Detecting Malicious Apps through Real-Time Behavior Monitoring for Android Phone

Eun Su Jeong

Korea University, Center for Information and Security Technologies (CIST),
Anam Dong, Sungbuk Gu, Seoul, Korea
eunsu.jeong@sk.com

Abstract. In the Android platform environment, various techniques to detect personal information leakage are being introduced recently but effective blocking is still long way off. The proposed scheme intends to securely protect personal information on smartphones by monitoring behaviors of various Apps. If an App violates any behavior-based rule, the proposed scheme blocks running the behaviors of the App. For this purpose, I classified the behaviors of smartphone applications and defined the behaviors to monitor. I also proposed the architecture to apply it in the Android framework and applied the proposed scheme in the Android smartphone.

Keywords: Behavior Detection, Malicious application, Android phone.

1 Introduction

Because most Android Apps are implemented in Java that is highly portable, malicious code injection and repackaging are not so difficult due to ease of reverse engineering. In addition, Android Play Store [1] of Google entrusts everything to developers. So the risk of distribution of malicious code is greater than Apple's App Store.

In the Android platform environment, various techniques to detect personal information leakage are being introduced currently but effective blocking is still long way off. Initial Android malware used a comparatively simple method that injects malicious code in a well-known App and repackages it. Thus the signature-based detection was possible that scans source code and content. However, recent malwares are applying obfuscation and encryption to source codes to free from the signature-based detection. In addition, they are installed as normal Apps but they download source code via Internet. Thus the signature-based detection in the application level is impossible.

This paper intends to securely protect personal information on smartphones by monitoring behaviors of various Apps. If an App violates any behavior-based rule, the proposed scheme blocks the App. For this purpose, I classified various App behaviors on the Android smartphone environment, defined vulnerable behaviors for monitoring, proposed the architecture to apply in the Android framework

This paper is organized as follows: Chapter 2 finds the characteristics of previous studies for detecting personal information leakage. Chapter 3 classifies vulnerable

James J. (Jong Hyuk) Park et al. (eds.), *Future Information Technology*,
Lecture Notes in Electrical Engineering 309,
DOI: 10.1007/978-3-642-55038-6_129,

App behaviors and the behaviors for monitoring. Chapter 4 introduces the architecture and operation techniques of the proposed scheme. Chapter 5 concludes the results and explains the study plan in the future.

2 Related Work

This chapter finds the characteristics of previous studies for monitoring personal information leakages. According to the detection level, this paper classifies previous studies into the Application Level and Kernel Level.

2.1 Detection of Application Level

In the Application Level, static analysis is possible as well as dynamic analysis. The static analysis in Android does not run an App but scans the content itself such as source code, permission information, layout, resource, and such. Di Cerbo et al. [2] used the Android permission in manifest.xml to predict leakage of personal information. Kim et al [3] disassembled DEX files as well as Android permission to create an API that tries to leak personal information. Using the API, they proposed a model that detects personal information leakage. Zhou et al [4] created a signature based on the permission and API to quickly detect well known malicious codes in the Android market. In order to detect new types of malwares that run by dynamically fetching codes, they proposed a heuristic detection scheme that detects both the original code and dynamic code. In addition, they verified the efficiency of the proposed scheme by implementing Droid Ranger. Dong-jie Wu et al [5] tested various clustering algorithms to effectively classify malwares that try to leak personal information by using a combination of permission, Android components, and API and found an optimal algorithm.

According to abovementioned studies, they performed the static analysis and dynamic analysis in the application level to detect personal information leakage. However, the static analyses in the application level are vulnerable to new types of attacks because it needs signatures of existing malicious codes.

2.2 Detection of Kernel Level

In the Kernel Level, it is possible to analyze behaviors in real time because system resources can be accessed but system modification is required. Burguera et al [6] recorded system call logs, created a vector by extracting the characteristics of personal information leaking behaviors, and proposed a malware detection method after applying the K-means algorithm. Takamasa Isohara et al [7] recorded the logs for system calls and parameters through system call hooking method and found the signature that is used during information leakage. Blasing et al [8] modified the system and created a virtual environment. They proposed a comprehensive analysis that detects malwares by analyzing the permission and recording system call logs. However, previous studies in the kernel level use log-based detection techniques. So it is difficult to detect and block malwares in real time.

3 Behavior-Based Detection

Let us classify the properties of smartphone applications based on the behavior and perform the risk analysis for each behavior. In addition, let us define vulnerable behaviors while running applications and specify the behaviors that require real-time monitoring.

Table 1. Behaviors to Monitor

Target Behavior	Device Status	App. Status	Blocking	Remarks
Taking Pictures	Disp. Off	-	Always	Privacy Issues
	Disp. On	background	Always	
Collection of Locations	Disp. Off	-	Optional	Sport Tracker
	Disp. On	background	Optional	
Access to Personal Information(Call Logs, SMS Logs, Address book)	Disp. Off	-	Optional	Indexing Service
	Disp. On	background	Optional	
Access to Private files(Photos, Videos)	Disp. Off	-	Optional	Virus Scan
	Disp. On	background	Optional	
Network Access	Disp. Off	-	Optional	Message App., VOIP
	Disp. On	background	Optional	
Phone Calling, SMS delivery	Disp. Off	-	Always	Not Intended Calling
	Disp. On	background	Optional	Background Service while Calling

The behaviors to monitor as access behaviors to personal information or malicious behaviors are selected as in Table 1 to determine whether or not to block them through real-time monitoring. Especially, the behaviors to monitor are taking photos, collection of position information, access to personal information, access to files of user, network access, calling phones, SMS delivery, and such. Furthermore, it requires a detection map that runs these behaviors to monitor while in LCD On/Off or background status. When the corresponding App calls an API for these behaviors, user must be allowed to block desired behaviors and other threatening behaviors must be blocked always.

4 Proposed Scheme: Android Malware Detection Scheme through Application Behavior Monitoring

In order to securely protect personal information from potential security threats that cannot be screened by existing anti-virus programs by extracting unknown security threats found in behaviors on an Android device, I propose the scheme that blocks running the corresponding application through real-time application behavior monitoring and calling pattern recognition.

4.1 Operational Principle of the Proposed Scheme

The proposed scheme intends to securely protect personal information on smartphones by monitoring behaviors of various Apps. If an App violates any behavior-based rule, the proposed scheme blocks the App.

Fig. 1 shows the operational principle of the proposed scheme. The Rule Check Layer in an Android device determines whether or not to block behaviors of an App running by comparing with the predefined black and white lists. User can decide whether or not to run the behavior when an alert is received that says the behavior is not allowed. In addition, the rule database that has malicious behaviors is periodically updated so detection is possible in real time.

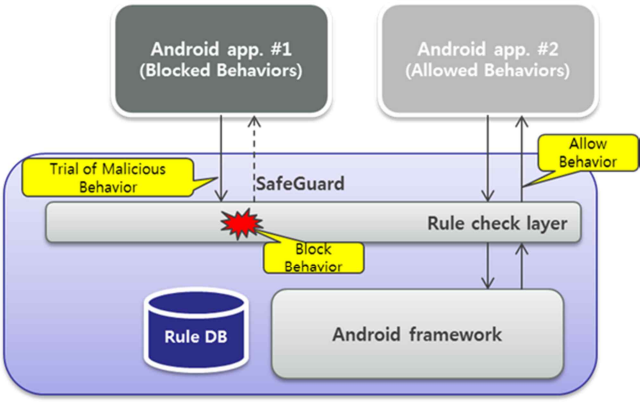

Fig. 1. Operational Principle of Behavior-based Malware Blocking System

4.2 System Architecture

In order to detect malicious behaviors of Android applications in real time, the system must interwork with the components in the platform. Accordingly, as in Fig. 2, I propose to add the components of the basic Android platform on the right.

The roles and functions of newly added modules are as the following. The SafeGuard library is added to Android RT to receive the call logs of core libraries and decide whether or not to run a behavior according to the predefined rules. If any rule violation happens, it notifies user of an alarm and an exception occurs to block the behavior. The rule provider stores allowed API calls for each application for management and the rule update service receives the rules that are pushed remotely and delivers them to the rule provider. The AppMonitor module displays alarms to user, provides an interface to change the rules, and notifies user of blocking results and blocking rules, and determines whether or not to use the rule for user deletion. It is provided in the form of App to be installed and it can be updated for support of additional rules.

Upon call of the core library of an Android App, it is required to insert a code to several methods in the core library and Content Provider. For this, collaboration is required from the device manufacturer. According to the behavior to monitor, the method to insert is determined. So at the initial stage, starting with several limited

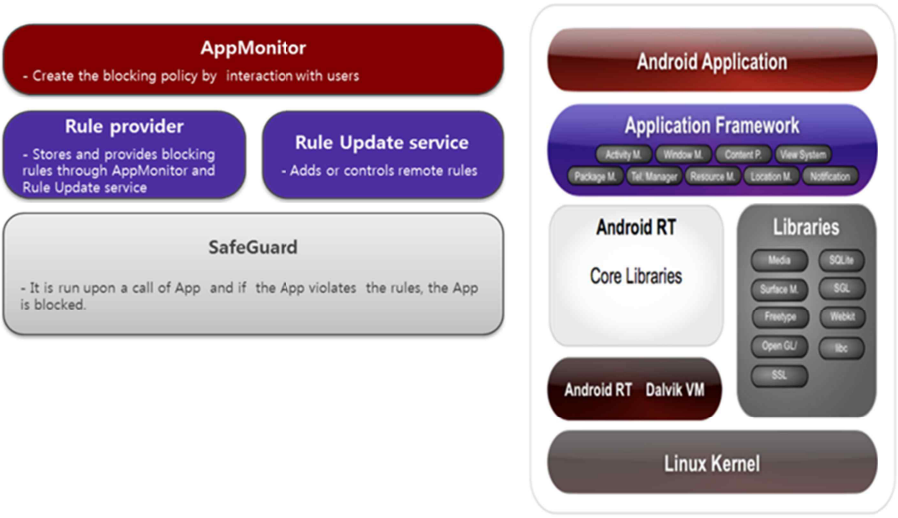

Fig. 2. System Structure

behaviors is recommended. The core library with an inserted code is dependent on the SafeGuard library. So it is required to preload the SafeGuard library along with the Android core library.

4.3 SafeGuard Implementation

Fig. 3 below shows the examples for the SafeGuard App implementation. The App has the blocking rule selection menu, which enables user to set the custom rule according to the danger level. Using the blocking rule update function, the changed or added rules are downloaded in real time while synchronizing with the server and the detection function is run according to the updated rules.

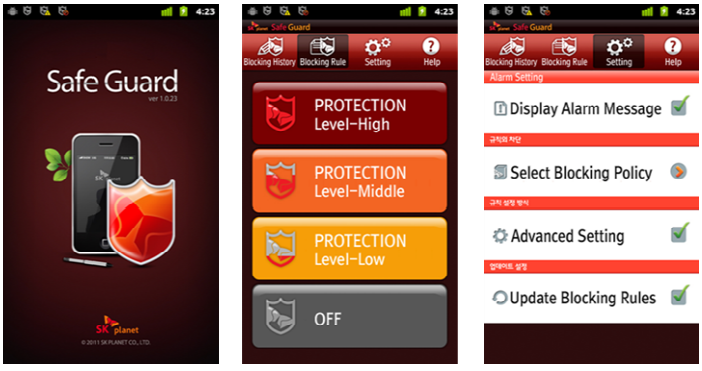

Fig. 3. User Interfaces of SafeGuard App

5 Conclusions and Future Work

This paper intends to securely protect personal information on smartphones by monitoring behaviors of various Apps. If an App violates any behavior-based rule, the proposed scheme blocks the App. In order to get over the limitation of static analysis and dynamic analysis in the previous application or kernel level, I proposed the behavior-based monitoring scheme that blocks malicious behaviors through real-time monitoring over risky behaviors. Furthermore, I verified that the scheme can be applied to smartphones through implementation.

In the future, I will study the different detection approach against privilege escalation attacks and potential threats that try to insert a hooking module to evade from the proposed technique.

References

1. https://play.google.com/store
2. Di Cerbo, F., Girardello, A., Michahelles, F., Voronkova, S.: Detection of malicious applications on android os. In: Proceedings of the 4th International Conference on Computational Forensics, IWCF 2010, pp. 138–149 (November 2011)
3. Kim, S., Cho, J.I., Myeong, H.W., Lee, D.H.: A study on static analysis model of mobile application for privacy protection. Computer Science and Convergence 114, 529–540 (2012)
4. 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 19th Annual Network & Distributed System Security Symposium (February 2012)
5. Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., Wu, K.-P.: DroidMat: Android Malware Detection through Manifest and API Calls Tracing. In: 2012 Seventh Asia Joint Conference on Information Security (Asia JCIS), pp. 62–29 (August 2012)
6. 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 (2011)
7. Isohara, T., Takemori, K., Kubota, A.: Kernel-based Behavior Analysis for Android Malware Detection. In: 2011 Seventh International Conference on Computational Intelligence and Security, pp. 1011–1015 (December 2011)
8. Blasing, T., Batyuk, L., Schmidt, A.D., Camtepe, S.A., Albayrak, S.: An Android Application Sandbox system for suspicious software detection. In: 2010 5th International Conference on Malicious and Unwanted Software (MALWARE), pp. 55–62 (October 2010)