A Malicious Application Detection Framework using Automatic Feature Extraction Tool on Android Market

Dong-uk Kim, Jeongtae Kim, and Sehun Kim

***Abstract*—**Android applications are becoming increasingly because android phones are widespread and steadily gaining popularity. Unfortunately, such phenomenon draws attention to malicious application developers so that malicious applications are increasing rapidly. To detect these malicious applications, several researches have been done using machine learning. In machine learning methods, there are two ways which are dynamic detection and static detection to detect these malicious applications. The dynamic detection has a high detection rate but uses a lot of resources and takes long time to detect malicious behaviors. In addition, it is possible to infect malicious code after implementing detection method. On the other hand, the static detection has a small amount of resources and quick detection time but has low detection rate.

To these weaknesses, we propose a malicious application detection framework on android market and an automatic extraction tool of malicious features for static detection. This framework is able to perform both detection methods using machine learning and is expected to perform a thorough analysis than previous framework. Also, the proposed automatic extraction tool is expected to help a code analysis for static detection.

***Keywords*—**Android, Malicious Application, Malware, Machine Learning

I. Introduction

Since Google developed an android platform, android phones have evolved over the years. The android phones are widespread and steadily gaining popularity so that android applications are becoming increasingly. The one of reason for this development is due to accessibility of android market. Anyone can develop an android application and upload the developed application on the android market. Unfortunately, such phenomenon draws attention to the malicious application developers so that malicious applications are increasing rapidly. If a user of android phone downloads these malicious applications, it can cause serious damages, such as making the phone unusable, stealing private information, and sending SMS or phone call for unwanted billing, to users. Therefore, it is important to detect the malicious android applications efficiently.

To detect the malicious applications, several researches have been done using machine learning. In machine learning methods, there are two ways which are dynamic detection and static detection to detect the malicious applications. In case of static detection methods [1]-[3], these detection methods based on permissions have a small amount of resources and quick detection time. However, since many android applications have various function, there acquire various permission so that these methods have low detection rate. In case of dynamic detection methods [4]-[6], these detection methods have a high detection rate but uses a lot of resources for implementing applications directly and takes a long time against intentionally changing execution time of malicious behaviors. In addition, it is possible to infect malicious code after implementing the detection methods.

To resolve these problems, several researches proposed detection methods using both static and dynamic method [7], [8]. However, these detection methods cannot solve all drawbacks of other researches. The study of Asaf Shabtai *et al.* [7] cannot fully solve a problem of resource limitation on android phone. Unlike this, the study of Yajin Zhou *et al.* [8] attempts to detect malicious applications using DroidRanger on android market and solves a problem of resource limitation on android phone. But they cannot solve a problem of static detection based on permissions.

In this paper, we propose a malicious application detection framework on android market to solve above problems. This framework is able to perform both detection methods using machine learning on android market so that the problem of resource limitation can be solved. In addition, this framework performs static detection based on methods of APIs [9], which is used to manage android services and is related to sending SMS, calling and information spill, so that the problem of static detection based on permission can be solved. The features based on methods of APIs can be extracted by automatic extraction tool, which is made by us, for static detection.

On June, 2012, the official android market had the 650,000

Dong-uk Kim is with the Department of Industrial and Systems Engineering & Graduate School of Information Security, Korea Advanced Institute of Science and Technology, South Korea (phone: +82-42-350-2954; fax: +82-42-350-3110; e-mail: donguk@kaist.ac.kr).

Jeongtae Kim is with the Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, South Korea (e-mail: jeongtae@gmail.com).

Sehun Kim is with the Department of Industrial and Systems Engineering & Graduate School of Information Security, Korea Advanced Institute of Science and Technology, South Korea (e-mail: shkim@kaist.ac.kr).

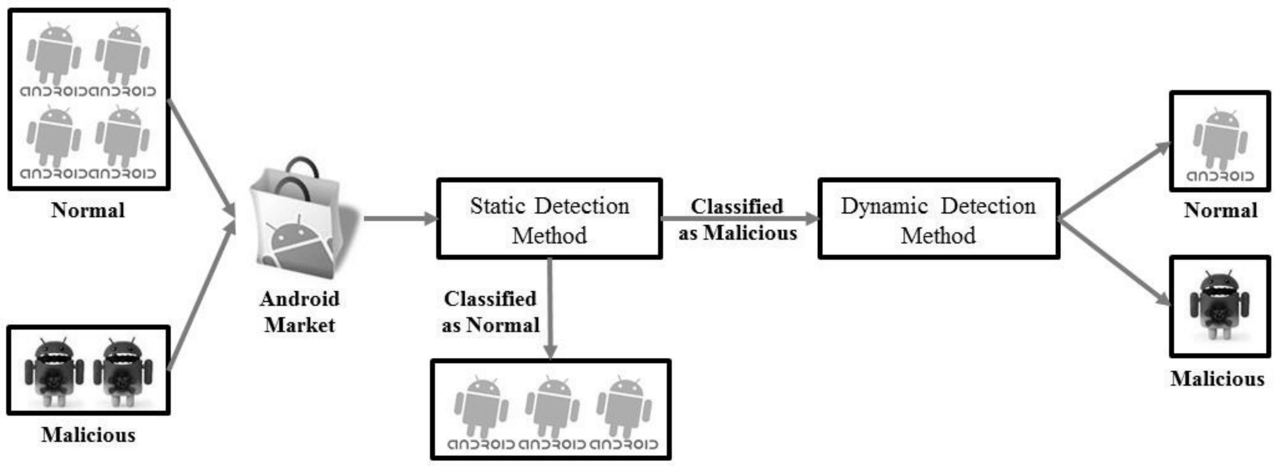

Fig. 1 The proposed malicious application detection framework

applications listed and 20 billion applications are downloaded to date [10]. In future, it is expected to develop many more android applications. If all android applications are listed on android market, these applications cannot be tasted by using both static and dynamic detection methods on android market because test process takes too long time. To resolve this problem, we apply to a filtering method on the proposed framework. First, all applications are quickly classified by static detection method. Then, only suspicious applications are tested by dynamic detection method. By using the filtering method, most normal applications are classified by static detection method and then a few rest applications are tested by dynamic detection method so that dynamic detection time can be reduced. Therefore, this framework can remain efficient even though dynamic detection may involve time-consuming methods for thorough analysis. Hence, this framework is expected to have fast and thorough analysis than previous framework. Also, the proposed automatic extraction tool is expected to help code analysis for static detection.

The remainder of this paper is organized as follows. In Section 2, the proposed framework is presented. After that, a brief simulation results are given in Section 3. Finally, we conclude the paper and suggest future work in Section 4.

II. Proposed Framework

A. Malicious Application Detection Framework

In this paper, we propose a malicious application detection framework on android market. This framework uses a filtering method [11] which filters normal application before a dynamic detection because the dynamic detection needs too much time and many resources for testing all applications thoroughly. Most of applications on android market are normal applications. If more time and resources are used to test normal applications than testing malicious applications, it is inefficient. Therefore, if the number of applications, which are tested by the dynamic detection method, is reduced by using the filtering method, it is efficient for detecting malicious applications and a detection performance will be increased.

Fig. 1 presents the proposed framework. In this framework, all registered applications on android market are firstly tested by the static detection method. As shown in Fig. 1, most of normal applications are initially classified according to extracted features. The extracted features are based on permissions and methods of APIs. Using features of methods of APIs, we can analyst functions of application in detail than using only features of permissions. This helps to increase a detection rate and decrease a false negative rate on the static detection. The detailed features are shown in Table I. The applications, which are classified to be clearly normal, will stop the test and are diagnosed as normal and remaining applications are suspicious of malicious applications. Then, remaining applications are thoroughly tested by the dynamic detection method. Hence, consuming time and resources can be reduced for detecting the malicious applications and the proposed framework is efficient than other detection frameworks.

TABLE I
Selected Features

Features based on	Feature used
Permissions	Number of each permissions
Methods of APIs	1. Number of each methods of APIs which are included in phone management 2. Number of each methods of APIs which are related to phone control and information of privacy

B. Automatic Feature Extraction Tool

To extract features before machine learning, we use an automatic feature extraction tool written by java script. Fig. 2 is overall process of automatic feature extraction. As an input, this tool takes an '.apk' file as android application package. The tool decompresses the '.apk' file because the '.apk' file is a compressed bundle of files. When the '.apk' file decompressed, this file is split into three main contents such as

'AndroidManifest.xml' file, 'Classes.dex' file and 'res' folder. The 'AndroidManifest.xml' file is an XML file including the set of permissions used in application. The 'Classes.dex' file is application source code which holds the complete bytecode to be interpreted by Dalvik Virtual Machine [12]. The 'res' folder is consist of files defining the layout, language, etc. Since the features in Table I are used to test, the automatic feature extraction tool extracts the 'AndroidManifest.xml' file and the 'Classes.dex' file related to permissions and methods of APIs respectively. Then, the tool decompiles the 'Classes.dex' file using 'Dedexer' tool [13] and 'AndroidManifest.xml' using 'AXMLPrinter' tool [14]. This process is that original files are converted into human readable format. Finally, using our program written by java script, the automatic feature extraction tool reads all files and extracts features. Then, we can get the data which is used in machine learning.

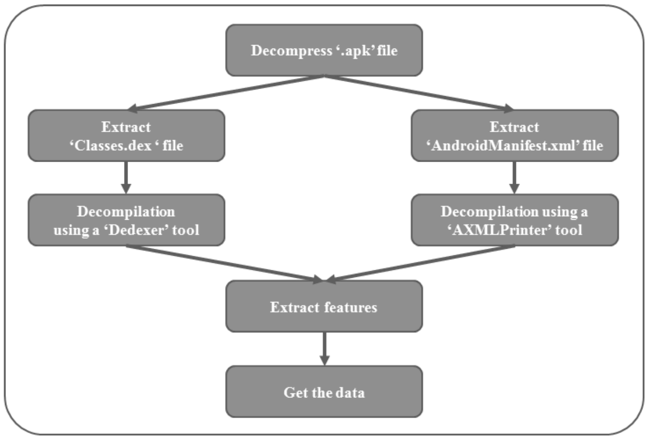
Decompress '.apk' file
Extract 'Classes.dex ' file
Extract 'AndroidManifest.xml' file
Decompilation using a 'Dedexer' tool
Decompilation using a 'AXMLPrinter' tool
Extract features
Get the data

Fig. 2 Automatic feature extraction tool

III. Simulation

A. Simulation Environment

To evaluate the proposed framework, we collected 893 normal applications from android market and 110 malicious applications from the internet site [15]. Then, features were extracted by an automatic feature extraction tool. The machine learning was done by WEKA which is a machine learning tool [16]. We used J48 decision tree classifier [17] and 10-folds cross validation scheme was used to training and testing data set. The simulation configuration is shown in Table II.

TABLE II
Simulation Configuration

Number of applications	1003
Number of normal applications	893
Number of malicious	110
Machine learning tool	WEKA 3-6-8
Classifier	J48 decision tree
Test option	10-folds cross validation

B. Results

This framework focuses on filtering out clearly normal applications so that the number of applications is reduced for thorough analysis of dynamic detection. To confirm efficiency of the framework, we compare detection rates between a framework which is based on permissions and our proposed framework based on methods of APIs. As shown Table III, the true positive rate of proposed framework is 82.7% and the false negative rate is 17.3%. The true positive rate of proposed framework is higher than that of permission based framework and the false negative rate of proposed framework is lower than that that of permission based framework. The low false negative rate means that normal applications are filtered out clearly. This is more important than lowering false positive rate due to the characteristics of the filtering method. As a result, we validate that a static detection method of our proposed framework is high performance for classifying malicious applications.

TABLE III
Detection Rates and False Positive Rates

	Proposed Framework		Permission based Framework	
	Predicted Normal	Predicted Malicious	Predicted Normal	Predicted Malicious
Real Normal	0.718	0.282	0.964	0.036
Real Malicious	0.173	0.827	0.609	0.391

Table IV presents the number of applications on each detection stage. On the static detection, 641 normal applications and 19 malicious applications are filtered out. Then, only 343 applications are tested on the dynamic detection. As shown in the result, only about 34.2% of all applications needed to be tested so that reduced applications can be tested thoroughly by a dynamic detection method. Through this result, we validate that our proposed framework can reduce efficiently consuming time and resources for detecting the malicious applications.

TABLE IV
Number of Applications on Each Detection Stage

	Static Detection	Dynamic Detection
Normal	893	252
Malicious	110	91
Total	1003	343

IV. Conclusion

In this paper, we proposed a malicious application detection framework using an automatic feature extraction tool on android market. To resolve a problem of static detection method based on permissions, the framework performs a static detection based on methods of APIs and performs both detection methods using machine learning on android market to resolve a problem of resource limitation. Also, this framework used a filtering method for operating efficient detection because the dynamic detection needs too much time and many resources for testing all applications thoroughly so that a static detection filters normal applications, which are most of applications on android market, before a dynamic detection. Hence, a few rest applications are tested by dynamic detection method so that detection time can be reduced. It causes to be efficient for detecting malicious applications and increase detection performance. In simulation, we confirmed that our proposed framework is efficient than previous framework using only features based on permissions through comparing detection rates. Also, we confirmed that many normal applications are filtered on stage of static detection.

When the false negative errors where malicious applications are misclassified as normal are occurred on process of filtering method, these errors may be critical problem. Also, if too many features are used in machine learning, detection rate may be decreased. Therefore, we will consider applying 'MetaCost' method and featuring selection methods on our proposed framework in the future.

ACKNOWLEDGMENT

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the CYBER SECURITY RESEARCH CENTER supervised by the NIPA (National IT Industry Promotion Agency), NIPA-H0701-12-1001.

REFERENCES

[1] F. Di Cerbo, A. Girardello, F. Michahelles, and S. Voronkova, "Detection of malicious applications on android OS," Lecture Notes in Computer Science, vol. 6540, pp. 138-149, Nov. 2011.
[2] D. Barrera, H. G. Kayacik, P. C. Van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to Android," Proceedings of the ACM Conference on Computer and Communications Security, pp. 73-84, Oct. 2010.
[3] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android Permissions Demystified," Proceedings of the ACM Conference on Computer and Communications Security, pp. 627-636, Oct. 2011.
[4] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale," Lecture Notes in Computer Science, vol. 7344, pp. 291-307, June 2012.
[5] L. Xie, X. Zhang, J.-P. Seifert, and S. Zhu, "PBMDS: A behavior-based malware detection system for cellphone devices," Proceedings of the 3rd ACM Conference on Wireless Network Security (WiSec'10), pp. 37-48, Mar. 2010.
[6] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "SCanDroid: Automated Security Certification of Android Applications," Technical report, University of Maryland, 2009.
[7] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, ""Andromaly": A behavioral malware detection framework for android devices," Journal of Intelligent Information Systems, vol. 38, no. 1, pp. 161-190, Feb. 2012.
[8] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. "Hey, You, Get off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets," In Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS '12), Feb. 2012.
[9] http://developer.android.com/reference/packages.html
[10] http://en.wikipedia.org/wiki/Google_Play
[11] J. Choi, G. Kim, T. Kim, and S. Kim, "An Efficient Filtering Method for Detecting Malicious Web Pages," The 13th International Workshop on Information Security Applications (WISA 2012), Aug. 2012.
[12] http://en.wikipedia.org/wiki/Dalvik_VM
[13] http://dedexer.sourceforge.net/
[14] http://code.google.com/p/android4me/
[15] http://contagiominidump.blogspot.kr/
[16] http://www.cs.waikato.ac.nz/ml/weka/
[17] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan kaufmann, 1993.

Dong-uk Kim received the B.S degree in industrial systems and information engineering from Korea University, Seoul, Korea, in 2007. Currently, he is working toward the integrated Master's Ph.D. degree in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. His research interests are security issues such as malicious application detection and intrusion detection system.

Jeongtae Kim received the B.S. and M.S. degrees in Industrial Engineering, from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea in 1999 and 2001, respectively. He is currently working toward the Ph.D. degrees in KAIST. Since 2003, he has been with Electronics and Telecommunication Research Institute (ETRI), Daejeon, Korea, where he is a researcher in the Technology Strategy Research Division. His research interests include the areas of mobile communication, network security and mobile computing.

Sehun Kim received the B.S. degree in physics from Seoul National University, Seoul, Korea, in 1972, and the M.S. and Ph.D. degrees in operations research from Stanford University in 1978 and 1981, respectively. In 1982, he joined the faculty of KAIST. He has published a number of papers in IEEE Trans. on Vehicular Technology, Computer Networks, Telecommunication Systems, IEICE Transactions on Communications, International Journal of Satellite Communications, and Journal of KIISC (Korea Institute of Information Security and Cryptology). He served as the chief editor of the Journal of KIISC from 1990 to 1993.