

Algorithm 1 Framework Building

\begin{algorithm}
\caption{IdentificationFramework}
\begin{algorithmic}[1]
\Require \text{Array}[n], \text{Name}[n]
\Ensure \text{IdentificationFramework}
\State \text{CClass}[x] \leftarrow 0
\State \text{Identification Framework} \leftarrow 0
\State \text{Cluster}[h] \leftarrow 0
\State \text{CenSamples}[n] \leftarrow 0
\For{$i = n; i > 0; i --$}
\State \text{CenSamples}[i] \leftarrow \text{GMM}(\text{Array}[i], K = 10)
\EndFor
\State \text{Cluster}[h] \leftarrow \text{OPTICS}(\text{CenSamples}[n])
\State \text{CClass}[x] \leftarrow \text{CClassRULE}(\text{Cluster}[h])
\State \text{FrameworkStructure} \leftarrow \text{FClassRULE}(\text{CClass}[x], \text{Cluster}[h], \text{Name}[n])
\State \text{IdentificationFramework} \leftarrow \text{TrainClassifier}(\text{XGBoost, FrameworkStructure})
\State \Return \text{IdentificationFramework}
\end{algorithmic}
\end{algorithm}

The input of this algorithm is the traffic feature set with the malware name label. The method consists of three major parts. (1) Each kind of malware data is input separately into the GMM algorithm for clustering, and the GMM algorithm is used to calculate $K$ Center Samples for each kind of malware. And then OPTICS algorithm is used to cluster all Center Samples($\text{CenSamples}[n]$), the clustering result is $\text{Clusters}[h]$. (2) In order to build $\text{FClass}$, we first propose a rule $\text{CClassRULE}$ to give each kind of malware a label named $\text{CClass}$. And then, we calculate the Distance among $\text{CClass}$, and use the rule $\text{CClassRULE}$ to define the $\text{FClass}$. $\text{FClassRULE}$ needs to input $\text{Clusters}[h], \text{CClass}[x]$ and the malware name label $\text{Name}[n]$. The identification framework structure is constructed based on the results of $\text{FClassRULE}$. (3) A set of binary supervised learning models are trained to form the final framework. The rest of this section discusses the algorithm in details. The process of proposed method is shown in figure 2, and we will discuss the process in detail below.

B. FEATURE ENGINEERING

1) Feature Design

There are many ways for malware to communicate encrypted, and in this paper, we took the most common TLS/SSL-based encrypted traffic as an example. The TCP flow over port 443 is the basic unit in our research, we consider traffic over port 443 as TLS/SSL protocol traffic. There are two kinds of traffic features that can be used to identify encrypted traffic. One of them is extracted from the unencrypted contents of traffic, like features from handshake of TLS/SSL protocol [11]. The other is to ignore the communication content and identify the traffic according to its statistical numerical features [24], [40].

The problem with the first kind of features is that we cannot always capture the complete TLS/SSL flow, in the real network traffic. Most of the time, we can only be sure that the captured packets are over TCP 443 port. In the face of such a network environment, a large amount of malware traffic is ignored because it does not meet the requirements of feature extraction. Therefore, we choose the second kind of features for training and traffic identification. These features are related to the state of the network between communication nodes and the running of the software. They are not affected by the encryption protocol, are not easily bypassed, and are somewhat stable even when the malware is updated.

There are 83 numerical features used in this paper, which can be divided into four categories. (1) The first category is the features of TCP and IP header, such as TCP internal ports and the sum of bit in the packet header; (2) time-based features, including the average arrival time of the packet; (3) length related features, like the length of the payload; (4) packet variation features, such as the number of TCP window change times, payload length change times. The extraction of these features does not depend on the specific content of the communication. All the captured data, no matter what encryption method it is, can successfully extract the above 83 features. And the full list is given in [41].

2) Feature Selection

However, not all classifiers require these many features. We only entirely used these features in clustering. For supervised learning model training, we selected different subsets from 83 features for different models.

Because this paper uses a tree-based algorithm, we first tried Filter features selection approach by calculating the information gain of features. However, in the experiment, we found that the model trained by such selected features was not accurate in traffic identification. For example, in the feature selection of $\text{FClass}$ model, the top 30 features of information gain are retained for training, and the accuracy is only 83.56%, but 87.28% if all features are used for training.

The purpose of feature selection in this paper is to select the feature set with high accuracy and make the number of features in the feature set as small as possible to improve the efficiency, so this paper has adopted a Sequential Forward Selection (SFS) [42]. SFS can increase the number of features in the feature set gradually, which perfectly fits our feature selection purpose.

However, SFS is a greedy algorithm, easy to fall into the local optimum, and its running time is long. Hence, sequential forward selection was improved to increase the selection efficiency. We randomly select 5 features from the feature set for training and evaluation in each round. The evaluation criterion is the accuracy of cross-validation. The number of rounds and sub feature sets are pre-set value. After several rounds, the several top-performing feature sets are combined to form the final feature set which are used to train the model. In this way, we can quickly select suitable features for training.

Because the proposed method will use a set of models, based on the analysis in section 3, to improve the identi-
fication efficiency and accuracy, each model uses different features in training process. It ensures that each model has the best identification capability. Table 1 shows the top 10 features used to supervised learning training. We can see the 10 most frequently used features in this paper, and these features also have the excellent ability to distinguish malware traffic.

### C. TRAFFIC CLUSTERING

Because the amount of encrypted malware traffic data varies greatly, if the original data is clustered directly, the clustering result will be biased to the malware with large data amount. The traffic behaviour of malware with small dataset may be ignored. In the collected dataset, most of the malwares generate a variety of communication traffic, such as confirm heartbeat packets with C2, or communication with C2 to receive commands, the transmission of sensitive information to C2, and so on. These behaviours are all equally crucial for traffic identification, but the amount of data from these behaviours varies widely. For example, the amount of data involving the transmission of sensitive information is much larger than the amount of data involving accepting command. Therefore, we design a method to avoid the result deviation caused by the data imbalance.

1) Center Samples calculation based on GMM algorithm

Each malware dataset will be separately input into the GMM algorithm, and calculate \( K \) Center Samples of each malware, as shown in figure 2. This step normalizes the amount of data in the malware traffic dataset so that each kind of malware has the same amount of data.

The GMM algorithm assumes that the dataset is a linear combination of several Gaussian Distributions. Its function is to split the sample set into several Gaussian Distributions and calculate the parameters of each one.

The Gaussian Distribution:

\[
N(x | \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)\right\}
\]  

(1)

The Gaussian Mixture Distribution:

\[
p(x) = \sum_{k=1}^{K} \pi_k N(x | \mu_k, \Sigma_k)
\]  

(2)

Where \( \mu_k \) is the center of the \( k^{th} \) cluster, \( \pi_k \) is the weight of the \( k^{th} \) cluster, \( K \) is the number of clusters. To ensure that the number of Center Samples of each malware is the same, we set \( K \) to the same value. The value of \( K \) should be slightly more than the number of traffic types of malware, to ensure that the traffic types with a small number are not ignored.

2) OPTICS clustering

After calculated \( K \) Center Samples of each malware, we cluster these Center Samples to help to expose the similarity between malwares and construct the FClass. As shown in figure 2, all Center Samples will be input into the OPTICS algorithm at the same time, and then the clusters of each Center Samples will be obtained. In figure 2, different color represent the different OPTICS cluster of Center Samples. Clustering algorithms usually require a pre-set number of clusters to perform operations. However, we cannot predict how many clusters in the dataset. So we selected an unsu-
s supervised learning algorithm that can automatically find how many clusters there are.

OPTICS algorithm is an unsupervised learning algorithm based on hierarchical density, which is not sensitive to parameters. It can deal with the problem of non-uniform density clusters in the dataset and can visualize the cluster structure of dataset. This feature can show the similarity of samples and clusters, providing a good reference for the building of identification framework. OPTICS algorithm has two parameters $eps$ and $MinPts$.

In the proposed method, we input the Center Samples of all malwares into the OPTICS algorithm with appropriate parameters to get the clusters. We implemented the algorithm by using python’s scikit-learn library [43]. In this way, the Center Samples has two labels: one for the malware names they belong to, and the other one for the cluster labels obtained from OPTICS algorithm.

### D. DISTANCE-BASED FRAMEWORK BUILDING

OPTICS algorithm divides all Center Samples into multiple clusters. Each such cluster may contain only one kind of malware behaviour. Hence, Center Samples of the same malware are subdivided into different clusters, making it impossible to build the identification framework directly by the OPTICS cluster results. Therefore, we proposed the following two rules to build the identification framework.

1) **CClassRULE**

   According to the OPTICS clusters of Center Samples from each malware, we designed CClassRULE to define a subclass called CClass within FClass. Center Samples of each malware will be input separately into the CClassRULE, as shown in figure 2. According to the OPTICS clustering results of Center Samples, CClassRULE will determine which kind of CClass each malware is. Furthermore, the relationship between malwares of the same CClass label needs to be determined based on its Distance and FClassRULE.

   The CClassRULE is described below:

   1) **CenterClass**: All $K$ Center Samples of this malware belong to the same OPTICS cluster.

   2) **SubCenterClass**: More than half but not all Center Samples of this malware belong to the same OPTICS cluster.

   3) **EdgeClass**: Its Center Samples belong to more than two OPTICS clusters, but none of them is more than half.

   4) **NoiseClass**: More than half of the Center Samples of this malware are noise. The noise is the sample that OPTICS algorithm believes does not belong to any cluster.

   The classes above are a relative concept. They relate to the OPTICS results of Center Samples. Perhaps a CenterClass malware will be defined as EdgeClass in another dataset. We use CClassRULE to give each malware a CClass label, and at this point, Center Samples of malware have three labels, OPTICS clustering results, malware name label Name and malware CClass.

   And then we utilize the labels above to measure the similarity between malwares through the Distance designed by this paper, to define the FClass. Based on the FClass, we could build an efficient and accurate identification framework. We proposed a rule called FClassRULE to define the FClass.

2) **Malware Distance and FClassRULE**

   We define the concept of Distance between malwares to measure their similarity based on the OPTICS clustering results of Center Samples.

   **Distance**: We first set the initial Distance to be the number of the Center Samples, $K$. Between two malwares, if there are $h$ Center Samples belong to the same OPTICS cluster except the Center Samples belonging to the cluster containing CenterClass, and the NoiseClass Center Samples, then the Distance is $K - h$.

   The reason why the Center Samples belonging to the cluster containing CenterClass are not counted is that the CenterClass generally takes up a large proportion and is the main body of the entire sample space. If this part of behaviour is removed, the rest of the sample is more representative, and we can discover the real similarity between malware through it. The noise Center Sample does not belong to any cluster, so it is not counted either.

   Based on the Distance between malwares, we define the Distance between CClass as follows: The Distance of CClass is the Distance of its contained malware, when the CClass contains more than one malwares, the Distance between these CClasses is the minimum Distance of their containing malwares. When calculating the Distance of NoiseClass between other CClass, if the NoiseClass malware has only the Center Samples belonging to the cluster of CenterClass in addition to the noise point, the Center Samples of CenterClass should be counted.

   According to the Distance between malwares, we design FClassRULE to define the FClass. The FClass only relates to traffic features, and it contains CClass. The FClassRULE is described below:

   1) The CenterClass and SubCenterClass malwares belong to the cluster of its most Center Samples belonging to. Malwares belonging to the same cluster are then defined as a FClass.

   2) Center Samples of one EdgeClass malware are divided into different clusters by OPTICS algorithm, so it’s hard to tell which cluster the EdgeClass malwares belong to. If they are all classified into independent FClasses, the identification framework will be too complicated to identify efficiently. Therefore, we design the following method to combine the EdgeClass. We first calculate the Distance between EdgeClass, and combine the EdgeClass to form a larger EdgeClass including malwares as long as whose Distance with one of the malware in the EdgeClass is less than $D$. Where $D$ is a pre-set value, we can control the similarity of malwares within the CClass by adjusting the value of $D$. Then, EdgeClass malware belongs to the FClass of the closest SubCenterClass. If there is no SubCenterClass meets the condition, each EdgeClass malware belongs to...
its own independent FClass. If more than two FClass meet the condition, then the malware should belong to the smaller FClass, because this can make the identification framework more balanced.

3) NoiseClass malware belongs to the FClass of the closest malware. If the condition is not met, they belong to a noise FClass. If more than two FClass meet the condition, then the malware should belong to the smaller FClass.

As shown in figure 2, we use the above rules, determine which malwares should belong to the same FClass, and the internal structure of each FClass. First, we determine the FClass to which SubCenterClass and CenterClass malwares belong. Then we combine the EdgeClass, and determine whether they belong to the FClass of SubCenterClass or a new FClass. Finally, we determine the NoiseClass malwares belong to the existing FClass or a new FClass. Through the above steps, the constructed FClass contains several CClass and each CClass contains several kinds of malware.

3) Building Framework
After FClassRULE executed, each malware belongs to a certain FClass, then the identification framework is built based on the FClass. First of all, a set of FClass identification models are trained; each model identifies one FClass. These models form the first layer of the identification framework. The suspicious samples need to be examined by all models in this layer. When only one model determines that the sample belongs to its identifying FClass, the sample will be sent into the next layer for further identification, and the rest of the results would be considered that the sample is an unknown class.

In the second layer, we train coarse classification models for identifying malware in a certain FClass; each coarse classification model identifies malware belonging to the same CClass in the FClass. Similar to the first layer, samples are examined by all the coarse classification models in the same FClass but not the model of other FClass. When only one coarse model determines that the sample belongs to its identifying CClass, the malware is then sent to the third layer for malware judgement. Under the coarse classification model, several leaf models are trained, and each leaf model identifies one malware in the CClass, and also the samples will be examined by all leaf models in the same CClass and when only one leaf model determines that the sample belongs to its identifying malware, this sample is considered to belong to the malware.

E. TRAINING MODELS
After building the identification framework, a set of supervised learning models need to be trained to form the final framework. In this paper, we chose the binary model to identify each traffic type. The supervised learning algorithm adopted in this paper is gbtree-based XGBoost [44] algorithm whose objective function is binary:logistic(binary logistic regression). XGBoost was chosen because it is the most popular classification algorithm in recent years and performs well in a variety of tasks.

To test the performance of the binary classification model in identifying encrypted malware traffic, we designed an experiment. Taking the XGBoost algorithm as an example; it can implement both dichotomies and multi-classifications. And the experiment was carried on the same dataset and the same features mentioned above. The dataset contains samples of 12 kinds of malware from 2 FClass defined by FClassRULE. Each model was tested in a few rounds, each with a different test set. The accuracy is the average result of 10-fold cross-validation. The amount of dataset used to train is the same.

The experiment running: Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz, 8GB of RAM. This experiment evaluated
the accuracy and training time of the model. The results are shown in figure 3. Under the same conditions, the binary classification model has the shortest training time and the highest average accuracy. At the same time, the experiment found that the number of features in 20 or so is the most appropriate. Although the binary classification model cannot identify a variety of malware traffic, through the cooperation of several layers and models, it can implement multiple classification tasks.

In order to train binary models that can accurately identify malware traffic, this paper constructs an appropriate training set based on the identification framework. The positive sample set in the training set is the traffic of FClass, CClass or one kind of malware that needs to be identified. The negative sample set is other malware samples in the same layer or the same FClass. To improve the robustness of the model, a part of benign traffic samples in the background environment are added to the negative sample set in the first layer. When there is a large gap between positive and negative datasets, according to GMM clustering results, the data was selected through stratified sampling by GMM clusters. At the same time, in order to improve the identification accuracy, the feature set of each model is selected separately by using the feature selection method mentioned above. Each model uses different features. The whole framework selected 35 features from the 83 alternative features. Therefore, after the framework is built, only 35 features need to be extracted from each flow, when identifying the traffic.

V. EVALUATION
A. EXPERIMENTAL SETUP

In our experiments, we combined three datasets as our experimental dataset. The CICIDS2017 [14] dataset contains benign and the most common attacks, such as SQL injecting, collected by the University of new Brunswick. In our experiment, we use only the benign flow from this dataset to improve the robustness of our framework. Another part of the benign data came from a controlled environment, the authors’ laboratory containing 15 PCs with a bandwidth of 100Mb/s. The malware-traffic-analysis [15] dataset mainly includes the malware infecting type traffic and has been collected since 2013. The Stratosphere IPS [16] dataset mainly includes the traffic flow of connecting C2 type. All of the above datasets provide raw communication data stored in PCAP format. We first extracted the TCP flows that communicate on port 443 from these PCAP, and extracted the features from it. The feature vector extracted from one TCP flow is taken as one sample.

Theoretically, our method can handle both encrypted traffic and non-encrypted traffic, but in order to verify its ability to identify encrypt traffic, this paper only uses encrypted traffic for testing.

The total data is shown in table 2. But not all the data is used to train models. The size of the training set should be determined according to the data quantity of positive and negative samples. Because multiple models are trained to identify different traffic, each model uses a different training set.

Our experiments are performed on the following configuration: Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz, 8GB of RAM. All models including comparison methods are trained by XGBoost algorithm.

To test the performance of the proposed method, our experiment set up two common multi-classification methods, and two advanced methods (MalClassifier, CluClas) for comparison. The first of the two common methods is to use the XGBoost algorithm with objective function multi:softmax to train a multi-classification model. The other one is to train a group of binary models, each models identify one kind of malware, and these binary models are arrange into one layer, the samples are identified by all the models. P.M. Comar et al [45] have used similar identification structures. The dataset adopted by this two comparison methods was adjusted according to data size differences between different malware. Similarly, the features used in the comparison methods are not completely the same, but they are selected from 83 alternative features by the same feature selection method mentioned above.

When implementing the MalClassifier method, we used Zeek [46] to generate the logs. We selected the similar parameter as in the paper [18], $k = 10$, $n = 5$, same weight $w$, and the supervised learning algorithm was Random Forest. We didn’t use PCA to reduce feature dimension.

To implement the CluClas model, 16 features were selected by the feature selection method mentioned in the paper [17]. We have set the value of $k$ for K-means clustering algorithm to 100. 24 HMM models were trained, and the log-likelihood of these models was used to make judgement.

When evaluating the performance of these methods, the most commonly used metrics are Recall rate, Precision rate and F-Measure($F_1$). They are defined as in the following equations:

\[
Recall = \frac{TP}{TP + FN} \tag{3}
\]

\[
Precision = \frac{TP}{TP + FP} \tag{4}
\]

\[
F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}
\]

The identification of each class is treated as a dichotomy, each malware in turn as the positive samples(Mal) to be identified, while other malware as the negative samples(Other). Where true positive(TP) is the number of Mal identified as Mal, true negative(TN) is the number of Others identified as
Others, false positive (FP) is the number of Others identified as Mal, false negative (FN) is the number of Mal identified as Others.

B. EXPERIMENTAL RESULTS AND ANALYSIS

1) Framework building results

We first carried out an experiment to build the identification framework based on the 24 kinds of encrypted malware traffic collected. The samples with all 83 features extracted are input into algorithm 1. GMM algorithm extracts 10 center samples from each malware sample set. The parameter $\epsilon$ of the OPTICS algorithm is set to the default, and $\text{MinPts}$ is set to 5.

OPTICS clustering results are shown in figure 4. In the below subgraph of figure 4, the 24 malwares are in the following order: Angler, Styx, Terror, Nuclear, Magnitude, RIG, Neutrino, fiesta, Dyre, Cerber, Hancitor, DHL/USPS email, Boletor, Dridex, Papras, Upatre, CCleaner, Miuref, TrickBot, hbot, Emotet, Zeus, Artemis, Bunitu.

The OPTICS clustering results have proved our previous analysis that there are similarities between malware communication traffic, which is shown in the red points in figure 4. The red points are spread across most of the malware traffic. Meanwhile, many malware have their own behaviours, which are presented as points of other colors in the figure. It shows that malware can exhibit multiple behaviours at the same time, and the most unique behaviour is the noise because they can’t be categorized into any cluster. One of the purposes of the proposed method is to dig out the unique traffic behaviour generated by malware to improve the identification accuracy. From the results of figure 4, the proposed method can indeed discover unique traffic behaviours. Then these results are input into FClassRULE.

A total of 14 kinds of malware based on the CClassRULE are defined as $\text{EdgeClass}$. To build the optimal system structure, we need to set an appropriate $D$ value. In our experiment, the value of $D \in [0, 10]$. We tested the situation when $D$ was 3, 5, 7. The experimental results are shown in table 3.

- For $D = 3$: We found that there are 5 $\text{EdgeClass}$ that can merged into 2 $\text{EdgeClass}$, the rest of them are independent $\text{EdgeClass}$. Hence, we need to train a lot of coarse models to classify the $\text{EdgeClass}$ malwares, which makes the second layer of our identification framework too complex to complete the identification task efficiently.

- For $D = 5$: Four $\text{EdgeClass}$ still cannot be merged with the other $\text{EdgeClass}$, and the remaining 10 $\text{EdgeClass}$ are aggregated into three large $\text{EdgeClass}$. This result is better than $D = 3$. However, it still does little help to
improve the efficiency of identification, and the data volume of 4 independent \textit{EdgeClass} is too different to train an appropriate model.

\textbf{D = 7:} All the \textit{EdgeClass} are successfully incorporated into two larger \textit{EdgeClass}. If using this result to build identification framework, the amount of coarse models required by the system could be significantly reduced, which will improve the identification efficiency of the framework.

At the same time, since the CClassRULE has divided malwares into several \textit{CClass}, the merged \textit{EdgeClass} will not affect the identification accuracy of other \textit{CClass}. Hence, we set the FClassRULE pre-set value \textit{D} to 7. The FClass-RULE results are shown in table 4.

\textbf{FClass 1} is mainly composed of malware represented by red points which mainly correspond to infecting type malware. It indicates that the infecting behaviour of the malware is relatively stable and most of the malware will generate it. \textbf{FClass 0} mainly consists of other malware represented by other colors. These points mainly correspond to the connecting C2 type traffic. Figure 4 shows that this traffic is more cataclysmic, this indicates that when malware begins to communicate with C2, its communication behaviour will be more obvious and it will be easier to identify which kind of malware.

Different from the classes defined by expert experience, these two \textit{FClasses} were constructed according to the similarity of their traffic features and each \textit{FClass} contains multiple types of malwares. For example, Emotet, Hancitor and TrickBot mentioned earlier were divided into three \textit{CClass} in the two \textit{FClass}. Malware in the \textit{FClass} are further divided into several classes, and such \textit{FClass} can effectively help us to build the identification framework. Finally, About 80\% of the data was used for model training, and feature selection was carried out for each model. Then these models were used to construct the final framework.

The identification framework is shown in figure 5. These models will give uniform results according to the rules described in Section IV-D3 when identification. In the first layer of the framework, there are 2 models that identify \textit{FClass 0} and \textit{FClass 1}, respectively. In the second layer of the framework, there are 5 kinds of \textit{CClass} that need to be identified, so it needs 5 coarse models. In the last layer, 24 models are trained to identify 24 types of malware. Hence, the framework needs 31 binary models to identify the 24 kinds of malwares.

\begin{table}[ht]
\centering
\caption{FClassRULE Results}
\begin{tabular}{|c|c|c|}
\hline
\textbf{FClass} & \textbf{CClass} & \textbf{Malware} \\
\hline
0 & SubCenterClass & Upater, Mircel \\
0 & EdgeClass & Artemis, Bunitu, Ccleaner, Emotet, hbot, Papras, TrickBot, Zeus \\
1 & CenterClass & Cerber, Styx, Magnitude \\
1 & EdgeClass & Terror, Neutrino, Nuclear, fiesta, Dyre, Dridex \\
1 & NoiseClass & Boleto, DHL/USPS email, Hancitor, Angler-EK, RIG-EK \\
\hline
\end{tabular}
\end{table}

\textbf{2) Performance of the distance-based framework}

In our experiment, we evaluated the performance of the framework as a whole, and the performance of the models in the framework was not tested separately. The performance of the framework is evaluated mainly from two aspects; one is the accuracy of identification; the other is the identification speed.

We designed two experiments on identification accuracy.
One was to test the framework’s ability to identify known traffic with the unmodeled data left over from the 24 kinds of malwares. Another experiment was to test the performance of the framework on unknown traffic. We constructed an unknown test set consisting of 3 kinds of malware traffic that the framework has not encountered before to test the performance of the framework.

The confusion matrix for these two experiments is shown in figure 6, the X-axis is the true labels of the test set, and the Y-axis predicted label. The column entries represent the percentage of samples identified as that specific malware. In order to better analyse the experimental results, we drew the box-plot in figure 7.

Figure 6 and figure 7 show that the framework based on the distance-based method is better than the other four comparison methods. The distance-based framework has a
lower false positives rate. Most false positives are regarding the known traffic as unknown traffic. It is more stable in the face of different kinds of malwares, with fewer outliers. The framework also performs well in the face of unknown traffic, reaching a recall rate of 83.59%.

By analysing the results of the 24-classification model, it can be found that the false positives mainly appear in malware within the same FClass. This should be due to the influence of similarities in the behaviour of malware within the FClass on the model. The 24-classification model treats all traffic input into the model as one of the known 24 kinds of malware traffic. As a result, the model cannot handle unknown traffic, and the experimental results also prove it. This also makes the multi-classification model unusable in the real world, where there is always unknown traffic.

The 1-layer identification framework composed of 24 binary classification models performed the worst in the experiment, and a large number of samples were classified into the Unknown Class although it had the best performance against unknown traffic, with a recall rate of 85.32%.

MalClassifier as a single-model method, its performance has some similarities with the 24-classification model. The three metrics of these two methods are the closest in our experiment. The false positives of MalClassifier mainly appear in malware within the same CClass. This is a better result than the 24-classification model, because the misclassified samples somewhat is similar to its real class. It is worth mentioning that for all classes of malware traffic, its identification precision is above 80%, although its recall rate is relatively low and unstable.

When dealing with unknown traffic, it cannot clearly judge whether it is unknown or not, and can only divide it into existing classes. The overall performance of this method is not as good as that in the paper [18]. This may be partly because our scenario has only encrypted traffic and no traffic from other protocols. At the same time, as the kinds of malware that needs to be identified increases, the boundaries between different classes become fuzzy, which may affect the performance of this method.

CluClas is not a method for malware traffic identification, it is mainly used for network application traffic identification. We choose this method for comparison mainly for three reasons. First, this method combines supervised learning with unsupervised learning, which is similar to the method proposed in our paper. Second, the classification model used in this method is not based on decision tree, but HMM. We want to compare this algorithm and the tree-based algorithm. Third, we try to find out whether normal traffic identification methods can be used in malware traffic identification.

Experimental results show that the normal traffic identification method CluClas can not be used in encrypted malware traffic identification. CluClas can only successfully identify a small part of malware traffic, but not the majority. We believe that the main reasons are as follows: (1) In the normal traffic identification, there is a big difference between different traffic, for example, p2p traffic is obviously different from HTTP traffic. In the identification of malware traffic, the difference between samples will be much smaller. (2) The class imbalance problem in malware traffic is much more severe than that in normal traffic. (3) In this experimental environment, the malware traffic is encrypted, resulting in the identification difficulty is further increased.

The numerical results comparisons are listed in table 5. The data in table 5 is the result of the arithmetic average of the metrics of each malware. We performed a t-test of the identification results between the Distance-based framework and other methods. The results are also listed in table 5. The t-test results demonstrate that our method is significantly better than the others on most indicators. In particular, our method is significantly better than the other four methods in recall rate. Compared with MalClassifier and 24-classification model, the precision of our method is not statistically significant, which indicates that the precision of our method is not high enough. The F-Measure of our method is only not significantly better than the 24-classification model. Although our method is not better than the other four methods in all the indicators, our method still has obvious advantages on the whole.

3) Identification efficiency comparison

We also evaluated the identification efficiency of the three methods under the same computer configuration. The experimental result is that for every 10,000 samples, the 24-classification model takes an average of 53 seconds, the 1-layer identification framework takes 18 seconds, and the distance-based method proposed in this paper takes 15 seconds. The MalClassifier model takes 32 seconds for every 10,000 samples. And CluClas takes 50 seconds for every 10,000 samples.

We believe that the identification time of the models based on XGBoost algorithm is closely related to the booster number of models. 24-classification model contains 29,663 boosters, compared to an average of about 200 boosters for a binary model. For a framework based on a binary classifica-
tion model, the number of models required for identification directly determines its efficiency. Compared with the 1-layer identification framework, the distance-based framework reduces the number of models required to identify traffic, so the efficiency is further improved. MalClassifier is a random forest model, its identification efficiency may be related to the number and complexity of the trees it contains. We set the number of trees to 15,000 when training the model. CluClas is the only non-tree-based one of the five methods. In the process of its operation, every sample has to pass the judgement of all models, so its efficiency is not high. This indicates that the selection of classification algorithm is as important as the design of identification structure.

VI. CONCLUSION

In order to achieve efficient and accurate identification of encrypted malware traffic, a distance-based method based on OPTICS algorithm, GMM algorithm and supervised learning method has been proposed. The experiment showed that the proposed method was able to identify multiple encrypted malware traffic with higher accuracy and less identification time. Moreover, the proposed method can deal with unknown traffic.

For encrypted malware traffic identification based on supervised learning, the biggest problem is that a large number of malware traffic samples need to be acquired in advance, and features need to be extracted in advance during identification. This greatly limits the practical application of this method. In the future, we will further study the malware traffic samples and its representative features to improve the efficiency and accuracy of identification further.

REFERENCES


VOLUME 4, 2016

JIAYONG LIU received his B.Eng. degree in 1982, M.Eng. degree in 1989, and Ph.D. degree in 2008 from Sichuan University, China. He is currently a professor in College of Cybersecurity, Sichuan University. His research interests include network information processing and information security, communications and network information system.

ZHIYI TIAN received the B.S. degree in information security from the Sichuan University, Chengdu, China, in 2017. He is currently pursuing the M.A degree at the College of Cybersecurity, Sichuan University. His current research interest include malware traffic analysis and machine learning.

LIANG LIU received the M.S. degree from Sichuan University, Chengdu, China, in 2009. He is currently an Assistant Professor with the College of Cybersecurity, Sichuan University. His current research interests include malicious detection, network security, system security, and artificial intelligence.

RONGFENG ZHENG received the master's degree from Sichuan University, China, in 2016. He is currently pursuing the Ph.D degree in the Electronic Information College of Sichuan University since 2016. His primary research interests include cyber security, threat intelligence, and Internet of Things (IoT) security.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2930717, IEEE Access