LSTM Android Malicious Behavior Analysis Based on Feature Weighting

Qing Yang[1,2], Xiaoliang Wang[2,3*], Jing Zheng[2], Wenqi Ge[2], Ming Bai[1], Frank Jiang[2]
[1] School of Computer Science and Information Engineering, Guangzhou Maritime University,guangzhou ,510725,China
[2] Hunan University of Science and Technology, Xiangtan 411201, China
[3] School of Info Technology, Deakin University, Geelong, Australia
[email : fengwxl@hnust.edu.cn]
[*]Corresponding author: Xiaoliang Wang

Received March 26, 2021; revised May 5, 2021; accepted May 23, 2021; published June 30, 2021

Abstract

With the rapid development of mobile Internet, smart phones have been widely popularized, among which Android platform dominates. Due to it is open source, malware on the Android platform is rampant. In order to improve the efficiency of malware detection, this paper proposes deep learning Android malicious detection system based on behavior features. First of all, the detection system adopts the static analysis method to extract different types of behavior features from Android applications, and extract sensitive behavior features through Term frequency-inverse Document Frequency algorithm for each extracted behavior feature to construct detection features through unified abstract expression. Secondly, Long Short-Term Memory neural network model is established to select and learn from the extracted attributes and the learned attributes are used to detect Android malicious applications, Analysis and further optimization of the application behavior parameters, so as to build a deep learning Android malicious detection method based on feature analysis. We use different types of features to evaluate our method and compare it with various machine learning-based methods. Study shows that it outperforms most existing machine learning based approaches and detects 95.31% of the malware.

Keywords: Android Security, Deep Learning, LSTM, Static Analysis, TF-IDF.

This research was supported by a research grant from the Cooperative Education Fund of China Ministry of Education [201702113002, 201801193119] and the Scientific Research Fund of Hunan Provincial Education Department [20A191].

http://doi.org/10.3837/tiis.2021.06.014 **ISSN : 1976-7277**

1. Introduction

By the end of 2018, the market share of Android system in the leading position among global mobile terminals has reached 85% [1]. Due to the high openness and extensibility of Android system, malware makers can easily develop a large number of malicious software, making Android mobile terminals more vulnerable to malware attacks [2]. About 97% of mobile malware is developed for Android mobile terminals [3]. In order to prevent and mitigate the harm of malware, this paper studies the security of Android mobile applications.

In order to protect users' privacy [4], researchers at home and abroad have proposed various methods. These methods can be divided into static analysis and dynamic analysis [5]. Static analysis is to extract the corresponding characteristic information from the source file of the program for analysis [6]. The detection process is relatively simple. The overall analysis of the program from the global perspective can quickly and efficiently analyze the malware [7], but the application cannot be dynamically loaded and the intrusion detection cannot be suddenly found when the application is running. In dynamic analysis, the behavior features extracted by the application during the running time are compared with malicious samples [8]. In this process, a lot of overhead will be generated and the detection efficiency is low. It cannot detect the current massive amount of malicious software quickly and effectively. Considering that the number of malicious applications is increasing rapidly, this paper adopts the static analysis method to analyze the applications on a large scale.

In recent years, people commonly use machine learning methods to detect malware, such as support vector machines (SVM) and Naive Bayes, etc [9]. Moreover, in the field of image recognition and voice recognition, deep learning has shown the optimal detection performance [10], which has been widely concerned by researchers. Extracting a large number of behavioral features from software will contain irrelevant and noisy behavioral features, which will have a negative impact on malware detection [11]. we need to screen the extracted behavioral features, remove noise and irrelevant data [12] in the data set, and improve the performance of the classifier [13]. All in all, researchers have proposed many detection methods based on deep learning [14], which only extract feature information from applications, and insufficient analysis of the extracted features reduces the accuracy of application detection. It is also considered that the use of dynamic analysis causes additional overhead in the detection process, such as increasing the time for training the model, occupying a large amount of computing resources, etc., and it is impossible to quickly and effectively detect unknown applications. So, this paper proposes the detection method of deep learning Android malware based on feature analysis. This paper uses static analysis technology to propose the research and design of a deep learning Android malicious detection system based on feature weighting. On the one hand, it eliminates redundant features that are not related to detection, and enhances the ability to distinguish malicious applications to reduce additional overhead. On the other hand, it conducts comprehensive detection and analysis of applications, automatically digs deep features, and analyzes the information between features to quickly and effectively detect Android applications more accurately. The study shows that our method has better performance than most existing detection methods, and can detect 95.31% malware with 93.32% accuracy.

In summary, this paper makes the following contributions：

- Combining the TF-IDF algorithm, consider the importance of each behavior feature between malicious and benign applications, and generally consider the degree of association of each behavior feature to malicious or benign applications. Reduce the weights that evenly

exist in the behavior characteristics of malicious and benign applications, and enhance the behavior characteristics specific to malicious and benign applications.
- We built LSTM models with different network structures to detect Android malware, and selected the model structure with the best detection effect to detect Android malware through experimental analysis.
- We assessed our method with 95.31% accuracy using different types of behavioral characteristics and compared them with machine learning methods.

The rest of this paper is organized as follows: Related work is introduced in Section 2. Approach we proposed is presented in Section 3. Evaluation and comparison with related approaches are presented in Section 4. Section 5 concludes the paper.

2. Related Work

In the past years, Android malware detection has been a research area of concern. In order to protect personal information from being leaked, many new detection technologies have been proposed. At present, researchers at home and abroad mainly detect Android malicious applications from static analysis and dynamic analysis [15].

Detection method based on static analysis, which carries out static analysis on the code extracted from Android application through reverse engineering [16]. AppFA [17] evaluates the network behavior of the application and similar applications by selecting similar applications and setting a threshold. It can only roughly discriminate the detection applications, and how to choose appropriate similar applications is also a challenge. APK Auditor [18] is only grading the application based on authority to assess the threat level of the application, but the grading basis is too simple, and there is no effective feature learning method for mass behavioral features. Dini G et al. propose a reliability evaluator [19], which uses the analytic hierarchy process (AHP) to make multi-standard decision combinations for the application, without the need to analyze the code to greatly reduce its complexity. However, only a single type of attribute is selected to detect Android malware, which cannot comprehensively reflect the behavior features of malicious applications and only provides a reference for users. DroidDet [20] adopts a combination of multiple types of features and machine learning to conduct classification and detection of malware through cross validation, which reduces the cost of detection of malware, but cannot automatically select and learn behavioral features. Karina et al. [21] construct the permission map be required by the software and evaluated it according to the performance index of the central permission build in the permission map. This method cannot effectively evaluate unknown malicious applications accurately.

Detection method based on dynamic analysis. This method extracts various dynamic behavior features for analysis at application runtime. Rehman et al. [22] make a dynamic analysis on the basis of Sato et al. [23] research and combined it with machine learning algorithm to detect the application. However, it increased the extra cost and occupied a lot of resources to detect Shabtai et al. [24] analyze applications on the PC side, obtain various features and events from mobile devices, and combine with classification algorithm to detect unknown applications. However, detection is inefficient and requires considerable time to collect resources. TaintDroid [25] uses the private data as the source of pollution to analyze the private data with dynamic stains, and tracks multiple private data sources at the same time. However, during the detection process, a large amount of resources are occupied by the phone, which cannot provide an accurate security assessment. MalAware [26] dynamically monitors the memory and CPU usage of the application, and adopts Logistic Regression algorithm to

detect malicious softwares.

According to the current research situation, the traditional detection methods are all shallow structures, which do not have the ability to automatically select and learn massive behavioral features. However, Deep Neural Networks has shown advanced performance in image recognition and other fields. Therefore, the application of deep learning detection has been studied. Hasegawa et al. [27] use one-dimensional CNN to analyze a small part of information of APK to reduce the resource overhead of detecting malware. DeepClassifyDroid [28] uses convolutional neural network to classify and detect the feature set extracted from static analysis, which increased the cost of detection application and affected the effect of detection. This model greatly increases the cost of resources.

In recent literature, a lot of deep learning has been applied to identifying vulnerable code snippets. However, the proposed studies were evaluated based on self-built/collected data sets, extracting only feature information from the application and not fully analyzing the extracted features, thus reducing the accuracy of application detection [29]. If dynamic analysis is used to extract behavioral features, it will result in extra overhead in the detection process, and it cannot quickly and effectively detect unknown applications. This paper uses static analysis technology to propose a deep learning Android malicious detection system based on behavior characteristics. On the one hand, TF-IDF algorithm is used to analyze the extracted massive features, remove redundant and irrelevant behavior features, enhance the distinguishing ability of Android malicious and improve the accuracy of application detection. On the other hand, deep learning is used to transform the behavioral feature representation of application system calls into a new feature space for comprehensive detection and analysis of applications. It can automatically dig deep features and analyze information between features to detect android applications quickly and effectively, making the detection more accurate.

3. Proposed Approach

We design and implement a deep learning Android malicious detection method based on feature analysis. As shown in **Fig. 1**, the method is composed of four different modules:

1. Feature Extraction Module: a wide range of static analysis of applications, from which different types of behavioral features are extracted as feature sets through reverse engineering.
2. Feature Analysis Module: In order to eliminate the redundancy and irrelevance of behavioral features and improve the detection accuracy of malicious applications, we use the TF-IDF algorithm to calculate the importance of each feature to application detection, and select and reconstruct the feature vector.
3. Feature Embedding Module: select features from feature sets of different dimensions according to the weight to form a joint feature set and map it to a joint vector space.
4. Detection Module: the traditional detection model is lack of effective learning ability of behavior features, and cannot effectively analyze a large number of behavior features. Therefore, we use Long Short-Term Memory neural network (LSTM) to detect and classify malware after learning the essential behavior features.

In the next four sections, we will discuss the details of these modules and provide the necessary background information. The Android malicious detection framework is shown in **Fig. 1**.

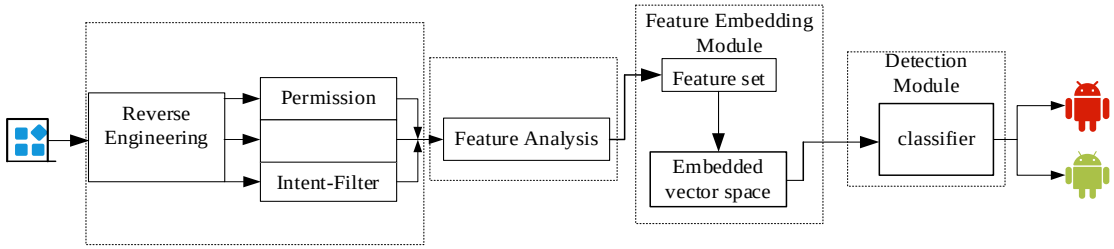

Fig. 1. Android malicious detection framework

3.1 Feature Extraction Module

In order to accurately detect Android malicious applications, this paper extracts different types of behavior features from Android applications, decompiles the installation files of Android applications using APKTOOL to generate the AndroidManifest.xml file, and extracts the static behavior features data of Android applications as follows:

Permission: Permission is one of the most important security mechanisms for Android applications. When installing an application, the user selects and authorizes it. Permission will provide Android applications with access to different types of security related resources and data. For example, if an application requests READ_CONTACTS and SEND_SMS permissions, this could mean that the application will send SMS to an attacker when it gets your communication information, which leads to a privacy breach. Current research shows that Android malicious applications will request more permissions than normal applications to obtain more resources and data. This paper will extract the permission list defined in Android manifest.xml as the behavior features of analysis.

Component of Application: There are four main components of Android application, that is Activity, Service, Content Provider and Broadcast Receiver. Each Android application can declare multiple different types of components in the Android manifest.xml file. Some Android malicious applications will quietly run some service processes in the background of the system through the service components to perform malicious behaviors. In this paper, we extract these component information as the behavior features of Android applications.

Intent-Filter: The Android system matches the Intent Filter configured by the Android application. It considers only three categories: Actions, Data, and Categories. To find the component or service that responds to the intent. We collect all the actions and categories in the manifest as feature sets, and Android applications declare the corresponding actions and categories in the androidmanifest.xml or source code. This mechanism makes it easy for Android malicious applications to monitor certain system events and launch malicious activities directly after the mobile terminal is restarted. We extract all these Intent-Filter as one of our feature sets.

3.2 Feature Analysis Module

TF-IDF is a commonly used statistical algorithm that is often used to assess the importance of a word to one of the documents in a document library. So we will use the static analysis method to extract different types of behavior features from the manifest file, including (Permission, Components, and Intent-Filter), and then extract to each behavioral feature is analyzed. The weight of each behavior feature is calculated by TF-IDF, and the importance of each behavioral feature to malware detection is evaluated to screen the features. The specific description is as follows:

We divided the collected application set into malicious application set M and benign application set B . Let N_i be the number of times each behavior trait i is invoked for application set M or B . N is the total number of times that application for application set B or M invokes all behavioral features. Then TF_i represents the frequency of invocation of a behavior feature in application, and its definition formula is as follows:

$$TF_{(i,M)} = \frac{N_{(i,M)}}{N_M} \tag{1}$$

$$TF_{(i,B)} = \frac{N_{(i,B)}}{N_B} \tag{2}$$

We define D_M and D_B as the total number of applications in the M and B training set respectively, $D_{(i,M)}$ and $D_{(i,B)}$ are the invocation behavior characteristics of the number of malicious and benign applications respectively, $IDF_{(i,B)}$ and $IDF_{(i,M)}$ as the number of applications in the system call behavior features divided by the logarithm of the total number of benign and malicious applications in the training set. The definition formula is as follows:

$$IDF_{(i,B)} = \log(\frac{D_B}{D_{(i,B)}+1}) \tag{3}$$

$$IDF_{(i,M)} = \log(\frac{D_M}{D_{(i,M)}+1}) \tag{4}$$

As for the weight formula of the above behavioral characteristics, analyze the importance degree of behavioral characteristics in the malicious application and the benign application to construct the absolute difference weight of the characteristics W_i 1 definition formula is as follows:

$$W_i = \left| TF_{(i,M)} * IDF_{(i,M)} - TF_{(i,B)} * IDF_{(i,B)} \right| \tag{5}$$

Through the above improved TF-IDF algorithm, this paper analyzes the call behavior weight of behavior characteristics in malicious and benign applications. The higher the weight, the more important the behavior feature is to the classification and detection of malicious software, and finally ranks it in descending weight order.

3.3 Feature Embedding Moudle

Malicious behavior of malware can reflect the features of system calls. Therefore, when detecting malware, we can use not only a single type of feature set, but also a combination of different feature sets. In order to use composite feature sets, we need to solve how to combine different dimensional feature groups into a unified representation.

We use TF-IDF algorithm to calculate the weight W_i for each behavior feature i of different feature sets. Through the size of W_i, we sorted different types of feature sets into a new feature set S as follows:

$$S = sort\left(W_1, W_2, \cdots, W_i\right) \tag{6}$$

We define feature set S as a Boolean expression with $|S_i|$ dimensions, and then embed the feature set into a vector space X to get a unified representation. If application X uses some features in the feature set, the feature set is a vector of 1 with a position of 0. Therefore, we can convert any application into vector space as follows:

$$X = \left\{S_1, S_2, \cdots, S_k\right\}, \left(k \in |S_i|, S_k \begin{cases} 1, F \in S_k \\ 0, F \notin S_k \end{cases} \right) \tag{7}$$

Let's consider a real example: when a malware steals a user's contact information, it needs to use group:android.permission-group.contacts, so that it can be embedded into a vector space, as shown below：

$$X = \begin{Bmatrix} \cdots & \cdots \\ 1 & android.permission.WRITE\_CONTACTS \\ 0 & android.permission.READ\_CALENDAR \\ 1 & android.permission.GET\_ACCOUNTS \\ 1 & android.permission.READ\_CONTACTS \\ \cdots & \cdots \end{Bmatrix} \tag{8}$$

In this way, we embed different feature sets into a unified joint vector space, and we can use feature sets to perform extensive detection.

3.4 Detection model Deep Learning Moudel

LSTM is proposed by Hochreiter [30], which can effectively overcome the gradient explosion and gradient disappearance caused by the long-term dependence of traditional RNN. It is composed of a forget gate, an input gate, an output gate and a memory cell. Through the forget gate, information is filtered, useful information is left behind, useless information is forgotten, and information is fully analyzed. Therefore, we design the LSTM network structure as a classification network, use the feature vector generated by the feature weighting method to classify, and deeply analyze the information between context features.

We use LSTM classification model to detect malicious applications. Its structure is shown in **Fig. 2**. Firstly, the model uses the method of feature weighting to construct the feature vector as the input, analyzes the relationship between the parameters of the input vector and the accuracy of system detection of malicious applications, and continuously adjusts the vector size to achieve a comprehensive detection of malicious applications. Then we consider the feature vectors of different attributes to analyze the performance of the system to detect malicious applications and select the best feature vectors. The hidden layer is composed of LSTM neural network, and Adam algorithm is used as optimization method in training neural network.

The algorithm corrects the mean value of gradient and the mean square of gradient by using the number of iterations and delay factors. It can predict the change of gradient more

accurately and has higher convergence speed. And the parameters are improved, and the number of hidden units is selected and debugged to achieve the best detection effect. After the hidden layer, a layer of full connection layer is constructed. Because the whole process is a classification and detection task, the output of the full connection layer is taken as the input of the classification layer, the application detection is classified by Sigmoid classifier, and the binary cross entropy loss function is used as the loss function to evaluate the prediction and actual effect of the detection model.

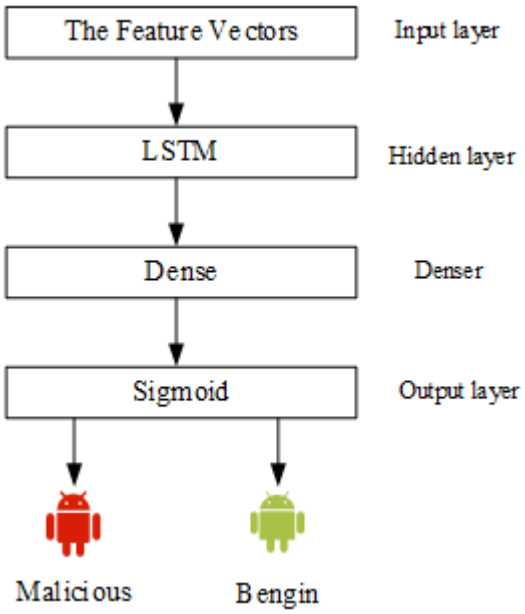

Fig. 2. LSTM detection model structure

3.4.1 LSTM Algorithm

In this paper, LSTM neural network is used for application classification detection. The TF-IDF algorithm is used to sort the features according to the weight and form the feature vector as the input sequence to analyze the application. The LSTM module includes Forget Gate, Input Gate, Output Gate and a cell to filter, save and update information. Next, we will introduce the LSTM module:

Forget Gate screens the cell state of the upper layer to leave useful information and Forget useless information. The formula is as follows:

$$f_t = \sigma\left(W_f * \left[h_{t-1}, x_t\right] + b_f\right) \tag{9}$$

W_f and b_f are respectively the weight and bias of the forgetting gate, h_{t-1} is the upper hidden state, and σ is the activation function of Sigmod. The forgetting gate is controlled by the sigmoid function, which generates a value of f_t based on the output h_{t-1} and the current input x_t at the previous moment, and decides whether to output the information C_{t-1} obtained at the previous moment.

Input Gate judge the information, send the important information to the update place of cell state, and complete the update of cell state. The formula is as follows:

$$i_t = \sigma\left(W_i * \left[h_{t-1}, x_t\right] + b_i\right) \tag{10}$$

$$C_t = f_t * C_{t-1} + i_t * \tanh\left(W_c * \left[h_{t-1}, x_t\right] + b_c\right) \tag{11}$$

W_i and b_i are the weights and biasing of the input gate, W_c and b_c are the weights and biasing of the cell state, C_{t-1} and C_{t-1} are the original cell state and the current cell state, respectively.

The process consists of two parts. One is to use the Sigmoid function to determine which information needs to be updated and added to the cell state. The other is to use the $\tanh$ activation function to transform the information that needs to be updated into a candidate vector that can be added into the cell state to generate a new candidate vector C_{t-1}. We combine the above two parts to generate a new cell state C_t.

The Output Gate contains the current input, the previous hidden state, the current cell state and so on to control the Output of the cell state of this layer. The formula is as follows:

$$o_t = \sigma\left(W_o * \left[h_{t-1}, x_t\right] + b_o\right) \tag{12}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{13}$$

Where W_o and b_o are respectively the weight and offset of the output gate. Sigmoid activation function is used to determine the desired output of o_t, then $\tanh$ activation function is used to process the contents of the cell state, and h_t to determine which cell state to output.

4. Experimental Analysis

4.1 Dataset

In order to verify the effectiveness of the proposed detection method, we use a data set consisting of 4000 malicious samples and 4000 benign samples. We have downloaded 7000 Android apps from the official Android Market, removed the unusable and duplicate apps, and got 4000 apps, and marked them as benign apps; downloaded Android apps from VirusShare (https://virusshare.com/), removed the unusable and duplicate apps, and got 4000 apps, and marked them as malicious apps. The application of training set and test set in the experiment consists of the above two parts.

4.2 Enviornment Metrics

The experiment in this paper was done on Windows 7 with Intel(R)Core (TM) i3-2130 CPU 3.40ghz, 16G of RAM. We used 4 different evaluation indexes, namely, Precision ($P=\frac{TP}{TP+FP}$), Recall ($R=\frac{TP}{TP+FN}$), F-score ($F=2\frac{(P*R)}{P+R}$) and Accuracy

($ACC = \frac{TP + TN}{TP + FP + TN + FN}$), as evaluation indexes to evaluate the effect of the detection model. TP—the number of malware samples that are correctly classified; TN —the number of benign samples that are correctly classified; FP —the number of benign samples that are incorrectly classified; FN —the number of malware samples that are incorrectly classified .The evaluation index formula is as follows:

4.3 Parameter Setting

We randomly selected 3000 ordinary applications and 3000 malicious applications and analyzed the structural performance of our test model from different perspectives. As shown in **Fig. 3**, we select the number of units in the hidden layer. Considering that the number of units in the hidden layer is too small, the data cannot be trained or the detection performance is poor, and the application cannot be accurately detected. When the number of hidden layers is too much, the training data is likely to fall into the local minimum and not get the optimal performance. So we choose the same features as the input vector length, found hidden layer unit number to 300 has high detection performance, and the hidden layer unit number from 300 to begin testing, we found all the hidden layer unit number from the Fig. of more than 93% accurate, and, when the hidden layer unit number to 900 accuracy reached 95.12%, higher than other performance of the unit number of hidden layers, so the paper selection for hidden layer unit number 900.

Secondly, we consider the impact of the length of the input eigenvector on malware detection. We test the relationship between the length of input eigenvectors and the accuracy of Android application classification. On the one hand, considering that the input feature set is too small to cover all malicious and normal behaviors, the input feature set is too long, which will cause additional overhead. On the other hand, we found that when the input feature vector length reaches 1000, it has better detection effect, so the input feature vector size starts from 1000. Using the results of these experiments, the optimal eigenvector of the classification task is selected. As can be seen from the figure, our classifier can distinguish malware from normal software, and achieve good accuracy and recall rate. It can be seen from **Fig. 4** that the best classification is achieved when the length of feature set is 8000, that is, the accuracy is 95.31%, and the recall rate is 95.86%. In other cases, the recall rate and accuracy index are 8000 lower than the length of feature set, but the accuracy rate and recall rate of detection results are still above 91%. It shows that the method in this paper can extract the applied behavior features well, and it will not lead to large fluctuations in the classification results, and achieve good accuracy.

We also analyze whether the number of hidden layers in different LSTM classification models would affect the detection results. We analyze the detection performance of LSTM classification models with different hidden layers. Each layer of all networks has 900 hidden units, and the input eigenvector length is 8000. As can be seen from **Fig. 5**, when the number of hidden layers is 3, the method in this paper can achieve the best classification result, that is, the accuracy rate is 95.12% and the precision rate is 94.67%.When the number of hidden layers is less than 3, not enough information can be obtained. If the number of hidden layers is greater than 3, then the network is over-fitting.

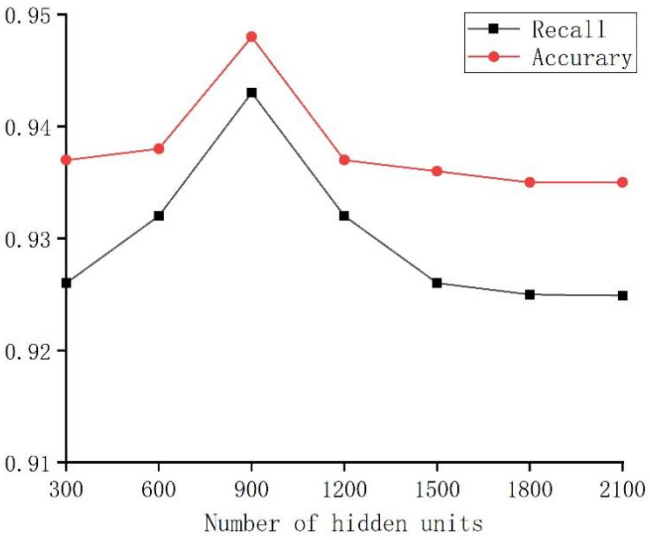

Fig. 3. Accuracy and recall under different numbers of hidden units

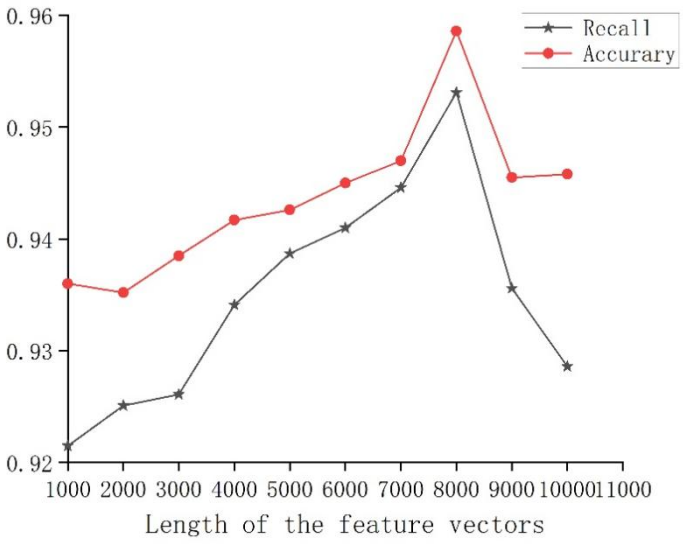

Fig. 4. Accuracy and recall under of different lengths of the feature vectors

In order to be able to accurately show malicious behavior in the application and get better detection results, researchers will choose different feature sets for detection research. In this paper, we use each feature set defined in the third section, we evaluate the effectiveness of the methods mentioned in this paper on different feature sets, and we embed each feature set into the joint vector space through TF-IDF algorithm to obtain comprehensive detection.

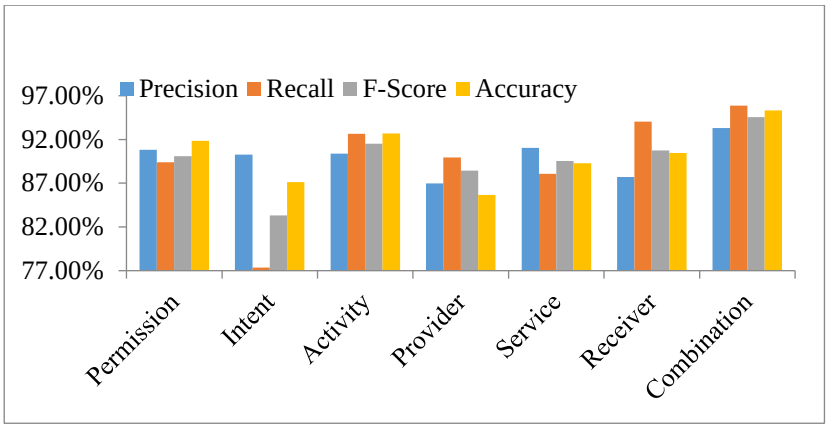

Fig. 5. Feature Vector Evaluation Based on Different Feature Sets

From **Table 1**, we can see that the combined feature set has the best effect in malware detection. Compared with other feature sets, the performance indicators differ by more than 3% and the detection effect of combined features cannot be achieved. There are many reasons for this situation. For example, the corresponding detection of permission feature set provides less information and cannot accurately detect applications.

Table 1. Metrics under a different number of hidden layers

Hidden Layer	Precision	Recall	F-Score	Accuracy
1	92.64%	93.15%	92.89%	93.89%
2	92.67%	94.47%	93.56%	94.83%
3	93.32%	95.86%	94.57%	95.31%
4	92.13%	96.30%	94.17%	94.42%

4.4 Experimental Result

In order to verify the proposed in-depth learning Android malicious detection method based on feature analysis, we use the same extracted feature set to compare with the current commonly used detection methods. The comparison results are shown in **Table 2**. When the machine learning algorithm is used for detection, it can also achieve good detection effect. The accuracy of detection is over 87%, and the accuracy of deep learning algorithm is over 95%. Other performance indicators are better than the above detection algorithm. Experiments show that the performance of the system using LSTM classification detection model is better than that of other malware detection models.

Table 2. Experimental results of different classifiers

Classifier	Precision	Recall	F-Score	Accuracy
SVM	90.89%	77.65%	83.75.%	87.48%
Logistic	91.04%	95.32%	93.13%	94.53%
Decision Tree	88.11%	92.22%	90.12%	91.59%
Naïve Bayes	89.20%	91.09%	90.14%	91.71%
LSTM	93.32%	95.86%	94.57%	95.31%

To sum up, the paper uses TF-IDF algorithm to process the behavior features, and uses long short-term memory neural network algorithm to form a deep learning Android malicious detection method based on feature analysis. Through experiments, it is verified that this model can detect malicious applications better than other detection models, can achieve more accurate and rapid detection, and achieve the expected detection effect.

5. Conlusion

In recent years, the prevalence of malicious applications on the Android platform and the existing Android malware detection technology have been studied and analyzed. This paper proposes a deep learning Android malware detection method based on feature analysis. This method combines TF-IDF algorithm with LSTM network algorithm to detect Android malicious applications accurately and effectively. This method analyzes the behavior features and optimizes the parameters to achieve the best detection performance. Experiments show that the model improves detection performance, has a high accuracy, and has a strong ability to identify malicious applications. This scheme analyzes the application from the overall perspective, which will produce certain errors, negatively affect the detection of malicious applications, and fail to detect malicious behaviors generated by the application running. In the future work, we will divide the applications into categories, analyze the applications of different categories in detail, extract the corresponding behavioral characteristics for analysis, improve the detection performance of malicious applications, and extend the extracted feature set to dynamic analysis for detailed analysis of applications.

Acknowledgement

The financial support for this work provided by the Cooperative Education Fund of China Ministry of Education (201702113002, 201801193119) and the Scientific Research Fund of Hunan Provincial Education Department (20A191) are greatly appreciated by the authors.

References

[1] EGHAM. Gartner says worldwide sales of smartphones recorded 1st ever decline during the 4th quarter of 2017.

[2] D. Kim, G. Shin and M. Han, “Analysis of feature importance and interpretation for malware classification,” *Computers, Materials & Continua*, vol. 65, no. 3, pp. 1891-1904, 2020. Article (CrossRef Link)

[3] A. Sadeghi, H. Bagheri and J. Garcia, “A taxonomy and qualitative comparison of program analysis techniques for security assessment of android software,” *IEEE Transactions on Software Engineering* vol. 43, no. 6, pp. 492-530, 2016. Article (CrossRef Link)

[4] W. Liang, D. Zhang and X. Lei, "Circuit Copyright Blockchain: Blockchain-based Homomorphic Encryption for IP Circuit Protection," *IEEE Transactions on Emerging Topics in Computing*, pp. 1-1, 2020.Article (CrossRef Link)

[5] Y. Li, G. Xu, H. Xian, L. Rao and J. Shi, “Novel android malware detection method based on multi-dimensional hybrid features extraction and analysis,” *Intelligent Automation & Soft Computing*, vol. 25, no.3, pp. 637-647, 2019. Article (CrossRef Link)

[6] Z. Wang, Y. Tang, J. Yao, R. Qian, Z. Zhang and P. Ma, “Large-scale Malware Automatic Detection Based On Multiclass Features and Machine Learnin,” in *Proc. of the 2nd International Conference on Computer Science and Application Engineering*, New York, NY, USA, pp. 1-5, Oct., 2018. Article (CrossRef Link)

[7] J. Liu, Y. Zeng, J. Shi and Y. Yang, “Maldetect: a structure of encrypted malware traffic detection,” *Computers, Materials & Continua*, vol. 60, no. 2, pp. 721-739, 2019. Article (CrossRef Link)
[8] Y. Zhang, Y. Yang and X. Wang, “A novel android malware detection approach based on convolutional neural network,” in *Proc. of the 2nd International Conference on Cryptography, Security and Privacy*, New York, NY, USA, pp. 144-149, 2018. Article (CrossRef Link)
[9] I. R. A. Hamid, S. Subramaniam and Z. Abdullah, “Classification of Polymorphic Virus Based on Integrated Features,” *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 6, pp. 2577-2583, 2018. Article (CrossRef Link)
[10] A. Zulkifli, I. R. A. Hamid, W. M. Shah and Z. Adbullah, “Android malware detection based on network traffic using decision tree algorithm,” in *Proc. of International Conference on Soft Computing and Data Mining*, Springer, Cham, Jan., vol.700, pp. 485-494, 2018. Article (CrossRef Link)
[11] P. Vinod, Z. Akka and M. Conti, “A machine learning based approach to detect malicious android apps using discriminant system calls,” *Future Generation Computer Systems*, vol. 94, pp. 333-350, 2019. Article (CrossRef Link)
[12] W. Liang, L. Xiao and K. Zhang, "Data Fusion Approach for Collaborative Anomaly Intrusion Detection in Blockchain-based Systems," *IEEE Internet of Things Journal*, pp. 1-1, 2021. Article (CrossRef Link)
[13] F. Ali, B. A. Nor, S. Rosli and W. A. W. Ainuddin, "A review on feature selection in mobile malware detection," *Digital investigation*, vol. 13, pp. 22-37, June, 2015. Article (CrossRef Link)
[14] W. Liang, S. Xie and D. Zhang, "A mutual security authentication method for RFID-PUF circuit based on deep learning," *ACM Transactions on Internet Technology*, pp. 1-20, 2020.
[15] A. Kelec and Z. Djuric, “A proposal for addressing security issues related to dynamic code loading on android platform,” *Computer Systems Science and Engineering*, vol. 35, no.4, pp. 271-282, 2020. Article (CrossRef Link)
[16] K. Lim, N. Kim Y and Y. Jeong, “Protecting android applications with multiple DEX files against static reverse engineering attacks,” *Intelligent Automation & Soft Computing*, vol. 25, no.1, pp. 143-153, 2019. Article (CrossRef Link)
[17] G. He, B. Xu and H. Zhu, “AppFA: A Novel Approach to Detect Malicious Android Applications on the Networ,” *Security and Communication Networks*, vol. 2018, pp. 1-15, Apr., 2018. Article (CrossRef Link)
[18] K. A. Talha, D. I. Alper and C. Aydin, “APK Auditor: Permission-based Android malware detection system,” *Digital Investigation*, vol. 13, pp. 1-14, 2015. Article (CrossRef Link)
[19] D. Gianluca, M. Fabio, M. Ilaria, P. Marinella, A. Saracino and D. Sgandurra, "Risk analysis of Android applications: A user-centric solution," *Future Generation Compute*r Systems, Mar., vol. 80, pp. 505-518, 2018. Article (CrossRef Link)
[20] H. Zhu, Z. You, Z. Zhu, W. Shi, X. Chen and L. Cheng, “DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model,” *Neurocomputing*, vol. 272, pp. 638-646, 2018. Article (CrossRef Link)
[21] S. Karina, P. Charles and L. Marc, “Android application classification and anomaly detection with graph-based permission patterns,” *Decision Support Systems*, vol. 93, pp. 62-76, 2017. Article (CrossRef Link)
[22] R. Sato, D. Chiba and S. Goto, “Detecting Android malware by analyzing manifest files,” in *Proc. of the Asia-Pacific advanced network*, vol. 36, pp. 23-31, 2013. Article (CrossRef Link)
[23] Z. U. Rehman, S. N. Khan and K. Muhammad, “Machine learning-assisted signature and heuristic-based detection of malwares in Android devices,” *Computers & Electrical Engineering*, vol. 69, pp. 828-841, 2018. Article (CrossRef Link)
[24] A. Shabtai, U. Kanonov and Y. Elovici, “Andromaly: a behavioral malware detection framework for android devices,” *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161-190, 2012. Article (CrossRef Link)
[25] W. Enck, P. Gilbert and B. G. Chun, “TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones,” *Communications of the ACM*, vol. 57, no. 3, pp. 99-106, 2014. Article (CrossRef Link)

[26] J. Milosevic, A. Ferrante and M. Malek, "MalAware: Effective and Efficient Run-Time Mobile Malware Detector," in *Proc. of 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pp. 270-277, 2016. Article (CrossRef Link)

[27] C. Hasegawa and H. Iyatomi, "One-dimensional convolutional neural networks for Android malware detection," in *Proc. of 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, pp. 99-102, 2018. Article (CrossRef Link)

[28] Y. Zhang, Y. Yang and X. Wang, "A novel android malware detection approach based on convolutional neural network," in *Proc. of the 2nd International Conference on Cryptography, Security and Privacy*, pp. 144-149, Mar. 2018. Article (CrossRef Link)

[29] G. Lin, W. Xiao, J. Zhang and Y. Xiang, "Deep Learning-Based Vulnerable Function Detection: A Benchmark," in *Proc. of International Conference on Information and Communications Security*, vol.11999, pp. 219-232, 2020. Article (CrossRef Link)

[30] H. Sepp and S. Jürgen, "Long short-term memory," *Neural Computation*, vol. 9, no.8, pp. 1735-1780, 1997Article (CrossRef Link)

Qing Yang received the B.E. degree in computer engineering from Hunan Normal University, China, and the master's degree of computer science from National University of Defense Technology, China. He received the Ph.D. degree from Wuhan University of Technology, China. He had been a visiting researcher at the University of Alberta, Canada. Now, he is a professor of information technology, Guangzhou Maritime Institute, China. He leads a team of researchers and students in the areas of Network Security and Internet of Things. His has published more than 10 highly reputed SCI/EI indexed journals/ conferences articles and his research has been funded by Natural Science Foundation Committee and Ministry of Education of China.

XiaoLiang Wang received the B.E. degree in computer engineering from Xiangtan University, China, and the master's degree of computer science from the joint education of Xiangtan University and the Institute of Computing Technology of the Chinese Academy of Sciences, China. He received the Ph.D. degree from Hunan University, China. He had worked at Xiangtan University and the Nanjing Government of China, and had also worked as a postdoctoral researcher at the University of Alabama, USA. Currently, he is a professor of information technology and the director of the Department of Internet of Things Engineering, Hunan University of Science and Technology, China. He leads a team of researchers and students in the areas of Information Security and Internet of Things, such as VANET security, Anonymous Authentication in Ad Hoc Networks. His has published more than 30 highly reputed SCI/EI indexed journals/ conferences articles and his research has been funded by Natural Science Foundation Committee and Ministry of Education of China.

Jing Zheng, born in 1997, computer science and technology graduate student, Hunan university of science and technology, member of CCF. His research interest includes the application of block chain in the Internet of Things and privacy preservation.

Wenqi Ge received his master's degree from Hunan University of Science and Technology in 2020. His main research interests are mobile communication security and privacy protection.

Ming Bai received the master's degree from Sun Yat-sen University, China. He is a professor of information technology and the director of the College of Innovation and Entrepreneurship, Guangzhou Maritime Institute, China. He has published 10 SCI/EI indexed journals/ conferences articles. His main research interests include data-driven cyber security, predictive analytics, biologically inspired learning mechanism, and its application in the complex information security system.

Frank Jiang received the Ph.D. degree from The University of Technology Sydney and the master's degree in computer science from the University of New South Wales (UNSW), Australia. He gained the 3.5 years of post-doctoral research experiences UNSW. He has published over 100 highly reputed SCI/EI indexed journals/ conferences articles. His main research interests include data-driven cyber security, predictive analytics, biologically inspired learning mechanism, and its application in the complex information security system.