International Journal of Security and Its Applications
Vol. 10, No. 12 (2016), pp.251-260
http://dx.doi.org/10.14257/ijsia.2016.10.12.20

Static Anomaly Detection Framework for Android-based Mobile Phones

Xiaobo Ji[1], Fan Zeng[1,*] and Bangxian Ye[2]

[1]*Department of Information, Research Institute of Field Surgery, Daping Hospital, Third Military Medical University, Chongqing, China*
[2]*Faculty of Software, Fujian Normal University, Fuzhou 350007, China*
zengfan215@163.com

Abstract

With the rapid development of wireless communication, mobile network, and embedded system technologies, android-based mobile devices show a number of useful functions and then they are attacked by hackers for obtaining some useful information. In this paper, an efficient static anomaly detection framework is shown for android-based mobile phones to improve their security. The proposed framework uses support vector machine to perform the anomaly detection and exploits the cloud computing platform to reduce the impact on android-based mobile phones. Experimental results show that the proposed framework is better than existing anomaly detection frameworks in terms of the detection precision and the detection time.

Keywords*: Android-based mobile devices, Support vector machine, Cloud computing, Static anomaly detection*

1. Introduction

In recent years, the mobile internet develops very quickly and then mobile phones have more and more functions. People's daily life depends on their mobile phones significantly. For example, people use mobile phone to buy goods and play games. IDC reports that the android-based mobile phones have 75% of the global smart phone market. Compared with other smart phones such as close source IOS, the android-based mobile phones have the following features, which make them be the attacked target easily [1].

(1) There are a large number of third part application markets, which do not detect the applications developed for android-based mobile phones. Then the android-based mobile phones are attacked by malicious applications [2].

(2) Android-based mobile phones users are lack of security consciousness. For instance, something about user authorization protection and traffic management [3].

(3) Android-based mobile phones have low security themselves [4].

Until now, android-based mobile phones have been attacked for many times. Data from a famous security company reports that about 90% of mobile malicious applications target on android-based mobile phones [5]. Therefore, it is very necessary to design an efficient anomaly detection framework for android-based mobile phones. In this paper, we propose a static anomaly detection framework for android-based mobile phones in order to detect malicious applications. The proposed framework employs support vector machine to find out the malicious applications and exploits to the cloud computing platform to reduce the detection time. Experimental results show that the proposed anomaly detection framework is better than existing anomaly detection frameworks in terms of the detection precision and detection time.

The remainder of this paper is organized as follows. Section 2 briefly reviews android operating system and then existing anomaly detection methods for android-based mobile

ISSN: 1738-9976 IJSIA

phones. Section 3 shows a static anomaly detection framework for android-based mobile phones. Section 4 presents the experimental results. Finally, Section 5 concludes the work.

2. Background and Related Work

In this section, the android operation system is first analyzed and then existing anomaly detection methods are briefly reviewed.

2.1. Android Operating System

Android operating system uses an efficient layer architecture, which is composed of operating system, middleware, and applications, respectively [6].

(1) Operating system layer

The lowest layer of android operating system is a Linux kernel that is customized by using the Linux kernel 2.6. The Linux kernel is responsible for managing system resources, such as process, main memory, and drivers. In addition, the Linux kernel also hides the detailed implementation of hardware layer and then provides common interfaces to the upper layer [7].

(2) Middleware

Android operating system introduces the middleware to encapsulate and abstract all the functions of the bottom layer Linux kernel. The middleware is responsible for coordinating the communication between applications and Linux kernel. The layer is composed of android runtime environment, function libraries, and some application frameworks, respectively [8].

The android runtime environment consists of two components, which are function libraries containing Java language core functions and Dalvik virtual machines. The android applications are developed by using Java programming language. However, the Dalvik virtual machines execute DEX format files of android applications, not Java applications directly. The Dalvik virtual machines is a kind of virtual machine, which is based on registers. In the android operating system, executing an android application will fork a process, which will execute a Dalvik virtual machine instance. Different applications run in the different processes. In this case, each application is independent with others.

Android operating system has a set of function libraries, which is written by using C/C++. The functions of this function libraries can be provided to the developers by using the application framework. The core components of the function libraries are composed of GLIBC, Web browser engine, SGL, OpenGL ES 1.0/2.0, and SQLite.

Application frameworks are used by android operating system to help developers build android applications. The developers can reuse application components easily by using application frameworks.

(3) Android applications

Android applications are on the top layer of android operating system architecture. It is the part, which bridges the gap between the android operating system and users. There are a large number of android applications, such as email client, map, browser, and contact. Users can also download android applications from application market such as games and social applications [9].

Figure 1 shows the architecture of android operating system.

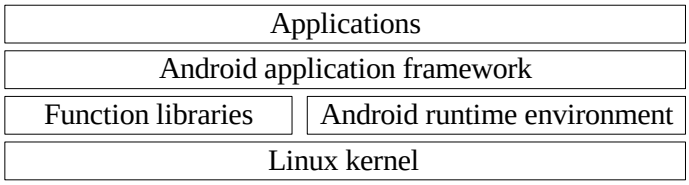


Figure 1. Architecture of android operating system

2.2. Anomaly Detection Techniques for Mobile Phones

In this section, a number of anomaly detection techniques for mobile phones will be briefly reviewed. As shown in Figure 2, there are two main kinds of anomaly detection techniques for mobile phones, which are static anomaly detection and dynamic anomaly detection, respectively. The static anomaly detection technique is also referred to as the active anomaly detection technique. It is used to detect an abnormal application before it is executed. The dynamic anomaly detection technique is also called as passive anomaly detection technique. It is used to detect the abnormal behavior of an application when the application is executed [10].

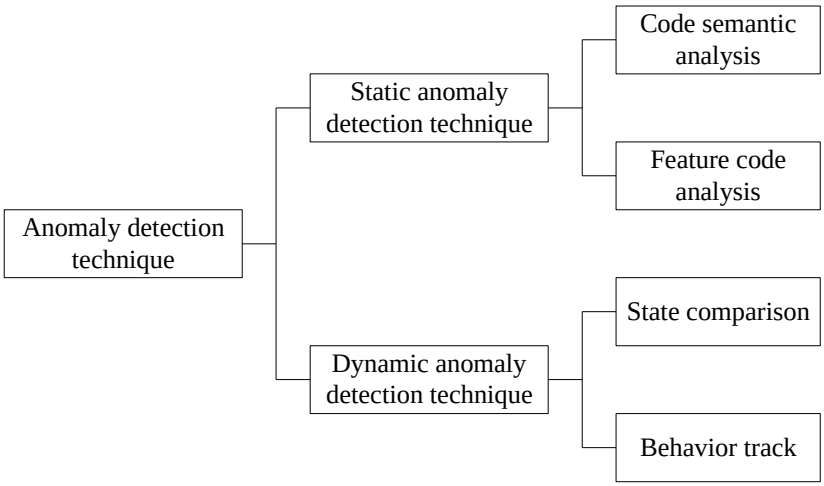


Figure 2. Anomaly Detection Techniques

(1) Static anomaly detection technique [11]

Static anomaly detection technique uses specific software analysis tools to analyze the files and execution code in software, then analyzes the run flow and system resource calls. Static anomaly detection technique often includes the following two key techniques.

a) Code semantic analysis

Code semantic analysis is a method that restores the main function and flow graphs of software code. It analyzes the source code of software by using specific software analysis tools and then understands the semantics and the logical flows of the source code. Most of software often do not open source code, so code semantic analysis method should be used with decompiling techniques [12].

The decompiling technique is an inverse process of software compiling. There are a lot of decompiling tools used to decompile APK installation packages, such as dex2jar. The APK installation packages are extracted as source code, resource files, and configuration files.

b) Feature code analysis

Feature code analysis employs the reverse engineering techniques to extract the feature information from applications such as the operating code sequence, binary sequence, and function call sequence [13].

The static anomaly detection technique do not need virtual machines and sandboxes, so it has the advantage of low power consumption. Moreover, the static anomaly detection technique has low risk and low requirement for real time.

(2) Dynamic anomaly detection technique [14]

Dynamic anomaly detection technique is a method that installs and executes software on smart phones, and then detects the system resource usage of this software by using the functions of software. These resources includes the connecting internet network condition, obtaining the contact information, sending short message without alerting, and requesting uploading and downloading files. Executing software is used to detect whether software has abnormal behaviors such as illegal content spread.

Dynamic anomaly detection technique can be categorized into the state comparison and behavior track, respectively.

The state comparison is mainly used to analyze the state conditions of operating system before executing the application and after executing the application. Then, it extracts the behavior of application by comparison such as the comparison of the file structure before executing and after executing application. However, the state comparison can be affected by other running applications and then the detection precision is very low [15].

The behavior track can be also categorized into lightweight level and instruction level. The lightweight level method tracks the dynamic behavior of applications by calling hook technique or filtering device drivers. The instruction level analysis technique modifies the state or value of register, CPU, and main memory and then changes the application flow when it is executed. However, this technique must deep into the lowest layer of operating system, so it consumes a lot of time when analyzing [16].

It is simple to implement the dynamic anomaly detection and its detection efficiency is very high. It is efficient for detecting abnormal behavior of applications, but it also has its shortages.

The advantages and disadvantages of these two anomaly detection techniques are listed in Table 1.

Table 1. Advantages and Disadvantages of Two Anomaly Detection Techniques

Detection technique	Advantages	Disadvantages
Static anomaly detection	Low detection energy consumption, Low risk, Low requirement for real time	Hard to obtain data sample
Dynamic anomaly detection	Simple to implement, High efficiency	Very high requirement for real time

Because of the low power of the smart phones, a static anomaly detection technique is proposed for smart phones in this paper.

3. Static Anomaly Detection Framework

In this section, an efficient static anomaly detection framework for smart phones will be presented in detailed.

3.1. Overview of Framework

As illustrated in Figure 3, the proposed static anomaly detection framework consists of two components, which are the data collection module and the anomaly detection module, respectively. The data collection module is deployed on android-based smart phones and the anomaly detection module is deployed on a cloud computing platform. If the anomaly detection module was also deployed on the android-based smart phones, it will consume a large amount of energy and influence the performance of smart phones. So, the anomaly detection module is deployed on the cloud computing platform. It can not only reduce the energy consumption of smart phones, but also exploit the strong computing capability and storage capacity of cloud computing platforms to improve the detection performance.

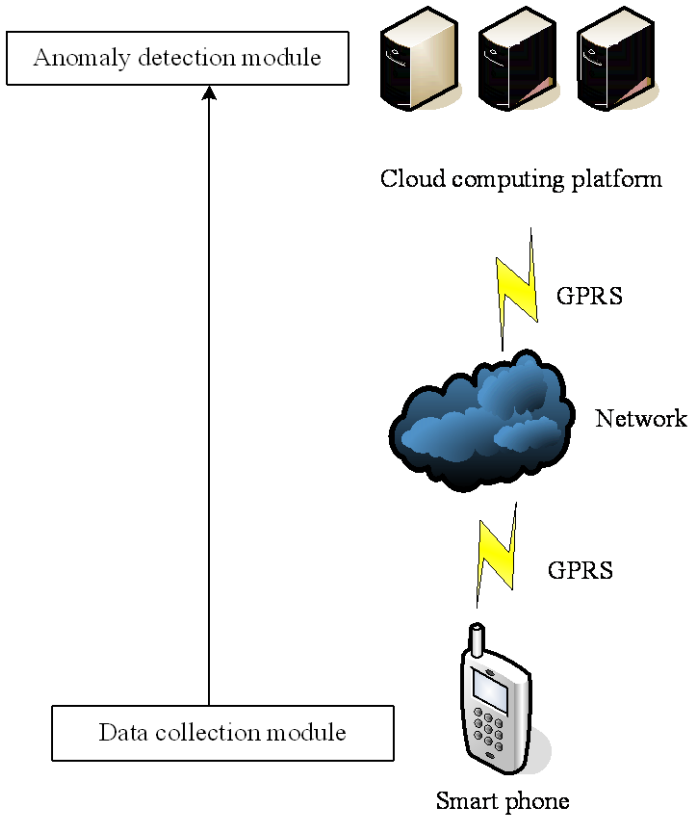


Figure 3. The Proposed Anomaly Detection Framework

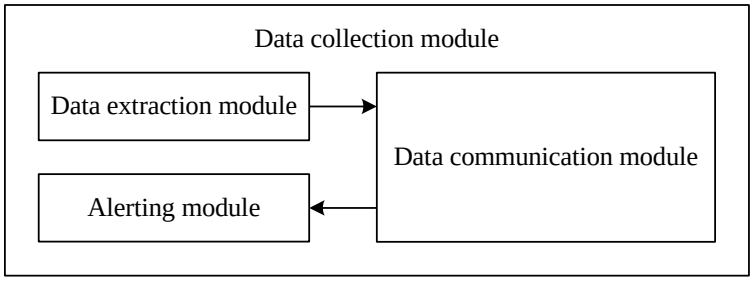


Figure 4. Data Collection Module

The data collection module runs on the android-based mobile phones in the form of a daemon. As illustrated in Figure 4, the data collection module contains three components, which are the data extraction module, data communication module, and alerting module, respectively. The data extraction module monitors the android operating systems and then

checks whether new software is installed. If new software is installed, the data extraction module will extract the feature from the android APK file and then transfer the feature to the data communication module. The data communication module sends the feature to the anomaly detection module through the GPRS and Internet network. The detection results are sent to the data communication module via the GPRS and Internet network and then the data communication module transfers the detention results to the alerting module. The alerting module alerts the users of android-based mobile phones with the detection results.

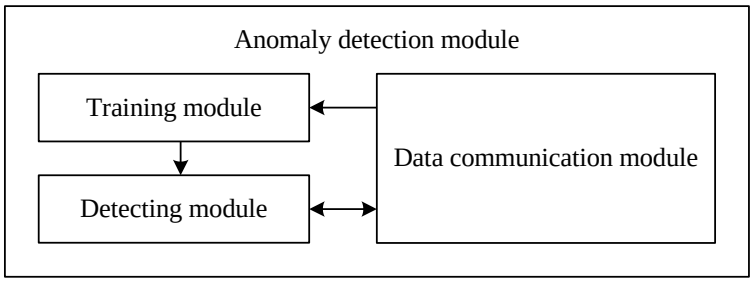


Figure 5. Anomaly Detection Module

As shown in Figure 5, the anomaly detection module contains three components, which are the data communication module, training module, and detecting module, respectively. The data communication module receives the feature from android-based mobile phones. The training module adopts the support vector machine to train two classifiers, which are the normal feature classifier and the abnormal feature classifier. The detecting module is in charge of detecting the abnormal behavior of software by using two classifiers and then sending the detection results to the data collection module.

3.2. Data Extraction Module

(1) Introduction of APK File

APK is an abbreviation of android package, namely the android installation package. It is developed by using Java language. Its source code is compiled and then packed into the APK file. In fact, specific files and directories are compressed into a APK file in the form of Zip format, and then the file extensions are changed into the APK format. Hence, file extensions of APK files are changed to Zip and then Zip files are uncompressed to obtain its contents. In this case, the data extraction module uses the zipfile module to extract the files and directories from the APK files.

Figure 6 shows the internal directory structure of APK file after being uncompressed.

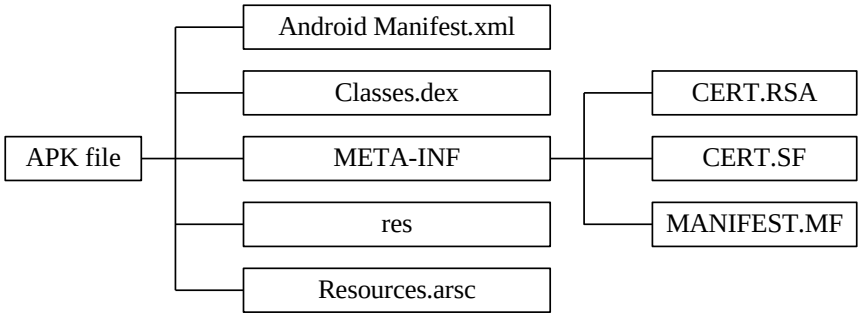


Figure 6. Internal Directory Structure of APK File

Files and directories of APK file are described in detailed in Table 2.

Table 2. Descriptions of Each File and Directory in APK File

File or Directory	Description
Android Manifest.xml	Global configuration file of application. It is used to describe the name, version, declaration of rights, and component information of application.
META-INF	Jar package information directory. It is used to store signature information such as CERT.RSA, CERT.SF, and MANIFEST.MF, and it is employed to ensure the integrity of APK file and then protect APK file from being changed maliciously.
Classes.dex	DEX is the format of Dalvik virtual machine execution file. DEX has been optimized for embedded devices.
res	This directory is used to store resource files.
Resources.arsc	It is a binary resource file.

Table 2 shows that a APK file often includes two directories and three files. The main information of the APK file are stored in the classes.dex file, which is the main file that implements the functions of APK file. Therefore, the data extraction module extracts the functions and class reference information from classes.dex file as the feature of APK file.

(2) Extracting Feature from DEX File

As illustrated in Figure 7, the data extraction module employs the Python programming language to extract the DEX file from the APK file and then gets the functions and class reference information from the DEX file. Finally, the DEX file feature in the form of the numerical vector is generated according to the android SDK class and function dictionary.

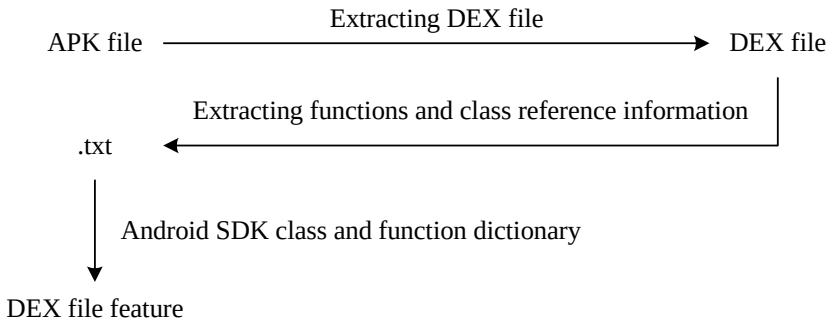


Figure 7. Process for Extracting Feature from DEX File

4. Experimental Results

In this section, experimental results are shown to assess the performance the proposed static anomaly detection framework.

In order to assess the performance of the proposed static anomaly detection framework, this paper compares the proposed static anomaly detection framework with the common static anomaly detection framework, which runs the data collection module and anomaly detection module on the android-based mobile phones and detects the abnormal behavior by using the k-Nearest Neighbor algorithm. The performance metrics that are measured in our experiments are the detection precision and the detection time, respectively.

Table 3. Experimental Results

Framework	Detection precision	Detection time
Static anomaly detection framework	89.5%	5s
Common anomaly detection framework	80.2%	30s

As listed in Table 3, our proposed static anomaly detection framework can significantly to decrease the detection time. That is because our proposed static anomaly detection framework deploys the anomaly detection module on the cloud computing platform and then exploits the strong computing capability and the storage capacity of cloud computing platform to reduce the detection time. In order to improve the detection precision, the proposed static anomaly detection framework employs the support vector machine to train a normal feature classifier and an abnormal feature classifier to detect abnormal behaviors of application, so it is better than common anomaly detection framework in terms of the detection precision.

5. Conclusion

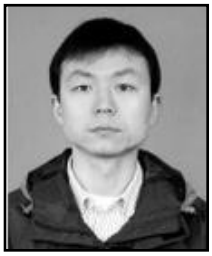
In this paper, an efficient static anomaly detection framework is proposed for android-based mobile phones. The proposed static anomaly detection framework consists of a data collection module and an anomaly detection module. The data collection module runs on the android-based mobile phones and it is in charge of extracting DEX file features from the APK file and communicating with anomaly detection module. The anomaly detection module is deployed on the cloud computing platform. The anomaly detection framework is responsible for training a normal feature classifier and an abnormal feature classifier by using the support vector machine, detecting the DEX file features from the android-based mobile phones, and communicating with the data collection module. Experimental results show that the proposed static anomaly detection framework has higher detection precision and shorter detection time than the common anomaly detection framework.

References

[1] W. Hu and K. Zhang, "Research and Implementation of Android Embedded Code Generation Method Based on Rule Model", International Journal of Multimedia and Ubiquitous Engineering, vol. 9, no. 11, **(2014)**, pp. 273-282.
[2] Q. Qian, J. Cai, M. Xie, and R. Zhang, "Malicious Behavior Analysis for Android Applications", International Journal of Network Security, vol. 18, no. 1, **(2016)**, pp. 182-192.
[3] V. Moonsamy, J. Rong, and S. Liu, "Mining Permission Patterns for Contrasting Clean and Malicious Android Applications", Future Generation Computer Systems, vol. 36, **(2014)**, pp. 122-132.
[4] Y. J. Ham, D. Moon, H.-W. Lee, J. D. Lim, and J. N. Kim, "Android Mobile Application System Call Event Pattern Analysis for Determination of Malicious Attack", International Journal of Security and its Applications, vol. 8, no. 1, **(2014)**, pp. 231-246.
[5] H. Han, R. Li, and X. Gu, "Identifying Malicious Android Apps Using Permissions and System Events", International Journal of Embedded Systems, vol. 8, no. 1, **(2016)**, pp. 46-58.
[6] S. Mansfield-Devine, "Android Architecture: Attacking the Weak Points", Network Security, vol. 2012, no. 10, **(2012)**, pp. 5-12.
[7] C. M. Li, L. Huang, and R. Wang, "The Design and Implementation of Android-based Mobile Learning Platform", Applied Mechanics and Materials, vol. 672-674, **(2014)**, pp. 1981-1984.
[8] W. Hu and Y. Zhao, "Analysis on Process Code Schedule of Android Dalvik Virtual Machine", International Journal of Hybrid Information Technology, vol. 7, no. 3, **(2014)**, pp. 393-403.
[9] M. Butler, "Android: Changing the Mobile Landscape", IEEE Pervasive Computing, vol. 10, no. 1, **(2011)**, pp. 4-7.
[10] A. Amamra, J.-M. Robert, and C. Talhi, "Enhancing Malware Detection for Android Systems Using A System Call Filtering and Abstraction Process", Security and Communication Networks, vol. 8, no. 7, **(2015)**, pp. 1179-1192.

[11] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, and M. Conti, M. Rajarajan, "Android Security: A Survey of Issues, Malware Penetration, and Defenses", IEEE Communications Surveys and Tutorials, vol. 17, no. 2, **(2015)**, pp. 998-1022.
[12] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of Machine Learning Classifiers for Mobile Malware Detection", Soft Computing, vol. 20, no. 1, **(2016)**, pp. 343-357.
[13] H. Kang, J.-W. Jang, A. Mohaisen, H. K. Kim, "Detecting and Classifying Android Malware Using Static Analysis Along with Creator Information", International Journal of Distributed Sensor Networks, vol. 2015, **(2015)**.
[14] S.-H. Seo, A. Gupta, A. M. Sallam, E. Bertino, and K. Yim, "Detecting Mobile Malware Threats to Homeland Security Through Static Analysis", Journal of Network and Computer Applications, vol. 38, no. 1, **(2014)**, pp. 43-53.
[15] M. Spreitzenbarth, F. Freiling, F. Echtler, T. Schreck, and J. Hoffmann, "Mobile-sandbox: Having A Deeper Look into Android Applications", in Proceedings of the 28th Annual ACM Symposium on Applied Computing, **(2013)**, pp. 1808-1815.
[16] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution", in Proceedings of the 33rd IEEE Symposium on Security and Privacy, **(2012)**, pp. 95-109.

Authors



Xiaobo Ji, He received his B.S. degree in Command Automation and the M.S. degree in Computer Science from Logistical Engineering University of China, Chongqing, China, in 2002 and 2005, respectively. He also received his Ph.D. degree in Computer Science from Chongqing University, Chongqing, China, in 2011. Currently, he is with the Department of Information, Research Institute of Field Surgery, Daping Hospital, Third Military Medical University, Chongqing, China. His current research interests include distributed system and flash memory.

Fan Zeng, She received the B.S. degree in Computer Science from Chongqing Communication Institute, Chongqing, China, in 1997 and the M.S. degree in Science of Educational Management from Southwest University, Chongqing, China, in 1999. Currently, she is with the Department of Information, Research Institute of Field Surgery, Daping Hospital, Third Military Medical University, Chongqing, China. Her research interests include distributed system and flash memory.

Bangxian Ye, He is pursing his B.S. degree from Faculty of Software, Fujian Normal University, China. His current research interests include wireless sensor network and flash memory.