Detecting Mobile Malware with TMSVM

Xi Xiao, Xianni Xiao, Yong Jiang, and Qing Li(✉)

Graduate School at Shenzhen, Tsinghua University, Shenzhen, China
{xiaox,jiangy,li.qing}@sz.tsinghua.edu.cn,
sunny13940512@gmail.com

Abstract. With the rapid development of Android devices, mobile malware in Android becomes more prevalent. Therefore, it is rather important to develop an effective model for malware detection. Permissions, system calls, and control flow graphs have been proved to be important features in detection. In this paper, we utilize both static and dynamic strategies with a text classification method, TMSVM, to identify the mobile malware in these three aspects. At first, features have to be selected. Since the sum of control flow graphs is very large, Chi-Square method is used to get the key graphs. Then features are transformed into vectors and TMSVM is subsequently applied to get the classification result. In the static method, we firstly analyze permissions and control flow graphs respectively and then think of the combination of them. In the dynamic method, the system calls are considered. At last, based on the results of the static method and dynamic method, a hybrid classification model of three layers classification is proposed. Compared with the other methods, our method increases the TPR and decreases the FPR.

Keywords: Mobile malware · TMSVM · Dynamic analysis · Static analysis · Permission · Control flow graph · System call

1 Introduction

With the mobile device becoming more and more popular, the malware enjoyed a prevalence in the market, especially in the Android market. With the special policy of Google Android Market, every developer has to declare permissions to get access to the important resources [1]. Hence, permissions have become an important factor to identify whether an application is malicious [2]. However, many developers may have bad habits that they would tend to announce more permissions than they need. Thus only using the permissions is not enough to identify the application's behaviors. So we think of two more factors: Control Flow Graphs(CFGs) and system calls of the applications. Control flow graphs have been proved to be a very prominent feature in detecting the malware [3]. Nevertheless, the control flow graphs may neglect the semantic meaning of every program. The extra factor, the system call, could make up for this drawback.

In this paper, we distinguish the malware from the benign applications in three terms: permissions, control flow graphs, and system calls. We use both

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 507–516, 2015.
DOI: 10.1007/978-3-319-23829-6_35

the static analysis and dynamic analysis method to obtain a hybrid multi-layer classification model. In our work, the basic classification model is the Support Vector Machine and the elementary classification method is the text classification method. The text classification method we use is TMSVM, which can be got from https://code.google.com/p/tmsvm/. In the static method, we first do the classification with only permissions or control flow graphs, and then do the classifications with the combination of them. In the dynamic method, the system calls of one application during its execution become the detection feature. After analyzing the results of the dynamic method and the static method, a hybrid multi-layer classification model is proposed. In this hybrid multi-layer model, applications would be classified by three classifiers chosen from the dynamic analysis experiment and the static analysis experiments. The detection result of this model is much better than the result of SCSDroid [4] and Peirvavian [5].

The major contributions of this paper can be summarised as follows:

1. We use TMSVM to do the malware classification. Different with the ordinary SVM model, TMSVM could automatically choose the best kernel trick to build the SVM model for different datasets.
2. We identify the malware in all the three features: permissions, system calls, and control flow graphs. Researchers have analyzed the features respectively, but the combination of these features has not been used for malware detection.
3. We examine the role of control flow graphs in detecting malware. However, control flow graphs were only used in identifying the malware variants before.
4. We propose a hybrid multi-layer classification model. Compared with the result of the one layer classification, the hybrid multi-layer classification increases the TPR and decreases the FPR.

2 Related Work

For malware detection, it's important to get detailed knowledge of application's characteristics. Static analysis and dynamic analysis of software are the two common practices recently [6]. In static analysis, various binary forensic techniques are used, and applications don't need to be executed. However, in dynamic analysis, it involves running an application in a controlled environment and monitoring its behavior. Both of the two analysis strategies have advantages and disadvantages [7], and many approaches using both of the two methods exist.

The principal skill of the static analysis is identifying the malicious code by unpacking the samples and looking into the result codes [8,9]. Static analysis is also used for detecting vulnerabilities or information leakages of the applications [10–13]. For example, Lu et al. [14] split Android apps by the component entry points and used static analysis method to detect the Android apps for component hijacking vulnerabilities. Felt et al. [15] created the Stowaway system to make a map between the API set and the permissions to detect the over-privileged attack. Many researchers work on distracting the distinguishing features and utilizing the similarity distance to identify the malware [16].

The static analysis is convenient and fast, while it could not achieve the real time detection of malware [17]. The dynamic detection could address this issue. The CrowDroid system [18] collected the kernel system call to detect the malware in the form of Trojan horses. Authors in [19] logged the activity of all applications and used the signature matching approach to detect the personal information leakage. Enck et al. [20] proposed TaintDroid, an efficient, system-wide dynamic taint tracking and analyzing system capable of simultaneously tracking multiple sources of sensitive data.

The dynamic analysis also has a few flaws. The execution paths covered by the dynamic method are limited, making it difficult to fully cover all the running condition of malware, which would influence the detection precision. Thus many researchers try to find a way to combine both the dynamic method and the static method [21]. In [22], the author first decompiled the Android applications to find the suspicious model, then run the apps in AAsandbox. Finally, they analyzed the logs to find malware.

3 Background

Android is an open-source operating system built on Linux kernel. The security scheme of Android has its unique characters. Android protects the user data and the system resources by providing an isolation from other applications. Softwares run in the application sandbox. For additional capabilities not provided by the basic sandbox, applications need to declare the permissions they require. Users have to grant or deny all the requested permissions at a block before the application is installed. Control flow graphs in computer science is a visual representation of all paths the computer program can take during its execution. In this paper, we use the string form of control flow graphs defined in [3]. The process of Android is divided into user space and kernel space. The system call is the fundamental interface between an application and the operating system's kernel. Most operations interacting with the system require the use of system calls.

4 The Proposed Hybrid Model

We use the static and dynamic methods to detect whether an application is malware. TMSVM is employed in this paper to do the classifications. It is a text mining model which can choose a best SVM model in LIBSVM for different datasets. In TMSVM, we choose term frequency to calculate the weight of each feature vector. The malware dataset in this paper is from [23], including 1260 malware samples and the benign dataset, containing 1280 applications, is downloaded from the Google Play. The training set of the classifiers contains half of the malware dataset and half of the benign dataset. Accordingly, the test set contains the other half of them.

4.1 Static Analysis

In static analysis, the different roles of permissions and control flow graphs in identifying malware have been tested. Firstly, permissions and control flow graphs are set as the feature separately, and then combined by different weights.

Permission Analysis. Aapt is supplied by Google in Android SDK and can be used to dissemble the apk file and get its permission set. Before using TMSVM, the permission set would be turned into a numerical vector according to the index of every permission. The index of one permission is its index in the standard permission set which contains all the official 134 permissions in Android 4.3. Then with the training dataset, we could get a best SVM classifier using TMSVM. Test dataset would be classified with this classifier. The last row in Table 1 gives the result of the permission classification, since in this experiment, the weight of the permission is 1 in the static hybrid model.

Control Flow Graph Analysis. Androguard is used to get control flow graphs of every application. As the sum of control flow graphs of one application is large and the string format of a CFG can be very long, each CFG is firstly mapped into an integer value using Blizzard hash method. Since the Chi-square method has been proved to be an efficient feature selection method in text mining field [24], we use it to abstract the important control flow graphs. The first row in Table 1 shows the classification result with CFG.

The Static Hybrid Analysis. For further research, permissions and control flow graphs are joined together to form the application's detection feature. TMSVM use term frequency to calculate each application's frequency vector. Putting permissions and control flow graphs together directly is unfair for permissions, since each permission could appear at most once, while, control flow graphs can be repeated, and the sum of the permissions each application declares is much less than the sum of its control flow graphs. Hence, we need to repeat permissions enough times to make it achieve the weight we set below in the joined vector. According to the term frequency method, the repeated times R should satisfy the equation below.

$$\frac{R * P}{X} = \frac{C}{1 - X} \ . \tag{1}$$

where, X is the weight of permissions we set in the joined vector, P is the permission's sum in one application, and C is the sum of its control flow graphs. In this experiment, the weights of the permission in the combined vector range from 1/8 to 7/8 to do the comparison. Therefore, the weights of control flow graph range from 7/8 to 1/8. The TPR and FPR of different weights are shown in Table 1.

Table 1. Static Hybrid Classification Result

Weights of Permissions	TPR	FPR
0	0.973	0.05
0.125	0.973	0.030
0.25	0.975	0.038
0.375	0.962	0.038
0.5	0.959	0.022
0.625	0.957	0.013
0.75	0.965	0.038
0.875	0.964	0.028
1	0.958	0.021

4.2 Dynamic Analysis

In the dynamic experiment, every application runs in a real Android device, meanwhile, the system calls of every application are tracked. Monkey is used to randomly generate 1000 events for every application, including almost all the different kinds of events. Due to objective reasons, some application's system call has not yet been obtained. In the malware dataset, there are 1179 applications been tracked, and in the benign dataset, there are 1188 applications been tracked.

Similarly with permissions, the system call set would also be transformed into a digital vector according to the index of every system call in the standard system call vector. The standard system call vector contains all the 185 system calls in Android System 4.0.4. We calculate each application's frequency vector in both benign and malware dataset of the training set to obtain the average frequency vector for each dataset. Compared the two average frequency vector in Fig. 1, we find out that the average system call frequency vectors are dramatically different. Table 2 shows the FPR and TPR of the system call classification.

4.3 Hybrid Multi-Layer Model

The architecture of the hybrid multi-layer classification model is shown in Fig. 2, which is composed of three steps classification. All the classifiers are obtained in the experiments above in the training phase.As shown before, the system call classification could identify the malware with the relatively low FPR. In order to get low FPR, once the application is identified as malware in the system call classification, the hybrid multi-layer classification model. If the application is recognized as a benign application in the system call classification, it will be sent to the static hybrid classification model with the permission's weight being 1/8 which is represented as static hybrid classification in Fig. 2. In the second step, if an application is labeled as a benign application, the result is considered as its classification result in the hybrid multi-layer classification model. At the same time, the application classified as malware will be sent to the permission

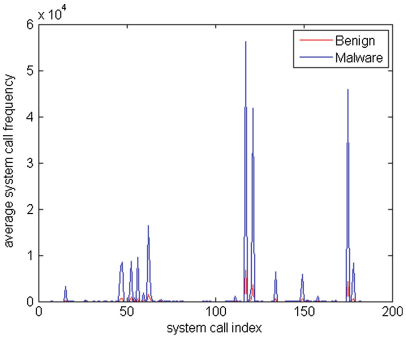

Fig. 1. System call frequency

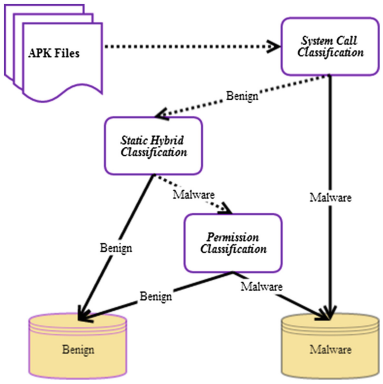

Fig. 2. The hybrid multi-layer classification model

classification. At last, its classification result in the third step will be its final result in the hybrid multi-layer classification model.

5 Experiment Result and Discussion

In Fig. 3 we report the results related to the one-step classifiers and the two-step classifiers. The one-step classifiers represent the static hybrid classifications that have been introduced above in Sect. 4.1. It is shown that among all the one step classification models, the static hybrid model with the permission's weight, 5/8, gets the best result. This classification is called Static/0.625. In the two-step classifications, the first step classification is the system call classifier and the second step is the static classifications with different permission weights. Among all the two-step classification models, the ideal result could be got if the second classification is the permission classification. Two-Step/1 is used to represent the best two steps classification.

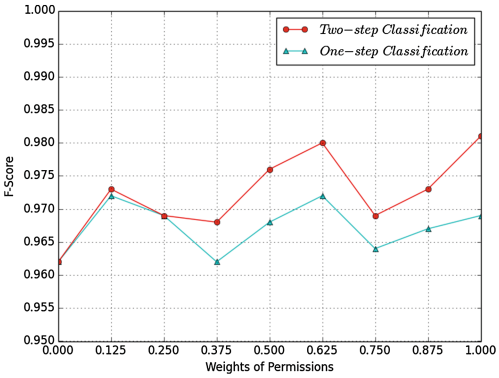

Fig. 3. Static Result of Different Classifications

Table 2. Result of Different Classification Model

Classification Model	TPR	FPR	F-score
Static/0	0.973	0.05	0.962
Static/0.625	0.957	0.013	0.972
Static/1	0.958	0.021	0.969
Dynamic	0.901	0.018	0.939
Two-step/1	0.997	0.037	0.982
Hybrid	0.989	0.017	0.986
SCSDroid [4]	0.96	0.02	0.941
Peirvavian [5]	0.941	0.025	0.945

Table 2 compares the result of different classification models. These models are the control flow graph classification(Static/0), the best static hybrid classification(Static/0.625), the permission classification(Static/1), the system call classification(Dynamic), the best two steps classification(Two-step/1), the hybrid multi-layer classification(Hybrid), SCSDroid [4],and Peirvavian [5]. As shown in Table 2, the hybrid multi-layer classification model increases the TPR as well as decreases the FPR, leading to the best F-score result of all the above methods. Comparing the results in Table 2 and Fig. 3, it can be seen that, all the one step classifications have a much better result than SCSDroid and Peirvavian. From Table 2, it shows that, the combination of permissions and control flow graphs is more accurate than the combination of permissions and API calls [5]. Taking the permissions, system calls and control flow graphs into account, it can be concluded that the permission is a more effective feature to identify the malware than the two other features.

6 Conclusion and Future Work

In this paper, permissions and control flow graphs are used as static analysis of the detection. We also do the dynamic analysis by means of system calls. According to the static analysis result, it can be drawn a conclusion that the combination of permissions and CFGs could improve the malware detection rate. Based on the static method and the dynamic method, we propose a hybrid multi-layer classification model which could increase the TPR and decrease the FPR. Compared with SCSDroid and Peirvavian, our results are much better. But the structure of the hybrid multi-layer model still needs some theoretical evidences. Meanwhile, some improvements are also required for the classification by system calls. A malicious application could be identified if it has suspicious actions in the form of system call sequences, which could be a research focus in the mobile malware detection in the future.

Acknowledgements. This work is supported by the NSFC project(61202358), the National Basic Research Program of China (2012CB315803), the National High-tech R&D Program of China(2014ZX03002004) and the Shenzhen Key Laboratory of Software Defined Networking. We would like to thank the authors in [23] to provide the malware dataset for us. We also would like to thank Zhenlong Wang, Yi He, and Peng Fu for the helpful discussion.

References

1. Sarma, B.P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM symposium on Access Control Models and Technologies, pp. 13–22, ACM Press, June 2012
2. Moonsamy, V., Rong, J., Liu, S., Li, G., Batten, L.: Contrasting permission patterns between clean and malicious android applications. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds.) SecureComm 2013. LNICST, vol. 127, pp. 69–85. Springer, Heidelberg (2013)
3. Cesare, S., Xiang, Y.: Classification of malware using structured control flow. In: Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107, pp. 61–70. Australian Computer Society, Inc., January 2010
4. Lin, Y.D., Lai, Y.C., Chen, C.H., Tsai, H.C.: Identifying android malicious repackaged applications by thread-grained system call sequences. Comput. Secur. **39**, 340–350 (2013)
5. Peiravian, N., Zhu, X.: Machine learning for android malware detection using permission and API calls. In: IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), 2013, pp. 300–305. IEEE Press, November 2013
6. Xiao, X., Tian, X., Zhai, Q., Xia, S.: A variable-length model for masquerade detection. J. Syst. Softw. **85**(11), 2470–2478 (2012)
7. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Computer Security Applications Conference, ACSAC 2007, Twenty-Third Annual, pp. 421–430. IEEE Press, December 2007

8. Zhou, W., Zhou, Y., Jiang, X., Ning, P.: Detecting repackaged smartphone applications in third-party android marketplaces. In: Proceedings of the second ACM conference on Data and Application Security and Privacy, pp. 317–326. ACM Press, February 2012
9. Potharaju, R., Newell, A., Nita-Rotaru, C., Zhang, X.: Plagiarizing smartphone applications: attack strategies and defense techniques. In: Barthe, G., Livshits, B., Scandariato, R. (eds.) ESSoS 2012. LNCS, vol. 7159, pp. 106–120. Springer, Heidelberg (2012)
10. Chan, P. P., Hui, L. C., Yiu, S. M.: Droidchecker: analyzing android applications for capability leak. In: Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks, pp. 125–136. ACM Press, April 2012
11. Chan, P.P., Hui, L.C., Yiu, S.: A privilege escalation vulnerability checking system for android applications. In: 13th IEEE International Conference on Communication Techonologies (ICCT), pp. 681–686. IEEE Press (2011)
12. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in Android. In: Proceedings of the 9th international conference on Mobile systems, applications, and services, pp. 239–252. ACM Press, June 2011
13. Yan, L. K., Yin, H.: Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In: Proceedings of the 21st USENIX Security Symposium, August 2012
14. Lu, L., Li, Z., Wu, Z., Lee, W., Jiang, G.: Chex: statically vetting android apps for component hijacking vulnerabilities. In: Proceedings of the 2012 ACM conference on Computer and communications security, pp. 229–240. ACM Press, October 2012
15. 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, pp. 627–638. ACM Press, October 2011
16. Suarez-Tangil, G., Tapiador, J.E., Peris-Lopez, P., Blasco, J.: Dendroid: A text mining approach to analyzing and classifying code structures in Android malware families. Expert Syst. Appl. **41**(4), 1104–1117 (2014)
17. Xiao, X., Xia, S., Tian, X., Zhai, Q.: Anomaly detection of user behavior based on DTMC with states of variable-length sequences. J. China Univ. Posts Telecommun. **18**(6), 106–115 (2011)
18. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pp. 15–26. ACM Press, October 2011
19. Isohara, T., Takemori, K., Kubota, A.: Kernel-based behavior analysis for android malware detection. In: 2011 Seventh International Conference Computational Intelligence and Security (CIS), pp. 1011–1015. IEEE Press, December 2011
20. Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. OSDI **10**, 1–6 (2010)
21. Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D.: These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In: Proceedings of the 18th ACM conference on Computer and communications security, pp. 639–652. ACM Press, October 2011
22. Blasing, T., Batyuk, L., Schmidt, A. D., Camtepe, S.A., Albayrak, S.: An android application sandbox system for suspicious software detection. In: 5th International Conference on Malicious and Unwanted Software (MALWARE), 2010, pp. 55–62. IEEE Press, October 2010

23. Zhou, Y., Jiang, X.: Dissecting android malware: Characterization and evolution. In: IEEE Symposium on Security and Privacy (SP), 2012, pp. 95–109. IEEE Press, May 2012
24. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: ICML, vol. 97, pp. 412–420 (1997)