ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 531-538
doi: 10.18178/wcse.2018.06.091

A Novel Method of Android Malware Detection Based on Ensemble Learning Algorithm

Jingyi Zhao [1,+], Xiuliang Mo [2] and Qiao Zheng [3]

[1+] Tianjin Intelligent Computing and Software New Technology Key Laboratory, School of computer Science and Engineering, Tianjin University of Technology, China

[2] Tianjin Intelligent Computing and Software New Technology Key Laboratory, School of computer Science and Engineering, Tianjin University of Technology, China

[3] Tianjin Intelligent Computing and Software New Technology Key Laboratory, School of computer Science and Engineering, Tianjin University of Technology, China

Abstract. Android is popular for mobile devices in recent years. As the risk of malware is sharply increasing in Android platform, Android malware detection has become an important research topic. However, the current work shows that the detection of malware still needs to be reformed. Many current results suggest a modification of permissions or a combination of permissions and intents, but Android's fragmentation issues and requiring rooting, hindering the widespread adoption of those methods. Many approaches have been proposed to detect this attack by modifying the Android OS. Existing anti-viruses depend on signature databases that need to be updated from time to time and are unable to detect zero-day malware. The Android Operating system allows inter-application communication through the use of component reuse by using intents. A malware detection model is developed based on ensemble learning algorithms and on the basis of the random forest in WEKA. The method we proposed can take a better efficiency and precision.

Keywords: Android Malware, Ensemble Learning, Random Forest, Classification

1. Introduction

As the performance of mobile phone hardware continues to evolve, smart phones become more and more popular in people's daily lives. Currently on the smartphone market, operating system mainly includes Android, iOS, Windows Phone and Blackberry OS. According to IDC [1] statistics, in the first quarter of 2017, the market share of Android operating system up to 85.1% shares, much higher than i0S and several other operating systems. Android operating system continues to grow and develop, naturally led to the growing Android application. As of now, there are nearly 2 million apps available for downloading from Google Play, and more than 50 billion downloads to date with an average of more than 500 new applications entering the application market each day. Besides the Google Play market, there are many other third-party application download platforms with no supervision. However, the supervision of the relevant departments is often limited, leading to malicious applications flow to the market. Through the people's download crazy spread, application market causes huge losses to users. To efficiently detect malware from applications available on official and third-party sources, many efforts have contributed to studying the nature of smartphone platforms and their applications in the past decade. A malicious application can inject malicious messages to result in user data breaches and to violate application security policies, that may create room for attacker exploitation. Malicious applications disguise themselves among the hundreds of thousands of normal ones. Therefore, how to effectively detect malicious applications from a large number of programs to

[+] Corresponding author. Tel.: + 86 - 13031891818; fax: +86-02260216906.
E-mail address: 592753744@qq.com.

protect the security and interests of Android users is an essential and urgent challenge for researchers. This paper aimed at the Android application permissions and intents mechanism, proposed a method with a combination of permissions and intents features. First, extract the application in the set of permissions as a feature set; then, intents will be another prepared features; finally, combine Classification algorithm for malicious applications and ordinary applications of screening. Experiment result shows that it is compared with the method of discriminating using permission and intents as the feature. This method can distinguish malware more accurately and reflect more completely with the characteristics of the application.

Section 2 of this paper introduces the related work; Section 3 describes the permissions and intent features and the method of combining; Section 4 shows the effectiveness of the method through simulation experiments; finally, Summary of the full text.

2. Related Work

The increased usage of smart phones can be combined with tremendous increase in the security breaches due to the exploitation of the increasing number of android application related vulnerabilities [1]. Faruki et al. [2] presented an overview on the current malware trends. Malware analysis generally uses static, dynamic, and hybrid methods for analysis. In static malware analysis, properties of apps were extracted by analyzing different static features without running the code. In dynamic analysis, the runtime profiles of apps were generated by monitoring and collecting the memory consumption, CPU usage, battery usage and network traffic statistics Shabtai et al. [3] and Burguera et al. [4].

Currently, many classification algorithms in machine learning are applied to Android malware detection. Zhang et al. [5] proposed a naive behavior recognition method based on naive Bayes (NB) for Android applications. It extracted whether the software applied for too many permissions and whether there was a combination of sensitive permissions as a classification attribute. By extending the Android security framework, the method implemented the real-time analysis and the treatment to the malicious behavior. Zhang et al. [6]use the relation between Android right, Android privilege and malicious intention of software, and achieved the detection of malware by improving Bayesian algorithm. Xu et al. [7] used Android privilege information as feature and uses information gain (IG) algorithm to optimize it, and then used Laplace calculus and multiplier to take natural logarithm to improve NB algorithm, and analysis and test for Android malware. However, the above research only detects and analyzes the permission information of the Android application, and the detection range is not comprehensive enough. In [8], an SVM classifier was created using a combination of dangerous API calls and permissions to automatically identify malware. In [9], two kinds of clustering algorithms, k-means algorithm and mini batch k-means, are used to classify malware. Although the detection of Android malware has been implemented in literature [8,9], the detection accuracy is not high enough. In [10], Droid Detector, an online malware detection tool, is implemented by using depth learning algorithm, which realizes the online detection and analysis of Android applications. However, its algorithm is more complex and consumes a lot of computer memory. Literature [11] proposed collaborative malicious code detection scheme based on the mobile terminal and the server side. Literature [12] designed a three-layer hybrid ensemble algorithm (THEA), and realized the detection tool Androdect for malicious code However, these two methods are relatively complicated in terms of technical implementation.

Petra’s et al. [13] shows that dynamic program analysis alone is not enough to assure Android security. They present a broad range of anti-analysis techniques for malware to successfully evade dynamic analysis. DREBIN [14] combines static analysis and machine learning techniques to detect Android malware. The experimental result shows that DREBIN can achieve high detection accuracy by using as many features as possible to aid detection. However, using more features also increases the overhead of their system.

This paper this paper proposes a hybrid features method based on ensemble learning algorithm, to detect zero-day malware and improve the accuracy of malware detection. Firstly, we preprocess the apks which downloaded from application markets, and then extract the features like permissions and intents to build up the dataset for training and testing. Furthermore, we have used K-Means clustering to obtain k disjoint clusters on training datasets each cluster depicts a region of similar features instances in terms of Euclidean distances between the instances and their cluster centroids. Finally, the proposed ensemble algorithm is used for features classification, and the classification result is used to verify the effectiveness of the proposed method of android malware detection.

3. Model Establishment

The model aims at developing a combination that focuses on feature extraction and selection to measure and characterize the malicious applications on the basis of permissions and intents specified in the manifest file of the application. The features constructed from the selected feature sets are fed to the classifier using WEKA. According to the hybrid features, the model was proposed as depicted in Fig. 1.

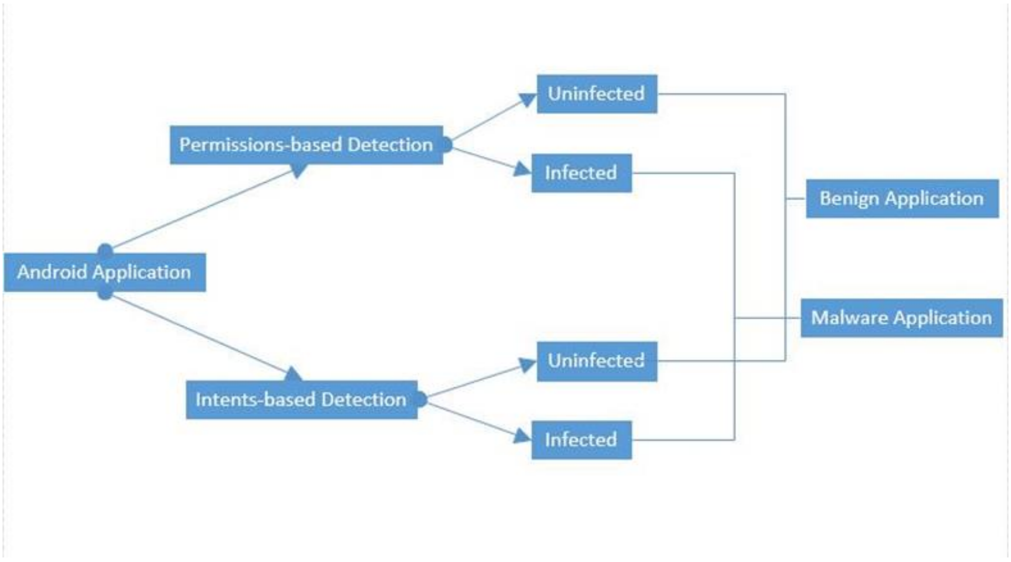

Fig. 1: Model of Hybrid features.

3.1. Features Clustering

The selected features are collected into the signature database and divided into training data and test data and used by standard machine learning techniques to detect the android malware applications. In the first step we have used K-Means clustering to obtain k disjoint clusters on training datasets each cluster depicts a region of similar features instances in terms of Euclidean distances between the instances and their cluster centroids. However, K-Means clustering may lead to anomalies in case of overlapping type of data. So we use decision tree classifiers to classify each cluster. K-means method is cascaded with decision tree learning by using the instances in each K-means cluster. However, K-Means clustering may lead to anomalies in case of overlapping type of data. So we use decision tree classifiers to classify each cluster. K-means method is cascaded with decision tree learning by using the instances in each K-means cluster. We have used K-means clustering as it guarantees at least a local minimum of the criterion function, thereby accelerating the convergence of clusters on large datasets. It is an iterative algorithm which organizes the training set in vectors with a dimension equal to the number of features to be evaluated leading to k number of clusters.

The algorithm K-means description as follow:

1. Calculate the centroid for each cluster.

2. Calculate the distance of the training set vectors from each cluster centroid

3. Assign the training set vector to the closest cluster centroid.

4. Move the cluster centroids to the mean of the respective cluster’s training sets. The steps are repeated until the algorithm converges. Convergence is achieved when the second step no longer assigns any vectors to new cluster centroids. The distance between the n-dimensional vectors (x) and a given cluster centroid is given by: $||x-\mu||$^2. where x is a training example, and μ is the cluster centroid.

3.2. Features Classification

3.2.1. Ensemble Learning

Ensemble Learning is a tool in machine learning algorithms that is very powerful. Due to its versatility and effectiveness, it is widely used in various machine learning and data mining competitions. Ensemble Learning is accomplished by building and combining multiple learners. It first generates base classifiers and then uses some sort of strategy to combine them. It is divided into two algorithms: Bagging and Boosting.

3.2.2. Random Forest

Random forest is a Bagging algorithm based on decision tree, but it introduces random attribute selection in the training of decision tree.

Random forests are more efficient, simpler, require less computing resources, and can demonstrate great performance in many display engineering tasks. So when choosing a classifier for training, this article used Random Forest as a training algorithm.

As the training set is usually very large, the branches and layers of generated tree using random forest algorithm are more, so it is necessary to prune the decision tree. We have also used random forest classification algorithms to obtain a relationship between the attributes that would make it possible to detect whether an app is infected or not.

3.3. Features Extraction

It's necessary that we should preprocess the dataset download from application market to collect relevant features. We used Dex2jar to disassemble the dataset. Features are extracted from AndroidManifest.xml and some of them would be listed in Table 1.

The AndroidManifest.xml file would be converted into .arff format and be analyzed in WEKA. The permissions requested that are specified in the <uses-permission> and the intents can thus be extracted and used as features for classification. Dataset are created one containing the permissions requested by all the apps and the other containing the intents of the apps.

The permission dataset include the various attributes such as standard permissions, the name of the malware in case of malicious app, infected attribute indicates whether the app is infected or non-infected app. The intent dataset consists of the flag to indicate the presence or lack of an intent marked as "1" or a "0" in the respective column. Figure 2 shows some of the Android applications permissions used for malware detection.

Table1: Features used for classification

Descriptions	Permissions	Intents
Message	Send_SMS	SMS_recieved
	Receive_SMS	
	Read_SMS	New_outgoing_call
	Write_SMS	SMS
Install and delete packages		Install_application
		Install_packages
		Delete_packages
	Internet	Connectivity
Read Phone State	Call_Phone	Phone_state
	Read_phone_state	Modify_Phone_State
	Change_Configuration	Boot_completed

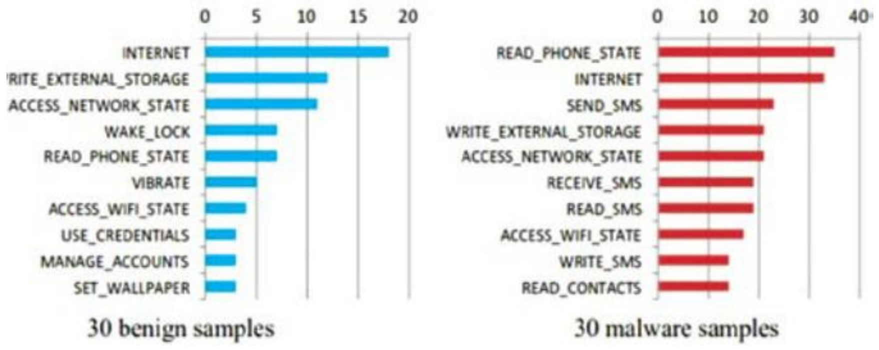

Fig. 2: Android applications permissions used for malware detection.

3.3.1. Permissions

Some of the malicious permissions are explained as follows: Text messaging permissions, Read Phone State, Install and delete packages, Internet, Change Configuration.

3.3.2. Intents

Intent is the basic communication mechanism used to exchange the inter- and intra-app messages. An intent conveys the intention of the app to perform some action. It specifies the label for a component, its category and action to be performed. Intents are of two types: Explicit and Implicit. Explicit intent specifies the component exclusively by the class name. These are generally used by apps to start their own components. Implicit intent does not specify the component by name. It states the required action only; system selects the app that has the component to handle the stated action. With explicit intent, the system launches the specified component immediately while with implicit intents, system looks for the appropriate component by comparing the intent filters. If there is any match between the intent and intent filter, the component of that app is launched. In case of multiple matching intent filters, users are sent with a dialogue box to select the app. Intents facilitate apps with same user ID to use each other's functionalities without separately declaring the permissions for them. This helps apps to gain extra privileges by augmenting the permissions.

Some of the common intents used in applications are analyzed to understand their functioning.

a) ACTION_DIAL: Perform a call to someone specified by the data. This Intent cannot be used to call emergency numbers or premium numbers by malicious applications.

b) ACTION _CALL _BUTTON : This intent is used to involve the dialer by pressing the "call button"

c) ACTION_CREATE_SHORTCUT Output: An Intent representing the shortcut is used to create a SHORTCUT_INTENT

d) ACTION_DATE_CHANGED: This intent is used to change the date of the system thereby disabling lot of applications such as browser to seize working.

e) ACTION_DELETE: Delete the given data from its container.

f) ACTION_INSTALL_PACKAGE: This intent is used to launch application installer. It can be used to install malicious applications.

g) ACTION_MAIN: Start as a main entry point, does not expect to receive data.

h) ACTION_MEDIA_REMOVED: External media has been removed.

The subsequent sections will describe the application of machine learning algorithms to identify malicious and suspicious applications in Android and their results.

4. Experiment and Analysis

4.1. Environment

The experiments are carried out on an Intel Core i7-3632HQ CPU@ 2.2 GHz machine with 16 GB RAM. Each of the classifiers is evaluated with two methods: 10-fold cross-validation and 80% split. In 10-fold cross-validation, the data set is divided into ten subsets, and the holdout method is repeated ten times. In each round, one subset is taken as test set and the remaining nine subsets are combined to form the training set. Errors of all the ten rounds are averaged out to obtain a final output. This method ensures that each instance is included at least once in the test set and nine times in the training set. The final model is the average of all ten iterations. The second method we use is 80% split, which uses 80:20 ratio (80% of a dataset for training and 20% for testing). This method is efficient but is less accurate than the 10-fold method.

4.2. Features Extraction

Preparation for the experiment, we used python web crawler to download 1500 apks from app market. 850 apks of them were benign apks, others were malicious apks. All the doanloaded files were scanned using antivirus virustotal to classified as malicious or benign.

Performance of our classifier is compared in terms of True Positive Rate (TPR), False Positive Rate (FPR), accuracy, (F1-Score) and Area Under Curve (AUC). These metrics are calculated using the following formulas (1), (2), (3), (4), (5):

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

$$ACC = \frac{TP + TN}{TP + FN + FP + TN} \tag{3}$$

$$F1\text{-Score} = 2 * \frac{precison * recall}{precison + recall} \tag{4}$$

$$AUC = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{5}$$

The above formulas are consisting of four parameters, we put it in a confusion matrix like Table 2.

Table 2: Confusion matrix

Actual class	Classified as malware	Classified as benign
Malware	TP	FN
Benign	FP	TN

We also make a comparison with other detection methods. The result shown as following Table 3:

Table 3: Comparison of other methods

Method	TPR	FPR	ACC	F1-Score	AUC
Sequential Minimal Optimization (SMO)	0.952	0.035	0.956	0.956	0.979
Naïve Bayes (NB)	0.982	0.012	0.985	0.985	0.995
Random Forest (RF)	0.985	0.008	0.988	0.988	0.989

According to the comparison result, Naïve Bayes and Random Forest are more effective than Sequential Minimal Optimization. Overall, Random Forest would give the better result.

5. Conclusion

We have implemented a novel method for detecting malicious android applications using permission and intent based analysis. After extracting several permissions and intents as features from several downloaded, features set was reduced using information gain algorithm. The database is then divided into training dataset and the testing dataset. We then used k-means clustering followed by the random forest using the training data to correctly detect malware among the given applications. Performances were compared in terms of accuracy and error percentage derived from the confusion matrix. The hybrid approach for malware detection of android applications will enhance the security of the mobile device. Future work may include evolving and comparing improved classification algorithms for detection of malicious applications.

6. Acknowledgements

This work was supported by the Student's Platform for Innovation and Entrepreneurship Training Program of China under Grants No. 201710060085.

7. Reference

[1] Smartphone OS Market Share[OL]. http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[2] Verma, S., & Muttoo, S. K. An android malware detection framework-based on permissions and intents. Defense Science Journal, 2016, 66(6), 618.

[3] Faruki P, Bharmal A, Laxmi V, Ganmoor V, Gaur M, Conti M, et al. Android security: a survey of issues, malware penetration and defenses. IEEE Commun Surv Tutorials 2014;17(2):998–1022.

[4] Shabtai A, Kanonov U, Elovici Y, Glezer C,Weiss Y. Andromaly: a behavioural malware detection framework for Android devices. J Intell Inf Syst, 2012, 38(1):161–90.

[5] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behaviour based malware detection system for Android. In: Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices. ACM, 2011, 15–26.

[6] ZHANG Y T, ZHANG Y, ZHANG T, et al. Intelligent identification of malicious behaviour in Android applications based on naive Bayes[J]. Journal of Southeast University: Natural Science Edition, 2015, 45(2):224-230.

[7] ZHANG R, YANG J Y. Android malware detection based on permission correlation[J]. Journal of Computer Applications, 2014,34(5): 1322-1325.

[8] XU Y P, WU C H, HOU M J, et al. Android malware detection technology based on improved naive Bayesian[J]. Journal of Beijing University of Posts and Telecommunications, 2016, 39(2):43-47.

[9] LI W, GE J, DAI G. Detecting malware for Android platform: an svm-based approach[C]// IEEE, International Conference on Cyber Security and Cloud Computing. New Jersey, USA: IEEE, 2015:464-469.

[10] FEIZOLLAH A, ANUAR N B, SALLEH R, et al. Comparative study of k-means and mini batch k-means clustering algorithms in Android malware detection using network traffic analysis[C]// International Symposium on Biometrics and Security Technologies. New Jersey, USA: IEEE, 2014: 193-197.

[11] YUAN Z, LU Y, XUE Y. Droid detector: Android malware characterization and detection using deep learning [J]. Tsinghua Science & Technology, 2016, 21(1):114-123.

[12] WEN W P, MEI R, NING G, et al. Malware detection technology analysis and applied research of Android platform[J]. Journal on Communications, 2014, 35(8): 78-85.

[13] YANG H, ZHANG Y Q, HU Y P, et al. A malware behaviour detection system of Android applications based on multi-class features[J]. Chinese Journal of Computers, 2014, 37(1): 15-27.

[14] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, and S. Ioannidis, Rage against the virtual machine: hindering dynamic analysis of android malware, in Proceedings of the Seventh European Workshop on System Security. ACM, 2014, p. 5.

[15] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, and C. Siemens, Drebin: Effective and explainable detection of android malware in your pocket, in Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS), 2014.