ANDROID MALWARE DETECTION BASED ON PERMISSIONS

Zhao Xiaoyan*,Fang Juan,Wang Xiujuan

*College of Computer Science, Beijing University of Technology, Beijing, 100124
zhaoxy325@emails.bjut.edu.cn; {fangjuan,xjwang}@bjut.edu.cn

Keywords: Android; Malware Detection; SVM; PCA

Abstract

In this paper, we propose a permission-based malware detection framework for Android platform. The proposed framework uses PCA(Principal Component Analysis) algorithm for features selection after permissions extracted, and applies SVM(support vector machine) methods to classify the collected data as benign or malicious in the process of detection. The simulation experimental results suggest that this proposed detection framework is effective in detecting unknown malware, and compared with traditional antivirus software, it can detect unknown malware effectively and immediately without updating the newest malware sample library in time. It also illustrates that using permissions features alone with machine learning methods can achieve good detection result.

1 Introduction

With the development of modern network communication technology, intelligent terminal including mobile phone, pad, multimedia center, home appliance and so on infiltrate into people's daily life, implementing intelligent, diversified, and personalized service and experience. Intelligent terminal is carrying a large number of privacy information and personal account information, etc, therefore it becomes a target for malware[1,2];Because of its' openness, Android platform is under attack on all sides. In recent years, Android malware emerge in endlessly, and have different methods of attack, research work for the Android malware detection have also developed.

In this paper, firstly, in order to deal with smartphones' restriction in resources, capacity ,etc,we propose a lightweight static malware detection framework and using machine learning methods in the process of detection; Through the study of the Android system permissions mechanism, we determine to extract permission as the detection feature; Secondly, to further reduce classifier's the amount of computation we use PCA (Principal Component Analysis) algorithm to reduce the feature dimension; Finally, as it is difficult to collect samples of Android platform malware , while SVM (Support vector machine) is able to maximize learning machine's generalization ability by following the Vapnik's structural risk minimization principle[11] , that is to say, in the case of a limited number of training samples, still be able to ensure small errors for an independent test set,we choose SVM in detection process to ensure a still high correct classification rate in case of lacking prior knowledge.

2 Related Work

2.1 Android Malware Detection

Through it is generally difficult to distribute malicious applications through the official Android Market because of strict application checking, but hackers can upload any malware to the third-party markets, given that the Android Market may be banned by some countries such as China [3].At the same time, as the smartphones viruses and computer viruses have many similarities in behavior[4], the current Android malware detection system are mostly transplant from corresponding field of computer technology, without considering the characteristics of smartphones and Android system. Botha et al.[5] analyzed the common desktop anti-virus program, and assess their applicability to mobile platforms. However, because the resources of smartphones are limited, some of these solutions (such as antivirus software) are inadequate for use on smartphones as they consume too much CPU and memory and might result in rapid draining of the power source.

Android platform malware detection is broadly divided into two types：dynamic monitoring and static analysis. Dynamic monitoring often requires system modification so that an application can be monitored as it runs in the Dalvik Virtual Machine (DVM) or native environment. Some methods require stricter permission management for applications. The representatives include Crowdroid[6] and Andromaly[7], both of which monitor phones dynamically and collect related information. Crowdroid client monitors Linux kernel's system call information of mobile application, and sends the information to the remote server through the network . Remote server will collect the behavior data generated by different users on the same application. It constructs feature vectors with the behavior data, cluster the data with k-means algorithm to determine whether the application is malicious. This method requires users' participation, and needs to collect the user's behaviour data when they use the application data, so user privacy data leakage problem will appear during the process of implementation. Andromaly detects malware by monitoring the smartphone and the user's activities, recording the sensor activity, CPU usage rate and so on. However, the process of monitoring mobile phones as well as the information transmission may cost additional resources and traffic, and it cannot give detection result to mobile phone in time, with high time and resource consumption. Taintdroid[8] uses dynamic taint tracking to identify any privacy leakage in Android applications (a similar privacy leakage detection system PiOS[9] is designed for Apple iOS). Enck et al.[10] monitors the phone's sensitive data access with dynamic Taint analysis technologies and analyze the situation, but they did not put forward specific malicious code detection scheme.

2.2 Android Permission Mechanism

Android system has a strict permissions management mechanism to restrict the behavior of applications[11,12].Some system functions is not allowed to be called by default when an android application is running, such as phone calls, SMS, bluetooth, WIFI, etc. These functions are generally corresponds to the specific hardware devices. In order to use these functions, corresponding permissions must be granted to the program using the <uses-permission> tag in AndroidManifest.xml and prompts to users when installation. Users can choose refusing to install if there are sensitive permissions .Also, the actual permissions the program use can't beyond the category of the statement when installation, otherwise will trigger security exception. In other words, all the required permissions need to be granted to the application before it can run on the system.

Permission is not only the Android system's unique identity designed for safety, but also the premise to realize certain operations for Android applications. But the actual usage not fully meets the design objectives. Users don't attach as much importance to the system warning of sensitive permissions as expected .On the one hand, users have made the decision to use the App when system warning occurs, at this time ,users won't give up this decision and will ignore the warning. On the other hand, general users are not familiar with the real effect of the permissions, also don't know the dangers of sensitive permissions may bring. All the above factors result in serious consequences that malicious software still exploit in spite of system notifications. Therefore, it is unreasonable and unrealistic to judge whether or not it based on users' choice. For one thing, most users do not have related system safety knowledge; For another thing, users are more concerned with the actual function of App than the potential threat.

Tajin Zhou[13] established a dataset contained of more than 1,200 malware samples that cover the majority of existing Android malware families, ranging from their debut in August 2010 to later ones in October 2011.Results of the analysis and comparison of 1260 malware samples and benign software samples' difference in using permissions found that ,compared with benign software, malicious apps are more inclined to request following permissions: SMS-related permissions, such as READ_SMS,WRITE_SMS, RECEIVE_SMS,SEND_SMS,and RECEIVE_BOOT_COMPLETED permission and CHANGE_WIFI_STATE permission ,etc. At the same time, malicious apps clearly tend to request more permissions than benign ones. The important difference between malicious software and general ones is also the reason why we chose permissions as features.

3 Malware Detection Framework for Android

3.1 System architecture

Based on the above study, we designed a malware detection framework for Android platform based on the SVM algorithm. Our system architecture is shown in Figure 1.Our malware detection framework mainly contains following modules:

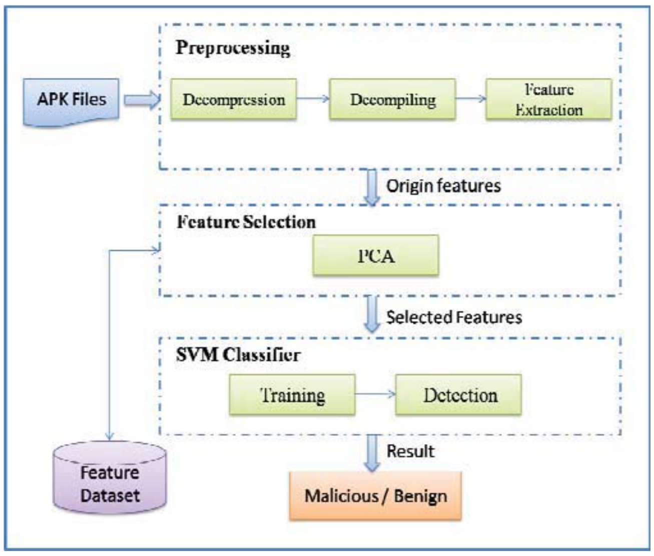


Figure 1: Malware detection Framework

Our malware detection framework mainly contains following modules:

Preprocessing module- It's role is to decompress each Android application package file, get AndroidManifest.xml file from the extracted content, and then decompile it. Finally get each APK's permission list from its' decompiled AndroidManifest.xml. All these permission vectors form the original feature set .

Feature selection- In this module, the original feature set get from preprocessing module will be processed by PCA algorithm to extract principal component.

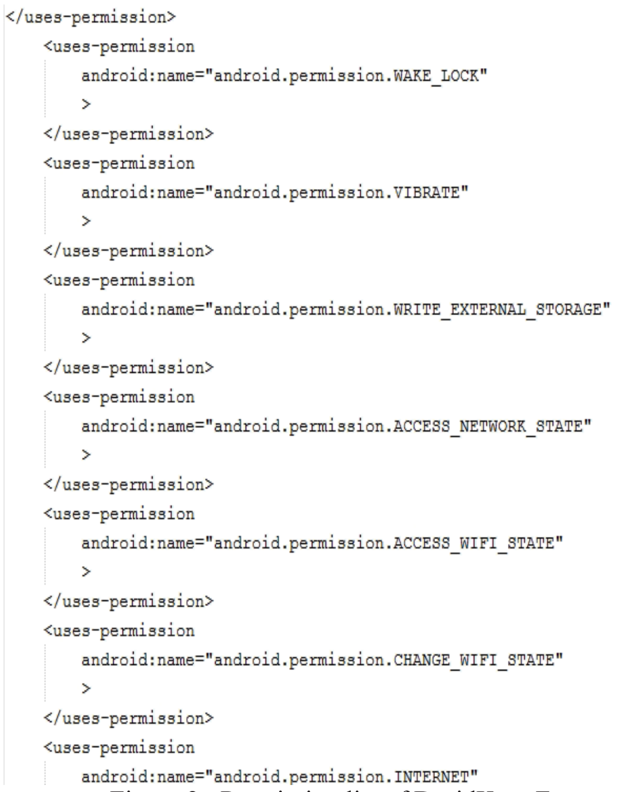
The SVM classifier - During training phase, a training set consist of feature vectors of malware samples and benign software samples is provided to the classifier. In detection phase, the trained classifier can classify the input feature vectors of an unknown APK.

Feature Dataset- This module is responsible for storing and updating features extracted from samples.

3.2 APK Preproccessing

In this module, the main task of the system is to decompile the software samples and get initial features set from it. As

APK(Android application package) is essentially a zip format file, so APK should be unzipped at first and then it can be decompiled . AndroidManifest.xml, Android application's global configuration file, will be seen in the extracted content after decompression .AndroidManifest.xml contains the Android application package' s name, the Linux user ID, permissions, the minimum API version that this application needs. In this article, we use the decompiler, AXMLPrinter2.Jar, to convert AndroidManifest.xml to a readable XML file. After decompiling XML file, permissions of the APK can be seen. The figure below is permissions list of a variant of DroidKungFu.



```
</uses-permission>
    <uses-permission
        android:name="android.permission.WAKE_LOCK"
        >
    </uses-permission>
    <uses-permission
        android:name="android.permission.VIBRATE"
        >
    </uses-permission>
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        >
    </uses-permission>
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE"
        >
    </uses-permission>
    <uses-permission
        android:name="android.permission.ACCESS_WIFI_STATE"
        >
    </uses-permission>
    <uses-permission
        android:name="android.permission.CHANGE_WIFI_STATE"
        >
    </uses-permission>
    <uses-permission
        android:name="android.permission.INTERNET"
```

Figure 2: Permission list of DroidKungFu

For each Android application, we will then compare the permission list retrieved from corresponding application package with total set of Android system's permissions, and stored the values of features as a binary number (0 or 1),which is represented as a sequence of comma separated values. The few sample features are described as the following:

0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0
,0,0,0,0,1,0,0,0,1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,1,1,0,0,0,
0,0,0,1,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0
,0,0,0,1,0,1,0,0,0,0,1,0,0,1,0,,0,0,1,0,1,0,0,0,0,0,0,1.

Figure 3: An example feature vector for malware application retrieved from its Android application package file

3.3 Feature Selection

When the classifier detects a software, if the selected feature dimension is high, calculation amount of the classifier will accordingly increase .Since computing power, storage space, power and other aspects resource of smartphones is limited, this kind of defect will be more prominent. Therefore, in order to improve the efficiency of malware detection without affecting the detection accuracy, the dimensionality of the feature should be as low as possible. In this paper, we use the PCA(Principal Component Analysis) method for feature selection.

Principal Component Analysis algorithm was proposed by Karl Pearson in 1901.It is a multivariate statistical method which, under the premise of little loss of information, transforms the original high-dimensional features into lower-dimensional features through the linear transformation. The generated comprehensive features usually called principal component. Since the process of dimensionality reduction preserves main information of the origin features and keeps new features independent of each other as far as possible, which makes the new generated features, that is the main component, has more advantages than the original feature. According to the basic principles of principal component analysis, feature extraction algorithm design steps are as follows:

Step1: Extract permissions feature vector from each APK: $A_i = (p_{i1}, p_{i2}, \cdots, p_{im})$ (i=1,2,…n)

Step2: Get original features matrix: $A = (A_1, A_2, \cdots, A_n)^T$

Step3: Calculate the covariance matrix of matrix A，record it as S;

Step4: Calculate eigenvalues of S:λ_1 , λ_2 , … , λ_m （$\lambda_1 \geqq \lambda_2 \geqq \ldots \geqq \lambda_m > 0$） ,and the corresponding orthogonal eigenvectors :u_1 ,u_2 , … , u_m;

Step5: Calculate the variance contribution rate $\beta_i = \lambda_i \Big/ (\sum_{i=1}^{m} \lambda_i)$,set the cumulative contribution rate threshold value γ.Select the first k eigenvalues that meet the formula:

$$\gamma \le \beta = \beta_1 + \beta_2 + \cdots + \beta_k = \sum_{i=1}^{k} \lambda_i \Big/ \sum_{i=1}^{m} \lambda_i \qquad (1)$$

Step 6: Calculate the first k principal components and a matrix X:

$$X_1 = u_1 A \, , X_2 = u_2 A \, , \ldots\ldots , \quad X_k = u_k A$$

$$X = (X_1, X_2, \cdots, X_k)^T = (u_1 A, u_2 A, \cdots, u_k A)^T \quad (2)$$

The matrix X is the feature set that can be used to train support vector machine.

3.4 SVM Binary Classification

Support Vector Machine is a binary classification model, and its basic model is defined as the maximum interval linear classifier in feature space, that is to say, the learning strategy of SVM is to maximize this interval, eventually this problem can be transformed to solve convex quadratic programming problems.

Malware detection problem is actually a two-type classification problem. In the training phase, for the n training samples APKs , $(x_1, y_1),(x_2, y_2),\cdots,(x_n, y_n) \in X \times \{1,-1\}$,where $x_i \in X$, X is a feature vector set which is obtained previous feature selection module, Its corresponding sample label is $y_i \in \{1,-1\}$, here -1 represents benign software, 1 represents malware. Train the Classifier, and Establish hyperplane:

$$w^x + b = 0 \qquad (3)$$

Here x is the input vector, w is a hyperplane normal vector whose weights can be adjusted, b is a bias. With the hyperplane, the sample space is divided into two parts which represent two different procedures. According to the Lagrange multiplier method, The problem is transformed into finding a Lagrange Operator α_i of maximization objective function :

$$\max W(a) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j M(x_i, x_j) \qquad (4)$$

$$\text{s.t.} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ ;}$$

$$\alpha_i \in [0, H] \text{ (i=1, 2, …N) ;} \qquad (5)$$

Here H is a selected positive parameter. Use the kernel function:

$$M(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) \qquad (6)$$

We can finally get the decision function:

$$f(x) = \mathrm{sgn}\left[\sum_{i=1}^{n} \alpha_i y_i M(x_i, x) + b\right] \qquad (7)$$

After the training process ,an unknown android application can be classified according to equation (7) based on its' feature vectors.

4 Experiment and Results

The experimental samples consist of 220 malware samples and 234 samples of benign software. Malware samples contain typical Android platform malware like DroidKungFu, AnserverBot, etc; Benign samples come from official Android Market and Android applications get 'Wan Dou Jia Design Award', including games , tools, and other categories. All of these applications have been officially identified, therefore they can be considered benign. This article completes coding to extract permission from APK and get original feature set. Then form the final feature set by implementing PCA algorithm to select principal component.

Let TP (true positive) be the number Android malware that are correctly detected; FN (false negative) be the number of Android malware that are not detected (predicted as benign application); let TN (true negative) be the number of benign applications that are correctly classified; and let FP (false positive) be the number of benign applications that are incorrectly detected as Android malware. We use the following parameters to evaluate the detector:

FPR (False Positive Rate): Percentage of wrongly identified malware applications

$$\text{FPR} = \text{FP} / (\text{TN}+\text{FP}) \qquad (8)$$

ACC (Overall Accuracy): Percentage of correctly identified applications

$$\text{ACC} = (\text{TP}+\text{TN}) / (\text{TP}+\text{TN}+\text{FP}+\text{FN}) \qquad (9)$$

Experiment I: The propose of this experiment is to evaluate the effect of PCA algorithm for dimensionality reduction. Table 1 shows the dimensions of selected features when we choose different cumulative contribution rate threshold γ . Figure 4 shows different feature selection parameters for classification resulting different performance of the SVM classifier.

γ	Feature dimensions
0.8	33
0.85	41
0.9	54
0.95	76

Table 1. Cumulative Contribution Rate threshold γ and Feature dimensions

From the result of Experiment I (Figure 4) we can see that, when the cumulative variance contribution threshold is 0.85, i.e., when the feature dimension is 41, SVM classifier achieves a higher detection accuracy (90.08%), and a lower false alarm rate (10.4%) . Therefore, in the system's feature selection module we will set the cumulative contribution rate threshold γ is 0.85.

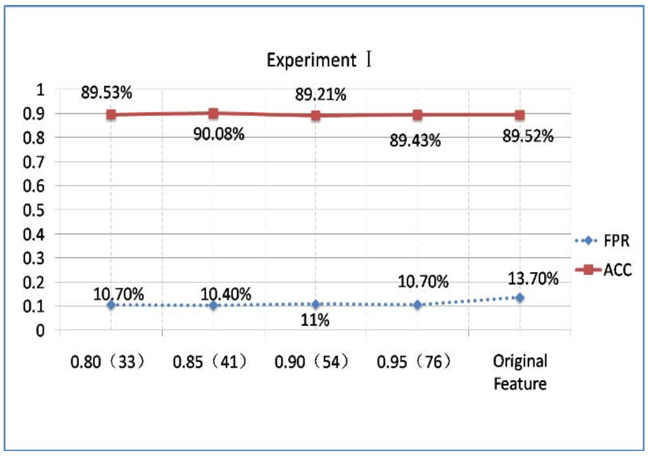


Figure 4. SVM Classifier's performance under various Feature dimensions

Experiment II: Based on previous experiment ,we choose the appropriate cumulative contribution rate threshold γ (0.85),in this experiment, all samples first feature after feature selection process will be compressed to 41 dimensions, then tests the system's performance on detecting an new application. Meanwhile, in order to further illustrate the effect of SVM algorithm, we compare it with some other classifiers:BayesNet,NaiveBayes,NBTree,CART,Random Tree,J48. Results in Figure 4 clearly illustrates that, compared to other classification algorithm, SVM can achieve better results. In summary, this system can achieve a high detection rates for unknown malware through sample library is limited. In practice, it is equivalent that the system can detect unknown good software without update sample library in real time.

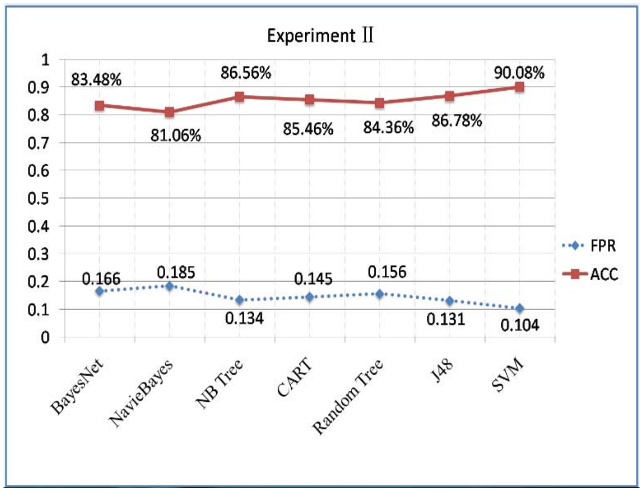


Figure 5. Average Accuracy and FPR of each Detectors

5 Conclusions

Android permissions, as an important security identity system, is also meaningful for distinguishing Android malware and benign software. Support Vector Machine algorithm is a machine learning method that has a good learning performance especially for a small number of samples ,and it has been used in many aspects like web recognition, face recognition and so on. This paper discusses extracting permission features and using SVM algorithm in Android malware detection, and proposes an malware detection framework for Android platform based on SVM . Simulation results show that using permission features with machine learning methods can achieve good detection results. In practice, it is equivalent that the system can detect unknown good software without update sample library in real time.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grant No. 61202076) and by the General Program of Science and Technology Development project of Beijing Municipal Education Commission (Grant No. KM201210005022), along with other government sponsors. The authors would like to thank the reviewers for their efforts and for providing helpful suggestions that have led to several important improvements in our work. We would also like to thank all the teachers and students in our laboratory for helpful discussion.

References

[1] Leavitt, N.: Mobile phones: the next frontier for hackers? Computer 38(4), 20–23 (2005)

[2] Shih, D.H., Lin, B., Chiang, H.S., Shih, M.H.: Security aspects of mobile phone virus: a critical survey. Industrial Management & Data Systems 108(4), 478–494 (2008)

[3] S. Ye. Android Market is Currently Blocked in China. Here are your Alternatives, Sep 2011. http://techrice.com/2011/10/09/android-market-is-currentlyblocked-in-china-here-are-your-alternatives/

[4] Muthukumaran, D., et al.: Measuring integrity on mobile phone systems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (2008)

[5] Botha, R.A., Furnell, S.M., Clarke, N.L.: Fromdesktop to mobile: Examining the security experience. Computer & Security 28, 130–137 (2009)

[6] BURGUERA I, ZURUTUZA U, NADJM-TEHRANI S. Crowdroid: behavior-based malware detection system for Android[C]//Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York, USA: ACM, 2011: 15-26.

[7] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss: Andromaly: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38(1) (January 2011) 161-190

[8] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In Proc. of USENIX OSDI, 2010.

[9]M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In Proc. of NDSS, 2011.

[10] ENCK W, GILBERT P, CHUN B G, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones[C]//Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Berkeley, CA, USA: USENIX, 2010: 1-6.

[11] Song Jie, Party Li Cheng, Guo Chao, Zhao Meng, Security Mechanisms Analysis and Appliction Research of Android mobile platform [J]. Computer Technology and Development.2010,20(6):152－155.

[12] Liao Ming-Hua, Zheng Liming. Android Security Analysis and Solutions Study, [J]. Science Technology and Engineering.2011,11(26):6350-6355.

[13] YAJIN ZHOU, XUXIAN JIANG. Dissecting Android malware: characterization and evolution. North Carolina State University. Security and Privacy (SP)[J].2012IEEE Symposium on Date of Conference,2012: 95-109