Android Malicious App Detection Based on CNN Deep Learning Algorithm

Peng He and Gang Gan
School of Cyberspace Security, Chengdu University of Information Technology, Chengdu 610225, China.
Email: 592921931@qq.com; test_me@cuit.edu.cn

Abstract. In recent years, with the increasing number of Android applications, Android malicious apps have also increased exponentially. This paper proposes a detection scheme based on CNN deep learning algorithms. The scheme uses static analysis and detection methods. CNN deep learning algorithms have become very mature in the field of image classification. Therefore, this article uses a conversion algorithm to convert Android APK binary files to RGB PNG images, and then uses the algorithm to extract permissions in the APK, and converts the permission matrix to a transparency matrix, combines the previous RGB images, and finally generates RGBA images. RGBA images contain more information than RGB images, and the use of CNN deep learning algorithms works better and the final detection rate reaches 93.4%.

1. Introduction
At present, Android, ISO and windows systems are the main systems used in the mobile device market. With the breakthrough of communication technology and mobile device hardware, the number of mobile devices is increasing, and the corresponding applications are also increasing. According to the Statistic [1] report, as of 2017, the number of Smartphone users has reached 4.77 million since 2013, and it is predicted that by the end of 2019, the number of users will reach 5 billion. In 2018, the CEO of Google revealed that Google's Android operating system has made great achievements, with more than 2 billion online active Android users every month [2]. The profits of the corresponding Android black industry are also growing, attracting a large number of hackers to participate in it. The common Android malicious apps in the first stage market are mainly divided into virus, worm, Trojan horse, ransomware, rootkits and botnet. The proportion of them in the mobile application market is also very high. It is reported that they have reached 10% of the existing app market.

Android malicious app refers to any android app with malicious intention, which will cause many problems, such as spreading malicious behavior through short-distance communication hardware such as Bluetooth of the device. In many cases, they will attack the authority security system to get the root authority of the system, and then steal the user's private information. At the same time, they will send some tariff messages to consume the user's assets. Recently, there is a lot of extortion software, through malicious software to lock the storage data of the mobile device, so that the mobile device cannot operate normally, it needs to pay a certain amount of money to complete the unlocking of the storage data. There are also some malware, through the monitoring of GPS and other functions, to obtain the location of users, and then carry out a series of social engineering attacks.

In recent years, APP market such as Google play and Huawei app market are also reducing the number of Android malicious apps and their harm to Android users through various ways, but they do not solve the problem of Android malicious app detection. New circumvention detection technology is developing.
various malicious app variants are still the bottleneck of detection technology at this stage, and Android malicious app detection technology is in urgent need of development.

2. Related Work
At this stage, the mainstream detection methods of Android malicious app are divided into three kinds, namely dynamic detection method, static detection method and machine learning method. In the field of Android malicious app detection, because the number of malicious app increases, machine learning and deep learning are more and more tolerated by people. This section will study the machine learning algorithm and deep learning algorithm in the field of Android malicious app detection Related work, and describes the breakthrough and improvement of this method.

Flowdroid [3] uses static analysis technology to analyze the called callback, extract data such as data flow, context, field sensitive information data and objects, model Android's life cycle method, and handle and analyze the pollution transmission of user interface objects and callback behavior. Drodimat [4], through extracting configuration files and log files, we can understand API calls and track the process of API calls, and finally get a good detection rate. Aziz Makandar, etc [5], converts binary files of malicious code to grayscale images. Then we use the machine learning algorithm to classify the malicious app. The main task is to classify the malicious app family. R. Vinayakumar, etc [6] proposed to use LSTM model to detect Android malware, which can get good results by analyzing a small number of samples, but there are problems of long training time and detection time. Droiddolphin [7], first pre-process the log, then carry out simulation and test experiments, then extract features, and finally use machine learning method to classify and detect. It mainly analyzes the log at runtime, traverses the application code path, and analyzes the path. Taindroid [8] is a system for tracking the dynamic pollution of privacy violations. Its analysis includes methods, inter process messages, variables and file levels. Vetdroid [9] based on the Taindroid, mining all permissions, highlighting the connection between methods, and finally connecting the relationship between methods into function call graph, and using machine learning method for identification.

Through the research of Android malicious app, the method in this paper has the following advantages:
• In this paper, we use the conversion method to convert Android APK binary files into color pictures. Many detection methods do not detect Android so library and res files, but in the first stage, many malicious codes exist in these files. This method detects the whole file, and solves the problem that Android so files and res files are ignored.
• The color image used in this paper carries more information than the gray image, so it is better to use CNN deep learning algorithm for feature extraction and classification.
• In this paper, we combine APK features with permission features, and use RGBA images to improve the accuracy of CNN deep learning algorithm.

3. Malicious App Detection Method
This section will introduce the improved method based on CNN deep learning algorithm, including the following two aspects: (1) in this paper, APK binary file is transformed into RGB color image, and the authority matrix in app is extracted, and the image and authority matrix are transformed into RGBA color image. (2) Using CNN improved deep learning algorithm to extract the feature of RGBA color image, detect and classify android app. According to the result of classification, we realize the automatic detection of Android malicious app.

3.1. Image Conversion
In the process of transforming Android APK into RGBA color picture, there are three parts: Android APK transforms RGB color picture; obtains the permission directory of app and transforms it into permission matrix; generates RGBA color picture by combining RGB color picture and permission matrix.
3.1.1. **APK to RGB Picture.** At this stage, there are many ways to convert Android APK binaries into RGB images. In this paper, we refer to the visualization method of executable malware binary [10]. In this paper, the binary bit string of malware can be divided into several 8-bit-long substrings. Each of these substrings is regarded as a pixel. In the storage of gray-scale image, the storage space of a pixel is exactly 8 bits. At the same time, 8 bits can be interpreted as unsigned integers in the range of 0 to 255. Using the above method for reference, RGB color picture pixel points are divided into R, G, B three bits, and 24 bit data can be regarded as a pixel point. For example, if the string is 010010111101010111001100, the process is 010111101010111001100 \( \rightarrow \) 75213204 \( \rightarrow \) RGB pixel point (75213204).

After RGB pixel matrix transformation, the matrix needs to be saved as a PNG picture. Through experimental analysis, it is found that the size gap of Android APK file is relatively large, the size of malicious app is generally small, and the size of normal app file is relatively large, so if you use the same width to save the picture, there will be a problem of too large picture gap. Therefore, according to the experience in the paper [10], table 1 gives an experience based view. Observe the recommended image width for each file size. According to the research and observation, the equal scale compression of image file will not have a great impact on the detection results. In this paper, all RGB images are finally scaled to the same size.

3.1.2. **Authority Matrix Extraction.** Because Android OS system has used new permission mechanism since version 6.0, many app permission requirements can be applied dynamically. In order to obtain more accurate app permission directory, two methods are used for permission extraction. The first method is to extract the permission list using Andrograd, and the second method is to extract the Android manifest.xml file using aapt, a tool in Android SDK. At the same time, Apktool is used to decompile APK file, and then regular expression is used to extract the permission list dynamically applied in the file. Finally, the permission matrix of each app is a one-dimensional vector matrix. At the same time, because of the use of user-defined permission in APK, this paper classifies all other permissions into other permissions. The permissions of a single file are based on all the permission types in the statistics. Fill 153 for the positions with permissions and 76 for the positions without permissions. Finally, the number of rows and columns in the permission matrix of all apps is the same.

3.1.3. **Composite RGBA Picture.** Because the permission matrix is a one-dimensional vector matrix and the RGB image is a two-dimensional matrix, the permission matrix needs to be processed. Through statistics, it is found that the permission types of all files are smaller than the width of RGB image, so the permission matrix is filled in first, with a filling of 255, so that the number of elements
of one-dimensional matrix is the same as the width of RGB image, copy this one-dimensional vector, copy the number. Finally, a two-dimensional matrix with the same structure as RGB is generated. Combine the generated permission matrix with the pixel matrix of RGB image. For example, an RGB pixel is represented as (100220105), plus one (255), which is transformed into an RGBA pixel (100220105, 255). By this way, the final RGBA image can be generated.

| File size | Image width |
|-----------|-------------|
| <10kB     | 32          |
| 10kB~30kB | 64          |
| 30kB~60kB | 128         |
| 60kB~100kB| 256         |
| 100kB~200kB| 384        |
| 200kB~500kB| 512        |
| 500kB~1000kB| 768        |
| >1000kB   | 1024        |

3.2. Neural Network Algorithm

CNN is a kind of neural network specially used to process data with similar network structure, such as one-dimensional grid formed by regular sampling on the time axis, such as log files, and two-dimensional pixel grid. CNN refers to those neural networks which use convolution operation to replace the general matrix multiplication operation at least in one layer of the network. CNN has excellent performance in many fields. Its main composition is generally the superposition of one or more layers of convolution and pooling operations. On some problems that can't extract features, CNN has a good realization. In the field of Android malicious app detection, CNN often participates. In this paper, three layers are used, namely convolution layer, pooling layer, normalization layer and nonlinear processing layer. The specific implementation is shown in Figure 2.

![CNN Architecture Diagram](image)

Convolution layer, in general, convolution is a kind of data operation for two real variable functions. Convolution layer can reduce the number of image parameters. By convolution operation on a local area, the eigenvalues of this local area are transmitted to the lower layer, which greatly improves the ability of neural network to extract features, and also reduces the size of data. The specific mathematical formula is as follows:
\[ x_j^l = \sum_{i \in M_j} x_{ij}^{l-1} * k_{ij}^l + b_j^l \quad (1) \]

Among them, \( M_j \) is the set of input graphs, \( k_{ij}^l \) is the convolution kernel for the connection between the \( i \)th input characteristic graph and the \( j \)th output characteristic graph, \( b_j^l \) is the deviation corresponding to the \( j \)th characteristic graph.

In the pooling layer, the pooling function uses the total statistical characteristics of the adjacent output at a certain location to replace the source of the network at that location. In this paper, the maximum pooling function is used, and the maximum value in the adjacent matrix area is given. The convolution kernel size is 2 * 2, and the step size is 1. Its mathematical principle is as follows:

\[ y_{ij} = \max(x_{ij}, x_{i,j+1}, x_{i+1,j}, x_{i+1,j+1}) \quad (2) \]
\[ i \leq m - 2 \quad (3) \]
\[ j \leq n - 2 \quad (4) \]

In the first and non-linear processing layer, the non-linear processing function used in this layer is the ReLU function, whose function is to increase the non-linear relationship between each layer of the neural network. In fact, the ReLU function is a piecewise linear function, which changes all negative values to 0, while positive values remain unchanged. This operation is called unilateral suppression. With unilateral inhibition, the neurons in the neural network also have sparse activation. The normalization layer can normalize the input layer or convolution layer data, so that the data will not cause the network performance instability due to the large data before ReLU. The mathematical principle of the normalization layer function is as follows:

\[ y = \frac{x - \text{mean}(x)}{\sqrt{\text{var}(x) + \text{eps}}} \cdot \text{gamma} + \text{beta} \quad (5) \]

Where \( x \) is the input matrix and \( y \) is the output matrix.

In this paper, the convolution pooling layer extracts the features of RGBA images, and uses the extracted feature matrix for the full connection layer. Finally, a one-dimensional matrix of 120 elements is generated and classified. The features extracted by CNN are unreadable, but very effective, and can automatically classify RGBA images.

4. Details and Results of the Experiment

In this section, we introduced the effective evaluation of this method. The malicious app data set used in the experiment is the app APK verified by Windows Defender and Google play. At the same time, the app APK downloaded from Huawei app store is used as the normal app data set, the neural network framework is the python framework, and the PC uses Intel(R) Xeon(R) CPU E3-1230 V2 3.30GHz.

4.1. Dataset and Experimental Setting

There are 3815 samples in this dataset, including 2179 malicious app samples and 1636 normal samples. There are 131 categories of permissions, including 130 system permissions and 1 other permissions. The pixel matrix of RGBA image is set to 224 * 224. Through the experiment, it is found that the learning rate of CNN deep learning algorithm is set to 0.001, which is the most appropriate. CNN finally generates one-dimensional vectors of 120 elements, and calculates the loss through the criterion function. The training uses CPU for training. According to the experimental test, it is found that the number of iterations is set to 100, the number of categories is 2, and the batch size is 4.

We use accuracy, precision and recall as evaluation indicators. These evaluation indicators are often used in machine learning methods to provide a comprehensive assessment of learning imbalances. These indicators are defined as follows:

\[ \text{Precision} = \frac{TP}{TP + FP} \quad (6) \]
\[ \text{Recall} = \frac{TP}{TP + FN} \quad (7) \]
Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (8)

True Positive (TP) and False Positive (FP) represent the number of file samples of malicious app in the correct and wrong classification. Similarly, True Negative (TN) and False Negative (FN) represent the number of samples of documents classified as good app for correctness and error.

4.2. Experimental Results
In this section, we will carry out a comparative test on our existing data sets, including the classification using LSTM method and the classification using RGB pictures as CNN input. Among them, the flow of LSTM detection algorithm is to run android app in sandbox experimental environment, collect system call logs, pre-process system call logs, and finally use LSTM algorithm for training and testing. In the experiment, it is found that the LSTM dynamic detection method has the worst detection effect on this sample set, and the longest training and detection time. When RGB image is used as CNN input, the training and testing time is short, but the detection effect is not as good as that of CNN which RGBA image is the most input. The experimental results are as follows:

| Table 2. Compare Experimental Results |
|--------------------------------------|
| Accuracy | Precision | Recall | Run time |
| LSTM     | 86.5      | 87.2   | 86.8     | 60ms     |
| RGB+CNN  | 90.6      | 90.5   | 90.3     | 20ms     |
| RGBA+CNN | 93.4      | 93.6   | 93.3     | 30ms     |

The final experiment shows that the RGBA image detection method based on CNN has a moderate running time, but the detection effect is better.

5. Conclusion and Future Work
This paper proposes an RGBA image detection system based on CNN deep learning algorithm, which adds a new method to the static detection method, and proposes a novel feature combination method, which can better combine the permission list with the RGB image transformed by android app. In the next stage, we can combine the static detection method with the dynamic detection method, using logs such as API calls, system method calls, etc. as a result, CNN or other algorithms extract features, and then combine RGB and feature matrix to further improve the detection rate.

6. References
[1] S.T.S. Portal 2018 Number of mobile phone users worldwide from 2013 to 2019 (in billions) Available from:https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/
[2] B. Popper 2017 Google announces over 2 billion monthly active devices on android Available from:https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users
[3] S. Arzt, et al. 2014 Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps *Acm Sigplan Not.* 49 (6) (2014) 259–269
[4] D.-J. Wu, et al. 2012 Droidmat: Android malware detection through manifest and api calls tracing *Information Security, Asia JCIS, 2012 Seventh Asia Joint Conference on IEEE*
[5] Aziz Makandar, et al. 2017 Malware Class Recognition Using Image Processing Techniques International Conference on Data Management, Analytics and Innovation (ICDMAI) Zeal Education Society, Pune, India, Feb 24-26
[6] R. Vinayakumar, et al. 2018 Detecting Android malware using Long Short-term Memory (LSTM) *Journal of Intelligent & Fuzzy Systems* 34 (2018) 1277–1288
[7] W.-C. Wu and S.-H. Hung 2014 Droiddolphin: a dynamic android malware detection framework using big data and machine learning, in: *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems ACM*
[8] W. Enck, et al. 2014 Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones ACM Trans. Comput. Syst. (TOCS) 32 5

[9] M.N.P. Pravin 2015 Vetdroid: Analysis using permission for vetting undesirable behaviours in android applications Int. J. Innov. Emerg. Res. Eng. 2 131–136

[10] L. Nataraj, S. Karthikeyan, G. Jacob and B. Manjunath Malware images: visualization and automatic classification Proceedings of the 8th international symposium on visualization for cyber security ACM p 4