International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:9, 2016

A Static Android Malware Detection Based on Actual Used Permissions Combination and API Calls

Xiaoqing Wang, Junfeng Wang, Xiaolan Zhu

***Abstract*—**Android operating system has been recognized by most application developers because of its good open-source and compatibility, which enriches the categories of applications greatly. However, it has become the target of malware attackers due to the lack of strict security supervision mechanisms, which leads to the rapid growth of malware, thus bringing serious safety hazards to users. Therefore, it is critical to detect Android malware effectively. Generally, the permissions declared in the AndroidManifest.xml can reflect the function and behavior of the application to a large extent. Since current Android system has not any restrictions to the number of permissions that an application can request, developers tend to apply more than actually needed permissions in order to ensure the successful running of the application, which results in the abuse of permissions. However, some traditional detection methods only consider the requested permissions and ignore whether it is actually used, which leads to incorrect identification of some malwares. Therefore, a machine learning detection method based on the actually used permissions combination and API calls was put forward in this paper. Meanwhile, several experiments are conducted to evaluate our methodology. The result shows that it can detect unknown malware effectively with higher true positive rate and accuracy while maintaining a low false positive rate. Consequently, the AdaboostM1 (J48) classification algorithm based on information gain feature selection algorithm has the best detection result, which can achieve an accuracy of 99.8%, a true positive rate of 99.6% and a lowest false positive rate of 0.

***Keywords*—**Android, permissions combination, API calls, machine learning.

I. Introduction

IN recent years, Android smartphones play a more and more important role in people's daily life. Android, which is a Linux-based operating system proposed by Google in 2007, has attracted a large number of mobile phone manufacturers, developers and users because of its good open source and user experience. Therefore, it has accounted for absolute dominance among the mobile operating system. As stated by Chinese Internet Data Information Centre [1], Android operating system occupies as high as 53.54% of Global mobile operating system market share in 2015. In addition, according to strategy analysis, the global sales of smartphones have reached 1.3 billion and among them Android has occupied 81% of absolute advantage [2]. However, Android has become the target of attackers due to the lack of strict security review mechanisms, which provides a great convenience for malicious attackers and leads to the rapid growth of malware. Attackers often choose popular applications and republish them with malicious codes to attract users to download, which brings a huge security risk for users. According to the statistics of Never Quit "cloud security" monitoring platform, the number of mobile malware reached 85970 in the first half of 2014 and the year to year growth rate reached 68.3%. The infected phones reached 37.54 million and the year to year growth rate reached 78.6%, and 96% of mobile malwares were from Android platform [3]. As an open source operating system, Android has brought a serious security problem to users. All the time, Android's mechanism has not any restrictions for the number of permissions which an application can request, which results in the abuse of permissions and some malware are not correctly identified by traditional Android malware detection methods. Therefore, it is important to detect Android malware and its variants effectively.

In this paper, we present a machine learning detection method which is based on the actually used permission combinations and API calls. To validate the performance of this method, several experiments are conducted with 1170 malware and 1205 benign samples. Moreover, five different machine learning classification algorithms and two feature selection algorithms are used for distinguishing between malware and benign by using 10-fold cross-validation.

X. Q. Wang is with the Institute of Beijing System Engineering, China (e-mail: wang_xiaoqing@126.com).

J. F. Wang and X. L. Zhu are with College of Computer Science, Sichuan University, Chengdu, China (e-mail: wangjf@scu.edu.cn, 986712452@qq.com).

II. Related Work

The malicious code of Android platform is mainly in binary. There are many detection methods which include static analysis methods and dynamic analysis methods.

Static analysis methods detect malware by extracting static features from the disassembled codes decompiled by specific tools. Static analysis process does not need to execute the application, thus avoiding some problems caused by execution of applications such as space, time and resource consumption. Dynamic analysis methods detect malware by monitoring the behavior during the execution of applications, such as the sequence of function calls, the loading process of program and system resource accesses. The methods determine whether applications have malicious behavior through comparative analysis of similarity between applications and the known malware behavior model. Besides, dynamic analysis methods need to simulate the execution of the application in a virtual machine or sandbox, which is not suitable for Android platform with small storage and low execution efficiency. Considering above conditions and limitations, we focus on the research of static analysis methods.

Zhou et al. [4] collected about 1200 Android malware samples which cover large number of malware families and conducted several experiments in view of various respects such as installation, activation mechanism and loading of malicious codes. Besides, they detected these samples with anti-virus software including AVG, LookOut, Norton and TrendMicro. As a result, the detection result was 79.6% and 20.2% in the best and worst case detection rate. What's more, these samples are widely used in security fields and lay the foundation for the analysis of malware in Android.

Wei et al. [5] found that the number of dangerous permissions will increase along with the expansion of permission sets. In addition, they found a large numbers of applications applied for more permissions than the permissions they actually used. In other words, these applications do not obey the "principle of least privilege." [6]

Felt et al. [7] proposed a tool called Stowaway, which is used to detect whether the application is against "principle of least privilege" in according with the permissions which are declared in AndroidManifest.xml. In their research, about one-third applications have broken the principle among 940 samples.

Au et al. [8] proposed Pscout automated framework to improve the method that was put forward in [7]. They provided the whole mapping information between permissions and API calls through analysis of the permission system from Android 2.2 to 4.0.

Enck et al. [9] proposed Kirin security mechanism by defining rules which can identify dangerous permissions combination. Kirin will warn the danger when user tries to install applications. This mechanism prevents the intrusion of malware to some extent.

Fuchs et al. [10] put forward an automated tool named ScanDroid based on data stream static analysis technology. ScanDroid can be used to detect malware based on the consistency of the data stream and permissions requested in AndroidManifest.xml.

Sanz et al. [11] distinguished applications with machine learning classification algorithm such as Native Bayes, Support Vector Machine, K Nearest Neighbor, Decision Trees and Random Forest. Experiments show that Random Forest classification algorithm achieves the best result with a detection rate of 94.83%.

Aafer et al. [12] proposed a detection tool named DroidAPIMiner which can distinguish malware and benign samples with machine learning methods based on the API calls.

Yerima et al. [13] used Bayesian classification algorithm to classify 1000 malware samples and 1000 benign samples by extracting API calls and system commands from 49 malware families. The detection result shows that the method can obtain the detection rate of 90.6% and the accuracy rate of 92.1%.

Wu et al. [14] proposed detection framework named DroidMat. The experimental results show that the DroidMat can distinguish malware effectively and the recall rate of this method is higher than malware analysis tool Androguard.

III. Methodology

In this section, a machine learning detection method based on the actually used permissions combination and API calls is put forward. The framework of this paper is shown in Fig. 1.

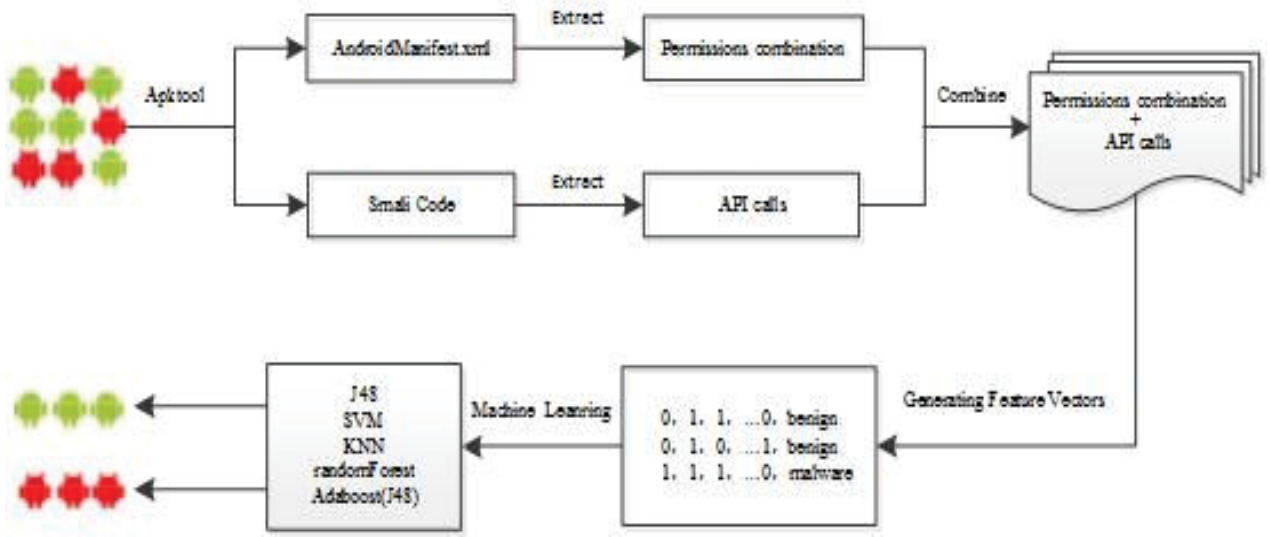

Fig. 1 The malware detection framework

The framework contains mainly four parts. The first part disassemble application with the Apk tool. The second part is designed to extract AndroidManifest.xml configure file and Smali codes from the previous dissembled codes. The third part aims to extract the actually used permissions combination from AndroidManifest.xml and API calls from Smali codes. As a result of this process, every application is represented as a feature vector with binary code 0 and 1. Finally, the last part makes full use of five classification algorithms with 10-fold

cross validation approach in machine learning to differentiate the applications to malware and benign categories.

Step 1: Extracting AndroidManifest.xml and Smali codes by Apk tool.

Step 2: Firstly, extracting the permissions that declared in AndroidManifest.xml with the package called xml.dom.minidom in Python. Secondly, extracting API calls through scanning Smali codes in according with the mapping relation between permissions and API, and get the actually used permissions. Finally, obtaining the actually used permissions combination based on the single permission.

Step 3: Generating feature vector, each application is represented as an instance. we uses 0 and 1 signifies whether this instance has current characteristic.1 means the current feature is appear in this application and 0 means the current application has not this feature. In addition, a class label is added in each instance to mark its category, a malicious sample is presented as malware and a benign sample is presented as benign.

Step 4: Using five machine learning classification algorithms, including J48, Random Forest, SVM, KNN and AdaboostM1 (J48), to realize the classification and evaluation for applications.

IV. Feature Extraction

A. Actually Used Permissions Combinations

Although permissions that requested in one application can reflect its function and behavior to a large extent. However, it is inaccurate to determine whether application contains malware behaviors in according with the declared permissions lonely. Therefore, this paper concentrates on the permissions that applications are used actually and then combines them with API calls based on PScout.

Through conducting statistical analysis between 1170 malware samples and 1205 benign samples in experimental data set, we found 873 samples do not actually used at least 1 declared permission and 524 samples do not actually used at least 2 declared permission. Figs. 2 and 3 give the top 20 permissions that are requested and actually used in these experimental samples in terms of the frequency of each permission.

Figs. 2 and 3 show that permissions abuse exists in both malware and benign samples. INTERNET, ACCESS_NETWORK_STATE and READ_PHONE_STATE permissions are widely used in both malware and benign samples. Therefore, it is inaccurate to determine the application class only rely on the permissions declared. For example, SEND_SMS permission is used for normal communication in benign samples, while malware samples use it to order some services related with charges automatically, thus bring economic losses to users. However, except these single permissions, their combination can be used to differentiate samples more accurately. For example, single permission READ_CONTACTS and INTERNET do not go against users generally, but great changes have happened when these two permissions are used simultaneously. Usually, malware often use READ_CONTACTS permission to read contacts and then use the INTERNET permission to send them to remote host, thus resulting in the revelation of users' privacy information. Therefore, we extract the top 20 permissions combination and their frequency in experimental data set with 1170 malware samples and 1205 benign samples, as is shown in Table I.

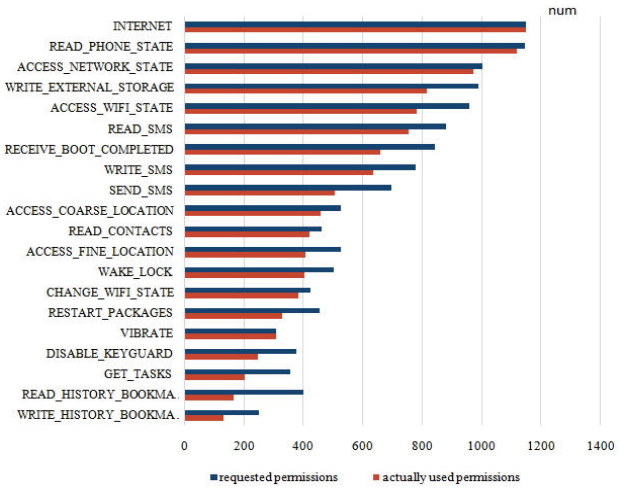

Fig. 2 Top 20 permissions that requested and actually used among malware samples

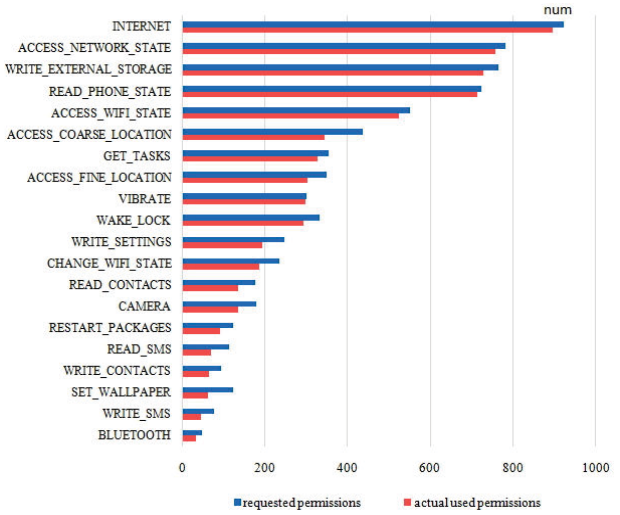

Fig. 3 Top 20 permissions that requested and actually used among benign samples

From Table I, we can conclude that the frequency of the same combination of permissions in malware samples is significantly higher than benign samples and their function is mainly concentrate on stealing of privacy data and Trojan control. For instance, the combination of INTERNET and READ_PHONE_STATE, as well as READ_PHONE_STATE and READ_SMS, can be used to reveal the IMEI of phone through internet or message. Besides, the PROCESS_OUTGOING_CALL, INTERNET and

RECORD_AUDIO permissions can be combined to monitor and record user conversations and transmit over the network.

B. API Calls

Android provides abundant API resources for applications to interact with the underlying operating system. Generally, an application is consisting of a large number of functions, and they often achieve its function by calling lots of system API functions. Therefore, in this section, we try to identify malware based on the features of API calls in an application.

First of all, extract the Smali codes with the Apktool. Then, scan these codes to extract functions that invoked in application and select those system API calls according to PScout. Through the statistical analysis of 1170 malware samples and 1205 benign samples, we display the top 20 API calls based on the frequencies as shown in Figure 4.

From Fig. 4, compared with benign samples, malware samples tend to call more particular APIs to implement their malicious purpose. These APIs mainly involves reading users' data, connecting with internet and tracking users' location, such as getDeviceId, getSubscriberId, getNetworkInfo, sendTextMessage. This fact illustrates that it is feasible to identify malware behavior in according with the call frequency of particular APIs.

TABLE I
TOP 20 PERMISSION COMBINATIONS BETWEEN MALWARE AND BENIGN SAMPLES

Permissions combination	malware	benign
INTERNET	1149	896
READ_PHONE_STATE	1118	714
INTERNET, READ_PHONE_STATE	981	568
ACCESS_NETWORK_STATE	972	757
WRITE_EXTERNAL_STORAGE	813	727
ACCESS_WIFI_STATE	780	523
READ_SMS	752	68
RECEIVE_BOOT_COMPLETED	659	243
WRITE_SMS	633	45
INTERNET, WRITE_EXTERNAL_STORAGE	597	550
SEND_SMS	504	67
INTERNET, ACCESS_NETWORK_STATE, READ_SMS	455	25
READ_PHONE_STATE, READ_SMS	425	26
INTERNET,ACCESS_WIFI_STATE, RECEIVE_BOOT_COMPLETED	422	90
INTERNET, ACCESS_WIFI_STATE, WRITE_SMS	422	28
INTERNET, READ_CONTACTS	420	135
INTERNET, READ_PHONE_STATE,ACCESS_WIFI_STATE	413	221
ACCESS_NETWORK_STATE, READ_SMS, WRITE_SMS	372	10
PROCESS_OUTGOING_CALL, INTERNET, RECORD_AUDIO	366	97
RECEIVE_SMS, SEND_SMS	360	16

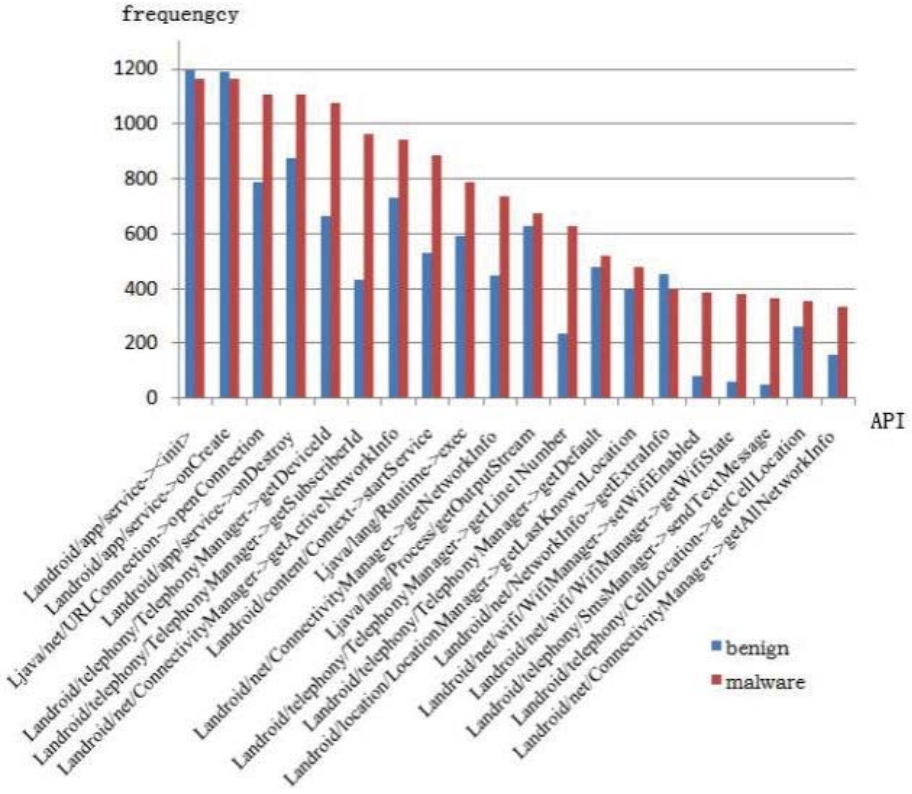

Fig. 4 Top 20 API calls between malware and benign samples based on their frequency

V. Experiments and Discussion

A. Data Set

In order to verify the effectiveness of this method, we collected a total of 2375 Android applications. Among them, the 1170 malware samples are composed of 23 families from genetic engineering [15]. Meanwhile, we get 1205 benign samples with web crawler technology from Google official market and Anzhi market to maintain the equivalent number of malware samples. Table II gives the structure of the data set that used in our methodology.

TABLE II
The Component of Data Sets

Data set	Samples	Malware Families/Categories
malware	1170	ADRD, AnserverBot, Asroot, BaseBridge, BeanBot, Bgserv, Crusewin, DroidKungFu1, DroidKungFu2, DroidKungFu3, DroidKungFu4, DroidDream, DroidDreamLight, FakePlyer, Geinimi, GoldDream, Pjapps, Plankton, KMin, SndApps, RogueSppush, YZHC, ZSone。
benign	1205	safety, tools, browsers, input methods, personalization, shopping, society, games, entertainment, reading, video, finance, life.

B. Evaluation Metrics

In this paper, we use the following four metrics in order to access the classification capability of different classification algorithms.

a. **True Positive Rate (TPR):** The number of malware samples that were correctly classified as malware in all of the malware samples.

$$TPR = \frac{TP}{TP + FN} \quad (1)$$

where TP is the number of malware samples that were correctly classified as malware and FN is the number of malware that were incorrectly classified.

b. **False Positive Rate (FPR)**: The number of benign samples that were incorrectly classified as malware in all of the benign samples.

$$FPR = \frac{FP}{FP + TN} \quad (2)$$

where FP is the number of benign samples that were incorrectly classified as malware and FN is the number of benign samples that were correctly classified.

c. **Accu (Accuracy):** The number of benign samples and malware samples that were correctly classified in all of the samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

d. **AUC (Area Under roc Curve)**: The area of ROC (Receiver Operating Characteristic), which is the part of the area that obtained under the curve that based on detection rate for the vertical axis while the false positive rate for the horizontal axis. Generally, the larger area indicates the better performance of classifier.

C. Results

In this section, we evaluate the classification performance of five different algorithms in terms of feature sets that have been extracted from applications, including API calls, permissions combination, the combination of actually used permissions combination and API calls, requested permissions. In addition, information gain and CFS feature selection algorithms are used to select the most useful features to improve the efficiency of classifiers. The purpose of this paper is to demonstrate that the feature of actually used permissions combination an API calls can achieve better performance. Therefore, four benchmarks were established to evaluate the ability of different classifiers. All experiments are on the base of 10 fold cross validation.

a. Actually Used Permissions Combination

The total number of feature set consists of 200 and 22 permission combinations after using information gain and CFS selection algorithms respectively. The detection results are shown in Table III. From Table III, we can conclude that the optimal detection result is obtained by AdaboostM1 (J48) classification algorithm based on the above two feature selection algorithms, achieving a detection rate of 99%, an accuracy of 99.3% and AUC of 0.994. Compared with CFS method, KNN classification algorithm can obtain a better detection result through information gain method, but a lower detection rate than CFS with respect to libSVM algorithm.

TABLE III
Performance of Classifiers Obtained When Using Permissions Combination Feature set with IG and CFS Methods

Algorithm	IG				CFS			
	TPR (%)	FPR (%)	Accu (%)	AUC	TPR (%)	FPR (%)	Accu (%)	AUC
J48	98.2	0.3	98.9	0.986	98.1	0.4	98.9	0.983
RandomForest	98.6	0.2	99.2	0.993	98.6	0.2	99.2	0.993
KNN	98.5	0.3	99.1	0.992	98.2	0.2	99.0	0.991
libSVM	97.2	0.2	98.5	0.983	98.0	0.4	98.8	0.987
AdaboostM1	99.0	0.3	99.3	0.994	99.0	0.3	99.3	0.994

b. API Calls

The total number of feature set consists of 100 and 6 permission combinations after using information gain and CFS selection algorithms respectively. The detection results achieved from this kind of feature are shown in Table IV.

TABLE IV
Performance of classifiers obtained when using API calls feature set with IG and CFS methods

Algorithm	IG				CFS			
	TPR (%)	FPR (%)	Accu (%)	AUC	TPR (%)	FPR (%)	Accu (%)	AUC
J48	95.5	5.4	95.0	0.958	95.1	7.1	94.0	0.954
RandomForest	96.4	4.4	96.0	0.987	93.8	5.0	94.4	0.982
KNN	96.1	5.6	95.2	0.983	94.4	6.2	93.7	0.98
libSVM	92.7	5.1	93.8	0.949	94.0	7.9	93.0	0.927
AdaboostM1	96.8	3.1	96.8	0.989	93.9	5.6	94.1	0.978

As shown in Table IV, the best classification algorithm is obtained by AdaboostM1 (J48) algorithm in according information gain feature selection method, which can achieve an accuracy of 96.8 and AUC of 0.989. Meanwhile, the best detection result is obtained by Random Forest algorithm after CFS method is adopted, achieving an accuracy of 94.4% and AUC of 0.982.

c. A Combination of Actually Used Permissions Combination and API Calls

The total number of feature set consists of 300 and 28 permission combinations after using information gain and CFS selection algorithms respectively. The detection results achieved from this kind of feature are shown in Table V. From Table V, we can conclude that these classifiers can achieve good performance when the actually used permissions combination and API class are integrated with respect to two feature selection methods: information gain and CFS. The best detection result is obtained by AdaboostM1 (J48) classifier with a detection rate as high as 99.6%, an accuracy of 99.8%, AUC of 0.999 while a false positive rate as low as 0. In general, the combined feature set can achieve higher accuracy while maintaining a lowest false positive rate, which can improve the ability to identity of the malware in the wild.

VI. Comparison with Related Work

In order to verify the efficiency of our method, we conducted several experiments to make comparisons with related methods, including permissions proposed in [11], single permissions combination and single API calls feature sets. At the same time, we implemented the detection method in [11] that is based on the declared permissions in applications. The total number of feature set is consisting of 100 and 30 after using information gain and CFS selection algorithms to combine the above feature sets respectively. The detection results achieved from this kind of feature are shown in Table VI. As shown in Table VI, the best performance is achieved by AdaboostM1 (J48) classifier based on the above two feature selection methods. However, this feature set can lead to a high false positive rate because that most applications have broken the "principle of least privilege".

The detection rate, AUC and false positive rate with the best performance based on information gain and CFS feature selection algorithms are shown in Figs. 5, 6 and 7, respectively.

TABLE V
Performance of Classifiers Obtained When Using the Combination of Permissions Combination and API Calls Feature Set with IG and CFS Methods

Algorithm	IG				CFS			
	TPR (%)	FPR (%)	Accu (%)	AUC	TPR (%)	FPR (%)	Accu (%)	AUC
J48	99.2	0.5	99.4	0.994	98.8	0.1	99.4	0.996
RandomForest	99.4	0	99.7	0.998	99.3	0.1	99.6	0.999
KNN	99.1	0.3	99.4	0.994	99.2	0.2	99.5	0.997
libSVM	98.1	0.1	99.0	0.988	98.6	0.2	99.2	0.991
AdaboostM1	99.6	0	99.8	0.999	99.3	0.1	99.6	0.998

TABLE VI
Performance of Classifiers Obtained When Using Declared Permissions Feature Set with IG and CFS Methods

Algorithm	IG				CFS			
	TPR (%)	FPR (%)	Accu (%)	AUC	TPR (%)	FPR (%)	Accu (%)	AUC
J48	90.8	5.9	92.5	0.949	91.0	6.1	92.5	0.948
RandomForest	95.6	4.6	95.5	0.987	92.5	4.2	94.2	0.98
KNN	96.8	7.5	94.6	0.984	92.8	6.0	93.3	0.97
libSVM	90.3	6.5	92.0	0.919	89.2	6.5	91.4	0.919
AdaboostM1	96.0	4.6	95.7	0.987	95.2	4.6	95.3	0.987

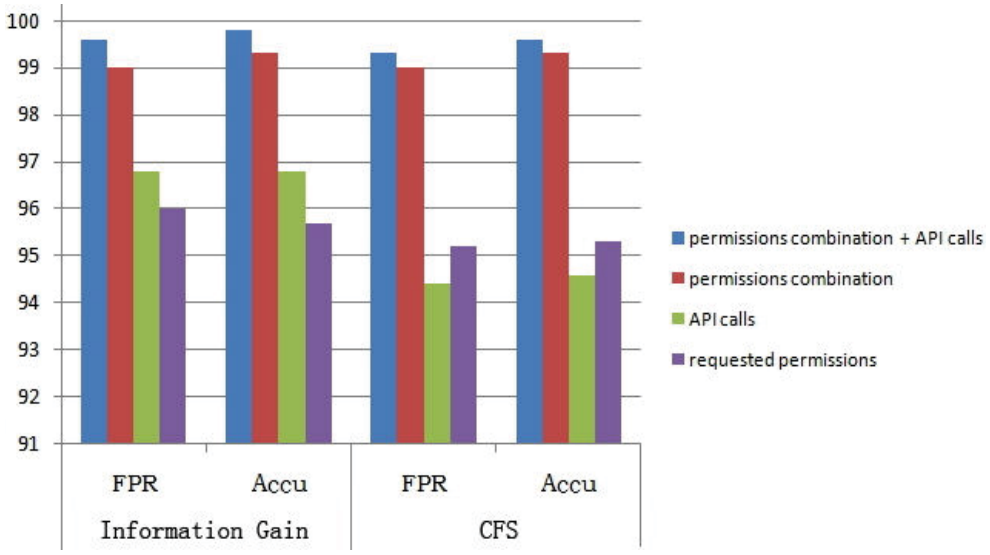

Fig. 5 Comparisons of TPR and Accu with best classifier under four kinds of data sets when information gain and CFS methods are used

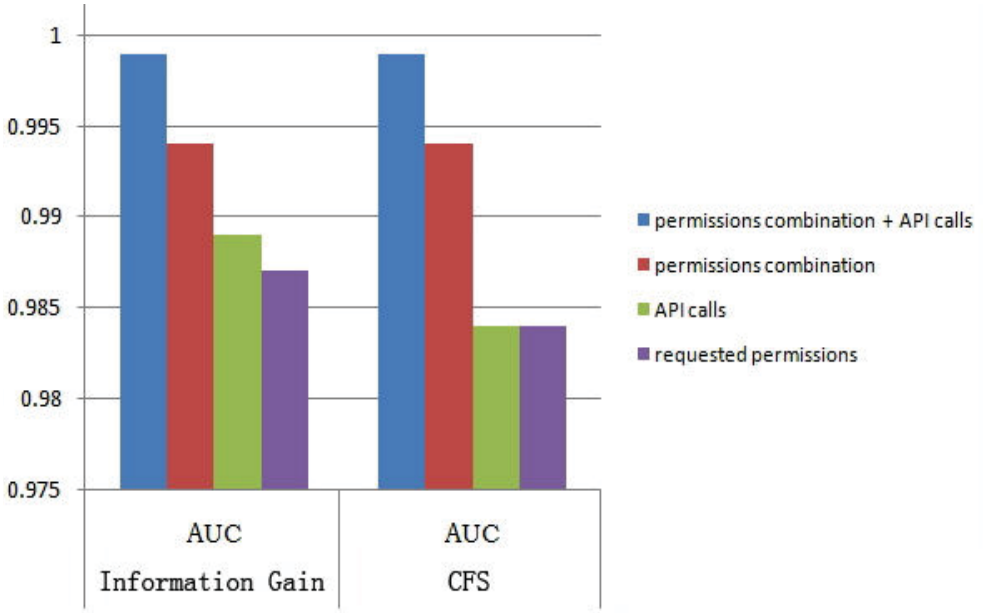

Fig. 6 Comparisons of AUC with best classifier under four kinds of data sets when information gain and CFS methods are used

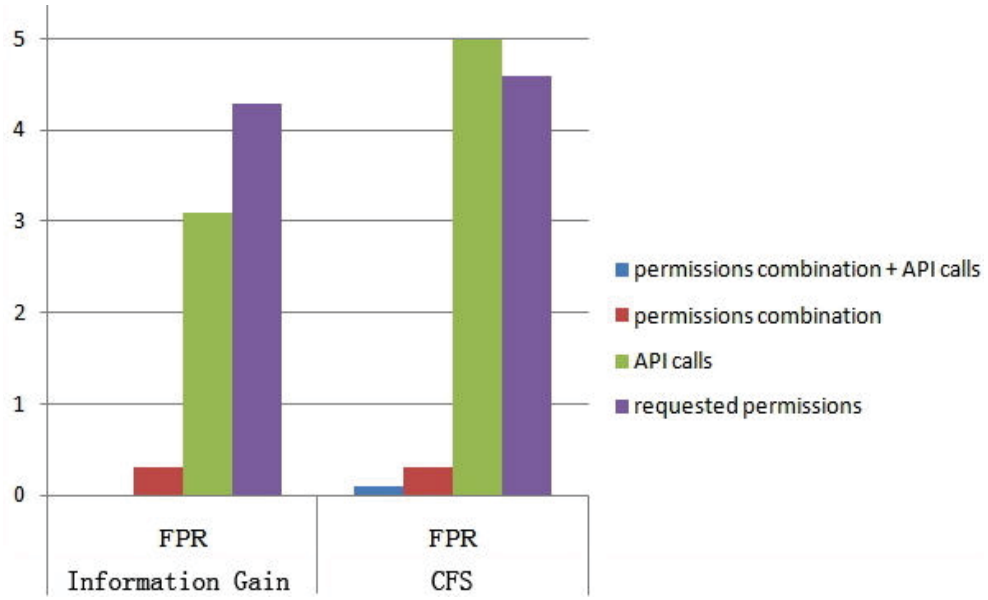

Fig. 7 Comparisons of FPR with best classifier under four kinds of data sets when information gain and CFS methods are used

Besides, Figs. 8 and 9 give the ROC curve based on the best classifier with information gain and CFS feature selection methods. The closer to the top left means the performance of this classifier is more excellent.

From above comparisons, we can draw the following conclusions: Compared with method that proposed in [11], our method can improve the performance of classifiers significantly. Secondly, the classifiers based on the combination of actually used permissions can achieve better identification. At last, no matter what kind of feature selection algorithm is chosen, our method can obtain higher detection rate, accuracy, AUC and lower false positive rate compared with other feature sets under same experimental conditions.

VII. Conclusion

In this paper, a machine learning detection method that based on the actually used permissions combination and API calls was put forward. Our main contributions are as follows: Firstly, we presented an Android malware detection method based on its actually used permissions combination and API calls rather than permissions just declared in AndroidManifest.xml. Secondly, various machine learning algorithms, feature selection methods and experimental samples are used to

validate the efficiency of our method. Finally, the detection results show that this method can improve the performance of classifiers significantly and is more accurate than the method that was put forward in [11]. In future studies, more useful characteristics could be extracted to achieve better results. In addition, integration of multiple classifiers could also be used to improve the identification of classifiers further.

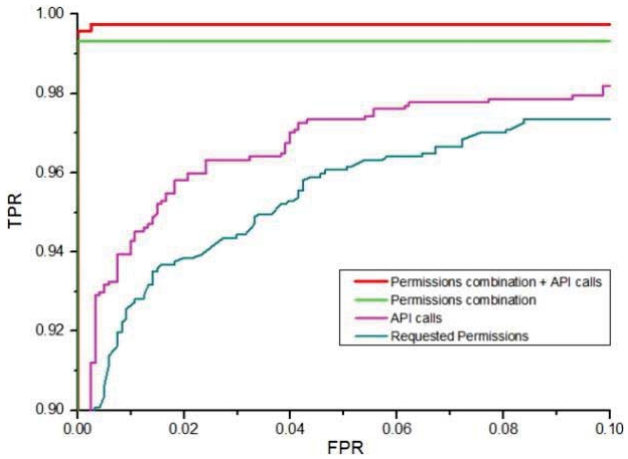

Fig. 8 Comparisons of ROC curve with best classifier under four data sets when Information Gain method is used

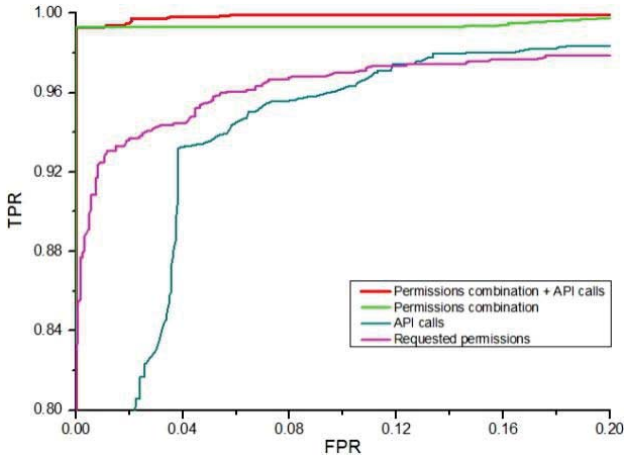

Fig. 9 Comparisons of ROC curve with best classifier under four data sets when CFS method is used

REFERENCES

[1] Chinese Internet Data Information Centre(199IT), http://www.199it.com/archives/390550.html
[2] Report of China Internet Network Information Center (CNNIC), The 37th China Internet Development Statistics(EB). https://www.cnnic.cn/hlwfzyj/hlwxzbg/201601/P020160122469130059846.pdf
[3] NetQin (Never Quit), 2014 in the first quarter of the global mobile security message(EB).
[4] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution(C)//Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012: 95-109.
[5] Wei X, Gomez L, Neamtiu I, Faloutsos M. Permission evolution in the Android ecosystem. In: Proc. of the 28th Annual Computer Security Applications Conf. (ACSAC 2012). 2012. 31−40
[6] Saltzer JH. Protection and the control of information sharing in Multics. Communications of the ACM, 1974,17(7):388−402.
[7] Felt A P, Chin E, Hanna S, et al. Android permissions demystified(C)// Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011: 627-638.
[8] Au K W Y, Zhou Y F, Huang Z, et al. Pscout: analyzing the android permission specification (C)// Proceedings of the 2012 ACM conference on Computer and communications security. ACM,2012: 217-228.
[9] Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification. In: Proc. of the 16th ACM Conf. on Computer and Communications Security (CCS 2009). 2009. 235−245
[10] Fuchs A P, Chaudhuri A, Foster J S. Scandroid: Automated security certification of android(J). 2009.
[11] Sanz B, Santos I, Laorden C, et al. MAMA: manifest analysis for malware detection in android(J). Cybernetics and Systems, 2013, 44(6-7): 469-488.
[12] Aafer Y, Du W, Yin H. DroidAPIMiner: Mining API-level features for robust malware detection in android(M)//Security and Privacy in Communication Networks. Springer International Publishing, 2013: 86-103.
[13] Yerima S Y, Sezer S, McWilliams G, et al. A new android malware detection approach using Bayesian classification(C)// Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on. IEEE, 2013: 121-128.
[14] Wu D J, Mao C H, Wei T E, et al. Droidmat: Android malware detection through manifest and api calls tracing(C)// Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE, 2012: 62-69.
[15] Zhou Y, Jiang X. Android malware genome project (EB/OL). IEEE,2012, (2014-02-27).