Android malware detection based on Multi-Class Features

Shuangwei Ye [a], Yue Zhang [b]

Dept. of Computer Science, Jinan University, Guangzhou 510632, China

[a] yoga__007@163.com, [b] zyuninfosec@gmail.com

Abstract

With the Android mobile phone market share rising and the number of its malicious software growing, an effective detection method of malicious software is very necessary. A malicious software detection algorithm considering the Android software multi-class features was proposed in this paper. This method extract the multi-class features sets of Android software and selects the corresponding optimal algorithm as the main algorithm and selects the sub-prime classification algorithm as the tuning algorithm for different feature sets. In each feature set, the classification results of main algorithm determine the adjustment extent of tuning algorithm. After that, the final result is based on the results of multi-class features sets so as to achieve more accurate recognition rate. The experimental results show that the accuracy of the method is 96%.

Keywords

malware application; machine learning; collaborative decision.

1. Introduction

In recent years, with the mobile Internet rapidly developing and intelligent terminal equipment being popular, smart phones become an indispensable part of people's lives. At present, the operating system of smart phone market are mainly iOS, Windows Phone and Android, Android is the fastest growing operating system of smart phone among them. Google has released a Linux-based operating system named Android in 2007, due to the open source and good compatibility of system, it has attracted a lot of program developers. According to a survey recently released by Statista, it show that Android mobile phone account for about 86.2% in the smart phone market by the end of July, 2016[1].Because of the openness of the system and the market, Android has become the target of malware developers at the same time.360 Internet Security Center has intercepted 14.03 million Android malicious program samples, it means increase 38,000 malicious programs and 700,000 mobile phones are infected by malicious programs per day[2].Once the user download the software containing malicious code, their mobile phone is likely to produce malicious charges, privacy theft and other hazards[3]. Therefore, it is very necessary to accurately and effectively identify Android malware.

At present, the research methods of malicious software detection based on Android platform can be divided into two types: signature-based detection and behavior-based detection which can be divided into static detection and dynamic detection[4]. Signature-based detection techniques is mainly based on the software signature, it has been widely used, but the technology must have some malware signature library before the detection of malware and this method will be invalid through a simple procedural trap. Feng.etl[5]proposed Apposcopy detection technology, which first sign the malicious software on the control flow and data flow. The component call graph and data flow graph of the application are obtained by static analysis technique, they can be matched with a known family signature to determine whether this application is malicious, but the method can’t detect software that uses obfuscation techniques. Zheng[6] proposed a detection method DroidAnalytics based on multi-level signature, which generate the signature through the look-up table and determine whether there is re-packing problem by judging the similarity of the two applications, but the detection efficiency is low.

Behavior-based detection techniques primarily use static or dynamic detection to monitor the behavior of software and match the behavior patterns of known malware. The static detection method extract grammatical features through source code analysis to compare with the syntax features of known malware in order to detect the malicious software. Felt[7]has developed a tool called Stowaway that automatically detects the permissions that the software applies and the actual used permissions in order to determine whether the software has applied for excessive permissions, but it can't solve the complex JAVA reflection mechanism. Yang Huan[8] put forward a kind of three-layer hybrid algorithm THEA which considers the behavior of Android and constructs different optimal classifiers for different characteristics to judge the behavior of malicious applications synthetically. But too few behavioral features are extracted and the dynamic simulation phase is too simple. Mu Zhang[9]proposed to use the function dependency graph to distinguish malicious software. By weighting the API dependency graph to prevent the malware byte code conversion from changing the result, the experimental results show that the accuracy of the method is 93%, but the method can't detect html5 software. Qian[10]achieve a behavioral analysis by adding dynamic monitor and request filtering model, but the method has the problem of correctness verification, which may lead to higher false alarm rate.

In addition, many researchers use machine learning to form classifiers in order to distinguish between benign and malware by analyzing the software's static or dynamic behavioral features recent years. Which machine learning algorithm are used, what features are extracted and how many features are extracted are the key factors that affect the validity of the classifier. Different classification algorithms have different detection effects on the same kind of features, and the same classification algorithm has different effects on different types of feature, so it is not ideal to choose a classification algorithm to detect many kinds of feature effects. In view of the above method, this paper proposes a multi-algorithm collaborative decision algorithm. Different feature set select the corresponding main classification algorithm and tuning classification algorithm according to the classification effect of the algorithm, the classification results of main algorithm determine the adjustment extent of tuning algorithm. Finally, the result will be obtained by merging classification results of many kinds of feature sets.

2. Related Information

2.1 APK structure

Android package is a file with ".apk" suffix name, we can get AndroidManifest.xml file, resource.arsc file, classes.dex file, META-INF folder, res folder by unzip the compressed package. Its internal structure shown in Fig.1. AndroidManifest.xml file is an essential configuration file and the configuration information is stored in the file for each Android software.res folder contains all the pictures, layout and other resource files. resources.arsc file store the compiled binary resource file. The classes.dex file is the byte code file generated by java source code.

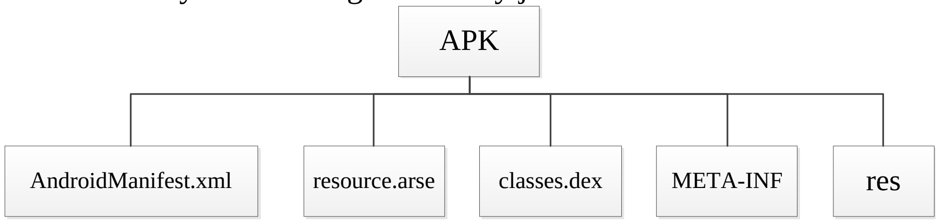

Fig.1 APK structure

The AndroidManifest.xml file contains information about the permissions, components, and Intents of the software application. Permission mechanism is one of the important security mechanisms for Android. When the application needs to access some user information or use some system functions, you must apply for certain permissions in the AndroidManifest.xml. An application that does not have any permission does not pose a threat to the user and malware usually applies for some permission that will not be applied for normal software, Such as reading the content of information, getting user location. In addition, the component information is also an important information in the

AndroidManifest.xml file. There are four different types of components for Android, they are Activity, Service, ContentProvider and BroadcastReceivers. Activity is usually responsible for managing the application's user interface and an application typically contains several Activity components. Service is mainly used to deal with the business logic which has nothing to do with the user interface, the implementation of the user specified the background operation and performs user-specified background operations. ContentProvider defines a standard mechanism that allows an application's specified data set to be available to other applications, other applications can obtain or store data from the Contentprovider through the ContentResolver class. BroadcastReceivers can filter external events and only respond to specific external events.

classes.dex is Dalvik byte code file and you can directly run it on the Dalvik virtual machine. we can get smali file and jar file by decompiling the Android software and get specific API calls by learning smali rules and reading jar source. Because some APIs have the authority to access sensitive information and resources in the phone, malware often calls these APIs to perform some malicious behavior which cause leak the user's privacy. For example the API for Internet traffic: execHttpRequest ().

The algorithm proposed in this paper is mainly based on the permissions, components and API used in the software. Therefore, it is very important to understand the file structure of APK and the characteristics of its analysis.

2.2 Machine learning

Large data technology continues to develop in recent years, researchers began using machine learning and data mining techniques in malicious code detection to identify benign and malware. Decision tree (DT), support vector machine (SVM), neural network (NN), logistic regression (LR) are commonly used.

The decision tree is a tree-like structure that is constructed from top to bottom. According to information gain measurement, the corresponding feature node is selected. Each internal node is a feature test. The branch represents the result of the test. The leaf node is a category, and the example of the same leaf node belongs to the same kind[11].

A neural network is a set of interconnected input and output units, each of which is associated with a weight. The correspondence between the input sample and its corresponding category is realized by adjusting the weight in the network learning stage.

Support vector machine is a two-class model, the core is to find a support vector in the training sample, so that it can build a best classification of the super-plane. That is, the distance of the support vector is maximum.

Logical regression is a very widely used classification algorithm which usually fitting the data into a logical function to predict the value of a discrete variable. The result is a predicted probability value, so the output value is between 0 and 1.

This paper mainly uses the above four classification algorithms. Each type of feature set is classified using four algorithms, according to the accuracy of the test results, we selects the corresponding optimal algorithm as the main algorithm and selects the sub-prime classification algorithm as the tuning algorithm for different feature sets.

3. Design

The malware detection process in this paper is divided into three parts: Apps sample collection module, feature extraction and processing module, detection module. The Apps sample collection module collects malicious apps and benign apps, and preprocesses the collected samples. The feature extraction and processing module mainly extracts the feature set and formats the features. The detection module uses the multi-algorithm collaborative decision proposed in this paper to classify the samples into malware and non-malware. as shown in Fig.2

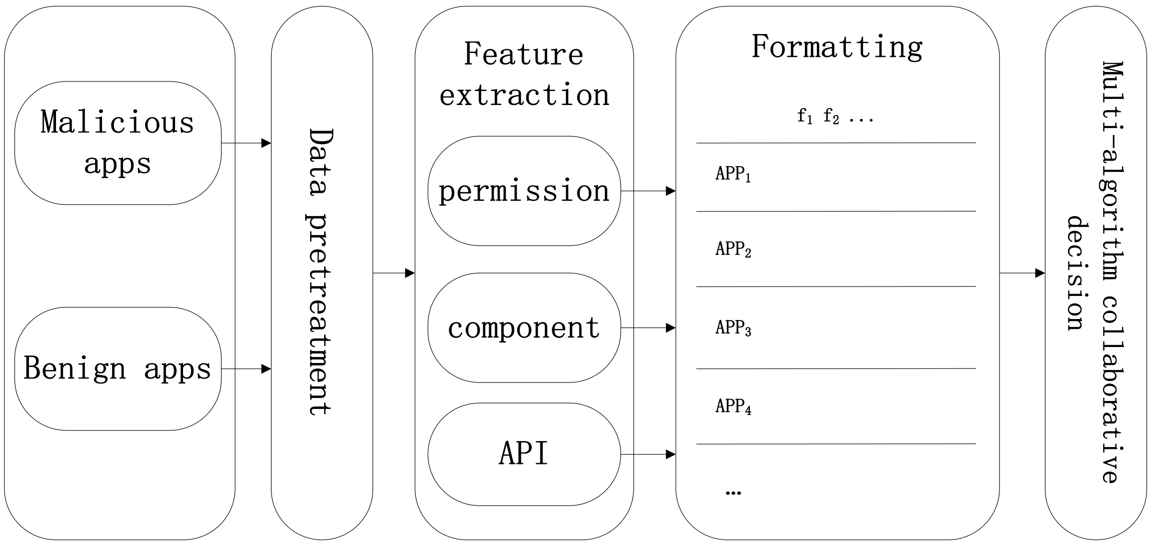

Fig.2 Basic workflow of design

3.1 Sample collection

This article creates the Apps sample database, which contains 746 benign software and 687 malware. Malware samples are available from virusshare.com. Benign software samples include social, video, life and other types of software were obtained by writing Python scripts. We have tested the benign software with malicious software detection tool in order to ensure the reliability of the sample, the results show that benign software samples are normal.

3.2 Feature extraction and processing

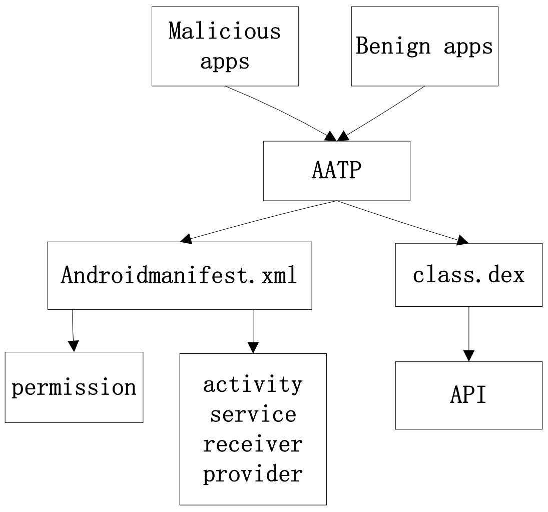

Fig.3 Feature extraction

This module mainly uses the static analysis technology to analyse the software as shown in Fig.3. We can extract the AndroidManifest.xml file and the Dex file for each software by using the AAPT that comes with the Android SDK.

Next, we extract the permissions from the AndroidManifest.xml file by writing the C ++ program. The tag <uses-permission/> describes the permission information that is required by the software. Most of the permissions are rarely applied by the literature [12], so the contribution of some permissions is very little in terms of detection and there may be misleading. In this paper, we carried out permissions feature extraction and statistics for the 746 normal software and 687 software malware, part of the statistical results shown in Fig.4. Frequently used permissions include: INTERNET, WAKE_LOCK, READ_PHONE_STATE, WRITE_EXTERNAL_STORAGE.

We can see that some applied permission indicate some behavior of malware through analysis. For example, some software leaks user privacy behavior, INTERNET and READ_CONTACTS may be frequently applied. Because the software get these permissions, it can read the user address book data and send it to the outside through the network. READ_PHONE_STATE permission is also frequently applied because it has the ability to obtain phone identification information, such as device ID.

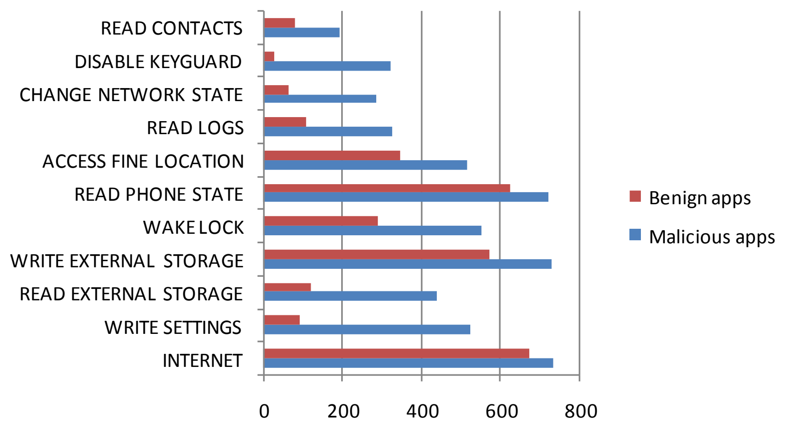

Fig.4 Percentage of the most frequently accessed permissions in malware and benign apps

The four components are defined in the AndroidManifest.xml configuration file and are managed by the system. it is possible to detect some known malware by extracting the names of these components, such as the malicious applications of the DroidKungFu family tend to use some specific Service components, so we also counted the number of four components used. The results are shown in Table 1, com.huajiao.lover.apk application declares 47 activities, 6 service, 3 receiver, and 1 provider.

Table 1 Usage of components

application	activity	service	receiver	provider
com.huajiao.lover.apk	47	6	3	1
com.homelink.apk	40	3	3	0
com.iqiuqiu.apk	25	5	5	1
...	...	...	...	...

API is the service interface provided by the operating system for the application, the application calls the API to achieve file access, network access, and other important resources. it will call a similar API function when the program implemente certain functions, so extracting API functions sequence features to identify malicious program behavior has become a common method[13]. Dex files are decompiled to get smali files and jar files and get the number of specific API calls by learning smali rules and reading jar source. For examplegetDeviceID (), setWifiEnabled () and so on.

The Weight is used when we filter some feature, the Weight is defined as

$$\mathrm{Weight} = \frac{|\mathrm{N_b} - \mathrm{N_m}|}{\mathrm{N_b} - \mathrm{N_m}} \tag{1}$$

$\mathrm{N_b}$ is the number of using certain feature of benign samples, $\mathrm{N_m}$ is the number of using certain feature of malware samples.

Each feature extracted by the static analysis technique contains multi-dimensional features. Beacause the large number of features will affect the detection time performance, and some features is easy to cause interference, so this paper uses Weight to filter the feature set. Finally, the remaining features are saved in vector form. For example the permission feature, a row vector represents permission feature of a software sample, each component attribute of the vector is called the permission name. the value of the attribute is 0 to indicate that the permission is not applied, the value of the attribute is 1 to indicate that the permission is requested.

3.3 MCD

Using a single algorithm to classify malware is too simple and different classification algorithms have different effects for different features. Therefore, this paper proposes a multi-algorithm collaborative decision algorithm for software detection, the algorithm is shown in Fig.5.

In order not to blindly choose the same algorithm to detect different feature sets, we have chosen a variety of classification algorithms for each feature set. Let $C_j(j = 1,2, \dots, n)$ be the set of classifiers, based on $F_i(i = 1,2, \dots, m)$, the j_{th} classifier makes a prediction that the software belongs to a class C_k with probability $P_{ij}(k)$. According to the accuracy of the test results, we choose the main classification algorithm $Z_i(i = 1,2, \dots, m)$ and the g optimal classification algorithm $Z_{ij}(i = 1,2, \dots, m, j = 1,2, \dots g)$ for each feature set $F_i(i = 1,2, \dots, m)$.

We define that using the corresponding tuning classification algorithm Z_{ij} and based on the feature F_i will makes a prediction that the software belongs to a class C_k with probability $P_{ij}(k)$, and using the corresponding main classification algorithm Z_i and based on the feature F_i will makes a prediction that the software belongs to a class C_k with probability $P_i^{init}(k)$.

$$P_i^{init}(0) = p(k = benign|F_i, Z_i) \tag{2}$$

$$P_i^{init}(1) = 1 - P_i^{init}(0) = p(k = malware|F_i, Z_i) \tag{3}$$

$1 \le i \le m$

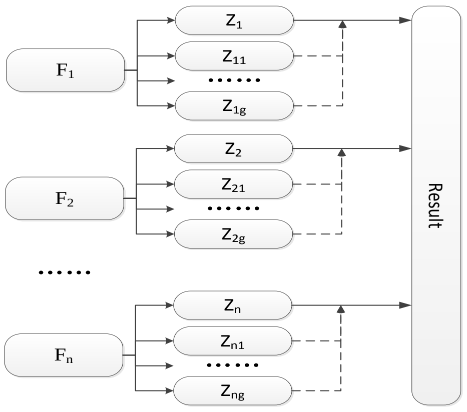

Fig.5 MCD

Each type of feature set gets g $P_{ij}(k)$ by the corresponding tuning classification algorithm. We will get a difference probability $P_i^{assist}(k)$ by comparing the difference between $P_{ij}(k)$ and $P_i^{init}(k)$ for the same type of feature set. The formula is as follows:

$$P_i^{assist}(k) = \frac{\sum_{j=1}^{g}(P_{ij}(k) - P_i^{init}(k))}{g} \tag{4}$$

The decision result of each feature set $P_i^{finally}(k)$ is not only determined by the main classification algorithm $P_i^{init}(k)$, but also by the difference probability $P_i^{assist}(k)$:

$$P_i^{finally}(k) = (1 - \omega_i)P_i^{init}(k) + \omega_i * P_i^{assist}(k) \tag{5}$$

$1 \le i \le m$, $\omega_i = \frac{1}{2} - |\frac{1}{2} - P_i^{init}(k)|$. we can see that when $P_i^{init}(k)$ is closer to 1 or 0 and the smaller ω_i will be, the results $P_i^{finally}(k)$ are mainly determined by the results of the main classification algorithm. On the contrary, $P_i^{finally}(k)$ will be adjusted by the tuning algorithm.

The final result of the software classification will be determined by $P_i^{finally}(k)$.

$$P(k) = \sum_{i=1}^{m}(\frac{acc_i}{acc} * P_i^{finally}(k)) \tag{6}$$

$$\mathrm{acc} = \sum_{i=1}^{m} \mathrm{acc}_i \tag{7}$$

acc_i indicates that the accuracy of using the main classification algorithm Z_i for the feature set F_i. The higher the accuracy is, the greater the final decision is. When $P(0) > P(1)$, the software will be determined as benign software.

4. Results and analysis

4.1 Data sets

The experiments were carried out on a computer armed with an Intel(R) i5-3337U CPU @ 3.30GHz, 8GB memory, 500GB hard disk space. In our experiments, we creates the Apps sample database, which contains 746 benign software and 687 malware. Malware samples are available from virusshare.com. Benign software samples be obtained by writing Python scripts. The sample database is shown in Table 2

Table 2 Apps sample database

sample	number	source
Malware apps	746	virusshare.com
Benign apps	687	Google Play

4.2 Evaluation

In this paper, benign software is defined as a positive tuple, and malware is defined as a negative tuple. True positive (TP) is a tuple that the classifier will correctly determine benign software as benign software. True negative (TN) is a tuple that the classifier will correctly determine malware as malware. False negative (FN) is a tuple that the classifier will determine benign software as malware. False positive (FP)is a tuple that the classifier will determine malware as benign software. In the experiments, the performance of each classification algorithm is evaluated using Precision, Recall, Accuracy and F-measure.

This experiment is divided into training part and testing part. First, the Apps database is divided into two parts: 80% of the software samples (including 597 benign software, 550 malware) as training sample set, 20% of the software samples (including 149 benign software, 137 malware) were used as test sample sets. Then, we need extract the feature for each sample, the original feature is filtered by Weight in order to reduce the operation time and the influence of some useless features.

Next, the main classification algorithm and the tuning classification algorithm of each feature set are selected by training the sample set. We use SVM, NN, LR, DT[14] algorithm to test the three types of features to get the average of the accuracy of the results. The results are shown in Table 3, The main classification algorithm of each feature set is selected: the DT algorithm performs better for the permission feature; the SVM algorithm works better for the component features; the SVM algorithm works better for the API feature.

Table 3 accuracy of various algorithms for three features

algorithms	Permission	Components	API
SVM	0.885	0.944	0.900
NN	0.899	0.936	0.871
LR	0.838	0.938	0.884
DT	0.904	0.936	0.894

According to the above experiments, the main classification algorithm and the tuning classification algorithm of each features are selected and the final result is obtained by multi-algorithm cooperative decision (MCD). In order to highlight the effectiveness of multi-algorithm collaborative decision method, the test results obtained by MCD are compared with the test results using a single classification algorithm as shown in Table 4. We can know that the multi-algorithm collaborative decision method proposed in this paper has obvious effect on the recognition of malware and the accuracy rate is 96.0%.

Table 4 Experimental result

algorithms	Precision	Recall	Accuracy	F-measure
SVM	0.934	0.890	0.910	0.911
NN	0.953	0.903	0.926	0.927
LR	0.897	0.869	0.881	0.882
DT	0.922	0.918	0.917	0.920
MCD	0.986	0.936	0.960	0.960

5. Conclusion

This paper presents a collaborative decision algorithm that takes into account features of Android software to detect Android malware. Extracting the multi-class feature set of Android software, the main classification algorithm and the optimal classification algorithm are selected for different feature sets and the classification results of main algorithm will determine the adjustment extent of tuning algorithm. After that, the final result is based on the results of multi-class features sets. This experiments show that this MCD improves the overall detection accuracy. The advantage of this approach is that the experiment not only uses three different types of features, but also detect each type of feature with the corresponding optimal algorithm, it focuses on samples that can't be determined by the main classification algorithm. But there are some shortcomings in the approach.

For future work, we plan to use more types of features such as hardware device information and utilize more types of algorithm to classify Android malware. Due to the approach is static, we hope to be able to combine dynamic detection techniques with static technology.

Acknowledgements

This work was supported by National Key R&D Plan of China (Grant No. 2017YFB0802203), National Natural Science Foundation of China (Grant Nos. U173620045, 61732021, 61472165 and 61373158), Guangdong Provincial Engineering Technology Research Center on Network Security Detection and Defence (Grant No. 2014B090904067), Guangdong Provincial Special Funds for Applied Technology Research and Development and Transformation of Important Scientific and Technological Achieve (Grant No. 2016B010124009), the Zhuhai Top Discipline--Information Security, Guangzhou Key Laboratory of Data Security and Privacy Preserving, Guangdong Key Laboratory of Data Security and Privacy Preserving.

References

[1] Information on http://www.199it.com/archives/509159.html.

[2] Information on http://www.360zhijia.com/360anquanke/178579.html.

[3] YANG Huan, ZHANG Yu-qing, HU Yu-pu, et al: Android malware detection method based on permission sequential pattern mining algorithm, Journal of Communication, Vol. 34 (2013) No.1, p.106.

[4] QING Si-Han: Research Progress on Android Security, Journal of Software, Vol. 27 (2016) No.1, p.45.

[5] Feng Y, Anand S, Dillig I, et al: Apposcopy: Semantics-based detection of android malware through static analysis, Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering(Hong Kong, China, November 16-21, 2014). p.576.

[6] Zheng M, Sun M, Lui J C S: Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware, Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on(Melbourne, VIC, Australia, July 16-18 2013). p.163.

[7] Felt A P, Chin E, Hanna S, et al: Android permissions demystified, Proceedings of the 18th ACM conference on Computer and communications security(Chicago, Illinois, USA, October 17-21, 2011). p.627.

[8] YANG Huan, ZHANG Yu-qing, HU Yu-pu, et al: A Malware Behavior Detection System of Android Applications Based on Multi-Class Features, Chinese Journal of Computer, Vol. 37 (2014) No.1, p.15.

[9] Zhang M, Duan Y, Yin H, et al: Semantics-aware android malware classification using weighted contextual api dependency graphs, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security(Scottsdale, Arizona, USA, November 3-7, 2014). p.1105.

[10] Qian Q, Cai J, Xie M, et al: Malicious behavior analysis for android applications, Int. J. Netw, Vol. 18 (2015) No.1, p.182.

[11] J.R. Quinlan, Induction of decision trees, Machine Learning, Vol. 1 (1986) No.1, p.81.

[12] BARRERA D, KAYACIK H G, Van Oorschot P C, et al: A methodology forempirical analysis of permission-based security models and its application toandroids, Proc of the 17th ACM Conference on Computer andCommunications Security(Chicago, Illinois, USA, October 4-8, 2010). p.73.

[13] Zhang Jia-wang, Li Yan-wei: Malware detection system implementation of Android application based on machine learning, Application Research of Computers, Vol. 34 (2017) No.6, p.1.

[14] Li Xiong-fei, Li Jun: Data mining and knowledge discovery (Higher Education Press, China 2003).