IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 February 2023, accepted 16 March 2023, date of publication 23 March 2023, date of current version 28 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3260977

RESEARCH ARTICLE

Detection of Android Malware Based on Deep Forest and Feature Enhancement

XUEQIN ZHANG[1,2], JIYUAN WANG[1], JINYU XU[3], AND CHUNHUA GU[1]
[1]College of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
[2]Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China
[3]School of Electrical and Information Engineering, Hubei University of Automotive Technology, Shiyan, Hubei 442002, China

Corresponding authors: Jiyuan Wang (wang13272462183@163.com) and Jinyu Xu (xujinyuhuat@qq.com)

This work was supported by the National Natural Science Foundation of China.

ABSTRACT Detecting Android malware in its spread or download stage is a challenging work, which can realize early detection of malware before it reaches user side. In this paper, we propose a two-stage detection framework based on feature enhancement and cascade deep forest. This method can detect the traffic generated in the encrypted transmission process of Android malware. The first stage realizes the binary classification of benign and malicious software. The second stage realizes the multi-classification of different categories of malware. To enhance data representation, convolutional neural networks is used to extract benign and malicious features in the first stage, and the principal component analysis method is used to extract the malicious features in the second stage. Theses extracted features are spliced with the payload part of the traffic to form fusion features for classification task. In order to adapt to different scale of samples, especially for the small-scale sample, cascaded deep forest method is proposed to construct the classification model. In this model, many layers that consist of base classifiers are cascaded and the number of layers can be automatically adjusted according to the scale of the samples. With different combinations of base classifiers in each layer, the optima detection accuracy is archived in the two stages. The experimental results on several datasets prove that the proposed method is effective for encrypted transmission detection of Android malware. It is also suitable for the detection of unknown attacks.

INDEX TERMS Android malware, anomaly detection, encrypted traffic, feature enhancement, deep forest.

I. INTRODUCTION

With the development of the Internet, large numbers of malicious applications have been generated, which bring a huge threat to users. Android system is currently the most widely used operating system and one of the most vulnerable platforms to malware attacks [1].

Android application is usually packaged as Android application package (APK) files. In order to avoid detection, malware makers can use TLS/SSL protocol to encrypt APK files when spreading them. Then these files perform decompilation when reaching the user side.

The current intrusion detection system can usually adopt two ways to analyze and detect Android malware. The first is host-based intrusion detection system (HIDS). HIDS implements detection by analyzing the static characteristics of APK files, such as permission, component and intent features, etc. [1] or the dynamic characteristics generated when the application is running, such as system call traces etc. [2]. Since HIDS residents on the host, it will bring extra consumption of user side, which is difficult to implement on mobile side. The second is network intrusion detection system (NIDS). NIDS implements detection by capturing the traffic when the infected side communicate with server side [3]. In this case, NIDS relies on whether the communication occurs. Furthermore, these two methods can only find attacks after the malware has reached or installed at the user side. At this time, the user side has already with high risk.

So, if intrusion detection system can detect malware in the stage of transmission or download, it is of great significance to realize the early detection of malware, and decrease the risk of user side. However, according to our literature review, the research about encrypted transmission detection of Android

The associate editor coordinating the review of this manuscript and approving it for publication was S. K. Hafizul Islam.

malware has not been reported yet. It is still a challenging work.

Therefore, focusing on detecting the encrypted traffic generated during the encrypted transmission of Android APK files, a two-stage detection framework, named FE-CaDF, is proposed in this paper. In this framework, the deep forest based model and feature enhancement technologies are used. The proposed method can analyze the security of Android APK files directly without decrypting and decompiling files, and does not need to run malware and occupy the host resources. This method can use traffic to achieve file-level detection. Also, it is suitable for small-scale and unbalanced samples and can detect unknown attacks.

The main contributions of this paper include:

(1) In order to realize early detection in malware spread phase, a static malware detection method based on traffic is proposed. The whole encrypted transmission traffic of an Android APK file is taken as analysis instance to get file-level detection result directly.

(2) To obtain the effective representation for the packaged and encrypted traffic, machine learning and deep learning technologies are comprehensively adopted to extract enhancement features from the traffic payload. For the binary classification task, convolutional neural network (CNN) is used to automatically extract high-dimensional nonlinear features. For the multi-classification task, the principal component analysis (PCA) method is used to analyze and extract the key features. These extracted enhancement features are combined with the original traffic payload to form fusion features.

(3) To achieve high accuracy multi-classification for malware, a two-stage framework is presented, which connects two classifiers in series. In this framework, the first stage realizes the binary classification of benign and malicious applications and the second stage realizes the multi classification of different categories of malware. Considering the case of small-scale samples (for example, the number of samples is less than 1000) and imbalanced data in the real network, an improved cascaded deep forest classifier is applied in the two stages. The classifier can adjust its number of layers automatically to fit the different scale of samples.

The rest of this paper is arranged as follows. The second section introduces the related work. The third section introduces the design and implementation of the proposed method. The fourth section mainly introduces the experiments and results. The fifth section summarizes the work of this paper and prospects the future work.

II. RELATED WORK

A. ANDROID MALWARE DETECTION

At present, there are three main methods to detect Android malware: file analysis and communication traffic analysis.

The traditional detection way of Android malware is analyzing APK files. Imtiaz et al. [4] proposed to extract the status code, authority, intention and other information from the decompiled APK file as detection features, and converted these character features into numerical vectors according to their statistical value. Deep neural network was used to build classifier. The accuracy of binary classification on CICInvesAndMal2019 dataset reached 93.4%, and that of four-classification reached 80.3%. Yadav et al. [5] proposed a deep learning based two-stage framework to detect Android malware. This method converted Android. DEX files into RGB images and used EfficefNetB0 convolution neural network to extract relevant features from the images. A stacking classifier which employed linear SVM and random forest methods as base-level classifiers and logistic regression method as the meta-level classifier is used for classification. The accuracy of four classification on MalNet and other datasets was 88.6%, and the accuracy of five classification was 92.9%. Zhu et al. [6] proposed a new malware detection framework named MSerNetDroid. In this framework, the application program was characterized by extracting permissions, API calls and hardware functions. A Multi-Head Squeeze-and-Excitation Residual block (MSer) was designed to learn the internal correlation between features, and a MSer based deep network MSerNet was used for classification. The experimental results on a self-built dataset showed that the accuracy of this method was 96.48%. Alam et al. [7] proposed a new method called DroidDomTree, which can mine the dominant tree of API calls, highlight the powerful path flow, identify the nested structure of API, and assign weights to each node in the dominant tree to achieve efficient function selection. This method is tested on eight machine learning based classifiers, and the highest accuracy of J48 classifier on binary classification is 99.7% in 2750 collected applications. Jeon et al. [8] implement detect during malware running and proposed to used Multiple Sequence Alignment (MSA) algorithm to construct transition probability matrix, and adopted the Profile Hidden Markov (PHM) model to detect malicious code. The accuracy of binary classification reached 96.3% on test dataset, which was composed of the Drebin dataset and 500 benign samples collected from Google Store. Abanmi et al. [9] proposed a hybrid malware detection framework named HyMalD, which combined static and dynamic analysis methods. In this framework, Bidirectional Long and Short-Term Memory Neural Network (BiLSTM) was used to analyze the opcode sequence, Spatial Pyramid Pool network (SPP-Net) was applied to analyze API calls. The experiments on KISA-data challenge 2019-Malware.04 dataset showed that the binary classification accuracy was 92.5% and the false negative rate was 7.67%.

Malware running at user side usually need to receive instructions from the attacker's server by means of network communication [10]. By analyzing the network traffic at the communication stage, malware can be found effectively. Wei et al. [11] proposed a traffic behavior modeling method based on one-dimensional convolutional neural network with Auto-Encoder (AE) and independent recurrent neural network. This method used the statistical features of network traffic as the detection features. And then it used

one-dimensional convolutional neural network to extract local features from flows, and used AE to extract the main flow features from the neural network. A recurrent neural network was used as classifier, the binary classification accuracy on CICAndMal2017 dataset was 98%. Liu [12] et al. proposed an Android malware detection method based on the statistical characteristics of Android application communication traffic. This method applied LSTM-based variational Auto-Encoder (LSTM-VAE) to extract the time sequence characteristics. Then Back Propagation (BP) neural network-based classifier was designed for initial classification task. The class vectors output from BP neural network was spliced with the original data, and fed to a cascaded forest-based classifier for final classification. Experiments on the CICAndMal2017 dataset showed that this method can achieve the highest 97.29% accuracy in 2-classification. Kamel and Khaled [13] proposed a feature enhancement model. Firstly, this method take conversation as analysis unit and removed the identification features in flow. Then an ensemble learning method was used to extract the effective features. Finally, extra trees, random forest, and decision trees are used to train classification models. Experiments on the CICAndMal2017 dataset showed that extra trees achieved the highest accuracy in binary classification and malware 4-classification, which is 87.75% and 79.97% respectively. Xu et al. [14] proposed a detection framework named Falcon. This method converted the network traffic generated during the running of the application program into 2D images, and extracted 32-dimension features from each image with a pre-trained CNN network. These feature vectors belonging to the same application were sent into a BiLSTM based detector in chronological order. Experiments on the CICAndMal2017 dataset showed that the binary classification accuracy of this method at file-level is 97.16%, and that of multi-classification is 84.70%.

B. DEEP LEARNING-BASED TRAFFIC DETECTION

In addition to the above research, there are many traffic detection methods based on deep learning, and achieve good results.

Li et al. [15] proposed to use information gain and PCA to extract the features from network traffic, and then apply DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to cluster data, and adopted Wasserstein GAN Divergence (WGAN-DIV) to generate data for each class. Then an XGBoost-based model is trained by enhancement data. The average accuracy of binary classification on NSL-KDD, UNSW-NB15 and other datasets was 96.70%. Samriya et al. [16] proposed a network intrusion detection cloud framework ACO-DNN, which used Ant Colony Optimization (ACO) to reduce the dimension of features, and applied a Deep Neural Network (DNN) based model for detection. At the same time, Dynamic Voltage and Frequency Scaling (DVFS) mechanism was adopted to reduce energy consumption. The accuracy of binary classification on NSL-KDD dataset reached 92.9%. Lotfollahi et al. [17] proposed a deep learning-based encryption traffic detection method called Deep Packet. This method firstly removed the fields of SYN, ACK, FIN, DNS etc. in the original traffic, and then used Stacked Automatic Encoder (SAE) and CNN based model to classify the processed network traffic, respectively. Experiments showed the performance of CNN based classifier is better than that of SAE, its detection accuracy of 12-classification on ISCX VPN-nonVPN dataset reached 93%.

In order to show the key points of the above research works more clearly, in Table 1, we summary them from 10 aspects: classification task, style of sample, classification level, pattern of classifier, type of feature, type of attack, sample scale, detection method, type of detection target and whether decompression is required.

For these methods, the premise of applying them is that the malware has already resided on the user side or the attack has been occurred. In the scenario of malware detection, HIDS needs to consume and occupy the resources of host, while NIDS is limited by whether there is communication between the server and the compromise user. The method proposed in this paper belongs to static detection method. It detects malware before it reaches the host. Compared with other static methods, one of the main advantages of this method is that it does not occupy host resources. Compared with other dynamic detection methods, one of the main advantages of this method is it does not need to rely on the running of malware.

III. ENCRYPTED TRANSMISSION TRAFFIC DETECTION BASED ON DEEP LEARNING FUSION FRAMEWORK

To achieve high accuracy detection for traffic generated in the encrypted transmission process of Android APK files, a two-stage detection framework FE-CaDF based on feature enhancement and deep forest is proposed in this paper. As shown in Figure 1, the whole FE-CaDF framework mainly includes a data preprocessing module and two classification modules. The captured encrypted transmission traffic of Android APK files is segmented and filtered in preprocessing module. The two classification modules are connected in series. They are the same framework but with different details. The first module of classification realizes the benign and malicious binary classification task, and the second realizes the multi-classification task for different categories of malware. In this first classification module, a self-developed deep neural network is used to extract the nonlinear features. Then the extracted features are fused with the preprocessed traffic to form the input of the first classifier. In order to detect the category of the malware, the traffic of the malicious samples detected in the first stage are sent to the second stage. In the second stage, a PCA model is used to extract key features of malicious samples. Then, the extracted features are also fused with the preprocessed traffic as the input of the second classifier. With this FE-CaDF detection framework, the predicted label of Android applications in file-level can be obtained.

A. DATA PRE-PROCESSING

At present, the user action of downloading file is one of the common ways to infect malware. The vast majority of android users download and install applications through the APP store. App Store can play as a measure of controlling the spread and effects of malware. But at this time, the malware has arrived at the server, which may cause thread to it. Therefore, detecting malware during their uploading process is very important. One of the mechanisms of APP store to manage files is through File Transfer Protocol (FTP) or Secure File Transfer Protocol (SFTP). Among them, SFTP can provide secure file transfer and prevent information disclosure with Secure Shell (SSH) protocol. Therefore, more and more users adopt SFTP to transfer files. So, the analysis object of this paper is the SFTP transmission traffic.

1) TRAFFIC SEGMENTATION AND FILTERING

This paper identifies SFTP traffic by port number, then the whole traffic generated during the encrypted transmission of an APK file is captured and taken as a sample or meta data. Considering that most of Android applications are transmitted through TCP (Transmission Control Protocol), the flow with other protocols is filtered firstly. Also, since session is an effective analysis unit of traffic detection [18], [30], the meta data is segmented according to the 5-tuple information [source IP address, source port, destination IP address, destination port, transport layer protocol]. In addition, to prevent attackers from spoofing and bypassing detection by forging IP addresses and port numbers etc., these kinds of information and the header of pcap are removed to reduce the interference to the classifier [17]. After preprocessing, only the payload part of original traffic is kept to form the input data.

Figure 2 shows an example of the traffic payload, which was obtained by packet capturing tool when transferring APK files by SFTP. The blue part marked in the figure is the encrypted traffic payload in the meta data.

Furthermore, reference [17] shows that when using deep learning method to detect traffic, the classification model can achieve high accuracy just with the first n bytes of traffic. Therefore, in data processing, the part of the payload exceeding n bytes is truncated, and the insufficient part is supplemented with 0×00. Referring to reference [18], here we adopt $n = 1024$.

2) TRANSFORMATION OF TRAFFIC PAYLOAD MAP

For adapting the input form of convolution neural network, the traffic is need to convert into gray-scale images. Network traffic is essentially a sequence of bytes. A byte is a two-digit hexadecimal number and its corresponding decimal value range is 0-255, which is exactly the range of pixel gray value. Therefore, in this paper, the traffic of an apk file after processing is converted into gray-scale images (called traffic payload map) as the input of the deep learning network. An apk file corresponds to a traffic payload map.

B. FEATURE EXTRACTION AND FUSION

1) THE FRAMEWORK OF FEATURE ENHANCEMENT AND FUSION

Many researchers have proved that feature enhancement is effective for improving detection rate [19], [20]. Considering the detection for the encrypted traffic of APK files is a challenging task, it is difficult to obtain a satisfactory detection rate by only using the original traffic. Therefore, a feature enhancement and fusion framework is presented as shown in Figure 3. In this framework, in order to obtain more effective features for binary classification and multi-classification task, different feature extraction methods are applied in the two stages.

For binary classification task, the traffic payload maps are input into a CNN network, and 1024-dimensional features are extracted. For multi-classification task, the n-dimensional malicious traffic payload obtained in the first stage is sent to PCA model to extract m-dimension key features (m is determined by experiments). Then these extracted features are spliced with the traffic payload to obtain the fusion features, which are used as the input of the corresponding classification model.

2) FEATURE EXTRACTION MODEL BASED ON CNN

For the encrypted traffic payload of APK file, it is difficult to rely on experts to extract explicit and semantic features. Therefore, in order to better analyze the hidden high-level nonlinear features, CNN is used to extract features in the first stage.

A typical CNN mainly includes input layer, convolution layer, pooling layer, full connection layer and output layer. Considering that too many hidden layer units in CNN cannot significantly improve performance, and too few convolution and pooling layers will make the model less robust [21], the CNN presented in this paper consists of an input layer, two convolution layers, two pooling layers and a full connection layer.

The structure is shown in Table 2. Among them, convolution layer and pooling layer are mainly used to extract features from traffic payload map, and the full connection layer is used to integrate those features.

3) FEATURE EXTRACTION MODEL BASED ON PCA

Compared with the binary classification task of benign and malicious software, in the malware categories classification task, the distinguish of classification features become smaller, and the high dimensional redundant features are not conducive to improving the classification effect. In order to obtain the key features that have a great impact on classification, pca method is proposed to extract features at the second stage of fe-cadf framework.

Suppose the traffic payload vector X of each malware consists of n numbers of p-dimensional data, which

TABLE 1. Summary of related work.

Paper	CT	SS	CL	CP	FT	AT	SS	DM	DTT	WDR
Imtiaz et al. [4]	BC&MC	F	FL	ML	EF	K	MD	S	NE	ND
Yadav et al. [5]	MC	F	FL	DL	EF	K	MD	S	NE	ND
Zhu et al. [6]	BC	F	FL	DL	EF	K	MD	S	NE	ND
Alam et al. [7]	BC	F	FL	ML	EF	K	SD	S	NE	ND
Abanmi et al. [8]	BC&MC	F	FL	ML	EF	K	MD	D	NE	DND
Jeon et al. [9]	BC	F	FL	DL	EF	K	LD	S&D	NE	D
Wei et al. [11]	BC	T	FL	DL	EF	K	-	D	E&NE	DND
Liu et al. [12]	BC	T	FL	DF	EF	K	-	D	E&NE	DND
Kamel et al. [13]	BC&MC	T	FL	ML	EF	K	-	D	E&NE	DND
Xu et al. [14]	BC&MC	T	FL	DL	EF	K	LD	D	E&NE	DND
Li et al. [15]	BC	T	TL	DL&ML	EF	K	LD	D	-	DND
Jitendra et al. [16]	BC	T	TL	DL	EF	K	LD	D	-	DND
Lotfollahi et al. [17]	MC	T	TL	DL	OT	K	SD	D	E	DND
Ours	BC&MC	T	FL	DF	EF&OT	K&UK	SD	S	E	DND

Classification task (CT): Binary classification **(BC),** Multi-classification **(MC)**
Style of sample (SS): File **(F),** Traffic **(T)**
Classification level (CL): File-level **(FL),** Traffic level **(TL)**
Pattern of classifier (CP): Deep Learning **(DL),** Machine Learning **(ML),** Deep Forest **(DF)**
Type of feature (FT): Extracted feature **(EF),** Original traffic **(OT)**
Type of attack (AT): Known **(K),** Unknown **(UK)**
Sample scale of minimum class (SS): less than 1000 samples **(SD),** 1000-10000 samples **(MD),** more than 10000 samples **(LD)**
Detection method (DM): Static **(S),** Dynamic **(D)**
Type of detection target (DTT): Encrypted **(E),** Non-encrypted **(NE)**
Whether decompression is required (WDR): need decompression **(ND),** Do not need decompression **(DND)**

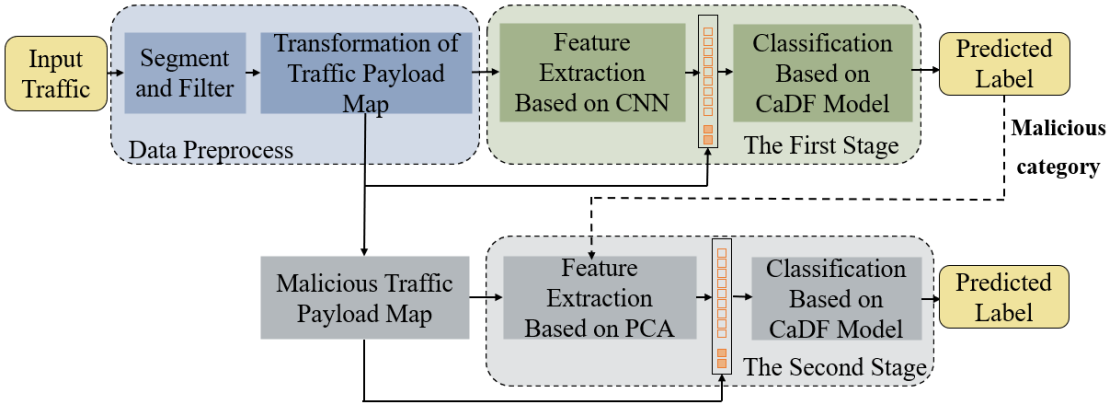

FIGURE 1. The framework of encrypted transmission traffic detection.

can be expressed as

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \quad (1)$$

According to PCA method, each term of the standardized matrix $\mathbf{Z}$ of $\mathbf{X}$ is:

$$\mathbf{Z}_{ij} = \frac{\left(x_{ij} - \overline{x}_j\right)^2}{s_j} i = 1, 2, \ldots, n; j = 1, 2, \ldots, p \quad (2)$$

Here, $\overline{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}, s_j = \frac{\sum_{i=1}^{n} (x_{ij} - x_j)}{n-1}$. The correlation coefficient matrix of $\mathbf{Z}$ is: $\mathbf{R} = \mathbf{Z}\mathbf{Z}^{\mathrm{T}}$. Calculating the characteristic equation of $\mathbf{R}$ with $\left|\mathbf{R} - \lambda \mathbf{I}_p\right| = 0$, p eigenvalues and eigenvectors can be obtained. The eigenvectors are arranged in rows from top to bottom according to the corresponding eigenvalue size to form a matrix, and the first k rows are taken to form matrix $\mathbf{P}$. After calculating $\mathbf{Y} = \mathbf{P}\mathbf{X}$, k dimension data can be got, which are the key features.

```
9d 01 00 16 da 74 4d bb  ba 60 60 e6 e6 5d 50 18
17 3d bc 3d 00 00 14 8f  96 51 e3 2e a0 5d 8a 17
d1 e9 32 80 3b fd 3c 83  17 e1 63 34 81 6c e7 75
e1 cd 10 b2 2a fd 97 92  54 a1 d6 77 5d 9b 1d 1a
94 c2 4f a5 11 ba 10 3d  e8 a0 46 26 4a 51 88 f8
5a 30 c3 ed 3f ed 95 87  f4 c4 29 97 47 79 ae 95
f7 5b 16 e9 07 5e
```

FIGURE 2. Example of encrypted payload data.

TABLE 2. CNN network structure.

Stage	CNN Network Structure
Conv1	Filter= (25, 1, 32), stride=1, padding='SAME'
Pooling1	Max pool
Conv2	Filter= (25,32,64),stride=1, padding='SAME'
Pooling2	Max pool
Dense	Output = 1024

C. DETECTION MODEL BASED ON CASCADE FOREST

In order to implement high accuracy classification, especially for the case of small-scale samples and imbalanced data in the real network environment, a deep forest based classifier is built.

gcForest (Multi-Grained Cascade Forest) [22] is a deep forest implementation framework proposed by Zhou etc. Unlike deep neural networks, which require a large amount of data to train models to obtain good performance, gcForest is a decision tree integration framework and the depth of model can be dynamically adjusted according to the scale of the data. It is not only suitable for large-scale data, but also suitable for small-scale and unbalanced data.

gcForest framework mainly includes two parts: multi-grained scanning module and cascading module. The multi-grained scanning module uses sliding windows with different grains to scan the original data to get input vectors. Considering that the traffic payload map is quite different from the general image, each byte of it has different meanings and less redundant information, extracting vectors with sliding windows will cause the loss of feature information and increase the time consuming [23]. Therefore, only cascading module are adopted here. The cascade model, named CaDF, is shown in Figure 4.

It can be seen from Figure 4, the model of CaDF consists of multiple layers, and each layer contains some base classifiers. The commonly used base classifiers are XGBoost, Random Forest, Extra Tree and SVM. Through the combination of different kinds and numbers of base classifiers, different classification models can be obtained.

In Figure 4, each layer of CaDF includes three base classifiers: XGBoost, Random Forest and SVM. Assuming that there are k classes need to be predicted, then each base classifier will output k-dimensional class vectors. This value is a k-dimensional probability vector. For example, in the first layer, when the model obtains the input data, the three base-classifiers in this layer performs the classification task separately and outputs the k-dimensional class vectors respectively. These class vectors are spliced, and connected with the original input data as the input of the next layer. And the following layers are also like that. In the last layer, the outputs vectors are the average value of the class vector generated by each base classifier. The maximum probability value in the class vectors represents the final prediction class of the input data.

In this model, the numbers of layers are self-adaptive. When expanding the next new layer, the accuracy of the model will be calculated. If there is no improvement compared with the previous layers, the training process will be terminated. Therefore, the model can automatically adjust the cascade depth according to the data scale, which makes it suitable for different scales of training data.

D. FE-CADF ALGORITHM PROCEDURE

1) THE ALGORITHM OF THE PROPOSED FE-CADF FRAMEWORK IS AS FOLLOWS

In the process of model training, to reduce the risk of overfitting, k-fold cross validation operation is applied for the base classifiers in each layer (Here $k = 5$). At the same time, for training strategies, when the accuracy of each layer is not increased compared with the previous two layers, the model stops training and outputs the classification results.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. DATASET AND EVALUATION INDEX

1) DESCRIPTION OF DATASET

Two datasets, named Dataset1 and Dataset2, are used in the following experiments. All the data are come from real world.

The Dataset1 is CICAndMal2017 [24]. CICAndMal2017 consists of 1700 benign applications collection from Google store, and 42 different families of malwares from multiple sources. There are benign applications and four types of malwares (Adware, Ransomware, Scareware and SMSmalware) in it. Among them, Adware is an advertising pop-up malware, which has little difference from the normal traffic. It is particularly difficult to be detected.

In order to show the universality of data, in Dataset2, the benign application is taken from Androzoo [25], and the three categories of malware (Banking, Botnet and Fakeinstaller) are from CICMalDroid2020 [26], Android Botnet dataset [27], and drebin dataset [28], [29], respectively. All the APK files of the three malicious classes in the datasets are selected.

The sample categories and numbers in the two datasets are shown in Table 3 and Table 4, respectively. It can be seen in Dataset1, the number of samples of malware of each malware is only about 100, which belongs to small-scale sample.

In the experiment, SFTP protocol is used to transfer APK files in Dataset1 and Dataset2 between virtual machines, and packet capturing tool is used to obtain the traffic. The whole traffic payload of an APK file is used as a sample. Each

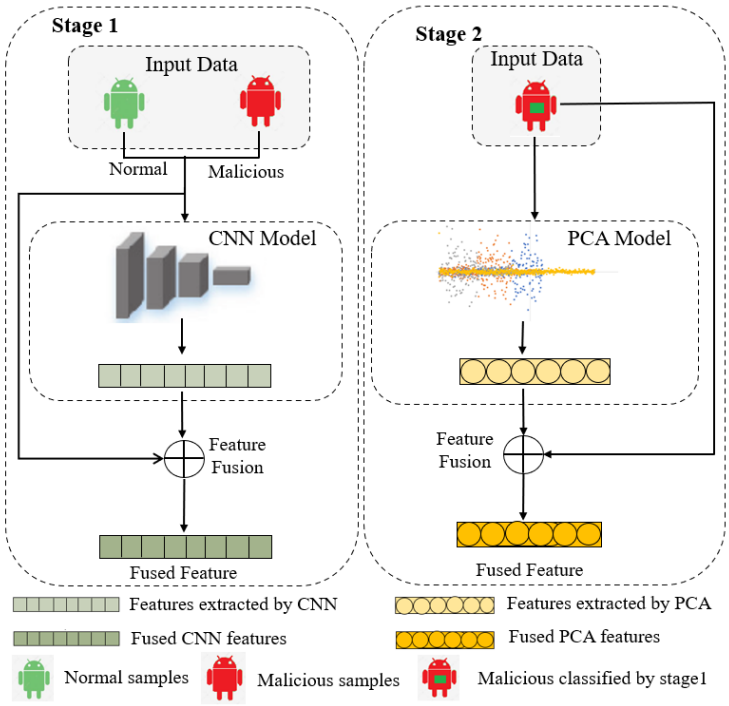

FIGURE 3. The framework of feature enhancement and fusion.

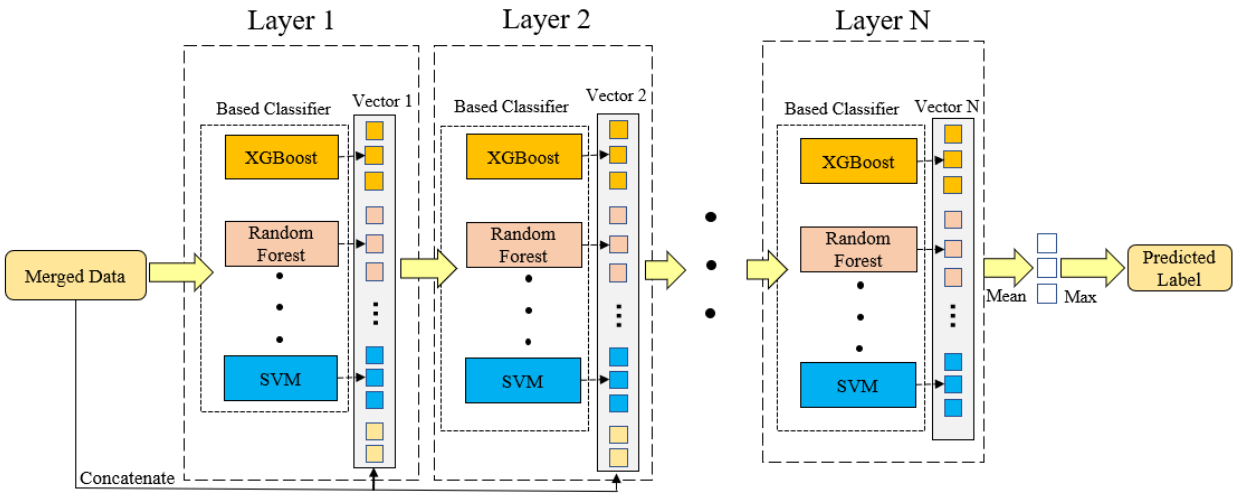

FIGURE 4. The structure of the cascade deep forest.

dataset is divided into training, validation and test set in the form of 3:1:1.

2) EVALUATION CRITERION

Let *TP* represents the number of instances correctly classified as X, *TN* represents the number of instances correctly classified as Not-X, *FP* represents the number of instances incorrectly classified as X, and *FN* represents the number of instances incorrectly classified as Not-X. ACC, DR, TPR, ROC curves, AUC values, Precision and F1-score are used to evaluate the performance of the classification model:

$$\text{ACC} = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (3)$$

$$\text{DR} = \frac{TP}{TP + FN} \times 100\% \quad (4)$$

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\% \quad (5)$$

$$\text{FPR} = \frac{FP}{TN + FP} \times 100\% \quad (6)$$

$$\text{F1} - \text{score} = 2 \times \frac{TPR \times Precision}{TPR + Precision} \times 100\% \quad (7)$$

TABLE 3. Category and Number of Samples in Dataset1.

Benign	Adware	Ransomware	Scareware	SMSmalware
1700	104	101	112	109

Among them, ACC is the accuracy of the detection model. DR is the detection rate of each class. AUC is the area under the ROC curve. FPR is False Positive Rate. F1-score is the average of Precision and DR.

3) EXPERIMENTAL ENVIRONMENT AND MODEL PARAMETER

The computer operating system is windows10 64, CPU is core i7-10750H, memory is 16.00 GB, GPU is NVINIA GeForce RTX 3060, and the development environment is python3.6, the version of Tensorflow is 2.1.0, the version of torch is 1.9.0, and CUDA is 11.1.

For the CNN feature extraction model used in the first stage, the batch size is 50, the learning rate is 0.000001, the dropout is 0.5 and the activation is ReLU. The learning rate is updated once after 20 rounds.

B. EXPERIMENTS AND RESULTS

In this part, the experiments include the binary classification and multi-classification experiments. Experiments 1 to 8 are conducted on Dataet1. In order to verify the generalization performance of the proposed method, Experiment 9 is conducted on Dataset2. Experiment 10 is a comparative experiment on two datasets.

1) EXPERIMENT 1: CLASSIFICATION MODEL SELECTION EXPERIMENT

Firstly, in order to prove the effectiveness of the proposed model of CaDF, it is compared with gcForest and several classical and novel deep learning algorithms: VGG, Densenet, DPN68 and Transformer.

In this experiment, the input is the original traffic payload, and the base classifier of CaDF adopts one XGBoost. The experiment results of binary classification and multi-classification are shown in Table 5 and Table 6, respectively. Here, DR_{nor} is the detection rate of benign application, DR_{mal} represents that of malware.

As can be seen from Table 5, CNN and VGG based detectors have the highest detection rate of 100% for normal traffic, but they can hardly detect malicious traffic. The detection rate of malicious traffic is 1.19% and 0%, respectively. The DR and ACC of Densenet, DPN68 and Transformer are lower than that of ours. The DR_{nor} of gcForest is 0.29% higher than ours, but the DR_{mal} is 13.94% and ACC is 6.13% lower than ours, respectively.

In addition, it can be seen from Table 6 that in the multi-classification experiment, the accuracy of our model is the highest, but the effects of all models are not satisfactory, the average accuracy of all models is less than 40%.

Therefore, the proposed model has better detection performance than gcForest and other deep learning model in both binary classification and multi-classification tasks.

2) BINARY CLASSIFICATION EXPERIMENT OF BENIGN AND MALICIOUS APPLICATIONS

a: EXPERIMENT 2: FEATURE EXTRACTION EXPERIMENT

It can be seen from Experiment 1, the detection rate of the model is not high enough when only using the original traffic payload as the input vector of the classifier, especially for multi-classification. This means that only relying on CaDF model to represent data is not enough.

This experiment is used to prove the effectiveness of the proposed feature extraction model and feature enhancement method. The proposed feature extraction models CNN is compared with Densenet, Resnet, VGG, PCA and gcForest's multi-grained scanning module (called MG-Scan). Here, the CaDF model is the same as experiment 1. The classification results with different feature extraction methods are shown in Table 7, and the ROC curves are shown in Figure 5.

It can be seen from Table 7, the value of DR_{mal} is 0 when Densenet, Resnet and VGG based feature extraction models are used, which means these models are invalidation for encrypted APK traffic. When using PCA and MG-Scan methods, although their DR_{nor} are higher than that of CNN, the value of DR_{mal} is much lower than CNN. Also, the accuracy of CNN is better than all the other models.

It can be further seen from the ROC curve in Figure 5, when using CNN feature, the classification model has the best comprehensive performance and the best value of AUC. Therefore, using CNN to extract benign and malicious features for binary classification is suitable.

b: EXPERIMENT 3: FEATURE ENHANCEMENT AND FUSION EXPERIMENT

This experiment is used to verify the effectiveness of the proposed feature enhancement and fusion methods. The CaDF classification model is the same as Experiment 1. Traffic payload (T-F), PCA feature (PCA-F), CNN feature (CNN-F), traffic payload © CNN feature (T-CNN-F), traffic payload © PCA feature (T-PCA-F) and traffic payload © PCA © CNN feature (T-PCA-CNN) are used to train the classification models, respectively. The performances of these models are shown in Table 8.

It can be seen from Table 8, when the traffic payload is used alone, the DR_{nor} and ACC perform well. However, the DR_{mal} is low, which only reaches 66.67%. When only PCA and CNN features are used, the index of DR_{mal} is

TABLE 4. Category and Number of Samples in Dataset2.

Benign	AndroidBonets	Banking	Fakeinstaller
6813	1929	2506	925

TABLE 5. The performance of different models on binary classification.

	DR_{nor} (%)	DR_{mal} (%)	ACC (%)
CNN	***100***	1.19	80.09
VGG	***100***	0	79.81
Densenet	72.35	40.70	66.04
DPN68	83.24	45.35	75.71
Transformer	23.24	74.42	33.49
gcForest	96.47	52.73	84.20
Ours	96.18	***66.67***	***90.33***

TABLE 6. The performance of different models on multi-classification.

	DR (%)				ACC (%)
	Adware	Ransomware	Scareware	SMSmalware	
CNN	19.04	61.90	4.76	0	21.43
VGG	0	0	***100***	0	25.29
Densenet	***5***	31.58	52.38	4.76	23.46
DPN68	***5***	***100***	0	0	24.9
Transformer	0	0	***100***	0	25.29
gcForest	0	42.85	39.13	4.55	21.84
Ours	0	42.86	43.48	***54.54***	***35.63***

TABLE 7. The comparison of binary classification under different feature extraction models.

	DR_{nor} (%)	DR_{mal} (%)	ACC (%)
Densenet	100	0	80.19
Resent	100	0	80.19
VGG	100	0	80.19
MG-Scan	96.47	52.73	84.20
PCA	***97.65***	32.14	84.67
CNN	86.18	***84.52***	***85.85***

TABLE 8. The performance of binary classification models with different features.

	DR_{nor} (%)	DR_{mal} (%)	ACC (%)
T-F	96.18	66.67	90.33
CNN-F	96.47	75.00	92.22
PCA-F	96.76	63.10	90.09
T-CNN-F	***98.02***	***98.57***	***98.11***
T-PCA-F	97.35	69.05	91.75
T-PCA-CNN	97.35	63.10	91.21

less than 80%. But when the traffic payload fused with the CNN or PCA features, the performance of the classification model is improved significantly. Compared T-CNN-F with T-F and CNN-F, the DR_{mal} increases by 31.9% and 23.57%, respectively. However, when CNN features, PCA features and the original traffic payload are combined, all indicators of the classification model decrease.

It is thus clear that combining CNN features with traffic payload can enhance the data representation in binary classifications effectively.

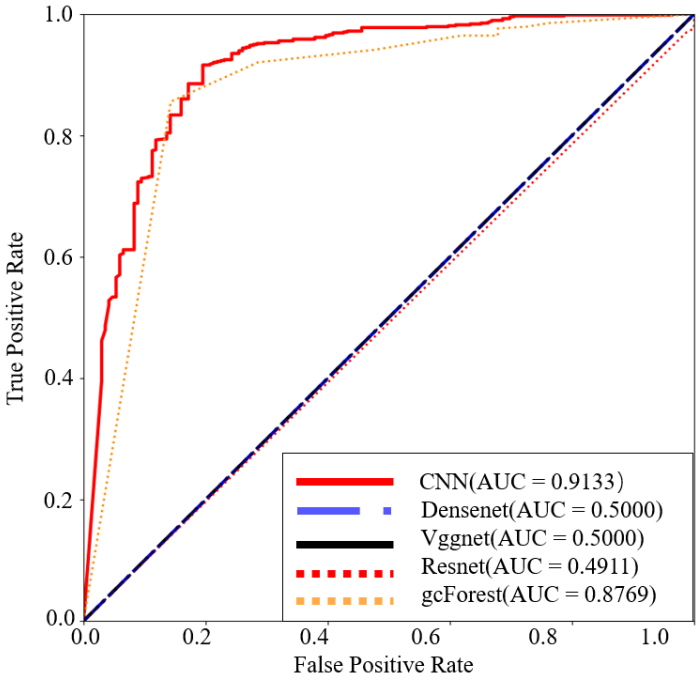

FIGURE 5. ROC curve of different models.

c: EXPERIMENT 4: BASE CLASSIFIER COMBINATION EXPERIMENT IN BINARY CLASSIFICATION

The main hyper parameters of CaDF are the category and number of base classifiers. The experiment is to find the optimal value of these two parameters. In this experiment, the fusion features T-CNN-F obtained in experiment 3 are used to train model. the classification results are shown in Table 9. Here, this result is the average value of 10 times.

In this table, 10, RF and ET represents xgboost, random forest and extra trees, respectively and the digit represents the number of base classifiers. For example, 1X $\oplus$ 1RF represents one XGBoost and one Random Forest are included in each layer.

As can be seen from TABLE 9, when take one XGBOOST as the base classifier, the values of DR_{NOR}, DR_{MAL} and acc are the best, DR_{nor} and DR_{mal} are all higher than 98%. Therefore, in the binary classification, xgboost is selected as the type of base classifier in each layer, and the number of it is 1.

3) MALWARE MULTI-CLASSIFICATION EXPERIMENTS

a: EXPERIMENT 5: MALICIOUS FEATURE EXTRACTION EXPERIMENT

Experiment 3 has proved that applying feature enhancement method can improve the detection rate of classification model. Therefore, this experiment mainly verifies the effectiveness of using PCA model to extract features. CNN, Densenet, MG-Scan and Singular Value Decomposition (SVD) are used to extracted features and compared with PCA, respectively. The features extracted by these models are combined with the original traffic payload as the input of detectors. Here, the CaDF classification model is the same as experiment 1, and the PCA feature dimension is 256. The experimental results are shown in Table 10.

It can be seen from Table 10, the features extracted by all the models are not ideal for the class of Adware. This is because the classifier confused the class of Adware and Ransomware. It is indicated that the characteristics of these two classes are very similar. Using the features extracted by MG-Scan, Densenet, SVD and CNN cannot distinguish them. Only with PCA features, Adware can be detected to a certain extent. Compared with other models, the detection rates for four types of malwares are greatly improved with PCA features. So, using PCA to extract the enhancement features are effective in the multi-classification task.

b: EXPERIMENT 6: THE FEATURE DIMENSION SELECTION EXPERIMENT

This experiment is to seek the optimal dimension of extracted PCA features. In this experiment, different dimensions are selected to form the fusion features. The classification model is the same as Experiment 1. The detection results are shown in Table 11.

It can be seen, with the increase of feature dimension, the accuracy of the classification model continues to improve. When the dimension is 512 or 1024, the classifier can distinguish different types of malwares with high accuracy, especially for Adware. The detection rate of each malware can reach more than 90%. But when the dimension is 2048, the detection rate of SMSmalware decreased.

Algorithm 1 FE-CaDF Algorithm

Input: Training set $\mathbf{D}^0 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{|D|}\}$ // |D| represents the number of training sample
Initialize: CNN model M^{C}, PCA model M^{P}, binary classification model M^{B},multi classification model M^{M}, the number of layers marked as n (n=0) in M^{B} and M^{M}, the maximum layers $N_{\max}$
Output: Classification results
//The output of stage1 is the result of binary classification. The output of stage2 is the result of multi classification.
Stage1:
(1) Input $\mathbf{D}^0$into M^{C} and extract 1024-dimension feature $\mathbf{F}^0$
(2) Fusing $\mathbf{F}^0$ with $\mathbf{D}^0$ to obtain a new binary classification training set $\mathbf{D}_{\text{new0}}$
(3) Send $\mathbf{D}_{\text{new0}}$ into the model M^{B}
For$n < N_{\max}$
do
$n+=1$
Use $\mathrm{D}^{n-1}_{\text{new0}}$ training to get M^B_n, output vector set $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots, \mathbf{y}_{|D|}\}$
$\mathrm{M}^{\mathrm{B}} = \mathrm{M}^{\mathrm{B}} \oplus \mathrm{M}^B_n$//$\oplus$ represents cascade
Calculating the accuracy of model M^{M} by k-fold Cross Validation ($k = 5$)
If the accuracy of the model is not improved for l consecutive times($l = 3$):
Output the prediction class, which has the maximum probability value
break
$\mathbf{D}^n_{\text{new0}} = \mathbf{D}^{n-1}_{\text{new0}} \copyright \mathbf{Y}$ // $\copyright$represents splicing. Splicing $\mathbf{D}$ vectors and $\mathbf{Y}$ vectors
End
Stage2:
(1) Input the malicious data in $\mathbf{D}^0$ predicted from **Stage1** into M^{P} and extract i-dimension feature $\mathbf{F}^1$
(2) Fusing $\mathbf{F}^1$ with malicious data in $\mathbf{D}^0$ to obtain a new multi-classification training set $\mathbf{D}_{\text{new1}}$
(3) Send $\mathbf{D}_{\text{new1}}$ into the model M^{M}
For$n < N_{\max}$
do
$n+=1$
Use $\mathrm{D}^{n-1}_{\text{new1}}$ training to get M^M_n, output vector set $\mathbf{Y}= \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots, \mathbf{y}_{|D|}\}$
$\mathrm{M}^{\mathrm{M}} = \mathrm{M}^{\mathrm{M}} \oplus \mathrm{M}^M_n$//$\oplus$ represents cascade
Calculating the accuracy of model M^{M} by k-fold Cross Validation ($k = 5$)
If the accuracy of the model is not improved for l consecutive times($l = 3$):
Output the prediction class, which has the maximum probability value
break
$\mathbf{D}^n_{\text{new1}} = \mathbf{D}^{n}_{\text{new1}}{}^{-1} \copyright \mathbf{Y}$ // $\copyright$represents splicing. Splicing $\mathbf{D}$ vectors and $\mathbf{Y}$ vectors
End

Therefore, 512 or 1024 are selected as the candidate dimension in the subsequent experiments.

c: EXPERIMENT 7: COMBINATION EXPERIMENT OF MULTI-CLASSIFICATION CLASSIFIERS

This experiment is to find the optimal type and number of base classifiers in multi-classification task. Here, the PCA based fusion feature is adopted, the dimension is 512, and XGBoost, Random Forest, Extra Trees and SVM are applied as the base classifier in each layer of CaDF. The experiment results are shown in Table 12 and the results are the average values of 10 times.

As can be seen from Table 12, When using 1XGB ⊕ 1RF ⊕ 1SVM as the base classifier, the detection rate of Ransomware and Scareware is 100%. The detection rate of Adware and SMSmalware also reaches more than 99%.

When 1ET is used as the base classifier, the detection rate of Adware, Ransomware and Scareware reaches 100%, and that of SMSmalware exceeds 95%. Furthermore, when we change the dimension of input vectors as 1024, the detection rate of SMSmalware increase to 99.09%.

When 1RF is used as the base classifier, the detection rate of Adware, Ransomware reaches 100%, and that of Scareware and SMSmalware exceed 95%. Furthermore, when the dimension of input vectors is changed as1024, the detection rate of Adware and Ransomware reaches 100% and that of Scareware and SMSmalware reaches 99.57% and 99.09%, respectively.

In order to further analyze the performance of these base classifiers, the training and test time of CaDF are shown in Table 13. In the table, Tr is the training time and Te is the test time for 84 APK files. For 1RF and 1ET case, the dimension of PCA feature is 1024, while for 1XGB ⊕ 1RF ⊕ 1SVM, the dimension of PCA feature is 512.

It can be seen from Table 13, when 1XGB ⊕ 1RF ⊕ 1SVM and 1ET are used for base classifier, the training time increases by 80.88% and 64.38% compared with 1RF, respectively, and the test time increases by 27.27% for both. So, 1RF is selected as the final base classifier and the PCA dimension is 1024.

Finally, considering the number of samples of each class is just about 100, the experiment also prove that this method is suitable for small-scale classification task.

4) EXPERIMENT 8: COMPARATIVE EXPERIMENT

In order to further verify the effectiveness of FE-CaDF, the binary classification methods proposed in reference [11], [12], [13], and [14] are compared with it on CICAndMal2017. Table 14 shows the comparison results.

It can be seen from Table 14 that the method proposed by reference [11] has best ACC and TPR. However, the FPR of our method is 0.18% lower than it. The F1-score and Precision are 0.39% and 1.16% higher than it respectively.

For multi-classification, since there are not multi-classification results are shown in reference [11] and [12], so we only compared our model with the methods proposed

TABLE 9. The performance of binary classification model with different combination of base classifiers.

	DR_{nor} (%)	DR_{mal} (%)	ACC (%)
1X	***98.02***	***98.57***	***98.11***
1RF	95.88	92.86	95.28
1ET	92.64	41.67	82.55
1X⊕1RF	96.76	89.29	95.28
1X⊕1ET	95	46.43	85.38
1X⊕1SVM	94.71	80.95	91.98
1X⊕1RF⊕1ET	97.94	85.71	95.52
1X⊕1RF⊕1SVM	93.53	92.86	93.40
2X	91.76	95.24	92.45

TABLE 10. The dr of multi-classification models with different feature extraction models.

	DR (%)			
	Adware	Ransomware	Scareware	SMSmalware
MG-Scan	0	19.05	39.13	68.18
CNN	4.76	33.33	43.48	36.36
Densenet	0	47.62	39.13	50.00
SVD	0	28.57	43.48	40.91
PCA	***52.38***	***95.24***	***100***	***95.45***

TABLE 11. The dr of multi classification models with different PCA feature dimensions.

	DR (%)			
	Adware	Ransomware	Scareware	SMSmalware
64	0	71.43	43.48	100
128	0	***95.24***	56.52	86.36
256	52.38	***95.24***	***100***	***95.45***
512	***90.48***	***95.24***	95.65	90.91
1024	***95.24***	***95.24***	95.65	***95.45***
2048	***95.24***	***95.24***	***100***	90.91

TABLE 12. The DR of multi classification models with different base classifiers.

	DR (%)			
	Adware	Ransomware	Scareware	SMSmalware
1XGB	90.48	95.24	95.65	90.91
1RF	***100***	***100***	95.65	95.45
1ET	***100***	***100***	***100***	95.45
2XGB	95.24	95.24	***100***	90.91
1XGB⊕1ET	***100***	***100***	***100***	95.45
1XGB⊕1RF	***100***	***100***	86.96	***100***
1XGB⊕1ET⊕1RF	***100***	***100***	95.65	95.45
1XGB⊕1SVM	***100***	95.24	95.65	86.36
1XGB⊕1RF⊕1SVM	99.05	***100***	***100***	99.09

in [13] and [14]. Reference [13] and [14] give out the 4-classificationand 5-classifiaction results on malware categories, respectively. Considering FE-CaDF is connected in series, we multiply the output values of two stages to obtain the final classification results. Figure 6 shows the comparison results of 4-classification, and Figure 7 shows that of 5-classification.

In these two figures, M-TPR and M-Precision are represented as the average value of TPR and precision, respectively. It can be seen, whether for 4-classification or

TABLE 13. The training and testing time of multi-classification model with different base classifiers.

	Tr(s)	Te(s)
1RF (1024)	5.91	1.60
1ET (1024)	11.01	2.20
1XGB⊕1RF⊕1SVM (512)	30.91	2.20

TABLE 14. The comparison of binary classification models with different methods on dataset1.

	ACC (%)	TPR (%)	FPR (%)	F1-Score (%)	Precision (%)
Wei et al. [11]	***98.26***	***98.39***	1.61	98.39	98.39
Liu et al. [12]	97.29	96.07	-	95.34	94.63
Kamel et al. [13]	87.75	85.33	-	87.29	89.35
Xu et al. [14]	97.16	97.16	2.87	97.14	97.13
Ours	98.11	98.02	***1.43***	***98.78***	***99.55***

TABLE 15. The DR of multi-classification on dataset2 with different base classifiers.

	DR (%)		
	AndroidBonets	**Banking**	**Fakeinstaller**
1XGB⊕1RF⊕1SVM	97.93	100	99.46
1ET	100	100	100
1RF	100	100	100

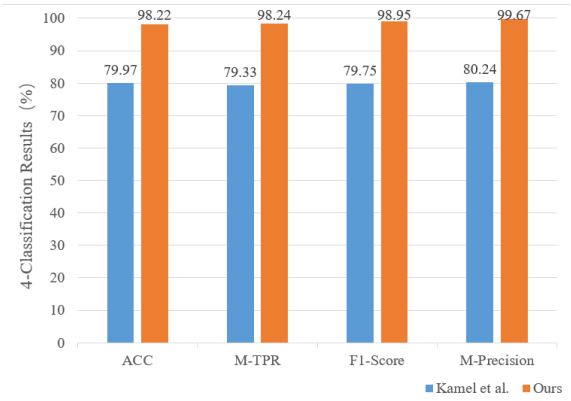

FIGURE 6. Comparison experiments of 4-classification with different methods on dataset1.

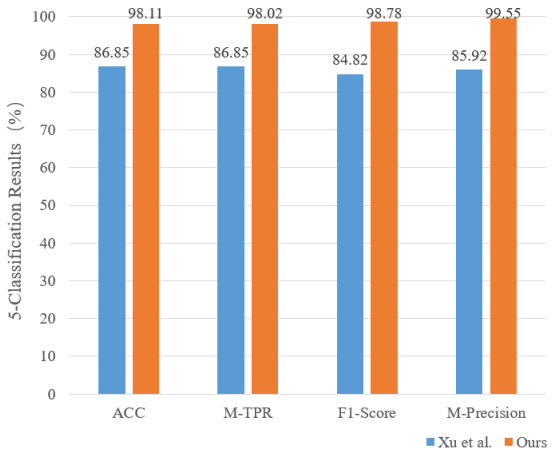

FIGURE 7. Comparison experiments of 5-classification with different methods on dataset1.

5-classification, the indications of our method are much better than those of compared methods.

It can be seen from the results that the method proposed in this paper is generally superior to the methods proposed in reference [11], [12], [13], and [14], whether it is binary-classification or multi-classification. In contrast, the method proposed in this paper belongs to static detection method, which realizes detection at the file-level by analyzing the traffic in the malware transmission stage. The other methods belong to dynamic detection methods, which achieve file-level detection by analyzing the communication traffic when the malware is running. These 4 methods need to execute detection after the malware is running. But at this time, the malware may have caused a threat to the user. Therefore, the method proposed in this paper has more advantages in realizing early detection of malware.

5) EXPERIMENT 9: MODEL GENERALIZATION PERFORMANCE EXPERIMENT

In order to verify the generalization performance of the proposed model, binary classification and multi-classification experiments are carried out on Dataset2.

Using the CNN feature enhancement method and the binary classification model established in section IV-B2, the

ROC curve is shown in Figure 8. DR_{nor}, DR_{mal} and ACC can be achieved 100%, 99.91% and 99.96% respectively.

From Figure 8, it can be seen, on Dataset2, the AUC value reaches 0.9995, which indicates excellent performance of the model.

Using the multi-classification model established in section IV-B3, the base classifier adopts 1XGB ⊕ 1RF ⊕ 1SVM, 1ET and 1RF respectively. Among them, the former adopts 512-dimension features while 1ET and 1RF use 1024-dimension features. The experiment results are shown in Table 15.

It can be seen from Table 15 that in multi-classification, when 1XGB ⊕ 1RF ⊕ 1SVM is used as base classifier, the average detection rate is 99.13%, while when ET and RF are used, the average detection rate reaches 100%. The performance of FE-CaDF is excellent.

To analyze the reason, it may be that we only use the traffic payload as the analysis object, which eliminates the interference of other information. At the same time, the proposed feature fusion method can effectively represent the data, and the established classification model can adapt to various sizes of data. Therefore, the FE-CADF framework has good generalization performance.

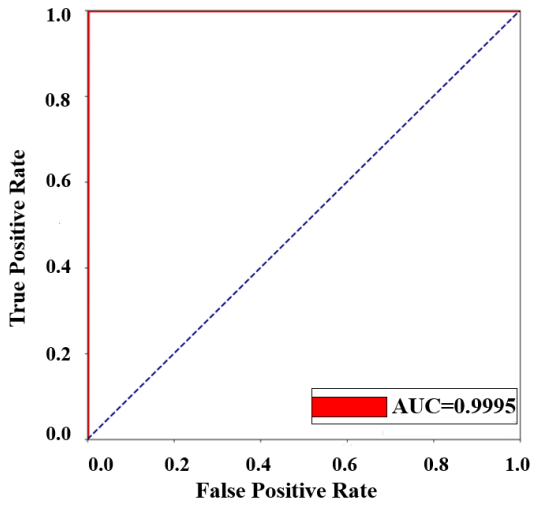

FIGURE 8. ROC curve on dataset2.

6) EXPERIMENT 10: UNKNOWN MALWARE DETECTION

This experiment further proves the detection capability of FE-CaDF for unknown malware. Here, the normal samples and the Scareware samples in Dataset1 are used to construct the training dataset. Adware, Ransomware and SMSmalware are used as unknown sample. The experiment results of binary classification are shown in Figure 9.

In this figure, normal_1 indicates the normal sample and mal_1 indicates the unknown malicious sample. As can be seen from Figure 9, the detection rate of for unknown malware reaches 93.55%.

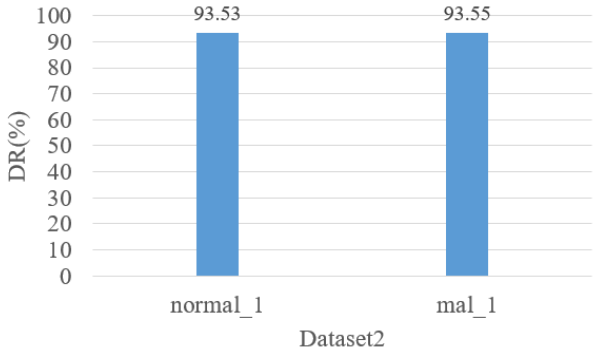

FIGURE 9. Unknown Malware binary classification results on dataset1.

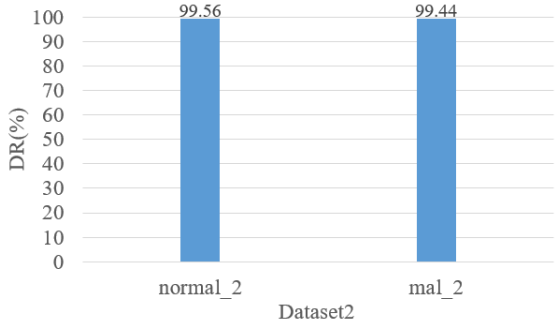

FIGURE 10. Unknown malicious binary classification results on dataset2.

Furthermore, Dataset2 is used as an unknown sample. The binary classification experimental results are shown in Figure 10.

In this figure, normal_2 and mal_2 indicate the normal and malicious samples in Dataset2, respectively. As can be seen from it, the DR_{nor} can reach 99.56% and the DR_{mal} can reach 99.44%. It shows that although the model is not trained on Dataset2, it still has good performance for those test samples in Dataset2.

To sum up, the FE-CaDF model proposed in this paper has strong generalization performance, which can not only detect known malware but also unknown malware with high accuracy.

V. CONCLUSION REMARKS

In this paper, a two-stage detection framework FE-CaDF based on feature enhancement and cascade deep forest is proposed to detect the traffic generated in the encrypted transmission process of Android APK files. The first stage undertakes binary classification task of benign and malicious applications. The second stage is responsible for malware muti-classification task. This method takes the complete traffic of an APK file as the detection unit. To obtain significant classification features, CNN is used to extract features in the first stage, and the PCA method is used to extract features in the second stage. These extracted features are fused with the traffic payload to form fusion features. In order to adapt to different sample scale, cascade deep forest is used to build

classification models in these two stages. With the appropriate base classifier, both of stages achieve high classification accuracy. The experimental results on several datasets show that the FE-CaDF framework also has good generalization performance, it is suitable for the classification of small-scale samples and unknown samples. In the future work, we will further study the transmission detection of iOS applications and Windows applications.

APPENDIX

See Table 16.

TABLE 16. Notation table.

$+$	Numerical addition
$-$	Numerical subtraction
$\times$	Numerical multiplication
Σ	Accumulation
$\oplus$	Cascade
©	Splicing
&	And
$\mathbf{Z}^{\mathrm{T}}$	Transpose of matrix $\mathbf{Z}$
$\mathbf{PX}$	Multiplying matrices $\mathbf{P}$ and $\mathbf{X}$

REFERENCES

[1] F. Tchakounte, "A malware detection system for Android," Ph.D. dissertation, Universitat Bremen, Bremen, Germany, 2015.
[2] T. Kim, B. Kang, and E. G. Im, "Runtime detection framework for Android malware," *Mobile Inf. Syst.*, vol. 2018, pp. 1–15, Jan. 2018.
[3] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah, and A. A. Ghorbani, "Towards a network-based framework for Android malware detection and characterization," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Calgary, AB, Canada, Aug. 2017, pp. 233–23309.
[4] S. I. Imtiaz, S. U. Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient deep artificial neural network," *Future Gener. Comput. Syst.*, vol. 115, pp. 844–856, Feb. 2021.
[5] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based Android malware detection and variant classification," *Comput. Intell.*, vol. 38, no. 5, pp. 1748–1771, Oct. 2022.
[6] H.-J. Zhu, W. Gu, L.-M. Wang, Z.-C. Xu, and V. S. Sheng, "Android malware detection based on multi-head squeeze-and-excitation residual network," *Exp. Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118705, doi: 10.1016/j.eswa.2022.118705.
[7] S. Alam, S. A. Alharbi, and S. Yildirim, "Mining nested flow of dominant APIs for detecting Android malware," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 107026.
[8] J. Jeon, B. Jeong, S. Baek, and Y.-S. Jeong, "Hybrid malware detection based on bi-LSTM and SPP-net for smart IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4830–4837, Jul. 2022.
[9] N. Abanmi, H. Kurdi, and M. Alzamel, "Dynamic IoT malware detection in Android systems using profile hidden Markov models," *Appl. Sci.*, vol. 13, no. 1, p. 557, Dec. 2022, doi: 10.3390/app13010557.
[10] J. Zhou, W. Niu, X. Zhang, Y. Peng, H. Wu, and T. Hu, "Android malware classification approach based on host-level encrypted traffic shaping," in *Proc. 17th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. (ICCWAMTIP)*, Dec. 2020, pp. 246–249.
[11] S. Wei, Z. Zhang, S. Li, and P. Jiang, "Calibrating network traffic with one-dimensional convolutional neural network with autoencoder and independent recurrent neural network for mobile malware detection," *Secur. Commun. Netw.*, vol. 2021, pp. 1–10, Feb. 2021.
[12] M. Liu, S. Wei, and P. Jiang, "A hybrid modeling of mobile app dynamics on serial causality for malware detection," *Secur. Commun. Netw.*, vol. 2021, pp. 1–10, Oct. 2021.
[13] M. Abuthawabeh and K. Mahmoud, "Enhanced Android malware detection and family classification, using conversation-level network traffic features," *Int. Arab J. Inf. Technol.*, vol. 17, no. 4, pp. 607–614, Jul. 2020.
[14] P. Xu, C. Eckert, and A. Zarras, "Falcon: Malware detection and categorization with network traffic images," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2021, pp. 117–128.
[15] D. Li, D. Kotani, and Y. Okabe, "Improving attack detection performance in NIDS using GAN," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 817–825.
[16] J. K. Samriya, R. Tiwari, X. Cheng, R. K. Singh, A. Shankar, and M. Kumar, "Network intrusion detection using ACO-DNN model with DVFS based energy optimization in cloud framework," *Sustain. Comput., Informat. Syst.*, vol. 35, Sep. 2022, Art. no. 100746.
[17] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, 2020.
[18] X. Zhang, M. Zhao, J. Wang, S. Li, Y. Zhou, and S. Zhu, "Deep-forest-based encrypted malicious traffic detection," *Electronics*, vol. 11, no. 7, p. 977, Mar. 2022.
[19] Y. Lai and M. Tariq, "Group enhancement for matching of multi-view image with overlap fuzzy feature," *Multimedia Tools Appl.*, vol. 79, nos. 3–4, pp. 2069–2084, Jan. 2020, doi: 10.1007/s11042-019-08173-0.
[20] T. Sun, C. Shao, H. Liao, S. Ding, and X. Xu, "Research on adaptive local feature enhancement in convolutional neural networks," *IET Image Process.*, vol. 14, no. 16, pp. 4306–4315, Dec. 2020.
[21] Y. Li et al., "Deep forest ensemble learning for classification of alignments of non-coding RNA sequences based on multi-view structure representations," *Briefings Bioinf.*, vol. 22, no. 4, Jul. 2021, Art. no. bbaa354, doi: 10.1093/bib/bbaa354.
[22] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. IJCAI*, 2017, pp. 3553–3559.
[23] Y. Dong, W. Yang, J. Wang, J. Zhao, Y. Qiang, Z. Zhao, N. G. F. Kazihise, Y. Cui, X. Yang, and S. Liu, "MLW-gcForest: A multi-weighted gcForest model towards the staging of lung adenocarcinoma based on multi-modal genetic data," *BMC Bioinf.*, vol. 20, no. 1, p. 578, Dec. 2019.
[24] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark Android malware datasets and classification," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2018, pp. 1–7, doi: 10.1109/CCST.2018.8585560.
[25] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting millions of Android apps for the research community," in *Proc. 13th Int. Conf. Mining Softw. Repositories*, May 2016, pp. 468–471.
[26] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic Android malware category classification using semi-supervised deep learning," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2020, pp. 515–522, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094.
[27] A. F. A. Kadir, N. Stakhanova, and A. A. Ghorbani, "Android BotNets: What URLs are telling us," in *Network and System Security* (Lecture Notes in Computer Science), vol. 9408, M. Qiu, S. Xu, M. Yung, and H. Zhang, Eds. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-25645-0_6.
[28] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. E. R. T. Siemens, "Drebin: Effective and explainable detection of Android malware in your pocket," in *Proc. 21st Annu. Netw. Distrib. Syst. Secur. Symp.(NDSS)*, Feb. 2014, doi: 10.14722/ndss.2014.23247.
[29] M. Spreitzenbarth, F. Freiling, F. Echtler, T. Schreck, and J. Hoffmann, "Mobile-sandbox: Having a deeper look into Android applications," in *Proc. 28th Annu. ACM Symp. Appl. Comput.*, Mar. 2013, pp. 1808–1815, doi: 10.1145/2480362.2480701.
[30] X. Zhang, J. Wang, and S. Zhu, "Dual generative adversarial networks based unknown encryption ransomware attack detection," *IEEE Access*, vol. 10, pp. 900–913, 2022.

XUEQIN ZHANG received the Ph.D. degree in detection technology and automation devices from the East China University of Science and Technology (ECUST), Shanghai, China, in 2007. In 2006, she was a Visiting Scholar with the University of Wisconsin–Madison. Since 1998, she has been with the Department of Electrical and Communication Engineering, ECUST, where she is currently a Professor. Her research interests include information security and pattern classification.

JINYU XU received the master's degree in signal and information processing from the East China University of Science and Technology (ECUST), Shanghai, China, in 2013. Since 2015, she has been with the School of Electrical and Information Engineering, Hubei University of Automotive Technology (HUAT), where she is currently a Lecturer. Her research interests include intelligent manufacturing, information security, and machine learning.

JIYUAN WANG received the B.S. degree from the Central South University of Forestry and Technology, in 2020. He is currently pursuing the M.S. degree in electronics and communications engineering with the East China University of Technology (ECUST). His research interests include intrusion detection, information security, and machine learning.

CHUNHUA GU received the Ph.D. degree in computer software from the School of Information Science and Engineering, East China University of Science and Technology (ECUST), Shanghai, China, in 2007. Since 1992, he has been with the Department of Computer Science, ECUST, where he is currently a Professor. His research interests include security and privacy issues for cloud computing, mobile social networks, and smart grids.