Hybrid Detection Using Permission Analysis for Android Malware

Haofeng Jiao[1,2], Xiaohong Li[1,2(✉)], Lei Zhang[1,2], Guangquan Xu[1,2], and Zhiyong Feng[1,2]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
{hfjiao, xiaohongli, lzhang, losin, zyfeng}@tju.edu.cn
[2] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin University, Weijin Road no. 92, Nankai District, Tianjin, China

Abstract. The growth of malicious applications poses a great threat to the Android platform. In order to detect Android malware, this paper proposes a hybrid detection method based on permission. Firstly, applications are detected according to their permissions so that benign and malicious applications can be discriminated. Secondly, suspicious applications are run in order to collect the function calls related to sensitive permissions. Then suspicious applications are represented in a vector space model and their feature vectors are calculated by TF-IDF algorithm. Finally, the detection of suspicious applications is completed via security detection techniques adopting Euclidean distance and cosine similarity. At the end of this paper, an experiment including 982 samples is used as an empirical validation. The result shows that our method has a true positive rate at 91.2 % and a false positive rate at 2.1 %.

Keywords: Android · Hybrid detection · Euclidean distance · Cosine similarity

1 Introduction

Android has become one of the most popular mobile platforms since it was released. With several hundred thousands of applications, it provides kinds of functionality to its users. Unfortunately, smartphones running Android are increasingly targeted by attackers and infected with malicious software. According to a study of F-Secure lab, the species of Android malware increased by 144 % from 2012 to 2013. This statistic shows that there is a need to do research for Android malware detection.

In this paper, a permission based hybrid detection method is proposed to detect Android malware. The method performs permission detection at first. Then suspicious applications are executed to obtain function calls related to sensitive permissions. Finally, applications are represented algebraically and security detection is employed to determine the security type of suspicious applications.

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 541–545, 2015.
DOI: 10.1007/978-3-319-23829-6_40

2 Methodology

To systematically detect malicious applications in Android markets, this paper proposes a hybrid detection method for Android malware which is shown in Fig. 1.

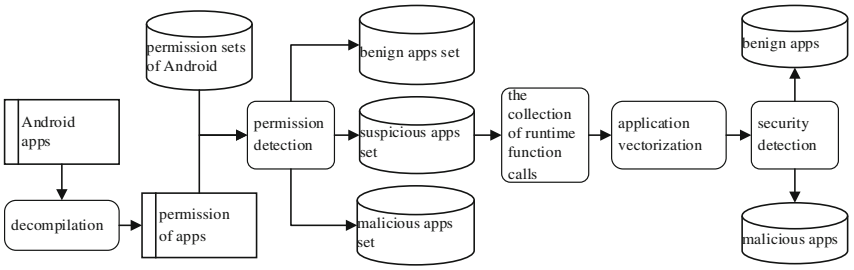

Fig. 1. The framework of hybrid detection

Permission Detection. If applications want to accomplish some tasks in Android, they have to explicitly request permissions. Thus, rules of permission can be employed for detection to discriminate benign and malicious applications.

All sensitive permissions are divided into 12 sets according to their classes. They can be represented by $perSet_i (1 \leq i \leq 12)$. $perSet_{13}$ and $perSet_i$ represent the set of permissions with a protection level of Normal and SignatureOrSystem, respectively.

Rules of permission detection can be given as following: If $AppPer \cap perSet_{13} = AppPer$, it can be judged as benign application. If $AppPer \cap perSet_{14} \neq \varnothing$, it can be judged as malicious application. If $AppPer \cap perSet_i \neq \varnothing$, $(1 \leq i \leq 12)$, it applies the sensitive permissions of the i^{th} class and can be judged as suspicious application.

The Collection of Runtime Function Calls. Suspicious applications need behavior tracking to complete their detection. In order to collect as many function calls as possible, monkeyrunner is used to run an automated start-to-finish test of applications. Then, function calls are filtered to reserve the function calls related to sensitive permissions. To relate functions with permissions, the method introduced by Felt et al. [1] is used.

Application Vectorization. Let $\varepsilon := \{f_1, f_2, f_3, \ldots, f_n\}$ be the set of function calls of application and $f_i (1 \leq i \leq n)$ stands for the i^{th} function of application, therefore every application can be represented by a set ε. Let C is the set of ε and stands for the set of all suspicious applications.

Define $w_{i,j}$ as the times of function f_i appearing in application ε_j. If f_i is not present in ε_j, $w_{i,j} = 0$. Therefore, an application ε_j can be represented as the vector $\varepsilon_j = \{w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{n,j}\}$.

To represent a function collection, VSM (vector space model) is employed. The $(i, j)^{th}$ element illustrates the value of f_i in application ε_j and is nonnegative.

To obtain the feature vectors of applications, TF-IDF algorithm is adopted to calculate the weight of every function in applications. Thus, as for the function f_i in application ε_j, $weight(i,j)$ is defined as:

$$weight(i,j) = tf_{i,j} \bullet idf_i.$$

where $tf_{i,j}$ represents the frequency of f_i in application ε_j and idf_i represents inverse term frequency of f_i in set C.

Security Detection Techniques. After application vectorization, applications can be represented as points in the feature space. When an application is being inspected, it is represented in the feature space and then compared with the points of benign and malicious applications. To this end, we use Euclidean distance and Cosine similarity as the distance measures. In order to obtain a final distance and give a result of security detection, three rules of calculating distance are employed. They are the mean distance, the max distance and the min distance.

3 Evaluation Measures

Let $n_{ben\to ben}$ be the number of benign samples classified as benign and $n_{ben\to mal}$ be the number of misclassified benign samples. $n_{mal\to ben}$ and $n_{mal\to mal}$ are defined similarly. Thus, the FPR and TPR (false and true positive rate) are given by:

$$FPR = \frac{n_{ben\to mal}}{n_{ben\to ben} + n_{ben\to mal}} \tag{1}$$

$$TPR = \frac{n_{mal\to mal}}{n_{mal\to ben} + n_{mal\to mal}} \tag{2}$$

4 Empirical Validation

An experiment is done to improve the effectiveness of the detection method. This section gives the data set and the results.

Data Set. The experiment employs a data set including 982 samples collected from Google Play, third-part markets and AMGP (Android Malware Genome Project). Every sample is inspected by F-Secure, Kaspersky, McAfee and Symantec. Table 1 shows the statistic result of data set.

Table 1. The statistic of experiment samples.

Source	Number of samples	Number of malware	Percentage of malware
Google Play	552	0	0 %
Third-part markets	232	51	22.0 %
AMGP	198	198	100 %

The Validation of Permission Detection. After permission detection, the samples of data set are separated into three groups. Table 2 shows the number and percentage of samples in different groups.

Table 2. The result of permission detection.

Application type	Number of samples	Percentage
Benign software	156	15.9 %
Suspicious software	784	79.8 %
Malicious software	42	4.3 %

The Validation of Behavior Detection. Three-fold cross validation is used for detecting of suspicious samples. Meanwhile, the experiment selects 10, 15 and 20 behaviors as the standard of classification. The result is shown in Tables 3 and 4.

Table 3. Results for the different combination measures using Euclidean distance.

Comb.	10 behavior features		15 behavior features		20 behavior features	
	TPR	FPR	TPR	FPR	TPR	FPR
Max length	0.703	0.226	0.754	0.214	0.756	0.198
Mean length	0.809	0.168	0.854	0.132	0.857	0.124
Min length	0.712	0.243	0.762	0.200	0.760	0.188

Table 4. Results for the different combination measures using cosine similarity.

Comb.	10 behavior features		15 behavior features		20 behavior features	
	TPR	FPR	TPR	FPR	TPR	FPR
Max length	0.778	0.194	0.828	0.168	0.832	0.124
Mean length	0.873	0.061	0.912	0.029	0.912	0.021
Min length	0.812	0.114	0.873	0.097	0.876	0.081

5 Related Work

A large body of research has been done to analyze and detect Android malware. Malware analysis focuses on the study of vulnerability in Android application. For example, Woodpecker [2] search for capability leaks of Android application. Comdroid [3] analyze the vulnerability in inner-app communication in Android applications. While malware detection puts emphasis on detecting the security type of applications. For example, TaintDroid [4], Crowdroid [5] and DroidRanger [6] are methods that can monitor the behavior of applications at runtime. Although very effective in identifying

malicious activity, they suffer from a significant overhead. However, methods such as Stowaway [7] usually induce only a small runtime overload. While these approaches are efficient, they mainly build on manually crafted detection patterns which are often not available for new malware instances.

6 Conclusion

Comparing the result of different combination measures, the result obtained by using cosine similarity and average length is the best of all. In particular, the best result has an accuracy of 95.8 %, with an FPR of 2.1 % and TPR of 91.2 %.

Future work is oriented in two main directions. The algorithm we used actually only weights the importance of one function based on its frequency of appearance. This may leads to FPR in some degree, other algorithms should be used to achieve a better result. Second, other distance measurements and combination rules could be tested.

Acknowledgements. This work has partially been sponsored by the National Science Foundation of China (No. 91118003, 61272106, 61340039), 985 funds of Tianjin University and Tianjin Key Laboratory of Cognitive Computing and Application.

References

1. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of ACM Conference on Computer and Communications Security (CCS), pp. 627–638 (2011)
2. Grace, M., Zhou, Y,. Wang, Z., Jiang, X.: Systematic detection of capability leaks in stock android smartphones. In: Proceedings of the 19th Annual Symposium on Network and Distributed System Security. NDSS 2012 (2012)
3. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in android. In: Proceedings of the 9th Annual Symposium on Network and Distributed System Security. MobiSys 2011 (2011)
4. Enck, W., Gilbert, P., Chun, B.-G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation. USENIXOSDI 2010 (2010)
5. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st Workshop on Security Privacy in Smartphones and Mobile Devices. CCSSPSM 2011 (2011)
6. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, You, Get Off of My Market: detecting malicious apps in official and alternative android markets. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2012)
7. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. CCS 2011 (2011)