IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON HUMAN-DRIVEN EDGE COMPUTING (HEC)

Received May 31, 2020, accepted June 26, 2020, date of publication July 8, 2020, date of current version July 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3008081

A Two-Layer Deep Learning Method for Android Malware Detection Using Network Traffic

JIAYIN FENG[1,2], LIMIN SHEN[1], (Member, IEEE), ZHEN CHEN[1], YUYING WANG[1], AND HUI LI[1,2]

[1]School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
[2]School of Mathematics and Information Science and Technology, Hebei Normal University of Science and Technology, Qinhuangdao 066004, China

Corresponding author: Limin Shen (shenllmm@sina.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772450, in part by the Hebei Natural Science Foundation under Grant F2019203287, in part by the Science and Technology Research Project of Colleges and Universities in Hebei Province under Grant QN2020183, in part by the Hebei Postdoctoral Research Program under Grant B2018003009, and in part by the Doctoral Fund of Yanshan University under Grant BL18003.

ABSTRACT Because of the characteristic of openness and flexibility, Android has become the most popular mobile platform. However, it has also become the most targeted system by mobile malware. It is necessary for the users to have a fast and reliable detection method. In this paper, a two-layer method is proposed to detect malware in Android APPs. The first layer is permission, intent and component information based static malware detection model. It combines the static features with fully connected neural network to detect the malware and test its effectiveness through experiment, the detection rate of the first layer is 95.22%. Then the result (benign APPs from the first layer) is input into the second layer. In the second layer, a new method CACNN which cascades CNN and AutoEncoder, is used to detect malware through network traffic features of APPs. The detection rate of the second layer is 99.3% in binary classification (2-classifier). Moreover, the new two-layer model can also detect malware by its category (4-classifier) and malicious family (40-classifier). The detection rates are 98.2% and 71.48% respectively. The experimental results show that our two-layer method not only can achieve semi-supervise learning, but also can effectively improve the detection rate of malicious Android APPs.

INDEX TERMS Android, malware detection, deep learning, network traffic.

I. INTRODUCTION

In recent years, Smartphones and tablets have been integrated into every aspect of people's lives. The advanced mobile technologies have made tremendous growth in mobile devices. The mobile device has many functions, such as browsing the Internet, making payments, taking a photo and sharing it. Mobile device is only a small part of the Internet of Things (IoT) device. According to Ericsson report 2019 [1], with mobile systems acting as a backbone for both the mobile phone and Internet of Things (IoT). The number of mobile subscriptions grew at 2 percent year-on-year and currently totals around 7.9 billion and 80 percent of traffic will be generated by mobile networks.

As a smart operation system, Android is the most popular used mobile platform for smartphones and IoT. With the ease of use, low-cost and portability nature to be the characteristics, Android occupies a top market share. The increasing users of Android have spawned a substantial of mobile applications (APPs). The total number of new Android APPs outnumbers IOS APPs by a ratio nearly 3:1 which reported by Mobile APP Trends Report 2019 [3]. People can download popular mobile APPs which access internet via Android for more personalized or complicated things, such as social network APPs, studding APPs, gaming APPs, information exchange APPs, financial transactions and cloud storage APPs etc. These various APPs generate huge amounts of mobile traffic data, which contain highly sensitive information. A statistic shows the number of available applications in the Google Play Store from December 2009 to March 2020 [4]. The number of available APPs in the Google Play Store was most recently placed at 2.87 million APPs, after surpassing 1 million APPs in July 2013.

Though various mobile APPs make our life more convenient, it also brings enormous burden to the mobile and IoT security protection. According to the G DATA report [5] in

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

2017, security experts discovered about 750,000 new Android malware during the first quarter of 2017. It is shown that a large number of mobile APPs carried out malicious code and spread to execute various cybercrimes on mobile devices. Android become the most targeted system by mobile malware not only because of its large market share and open source ecosystem of development, but also because Android operating system allows users to download APPs from third-party markets, and these APPs may include malicious or suspicious applications which seduce users to open its permissions to the APP if they want to use it. Malware is becoming more obscure and harder to eliminate. Consequently, how to protect Android devices from malicious APPs is of vital importance.

Google play is the official market store for android APPs and there are more than hundred third-party market to store android APPs. Because of the lack of effective verify method, malware developers can utilize multiple method to evade the detection provided by Android sand-boxing or other existed antivirus mechanisms and upload their malicious APPs to the market and even Google's official market. These evasion methods include dynamic execution, code obfuscation, repackaging or encryption [43]. Many android malwares which contain the mentioned evasion methods above have already been released in the market which caused great losses to mobile users, such as Trojans APPs can steal user account information, Ransomware can block user's common used software to blackmail money, and some repackaging APPs can send SMS messages without user's concern or implant mounts of ads, which brings great trouble to users. To prevent Android malware attacks, researchers and mobile security products use signature-based or heuristics-based method to detect malware [6]. However, because of the popularity of the mobile network, attackers can use powerful code obfuscation and repackage techniques to conceal their behavior into the large amounts of network traffic and evade the detection. The increasing sophistication of the network behaviors of android malware calls for new defensive techniques that can protect users from new threats.

To be more effective detect the Android malware, in this paper, we make the following contributions to detect Android malware:

1) Static malware detection based on permissions, intent and component information. The static features datasets are input into a fully connected neural network to detect the malware and test its effectiveness through experiments.
2) Network traffic based effective mobile malware detection. Our experimental results show that combining network traffic features with cascading deep learning CACNN methods can effectively identify malicious software in Android APPs.
3) Two-layer detection model. The first layer, applying a fully connected neural network to analyze static features, and input the results to the next detection layer. Second layer, network traffic features detection analyzed the final results to prove that CACNN model can effectively identify malware. At the same time, this models can also detect malware by its category and malicious family. Overall, combining two layer of detection model can further improve the detection efficiency.

The following paper is organized as follows: Section 2 discusses related works. Section 3 introduces the methodology of our models in detail. Experimental results are reported in section 4. Section5 gives the conclusions of the paper.

II. RELATE WORK

There are many proposed security mechanisms to detect Android malware and protect users' mobile phone from attacking. Most of the common mechanism is static analysis. Static analysis detects features from APP by unpacking or disassembling them without running the APP [7]–[11], these features are extracted from permission [12], [13], sensitive API calling or critical code segments in the source code [14]–[16]. However the static analysis method cannot detect certain source code tampering operations. Dynamic analysis examines the running APP and monitoring the execution behavior of APP, such as memory utilization, system calls, network connections and battery power [17]–[21], but it cannot fully traverse the execution path of the software and cannot detect certain malicious behaviors. In order to avoid the limitations of both static and dynamic analysis, researchers and commercial systems have used combination of both the mechanisms that has been termed as hybrid analysis. Hybrid analysis is a two-step process where initially static analysis is performed before the dynamic one [2], [22], [23]. However all these analysis methods above only focuses on the static detection, dynamic detection or both of them, and rarely considers the network traffic generated by malicious APPs.

Currently, almost all the attackers use mobile networks to obtain sensitive information of user or interact with its malicious APPs. Therefore, Android APPs can also be analyzed using network traffic. A number of studies have been conducted on area of Android malware detection using the network traffic. One of the first attempts was presented by Iland *et al.* in 2011 [24], where the authors presented a lightweight approach of detecting Android malware and violations of user privacy through the network traffic. But, this technique cannot be applied in the encrypted traffic. In 2013, Tenenboim Chekina *et al.* [25] proposed a novel network based behavioral analysis for detecting a new group of malware with self-updating capabilities which have been found in the Google Android marketplace. Their results showed that in most of the applications the threat was detected in the first five minutes after the infection occurred. In the same year, a new automatic network profile generator for detecting Android applications in HTTP was designed by Dai *et al.* [26], but the system needs a user seed path when login is involved and it cannot detect APPs which have no distinct network behavior and use the same service. Zaman *et al.* [27] in 2015 demonstrated a work-in progress detection method that was

effective against malware communicating with the remote server C&C servers, which is based on the logs of all remote locations. In 2017 [28] presented an algorithm to prioritize network traffic features with minimize number of features to be analyzed for better detection accuracy and processing time. Lashkari *et al.* [29] propose an Android Malware detection model based on a new network traffic feature set to expedite the efficiency of traffic classifier. It can classified the APPs into benign, malware and adware. Aceto *et al.* [30] aim to improve the performance of classification of mobile APPs traffic by proposing a multi-classification approach for mobile and encrypted traffic classification in 2018. In the same year, another paper [42] proposed a high-efficient hybrid-detecting scheme for Android malwares. They employed static and dynamic methods to analyze features. Additionally, they also presented some tools such as Com+ feature based on traditional Permission and API call features to improve the performance of static detection. Wang *et al.* [31] presented an effective malware identification and classification method called TrafficAV by combining machine learning algorithm and traffic analysis. And they added a section of the prototype system in TrafficAV in [32].

These studies we mentioned above involve many aspects of malicious traffic analysis. However, the above articles are rarely combined with deep learning method to analyze static features and network traffic features. Our approach is different from the above approaches in the following aspects. Firstly, a new novel two-layer model to detect malware is presented. It combines permission, intent and component information based static malware detection method and network traffic based effective mobile malware detection method. Secondly, the second layer combines network traffic analysis with a new cascading deep learning CACNN method that is capable of identifying Android malware with high accuracy. Moreover, our two-layer framework not only detect benign and malware, but also classify the APPs by category and malicious family. In the following sections, we discuss our two-layer model in more details and make the operation process of our method more clearly.

III. METHODOLOGY

In this section, we propose a two layer model for detecting malware. In the first layer, the static analysis method is used to extract features, then a fully connected neural network model is used to classify the datasets into benign and malware. In the second layer, mobile traffic analysis method is used to extract traffic features from all the benign in the first layer, then CACNN model is used to classify malware from the benign. CACNN model can also classify Malware by category and family. The complete framework is summarized as Figure 1.

A. STATIC FEATURE EXTRACTION LAYER

The first layer is a binary classification model between benign and malware sample based on static features.

1) ACQUIRING MANIFEST FILES

To acquire manifest files in batch, APKs are unzipped, and all the manifest.xml files are extracted and saved with the name of APK name.xml file in a folder first. The permission, intent and component information can be obtained from the manifest.xml file.

2) EXTRACTING PERMISSION INFORMATION

Among mobile operating systems, including Android, Windows Phone and Apple IOS, the naming of permissions gives intuitive and exact information on what it requests from OS [33]. Permission models have become one of the primary security mechanisms for Android systems to provide access control to sensitive information or components. Thus, the permission feature extraction is conduct to gain the component information or resources APPs need to share or use from the manifest.XML files. Through extracting the XML nodes information from Manifest iteratively, each node is checked to confirm whether the node includes user permission, system permission and components information.

3) FORMATION FEATURE VECTOR

The extracted features from all the APPs that contains benign and malware in process step 2 are used to format feature datasets. Setting the features attribute extracted from the APP to 1 in the corresponding attribute field, other attributes set 0 in the datasets to format feature vector. The structure of the datasets is described in Table 1.

TABLE 1. Extracted label and feature.

android.intent.action.FLEEXPLORE	ACTION_NEED_WRITE_PERMISSION	android.net.wifi.p2p.THIS_DEVICE_CHANGED	… …	Binary type
0	1	0	…	1
1	0	0	…	0
……	……	……		……

4) STATIC FEATURE DETECTION

The feature detection process is conducted to determine whether the application is malicious or not after all the features dataset is generated in step 3. Firstly, we choose to remove features with low variance, Chi-square test and extremely randomized tree method for feature selection. According to the quantity and quality of feature selection, the extremely randomized tree method is applied finally. Secondly, a fully connected neural network model is built to detect malware. The commonly used activation functions include sigmoid, ReLu, Tanh and Maxout. In our model, the rectified linear unit (ReLU) is taken as activation function after comparative experiment with those activation functions. And the binary_crossentropy loss function is used, so that the model continuously reduces the cross entropy between output and label during the training process. Thirdly, the model is used to classify the APPs into benign and malicious. Finally, all the benign APPs that detected in this layer are input into the network traffic analysis layer.

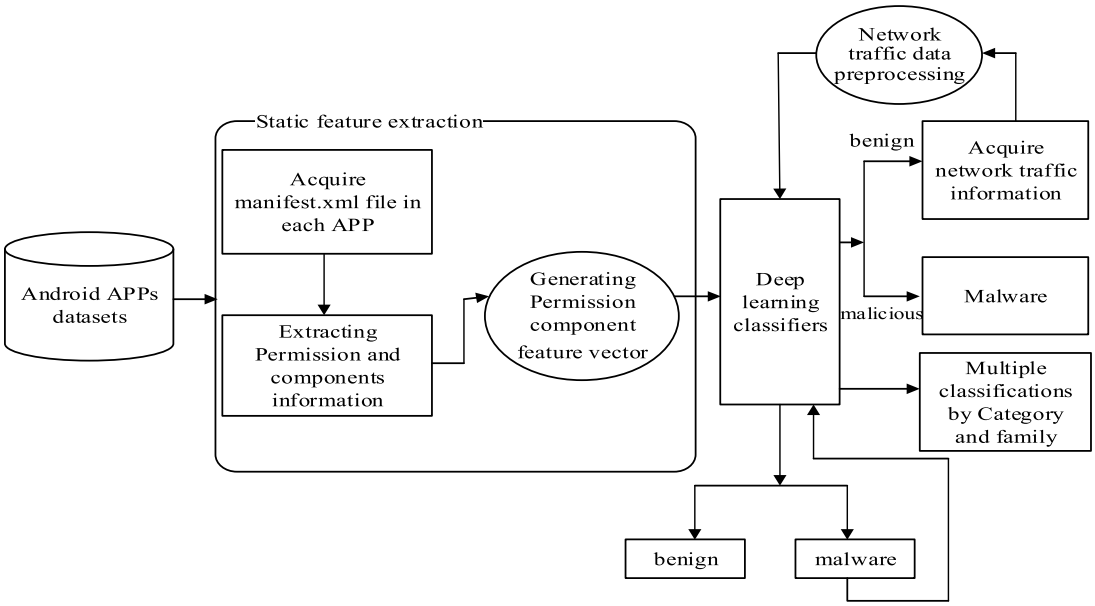


FIGURE 1. The overall architecture of the proposed mode.

B. NETWORK TRAFFIC ANALYSIS

All the network traffics are obtained from last output benign APPs of a fully connected neural network model. Then, the captured network traffic datasets are input into CACNN layer. CACNN layer contained two parts. One is binary classification model to determine whether the APP is malicious or not. Another is a multi-classification model, it can classify malware by category and malicious family. The detail process is described as follows:

1) TRAFFIC DATA COLLECTION

The traffic data were acquire from CICAndMal2017 [16].

CICAndMal2017 datasets collected 4,354 malware and 6,500 benign apps from VirusTotal [34], Contagio security blog [35] and previous researchers [16], [36] and [37]. Because of the sample errors and repeated label in different datasets, CICAndMal2017 retain 429 malware and 5,065 benign. Then, network traffic data are captured in three stages. Because of most advanced malware employed the evasion or transformation technique to dodge detection (i.e. code permutation, register renaming, idle activation) [2]. Some behaviors of malicious applications will be triggered only after connecting network update, other behavior of malware that only triggered over time after the restart process. In order to trigger all the malware behavior, network traffic data capture occurs in three minutes after the app is installed, and 15 minutes before and after restarting the phone.

2) NETWORK TRAFFIC DATA PREPROCESSING

This preprocessing contains three part. First, traffic segment. Second, traffic trim and generate training/testing datasets. Third, image generation and TFRecord transformation.

In the stage of malware traffic detection, because HTTP protocol is the most preferred protocol for most mobile APPs, TCP and UDP are also the most popular transport layer protocols for network, we handle HTTP, UDP and TCP connection as the main interaction granularity between the APP and the network. A TCP flow is a session which is defined as all packets that has the same 5-tuple (protocol, src_ip, src_port, dst_ip, dst_port) for which they contain traffic. A complete TCP flow begins with three-way handshake and ends in four-way waving. Therefore, flow is also the most basic unit in traffics. However, several flows generated by APPs contain comprehensive Meta information about all packets, and we are only interested in HTTP, TCP and UDP. Therefore, Wireshark is used first to filter the Pcap, then the Pcap was separated into the basic flow.

a: TRAFFIC SPLIT

The tool of USTC-TK2016 [38] is used to segment a PCAP into flows. The advantage of USTC-TK2016 is the ability to split a Pcap file based on flow, i.e. the frames from each TCP or UDP flow are placed in a separate Pcap file. All Pcap files are split into flows iteratively and saved into a folder named by the APP.

b: TRAFFIC TRIM AND GENERATE TRAINING/TESTING DATASETS

Traffic trimmed stage trims all flow files to 784 bytes. If the size of split Pcap file is beyond 784 byte, it is trimmed to 784 bytes. If it is shorter than 784 bytes, the file will be supplemented hexadecimal zero in the end to make the file size 784 bytes. Then all the trimmed Pcap files are divided into 80% training data and 20% testing data.

c: IMAGE GENERATION AND TFRecord TRANSFORMATION

The trimmed files in step (b) are converted to binary matrix, and then the binary matrix are converted to grayscale image in training data files and testing data files. Next step, the image path and corresponding labels are read into memory recursively, then labels and images are written into TFRecord files by 2-classifier, 4- classifier and 40- classifier.

3) CLASSIFICATION MODEL TRAINING AND TESTING

The TFRecord data are used to input into CACNN detection model to classify the malicious APPs from benign (2-classifier). The CACNN model could be used to classify the malicious APPs by category (4- classifier) and malicious family (40- classifier) too.

C. 2-CLASSIFIER, 4-CLASSIFIER AND 40-CLASSIFIER TRAINING

The model training mainly consists of two steps. One is compressing the data to obtain its two-dimensional features through deep convolutional Auto-Encoder model. Other is feeding the TFRecoder data into the convolutional Auto-Encoder cascading the convolutional neural networks to train the malware detection model.

1) CONVOLUTIONAL AUTO-ENCODER MODEL

After the traffic data preprocessing, all the traffic files are converted into TFRecord datasets. A convolutional Auto-Encoder (CAE) model is adapted to unlabeled images for testing whether the APP traffic data could cluster according to its lower-dimension feature. An Auto-Encoder is an unsupervised neural network where the input and the output have the same number of the nodes [39]. Hidden layers must be symmetric about center. And the number of nodes for hidden layers must decrease from left to centroid (encoder), and must increase from centroid to right (decoder). Convolutional Neural Network (CNN) is a special type of neural network. It contains a feature extractor composed of convolutional layer, activation layer and sub-sampling (pooling) layer. CNN is used to detect some patterns first, and feed these patterns to neural networks. In this way, it can process images in less complex way whereas get more successful results. However, it is tested for labeled supervised learning problems. As for unlabeled images for clustering, convolutional auto-encoder model is needed. The CAE model is similar to Auto-Encoder, Auto-Encoder network design is symmetric about centroid. The number of nodes reduce from left to centroid, and increase from centroid to right. Centroid layer would be compressed representation. It will apply the same procedure for CNN.

Convolutional layer - convolution: k convolution kernels (W) are initialized, each convolution kernel is matched with a bias b, and then it convolved with the input x to produce k characteristic graphs h. The formula is provided in (1)

$$h^k = \sigma(x^* W^k + b^k) \tag{1}$$

where the bias is broadcasted to the whole map in the Eq.1, the activation function σ is ReLU. * denotes the 2D convolution [4].

Convolutional layer – autoencoder: Convolution operation is performed in each feature graph h and its corresponding transposed convolution kernel. The formula is provided in (2)

$$y = \sigma(\sum_{k \in H} h^k * \tilde{W}^k + c) \tag{2}$$

where the convolution operation summed the result and add the bias c, $\tilde{W}$ is the transposition of W, the activation function σ is ReLU. * denotes the 2D convolution.

The cost function to minimize is the cross entropy which is portrayed in Eq. 3

$$H_{\theta'}(\theta) = -\sum_i y_i' log(y_i) \tag{3}$$

where θ is a function of W and b, y_i is the result of prediction. The weights are then updated using Adadelta.

It can call left to centroid side as convolution whereas centroid to right side as deconvolution. Deconvolution side is also known as unsampling or transpose convolution. The CAE is applied for unsupervised learning where partial APP traffic image data were used as unlabeled data training. APP traffic images were input into the model, and then compress it into two dimensions clustering data from left to centroid side. Then, deconvolution side is applied to reconstruct the original traffic image data. Finally, the weights are stored in the HDF5 file as the input of next model.

2) CACNN MODEL IS USED TO CLASSIFY MALWARE

CACNN model is an enhanced CNN which cascades CAE and CNN. The compressed clustering data were acquired from left to centroid side in step 1 as the input of the following classifier for supervised learning. CNN used for the supervised learning where part of traffic images data were used as labeled data and trained this model. Then the output data is converted into a 1D pixels array and input into the three layers of fully connected neural network. Finally, a softmax loss function which is portrayed in Eq. 4

$$softmax(y_j) = \frac{e^{y_j}}{\sum_{k=1}^{K} e^{y_k}} \quad for\ j = 1, 2, 3...k \tag{4}$$

is used to separate the malicious APPs from the datasets. k is the number of categories. k is 2 when the binary classification (2-classifier) is used to classify the malware and benign, k is 4 when the malwares are classified by category (4-classifier), and k is 40 when the malicious APPs are classified by malware family (40-classifier). y_i is the classification probability of it belongs to. The schematic of the CACNN is depicted in Figure 2.

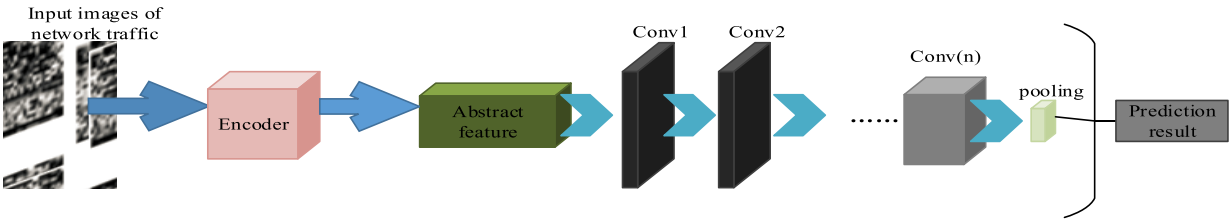


FIGURE 2. Architecture of CACNN.

IV. EXPERIMENT

We implement the proposed method using Keras, the Python Deep Learning library with Tensorflow backend. In this section, a detailed description of the dataset and experimented environment is provided first in section 5.1. Second, the data preprocessing is described in section 5.2. Third, the performance metrics evaluated in our experiment is recalled in section 5.3. Then, the number of training samples for the detection model were analyzed to achieve good detection result. Finally, our method was compared with other deep learning methods, static detection methods and traffic flow detection methods.

A. DATASETS

For the evaluation of our model, 5065 benign APPs and 4354 malware samples were used from CICAndMal2017 dataset [16]. The benign APPs were collected from Google play market published in 2015, 2016 and 2017. These APPs were collected based on their popularity and identified based on the detection results from VirusTotal [34]. Only those APPs that VirusTotal determined as benign are included to the benign APP set. Ultimately, 5065 of them were reserved as benign APPs and 4354 of them were reserved as malware APPs. And 1700 benign and 429 malware network traffics were captured in the installation of APPs, before restart and after restart. All the malware have four categories, they are adware, ransomware, scareware and SMS malware. Each categories have different family. Adware has Dowgin, Ewind, Feiwo, Gooligan, Kemoge, koodous, Mobidash, Selfmite, Shuanet and Youmi family. 440 malwares were collected, and 104 traffics were captured in this family. Ransomware has Charger, Jisut, Koler, LockerPin, Simplocker, Pletor, PornDroid, RansomBO, Svpeng and WannaLocker family. 1094 malwares were collected totally, and 101 traffics were captured in this family. Scareware has AndroidDefender, AndroidSpy.277, AV for Android, AVpass, FakeApp, FakeApp.AL, FakeAV, FakeJobOffer, FakeTaoBao, Penetho and VirusShield family. 1442 malwares were collected totally, and 112 traffics were captured in this family. SMS-malware has BeanBot, Biige, FakeInst, FakeMart, FakeNotify, Jifake, Mazarbot, Nandrobox, Plankton, SMSsniffer and Zsone family. 1269 malwares were collected, and 92 traffics were captured in this family.

B. DATA PREPROCESSING

After obtaining all the APPs and its network traffic data. The static features are extracted first. 8115 features of the permissions and intent actions were acquired from Manifest-File.xml of APK, and saved them in a CVS file. Then, the traffic network images feature data are produced which is introduced in methodology and saved as TFRecord files. The size of each grayscale image is 784 bytes. The visualization results are showed in Figure 3 and Figure 4.

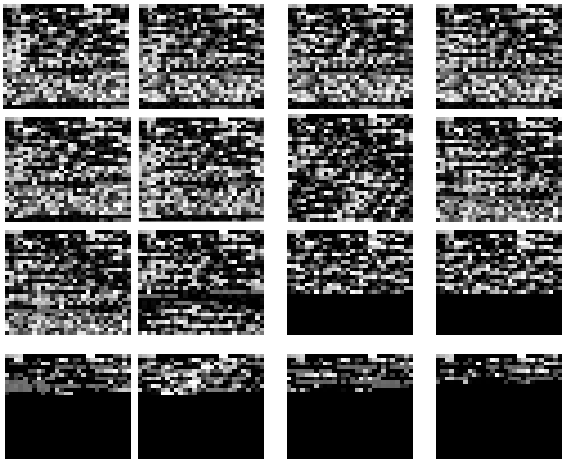


FIGURE 3. Benign traffic images.

Figure 3 shows the extraction sample images of benign traffic and Figure 4 shows the extraction sample images of malicious traffics. Meanwhile, Figure 4 is also shown the characteristic of different category of malicious traffic. The visualization result of all the network traffic shows that there are some different between benign traffic images and malware traffic images, and it is obvious that there are some different among malicious category traffic images data.

After obtaining the datasets and preprocessing the data, the final data statistics are shown in table 2. According to the characteristic of CVS file and the network traffic of APPs, it is considered to use fully connected neural network to detect malicious APPs based on permissions and intents in CVS, and use CACNN model to distinct different network traffic of APPs.

C. PERFORMANCE METRICS

As for a binary classifier of the detection model, there are four circumstance, if the APP is benign and is predicted to be a benign, it is a True Positive Rate (TPR); if the APP is malicious and is predicted to be a benign, and it is called False Positive Rate (FPR). Accordingly, if the APP is a malware and is predicted to be a malicious APP, it is called a true

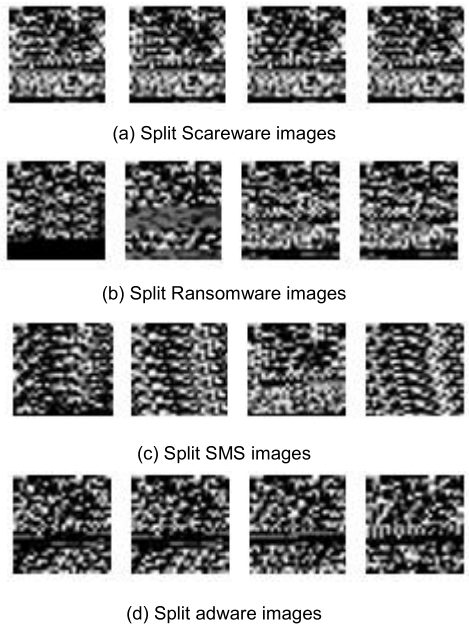


FIGURE 4. Malware traffic images.

TABLE 2. Dataset statistics.

items	Benign	Malware
APP numbers	5065	4354
The APPs which get its traffic	1700	429
Collected traffic data	18.5G	19.0G
Permissions and intents in CVS	8115	
Image number of training flow	241607	382373
Testing flow image data	500M	842M
Image number of test flow	26879	42492
Testing flow image data	55.2M	94.3M

Negative Rate (TNR), and if the benign is predicted to be a malware, it is False Negative Rate (FPR). A good detection model should identify as many malware APPs as possible and minimize the False Positive Rate. However, in multi-classifier and imbalanced problems, the detection rate or FPR cannot determine whether a model is good or not. And there are usually other performance metrics were used: accuracy (ACC), precision (P), recall (R) and f-measure value. Accuracy was used to evaluate the overall performance of the detection model. The formula portrayed in Eq. 5.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{5}$$

Precision is used to evaluate the rate of the number of true positive APPs to the number of predicted positive APPs. The formula portrayed in Eq. 6.

$$P = \frac{TP}{TP + FP} \tag{6}$$

Recall indicated the rate of the true positive APPs to the total number of APPs. The formula portrayed in Eq. 7.

$$R = \frac{TP}{TP + FN} \tag{7}$$

F-measure value is a relatively fair metric to understand and compare the performance of malware detection [41], it considers both the precision P and the recall R to compute the score. Due to our malware recognition is actually an imbalanced classification datasets (our dataset has 429 malicious samples and 1700 benign samples). F-measure comprehensively considers the detection rate and error rate of identifying malicious samples, and therefore is suitable for model evaluation. The formula is provided as follow:

$$F = \frac{(\alpha^2 + 1)P^*R}{\alpha^2(P + R)} \tag{8}$$

The formulas is F1-measure, when $\alpha = 1$.

To verify the accuracy and precision of the first static analysis layer, the fully connected neural network model was used to extracted datasets as input, there are 8115 features of permission and intent in the datasets. First, all the features were input into the model to observe the values of evaluation parameters, then the number of features were gradually decreased which is not important to the datasets, for instance, we delete the total number of APPs occupied in a feature is less than or equal to 1. The threshold was set in 1, 5, 10 and 20 respectively. The values of evaluation parameters are showed as follows in Figure 5.

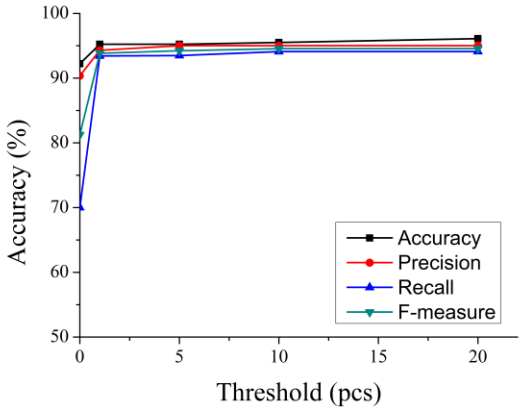
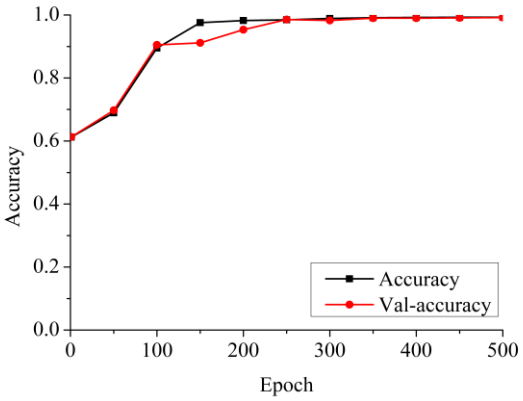
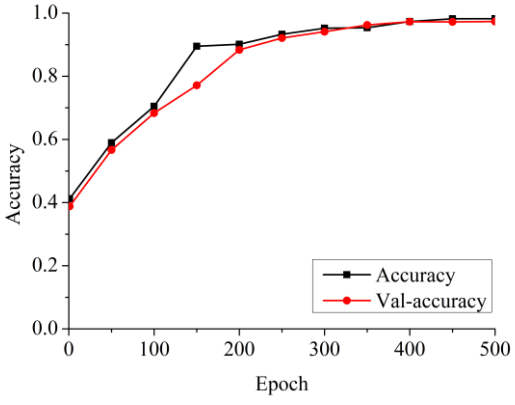


FIGURE 5. Evaluation value of static analysis layer.

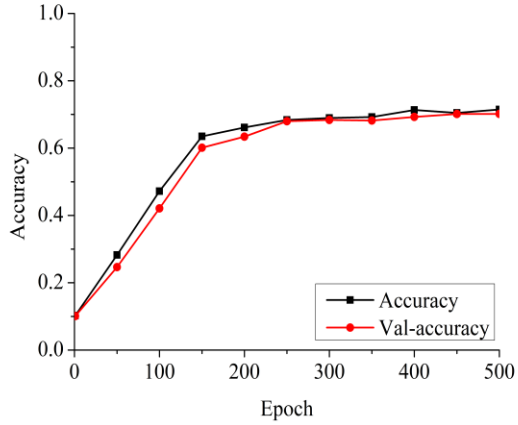
When all the datasets were input into the model, the accuracy is 0.9222, precision is 0.9035, and recall and F-measure are 0.7 and 0.8125 respectively. When the total number of APPs occupied in a feature is less than or equal to 1, it will be deleted. The accuracy is 0.9523, precision is 0.943, and recall and F-measure are 0.9348 and 0.9383. The precession increased dramatically because the features which less than or equal to 1 are definitely infrequent features and can be deleted in the feature datasets. And when the deleted features are less than or equal to 20. The accuracy is 0.961, precision is 0.9523, and recall an F-measure are 0.9411 and 0.9454 respectively. The performance metrics change stably, however the features we delete are 2423 which will be loss correction too much. So the threshold we select is 1 and delete the total number of APPs occupied in a feature is less than or equal to 1.



(a) Training and Test accuracy of 2-classifier

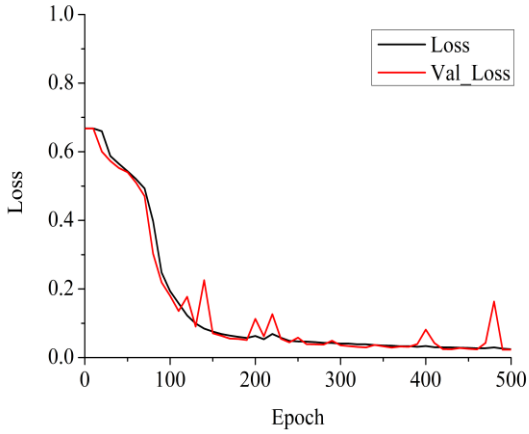


(b) Training and Test accuracy of 4-classifier

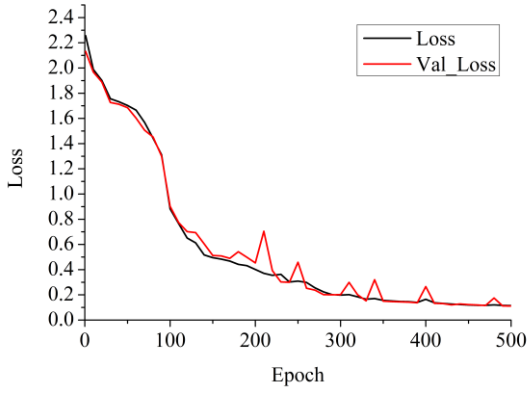


(c) Training and Test accuracy of 40-classifier

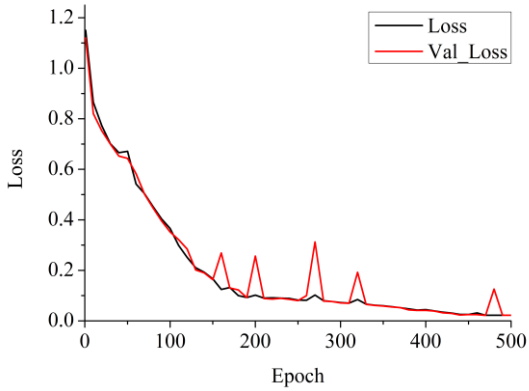
FIGURE 6. The accuracy of CACNN.



(a) Loss of 2-classifier



(b) Loss of 4-classifier



(c) Loss of 40-classifier

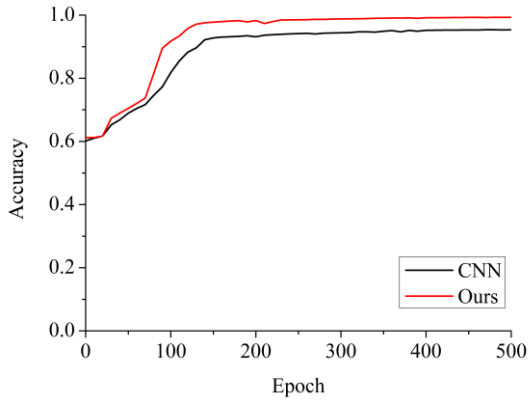
FIGURE 7. Loss in the classifiers.

Though the accuracy is 0.9522 when the threshold is 1, the false positive rate and false negative rate are 0.05488 and 0.25 respectively. This means that there is a one-fourth probability that the malware will be predicted as benign one, whereas the FPR is low enough that it rarely predict malware as benign one. Therefore, the next step detection is very important. All the benign APPs that the first model detected will be input into the CACNN model.

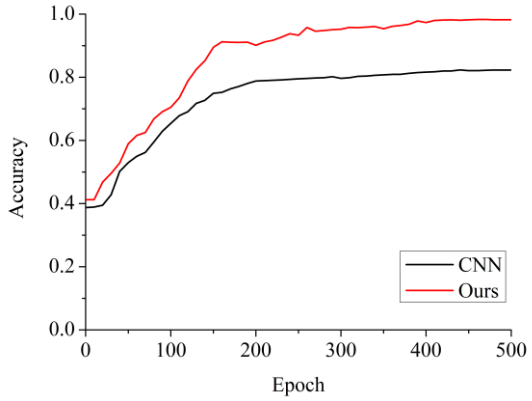
The second layer obtain network traffic data of APPs first, then the datasets will be classified into malware and benign after data preprocessing. Next, the input malware samples were classified into four malware categories (Adware, Ransomware, SMSMalware, Scareware) and 40 malware families(koler, svpeng, lockerpin, jisut, simplocker, wannalocker, charger, RansomBO, pletor, porndroid, mobidash, gooligan, dowgin, youmi, kemoge, feiwo, shuanet, ewind, selfmite, avpass, AndroidDefender, virusshield, android.spy.277, fakeapp, fakejoboffer, FakeApp.AL, fakeav, penetho, FakeTaoBao, AvForAndroid, mazarbot, fakemart, beanbot, jifake, zsone, fakeinst, smssniffer, nandrobox, biige, plankton).

D. CLASSIFICATION EVALUATION

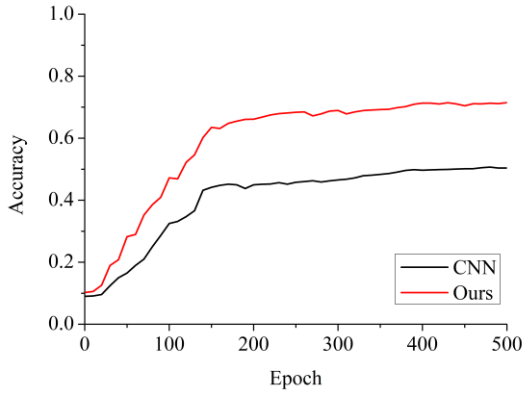
After data preprocessing in step B, there are 623980 training samples and 69371 test samples. Three experiments



(a) Comparison of our 2-classifier with CNN

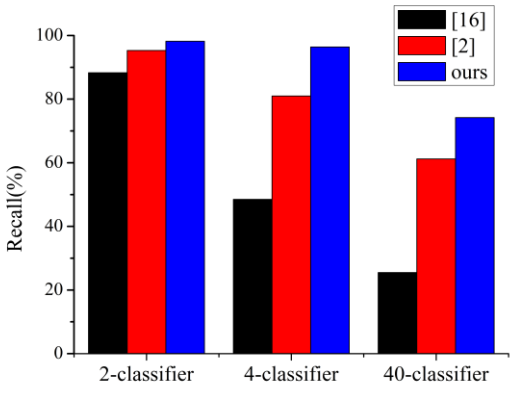


(b) Comparison of our 4-classifier with CNN

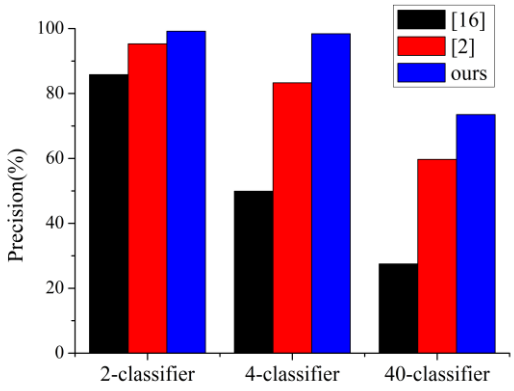


(c) Comparison of our 40-classifier with CNN

FIGURE 8. Comparisons of our method with CNN.



(a) Comparison of precision and recall



(b) Comparison of precision and recall

FIGURE 9. Comparison of our method with [16] and [2].

were carried out to test the accuracy of binary classification (2-classifier), category classification (4-classifier) and malicious family classification (40-classifier). The experiment results were showed as follows.

From Figure 6 (a), we can see that CACNN achieves an impressive 99.3% accuracy in training samples and 99.19% accuracy in testing samples by binary classification (2-classifier), and 98.22% accuracy in training samples and 97.3% accuracy in testing samples by category (4-classifier) in Figure 6 (b), and 71.48% accuracy in training samples and 70.16% accuracy in testing samples by malicious family (40-classifier) in Figure 6 (c). It is found that the effect of the classifier perform well in 2-classifer and 4-classifier, but it perform averagely in 40-classifier. This is because the images are similar in the same category but a little more different in malicious family. In future work, we should find ways to improve the accuracy of the classifier in family malware detection.

The experiment results of training and testing loss in binary classification (2-classifier), category classification (4-classifier) and malicious family classification (40-classifier) are showed in Figure 7.

Figure 7 depicts the training and testing value of loss, we can see that the loss of the classifier in figure 7 (a) is from 0.6681 to 0.0245 in training sample and from 0.6675 to 0.0231 in test sample respectively. The loss of the classifier in figure 7 (b) is from 1.1502 to 0.0221 in training sample and from 1.1201 to 0.0226 in test sample respectively. And the loss of the classifier in figure 7 (c) is from 2.256 to 0.115 in training sample and from 2.131 to 0.1113 in test sample respectively. From Figure 7, it is found that the classifier perform smoothly after 300 loop. And the test sample fluctuates more than the training sample because the test sample has a smaller amount of data.

Figure 8 is the accuracy comparisons of CNN and our method in binary classification (2-classifier), category classification (4-classifier) and malicious family classification (40-classifier).

From Figure 8 (a), the accuracy curves of CNN and our method are close in binary classification. And there are

obvious difference in Figure 8 (b) and Figure 8 (c). That means CNN and our method all have excellent performance in accuracy of binary classification, the two classifiers perform smoothly after 150 loops. However our method performs better than CNN in accuracy of multi-classification. In 4-classifier, our method reached 98.22%, and CNN only reached 82.26% after 500 loops. Our method and CNN reached 70.14% and 50.38% in 40-classifier respectively.

Figure 9 is the comparison of precision and recall in our method, [16] and [2] in CICAndMal2017 dataset.

As seen from the Figure 9, the precisions of our method is 99.2% in 2-classifier, 98.4% in 4-classifier and 73.5% in 40-classifier respectively. The results are 95.3%, 83.3% and 59.7% in [2] and 85.8%, 49.9% and 27.5% in [16]. The recalls are also increased into 98.2% in 2-classifer, 96.4% in 4-classifer and 74.2% in 40-classifer in our method. And the recalls are 95.3%, 81% and 61.2% in [2] and 88.3%, 48.5% and 25.5% in [16] respectively. As we can see from the experiments result, our method express better in multi-classification in precision and recall.

V. CONCLUSION

In this research, a two-layer Android malware detection model is presented. The first layer is permission, intent and component information based static malware detection model. The static features were combined with fully connected neural network to detect the malware, and the effectiveness was tested through experiments. Then, the results (benign APPs) were input into next layer. In the second layer, a new cascading deep learning CACNN method was used to detect network traffic features of APPs. The experimental results show that our methods can effectively identify malicious Android APPs. Moreover, the new two-layer model can also detect malware by its category and malicious family. Overall, combining two levels of detection work can further improve the detection efficiency.

REFERENCES

[1] (2019). *Ericsson Mobili Ty Report*. [Online]. Available: https://www.ericsson.com/en/mobility-report/reports

[2] L. Taheri, A. F. A. Kadir, and A. H. Lashkari, "Extensible Android malware detection and family classification using network-flows and API-calls," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.

[3] [Online]. Available: http://www.360doc.com/content/20/0122/20/33989007_887496741.shtml

[4] [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[5] C. Lueg, "8,400 new Android malware samples every day," G Data, Bochum, Germany, Tech. Rep., Apr. 2014. [Online]. Available: https://www.gdatasoftware.com/blog/2017/04/29712-8-400-newandroid-malware-samples-every-day

[6] A. Arora, S. Garg, and S. K. Peddoju, "Malware detection using network traffic analysis in Android based mobile devices," in *Proc. 8th Int. Conf. Next Gener. Mobile Apps, Services Technol.*, Sep. 2014, pp. 66–71.

[7] C. Urcuqui-López and A. Navarro Cadavid, "Framework for malware analysis in Android," *Sistemas y Telemática*, vol. 14, no. 37, pp. 45–56, Aug. 2016.

[8] N. Peiravian and X. Zhu, "Machine learning for Android malware detection using permission and API calls," in *Proc. IEEE 25th Int. Conf. Tools Artif. Intell.*, Nov. 2013, pp. 300–305.

[9] L. Onwuzurike, E. Mariconti, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "MaMaDroid: Detecting Android malware by building Markov chains of behavioral models (extended version)," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 14.1–14.34, 2019.

[10] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "HinDroid: An intelligent Android malware detection system based on structured heterogeneous information network," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2017, pp. 1507–1515.

[11] H. Kang, J.-W. Jang, A. Mohaisen, and H. K. Kim, "Detecting and classifying Android malware using static analysis along with creator information," *Int. J. Distrib. Sensor Netw.*, vol. 11, Jun. 2015, Art. no. 479174.

[12] Y. Zhang, M. Yang, B. Xu, Z. Yang, G. Gu, P. Ning, X. S. Wang, and B. Zang, "Vetting undesirable behaviors in Android apps with permission use analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2013, pp. 611–622.

[13] L. Sun, Z. Li, Q. Yan, W. Srisa-an, and Y. Pan, "SigPID: Significant permission identification for Android malware detection," in *Proc. 11th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2016, pp. 1–8.

[14] X. Wang, K. Sun, Y. Wang, and J. Jing, "DeepDroid: Dynamically enforcing enterprise policy on Android devices," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–15.

[15] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning," *Digit. Invest.*, vol. 24, pp. S48–S59, Mar. 2018.

[16] 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.

[17] B. Amos, H. Turner, and J. White, "Applying machine learning classifiers to dynamic Android malware detection at scale," in *Proc. 9th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jul. 2013, pp. 1666–1671.

[18] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker: Scalable and accurate zero-day Android malware detection," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2012, pp. 281–294.

[19] V. Rastogi, Y. Chen, and W. Enck, "AppsPlayground: Automatic security analysis of smartphone applications," in *Proc. 3rd ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, 2013, pp. 209–220.

[20] J.-W. Jang, J. Yun, J. Woo, and H. K. Kim, "Andro-profiler: Anti-malware system based on behavior profiling of mobile malware," in *Proc. 23rd Int. Conf. World Wide Web-WWW Companion*, 2014, pp. 737–738.

[21] A. Shabtai, U. Kanonov, and Y. Elovici, "Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method," *J. Syst. Softw.*, vol. 83, no. 8, pp. 1524–1537, Aug. 2010.

[22] R. Vinayakumar, K. P. Soman, P. Poornachandran, and S. Sachin Kumar, "Detecting Android malware using long short-term memory (LSTM)," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1277–1288, Mar. 2018, doi: 10.3233/JIFS-169424.

[23] (Oct. 25, 2016). *Malware Detection Methods*. [Online]. Available: http://www.avg.com/us-en/avg-software-technology

[24] D. Iland, A. Pucher, and T. Schauble, "Detecting Android malware on network level," Univ. California, Santa Barbara, CA, USA, 2011, vol. 12.

[25] L. Tenenboim-Chekina, O. Barad, A. Shabtai, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Detecting application update attack on mobile devices through network featur," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 91–92.

[26] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "NetworkProfiler: Towards automatic fingerprinting of Android apps," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 809–817.

[27] M. Zaman, T. Siddiqui, M. R. Amin, and M. S. Hossain, "Malware detection in Android by network traffic analysis," in *Proc. Int. Conf. Netw. Syst. Secur. (NSysS)*, Jan. 2015, pp. 1–5.

[28] A. Arora and S. K. Peddoju, "Minimizing network traffic features for Android mobile malware detection," in *Proc. 18th Int. Conf. Distrib. Comput. Netw. (ICDCN)*. New York, NY, USA: ACM, 2017, pp. 1–10.

[29] 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, doi: 10.1109/PST.2017.00035.

[30] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Multi-classification approaches for classifying mobile app traffic," *J. Netw. Comput. Appl.*, vol. 103, pp. 131–145, Feb. 2018.

[31] S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, and Z. Jia, "TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Service (IWQoS)*, Jun. 2016, pp. 1–6.
[32] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, "A mobile malware detection method using behavior features in network traffic," *J. Netw. Comput. Appl.*, vol. 133, pp. 15–25, May 2019.
[33] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "PScout: Analyzing the Android permission specification," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2012, pp. 217–228.
[34] *Virustotal.URL*. [Online]. Available: https://www.virustotal.com/
[35] V Total. (2016). *Contagio Mobile Malware Mini Dump*. [Online]. Available: http://contagiominidump.blogspot.ca/
[36] A. F. A. Kadir, N. Stakhanova, and A. A. Ghorbani, "Android botnets: What urls are telling us," in *Proc. Int. Conf. Netw. Syst. Secur.* Springer, 2015, pp. 78–91.
[37] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Droidkin: Lightweight detection of Android apps similarity," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Springer, 2014, pp. 436–453.
[38] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Da Nang, Vietnam, 2017, pp. 712–717.
[39] P. Baldi and Z. Lu, "Complex-valued autoencoders," *Neural Netw.*, vol. 33, pp. 136–147, Sep. 2012.
[40] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. 21st Int. Conf. Artif. Neural Netw. Artif. Neural Netw. Mach. Learn. (ICANN)*. Espoo, Finland: Springer-Verlag, Jun. 2011, pp. 52–59.
[41] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discriminative model for Android malware detection with decompiled source code," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 4, pp. 400–412, Jul. 2015.
[42] Y. Liu, K. Guo, X. Huang, Z. Zhou, and Y. Zhang, "Detecting Android malwares with high-efficient hybrid analyzing methods," *Mobile Inf. Syst.*, vol. 2018, pp. 1–12, Mar. 2018.
[43] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The dark side(-channel) of mobile devices: A survey on network traffic analysis," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2658–2713, 4th Quart., 2018.

JIAYIN FENG was born in Hebei, China. She received the B.S. degree in computer science and the M.S. degree in computer application science from Yanshan University, Qinhuangdao, China, in 2005 and 2008, respectively, where she is currently pursuing the Ph.D. degree. She has more than ten years of teaching experience with the Department of Computer Science, Hebei Normal University of Science and Technology. Her current research interests include mobile network security, deep learning, and information security.

LIMIN SHEN (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from Yanshan University, China. He is a Professor and a Ph.D. Supervisor with the College of Computer Science and Engineering, Yanshan University. His main research interests include service computing, collaborative computing, and cooperative defense.

ZHEN CHEN received the B.S. and Ph.D. degrees in computer application technology from Yanshan University, China, in 2010 and 2017, respectively. He is an Associate Professor with the College of Computer Science and Engineering, Yanshan University. He is currently working on service computing, cloud computing, and collaborative computing.

YUYING WANG received the M.S. degree from the Institute of Electrical Engineering, Yanshan University, Qinhuangdao, China, in 2018, where she is currently pursuing the Ph.D. degree in information science and engineering. Her current research interests include software formal methods and information security.

HUI LI is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Yanshan University. Her research interests include information security, mobile application security, and mobile information systems.