SCIS 2011 The 2011 Symposium on Cryptography and Information Security
Kokura, Japan, Jan. 25-28, 2011
The Institute of Electronics, Information and Communication Engineers

An Enhanced Security Policy Framework for Android

Yi Jae Park * Doyoung Chung * Made Harta Dwijaksara * Jangseong Kim * Kwangjo Kim *

Abstract— Googles Android, a smartphone operating system, is used by many users, and the number of its users is increasing. However, because in Android Market, the application store only for Android, there is no examination of security of applications and Android users can install applications without stores; malicious applications can circumvent the safety verification. As a result, Android users are exposed to malicious applications without any security privacy protection. Then, the verification process for the application security policy is an important issue, but there are static verification processes and inefficient dynamic verification processes so far. In this paper, to resolve these weaknesses, we propose an enhanced security policy framework for Android. In the framework, users can check dynamically on the permissions used by applications, allow the permissions of applications partially, and modify the permissions of the applications. With the framework, users can distinguish malicious applications.

Keywords: Smartphone security, Android security, Permissions

1 Introduction

After releasing Apple iPhone in 2009 Korea into customers, the number of smartphone users is growing dramatically. Especially, the market share of Android, a smartphone operating system of Google, is growing with its advantages: free release, multi-channel and multi-carrier OS. Gartner, Inc., an information technology research firm, forecast that the market share of Android in smartphone OS will be in second place [**?**]. As shown in Figure **??**, the representative smartphone operating systems are Apple's iOS, Nokia's Symbian, Google's Android, RIM's BlackBerry, and Microsoft's Windows Mobile.

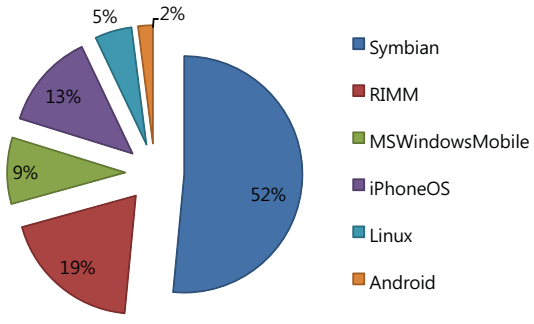

Figure 1: Worldwide Smartphone OS Marketshare as of Q2 2009 [**?**]

Along with the growth of smartphone market, the market of smartphone applications (called application store) is increasing. Two representative smartphone application markets are Apple's AppStroe and Google's Android Market. Applications sold at Apple's AppStore are registered only if it is verified by application-verification processes that there is no fault in the security and legitimacy of the applications. On the other hand, because, in Android Market, applications are released without application-verification processes for the security and legitimacy of the applications, Android smartphone users are exposed to malicious applications without any security privacy protection. Furthermore, since Android smartphone users can install applications without Android Market, the problem becomes more serious.

In addition, for the installation of an application of the Android operating system, users have to approve the permissions specified by the developers of the application. However, because there is no confirmation of the permissions of the applications after the installation, even though the permissions are used by the application for wrong purposes, users and the Android operating system cannot verify uses of the permissions [**?**, **?**].

Due to multi-purpose usage and multi-tasking features, the battery life of smartphones is shorter than typical mobile phones [**?**]. Because of the battery life of smartphones, antivirus applications for the security in the Android operating system are inefficient.

In this paper, for energy efficiency of the security policy verification process in the Android operating system, we propose an enhanced security policy framework for Android. Our proposed framework has three

* Department of Computer Science, KAIST, 291 Daehak-ro(373-1 Guseong-dong), Yuseong-gu, Daejeon 305-701, Republic of Korea, (krad, wordspqr, made.harta, jskim.withkals, kkj@kaist.ac.kr)

characteristics. First, users can check dynamically on the access permissions which applications use. Second, users can allow access the permissions of applications partially. Third, users can modify the access permissions of the installed applications. With the proposed framework, users can distinguish between malicious applications and general applications.

The organization of this paper is as follow: in Section 2, we discuss with the related work; then, we present our proposed framework in Section 3 and analyze the proposed framework; Finally, we conclude this paper with short summary and future work in Section 5.

2 Background and Related Work

2.1 Android

Android is a software-stack containing for mobile devices, operating system, middleware, and core applications. Its architecture comprises of four layers. Android applications are placed on top of the Android layer stack, which are supported by underlying three layers that include application framework, Android runtime, and Linux kernel.

2.1.1 Application Framework

Android provides an open development platform and helps developers to build rich an innovative applications. Developers are free to take advantage of the hardware of device, access location information (GPS), run background services, add notifications to the status bar, and so on.

Also, the developers can access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components. If an application allows the usage of its components subject to security constraints, any other application can utilze the components.

2.1.2 Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

2.1.3 Android Runtime

Android runtime has core libraries and Dalvik virtual machine. Besides libraries above, Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Dalvik virtual machine runs applications which have been converted into a compact Dalvik Executable format suitable for systems that are constrained in terms of memory and processor speed. Every Android application runs in its own process, with its own instance of Dalvik virtual machine.

2.1.4 Linux Kernel

Android relies on Linux version 2.6 for core system services such as hardware (display, camera, *etc.*), security, memory management, process management, and network. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

2.2 Comparison with Android and iPhone OS

In 2010, the most popular operating systems for smartphones are Android and iPhone OS. Although there are many differences with Android and iPhone OS, two most representative differences are application stores for developers and applications.

In the view of application store: Both Android and iPhone OS have their own application store - Android Market for Android and App Store for iPhone OS. In the situation of application developers, there are some differences. First, Android Market ask developers for 25$ as fee only once. However, App Store ask developers for 99$ per year [**?**, **?**].

Also, App Store needs some identification of developers like contact, a business registration and so on, but Android Market needs only developers gmail account.

Finally, the biggest difference in application stores is the verification processing of applications. In App Store, when a developer uploads its application to App Store, after a review for more than 5 days, users can download the application. If the application cannot pass the reviewing test, users cannot see the application. However, in Android Market, there is no verification test of applications. If developers upload their applications to Android Market, users can download them immediately. No verification test of application can be a security hole of Android.

In the view of applications: There are two differences between Android and iPhone OS: how to get applications and how to enforce restriction. In iPhone (along with iPod Touch and iPad), if users do not jailbreak their own iPhone, they can get applications from only App Store. However, in Android, users can get applications not only Android Market, but also application stores of 3rd parties and Internet directly. In order to get an application, users can download an Android package file (apk file) of the application and just install the application on their mobile devices.

Apple, the seller of iPhone, places some restrictions on applications. For example, developers cannot make an application which accesses important part of OS or data. But in Android, there is no restriction like iPhones. In addition to this, the source of Android is opened.

2.3 Related Work

W. Enck *et al.*, in 2009, proposed lightweight mobile phone application certification technique for static security policy verification in the Android operating system [**?**]. This technique judges whether an application has some potential risks or not when the application is installed. However, because this technique does not judge whether the application has some risks when the application is running, it is hard for this technique to assess risks of an application accurately.

A. Chaudhuri proposed a technique which evaluates the trustworthiness of applications. Their proposed technique provides static safety analysis of applicaion code on the basis of formal specifications of APIs provided the SDK [**?**]. The proposed technique anticipates data flow of applications by analyzing codes of applications. However, because there can be a debate that whether the application code analysis is precise, efficient and feasible to observe the data flow by analyzing codes of applications.

Unlike previously described techniques, M. Ongtang *et al.* proposed SAINT scheme to secure the Android platform [**?**]. They highlighted different aspects of security vulnerability in Android system and extended Android framework to provide a more controlled and secure environment for applications to interact with other applications and resources they can acquire. However, they are not ensuring application trustworthiness for trust and safety of Android users, but are focusing on application integrity to protect application itself from corrupting or malfunctioning.

M. Alam [**?**] proposes Android runtime security policy enforcement framework for dynamic security policy verification. The proposed framework by [**?**] assesses risks of an application based on the sequence of permissions the application needs in runtime. However, there are some vulnerabilities: attackers can break into the Android operating system by the sequences of permissions classified as secure sequences, and who will classify secure sequences of permissions.

3 Our Proposed Framework

Android operating system does not investigate application runtime behavior to ensure its trustworthiness and prevent any application from malfunctioning [**?**]. In order to solve these weaknessess, we propose an Enhanced Security Policy Framework for Android. The proposed framework is composed of 3 major parts: EDAPA (Efficient Dynamic Access Permission Analysis) which dynamically examines access permissions that applications use, pAPA (partial Access Permission Allow) which allows partially access permissions of applications, and to modify the permissions of the applications. Figure **??** shows the illustration of our proposed framework in the Android operating system. Compared to the previous approaches [**?**, **?**], which should perform additional computational overhead to check access permission of a service, our approach only checks access permission of the service by checking the corresponding system calls which are related to the registered access permissions by an end-user (*i.e.*, sending SMS and accessing web).

3.1 EDAPA

In Android platform, in order to examine the security policy of running applications, it requires a function that monitors continuously these applications, but the function causes an issue in which the battery of the smartphone can be reduced. In order to solve this issue and do efficient and dynamic analysis of the security policy of running applications, we propose EDAPA technique.

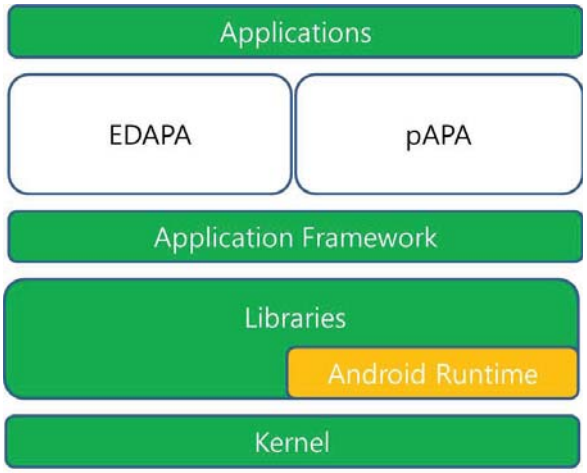

Figure 2: Our Proposed Framework in the Android Operating System

A user, first, sets up permissions he/she wants to protect. Next, if an application calls the permission the user sets up, the proposed framework reports to the user that the permission was called. By this technique, users can notice the applications that request the permissions the users want to protect and the malicious system call.

The dynamic verification process of EDAPA can be represented as in Figure **??**. For example, if a user sets up the permission to send SMS (SEND_SMS) and the permissions to make a call (CALL_PHONE) as the protected permissions, EDAPA informs the user when an application requests the system call for SEND_SMS or CALL_PHONE.

Because EDAPA informs users whenever the protected permissions are called, the convenience of users falls. In order to solve this issue, users can set up particular applications that EDAPA does not monitor although the applications call the protected permissions.

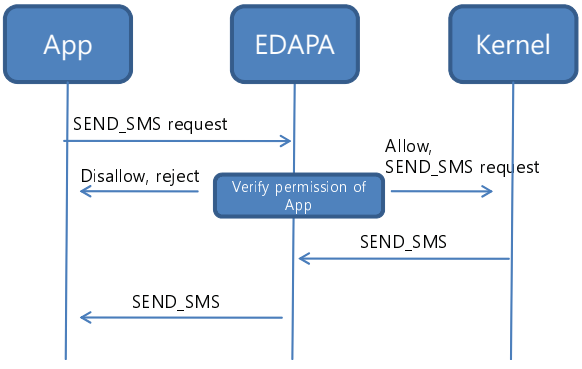

Figure 3: Dynamic Verification Process of EDAPA

3.2 pAPA

For the installation of an application in the Android operating system, users have to approve the permissions specified by the developers of the application. However, an instance that a user does not want an application to have all permissions specified by the developer may happen. By pAPA, users can allow partially access permissions of applications. Figure **??** shows interaction between pAPA and users at the installation time.

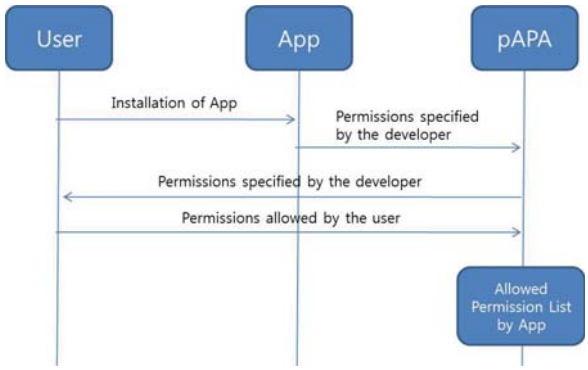

Figure 4: pAPA at Installation

For example, we suppose that there is a chatting application which, in the first run after the installation, searches phone numbers in the telephone directory of the user, finds persons who are in the telephone directory of the user and have the same application, and add the persons into the friend list of itself automatically. There can be a user of the application who does not want automatic adding the persons. In current Android operating system, because users have to approve all permissions specified by the developer of the application, they cannot prevent the application from adding the persons automatically. In the proposed framework, with pAPA, when a user installs the chatting application, if the user does not allow the permission to read the telephone directory (READ_contacts), the user can prevent the application from adding the persons automatically.

In pAPA, there are two cases of allowing the access permission partially: the permission is a monitoring target of EDAPA or not. Even though the permission called by an application is allowed by pAPA, if the permission is a monitoring target of EDAPA, the permissions are confirmed by the user. Figure **??** shows pAPA and a permissions which is not monitoring target of EDAPA, and Figure **??** shows pAPA, a permissions which is a monitoring target of EDAPA, and EDAPA.

By pAPA, users prevent malicious applications which are disguised as useful applications. When users install an application, they can install the application without doubtful permissions of the application with pAPA. For example, when a user installs a navigation application, we supposed that the navigation application needs the permissions to locate the user by

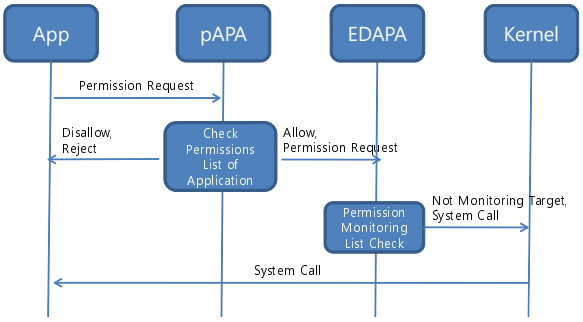

Figure 5: pAPA and Permission which is not monitoring target of EDAPA

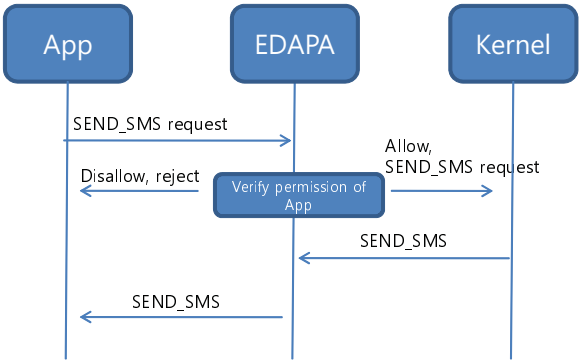

Figure 6: pAPA, Permission which is a monitoring target of EDAPA

GPS (ACCESS_FINE_LOCATION), to locate the user by Cell-ID (ACCESS_COARSE_LOCATION), and to make a call (CALL_PHONE). To users, the permissions to make a call of the navigation application can be doubtful. At this time, by pAPA, users can allow two permissions other than CALL_PHONE.

There can be a vulnerable point caused by pAPA: if a permission is essential for an application but is not allowed by pAPA, the application can make an error related to the permission. For example, let us look again the above navigation application. Assume that the navigation application needs CALL_PHONE permission at the beginning of the application and CALL_PHONE permission for the application is not allowed by pAPA, the navigation application might not run. In this case, a pseudo-permission can be a solution. To the application, which needs the permission the user does not want to allow, pAPA gives a pseudo-permission for the permission. Pseudo-permissions are recognized as real permissions by applications, but with pseudo-permissions, applications cannot perform functions related to the permissions.

3.3 Modification of the Permissions of the Applications

In the present Android operating system, the users cannot modify the permissions after the installation

Table 1: Comparison with other techniques

	Security Analysis	Detection	Battery Consumption	System Resource
W. Enck *et al.* [**?**]	Static	Policy	Low	Medium
Antivirus applications [**?**]	Dynamic	Signature of applications	High	High
M. Alam [**?**]	Dynamic	Sequence of permissions	Medium	Medium
Ours	Dynamic	System call	Low	Low

time. However, in the proposed framework, users can modify the permissions of the applications. Users can set more limits of permissions on applications which need more verifications and uncap limits of permissions on applications of which the verification process is concluded.

4 Comparison

In Table **??**, we compare the characteristics of our framework and existing security policy verification techniques [**?**, **?**]. Existing studies of Android security have analyzed applications statically. Lightweight mobile phone application certification [**?**], the static verification technique of W. Enck *et al.*, extract the security enforcement policy from corresponding Manifest file, and compare the extracted policy with their predefined invariants at installation in order to certify the application trustworthiness or maliciousness. In the static analyses, if the application passes the static verification process at the installation time, users cannot realize that the application does something malicious after the installation time.

By antivirus applications for the Android operating system [**?**], the users can solve this weakness of the static analysis. Because antivirus applications are not in the Android operating system, the applications must always occupy the system resources of the Android operating system. Then, the applications can cause shorter battery life.

M. Alam [**?**] tried to judge whether an application has some risks or not by adding a dynamic verification process of the sequence of the permissions the application needs into the Android operating system. However, for the analysis of the sequence of the permissions, the process have to always occupy the system resources, and although a sequence of permissions is already classified as a secure sequence, attacker can attack the Android operating system with the sequence.

In order to solve these weaknesses, our framework analyzes the access permissions of the applications dynamically by observing the corresponding system calls in Android operating systems. Compared to the previous approaches [**?**, **?**], we believe that our approach is more efficient based on the following observations. First, verification cost for access permission are determined not by the number of applications owned by an end-user but by the number of the corresponding system calls registered by the end-user. Second, when new type of the application is installed, our approach can enforce the current registered access permissions of the end-user without any additional security analysis of the application. Also, in our framework, partial access permissions make it possible for the end-users to verify application by themselves.

5 Conclusion and Future Work

Smartphone security is more important than the security of typical mobile phones because smartphones have two functions of a mobile phone and a computer. Android is an operating system for smartphones, and after its release, users of Android are rapidly increasing. However, because no verification of security of applications in Android Market is excuted, Android users are exposed to malicious applications without any security privacy protection. To resolve these issues, some frameworks have been proposed, but presented static verification processes or inefficient dynamic verification processes. In this paper, using system calls for an efficient dynamic security policy verification, we improve inefficiency of existing security policy verification processes. Also, in the proposed framework, users can allow the permissions of applications partially, and modify the permissions of the applications, but in the present Android operating system, users cannot do that. By the proposed framework, users can distinguish between malicious applications and general applications.

As our future work, we will implement this framework on real devices and verify its pros and cons.

References

[1] Kyle, "Worldwide Smartphone OS Marketshare as of Q2 2009," `http://www.handheldnow.com/2009/12/25/there-are-way-too-many-cell-phone-operating-systems/`

[2] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google Android: A Comprehensive Security Assessment ," *IEEE Security & Privacy* , 2009.

[3] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, and S. Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms ," 2009.

[4] Y. F. Chang, C. S. Chen and H. Zhou , "Smart Phone for Mobile Commerce," *Computer Standards & Interfaces*, 2009.

[5] Android Developer Signup, `http://market.android.com/publish/signup`

[6] iOS Developer Program, `http://developer.apple.com/programs/ios/`

[7] W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," *Proceedings of the 16th ACM Conference on Computer and Communications Security* , ACM, 2009, pp. 235-245.

[8] A. Chaudhuri, "Language-based security on Android," *Proceedings of the ACM SIGPLAN Fourth Workshop on Programming Languages and Analysis for Security*, ACM, 2009, pp. 1-7.

[9] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically Rich Application-Centric Security in Android," *2009 Annual Computer Security Applications Conference* , IEEE, 2009, pp. 340-349.

[10] M. Alam, "Android Runtime Security Policy Enforcement Framework," *Journal of Personal and Ubiquitous Computing*, Springer, 2009.

[11] D.H. Kang, J.H. Han, Y.K. Lee, Y.S. Cho, S.W. Han, J.N. Kim, and H.S. Cho, "Smartphone Threats and Security Technology (in Korean)," *Electronics and Telecommunications Trends*, ETRI, 2010.