A proposal to realize the provision of secure Android applications
- ADMS: an application development and management system -

Harunobu Agematsu\(^1\), Junya Kani\(^1\), Kohei Nasaka\(^1\), Hideaki Kawabata\(^2\),
Takamasa Isohara\(^2\), Keisuke Takemori\(^2\), Masakatsu Nishigaki\(^3\)

\(^1\) Graduate school of Informatics, Shizuoka University
3-5-1 Johoku, Naka, Hamamatsu, Shizuoka, 432-8011 Japan
\{gs11002,cs08028,gs10041\}@s.inf.shizuoka.ac.jp

\(^2\) KDDI R&D Laboratories, Inc.
2-1-15 Ohara, Fujimino, Saitama, 356-8502 JAPAN
\{ kawabata, ta-isohara, takemori\}@kddilabs.jp

\(^3\) Graduate school of Science and Technology, Shizuoka University
3-5-1 Johoku, Naka, Hamamatsu, Shizuoka, 432-8011 Japan
nisigaki@s.inf.shizuoka.ac.jp

Abstract—To realize the provision of secure Android applications, this paper proposes an application development and management system, or ADMS for short, that is operated and maintained by application developers and the market manager. ADMS requires (i) Android OS to be equipped with a “security manager”, (ii) all application developers to embed a code for event notification into applications to tell every event to the security manager whenever an application launches a security-related event, and (iii) market manager to remove all such applications that don’t include the event notification code.

Keywords-component: Android smartphone; malicious Android application; security manager

I. INTRODUCTION

The number of Android phone (smartphones equipped with the Android OS) users have exploded in recent years. Users who have Android phones can access market places, download and install application software freely. Everyone can make applications and upload to the market places because the development environment is also freely available to the public.

Some famous markets such as Google Play Store [1] carry out to check all applications and remove malicious ones. However, there are untrusted markets, in which many malicious Android applications called Trojans pretending to be a good application infiltrates the untrusted market and causes many problems. Gold Dream [2] and Droid Dream [3] are the Trojans which are reported to be recently spreading throughout the Internet.

Once such an application is installed, its behavior looks like a good application from the user’s point of view but it can cause a leak of personal information (IMSI: International Mobile Equipment Identity, IMSI: International Mobile Subscriber Identity, telephone number, SMS: Short Message Service message, and call log) and send it to an outside server without the user’s consent. These malicious Android applications can also call and send SMS messages to other people and access any web site without the user’s permission.

To decrease damage caused by these bad applications, there are two possible measures: counter-measure from the user side and counter-measure from the application provider (market side.)

As for malicious android application prevention from the user’s side, each user judges applications by carefully checking the installation permission which appears when applications are installed or by using antivirus software. However, due to varying degrees of knowledge and consciousness concerning these issues it is difficult to provide effective overall counter-measures from the user’s side.

As for market side safety measures, market manager tests all applications in advance and provides only safe applications into the market (like Google Play Store [1], Apple store [4] and au market [5].) However, there are many and a variety of malicious applications in the market and it’s difficult to detect all of them. So, to prevent such applications from entering the market we need a more powerful counter-measure as soon as possible.

In this paper, we propose an application development and management system, or ADMS for short, which is operated and maintained by application developers and the market manager. ADMS requires (i) Android OS to be equipped with a “security manager”, (ii) all application developers to embed a code for event notification into applications to tell all events to the security manager whenever the application launches any security-related events, and (iii) market manager to assume that all applications that don’t include...
the event notification code are malicious in nature and automatically remove them from the market. The application has to implement the event notification code if it wants to prevent deletion from the market. As a result, security manager could be aware of all security-related events and could determine whether applications should be executed or not by following the security policy.

To realize this security goal without adding unnecessary burden to developers, we propose to prepare a dedicated API called "ADMS API" and to create "Knowledge Database" which security manager could use to judge malicious behavior. In this paper, we will focus on the details of “ADMS API.”

II. RELATED WORK

Some similar countermeasures against such malicious applications have already been proposed. Enck et al proposed “TaintDroid”, a system-wide dynamic taint tracking system, in which multiple sources of sensitive data are tainted and the taint is used as a marker capable of real-time tracking of sensitive data [5]. They implemented “TaintDroid” and the evaluation result said that the overhead time for taint tracking was about 29% at most.

Takemori et al proposed the “white-list” measure which allows only secure and necessary applications to work [6]. In the white-list all approved applications are shown. In order to prevent malicious applications from intruding, any unlisted application will be immediately deleted even if it is installed.

Kawabata et al pointed out the risk that attackers could execute Java method by using JavaScript downloaded from the server and it may in turn cause malicious behavior [7]. To counter this, they proposed to conduct a static analysis for Android applications with JavaScript to ascertain its threat level.

Chin et al mentioned vulnerability of inter-application Communication in Android [8]. They pointed out that some malicious components can eavesdrop and tamper with the “Intent” while sending and receiving a message between applications. They surveyed 100 applications and revealed that they undoubtedly have such vulnerabilities.

III. ANDROID OS

In Android OS [9], each application runs in its own memory space and they are isolated from each other and can’t access each other’s data. To communicate with other applications, they use an original message passing method in Android. We will describe this structure below.

3.1 Intent

Android provides a message passing system between inter-application, in which “Intent” is used to link to other applications. Intent is a message that is composed of information regarding a recipient and data. In addition to that, Intent is used to communicate from OS to application. This Intent is called “Broadcast Intent” and it’s generally sent to multiple applications at the same time.

There are two types of Intent: Explicit and Implicit Intent. While an explicit Intent identifies the intended recipient by name, an implicit Intent leaves it up to the Android platform to decide which application should receive the Intent.

3.2 Component

Intents are delivered between application components. There are four types of components in Android. If a developer wants to use these components, the developer has to declare it in a Manifest file.

1) Activity
Activity is a visible process provided on screen to users with user interfaces. Activity is launched by the Intent.

2) Service
Service is a background process invisible to the user. For example, Service may perform the operations to play the music and download the file as a background process.

3) BroadcastReceiver
BroadcastReceiver is a component to receive the Broadcast Intent from the OS. For example, if the application wants to know the OS boot event, it has to receive the Broadcast Intent by using the Broadcast Receiver.

4) ContentProvider
ContentProvider is a database. For example, the contact list and phone log are managed by this component.

3.3 Security Model

Android OS has a unique security structure. Android has a virtual machine called Dalvik (Figure 1) and all applications are executed in this sandbox. Basically, each application executed on Dalvik runs in logically-separate memory spaced and they are tightly isolated each other. To communicate with other applications, the Android application uses the Intent as described at Sec.3.1. To communicate with OS, the Android application has to get the “Permission” from the user to access information stored in OS.
The Permission request is shown just when the application is installed, and thus the user is asked to respond (Figure 2). Once the Permission is granted by the user, the application can use the API (Application Program Interface) which provides various functions to access the OS resources including user’s personal information such as IMEI, IMSI, and telephone number. API is located in the Application Framework layer in Android OS (Figure 1).

There are variety types of Permission, and each Permission is linked to the related APIs. For an example, Table 1 provides a list of APIs which the application can use when the “READ_PHONE_STATE” Permission is granted by the user.

<table>
<thead>
<tr>
<th>API</th>
<th>Available Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>getLine1Number</td>
<td>Phone Number</td>
</tr>
<tr>
<td>getDeviceId</td>
<td>Device ID</td>
</tr>
<tr>
<td>getSimCountryIso</td>
<td>Sim Country</td>
</tr>
<tr>
<td>getSimOperator</td>
<td>MCC+MNC (mobile country code &amp; mobile network code)</td>
</tr>
<tr>
<td>getSimOperatorName</td>
<td>SimOperatorName</td>
</tr>
<tr>
<td>getSimSerialNumber</td>
<td>SimSerialNumber</td>
</tr>
<tr>
<td>getSimState</td>
<td>SIM state(communication is available or not, PIN is rocked or not)</td>
</tr>
<tr>
<td>getVoiceMailNumber</td>
<td>VoiceMailNumber</td>
</tr>
</tbody>
</table>

The problem here is that the Permission request (Figure 2) is very abstract to ordinary users. It is difficult to determine what application will do what, and it is also hard to judge what kind of information applications will access. Due to this, it could be hard for most users to understand an application’s potential threat.

IV. PROPOSAL SYSTEM – ADMS

4.1 Overview

In this paper, we propose an application development and management system, or ADMS for short. ADMS requires (i) Android OS to be equipped with a “security manager”, (ii) all application developers to embed a code for event notification into applications to tell all events to the security manager whenever the application launches any security-related events, and (iii) market manager to assume that all applications that don’t include the event notification code to be malicious and automatically remove it from the market. The application has to implement the event notification code if it wants to prevent deletion from the market. As result, security manager could be aware of all security-related events during the application runs and could determine whether applications should be executed or not following the security policy. If the security manager finds any application in violation of the policy, the security manager can send the notification to the market manager, which allows the market manager to remove the violating application from the market.

To realize this security goal without adding unnecessary burden to developers, we prepared the dedicated API set called "ADMS APIs" (Figure 3). The ADMS APIs are a replicated set of all the security-related Android APIs such as getLine1Number and so on and they all are extended by...
To evaluate our prototype system, we made an Android application for evaluation. This application declares the READ_PHONE_STATE permissions to access phone number and when the user presses the button, the application calls the API. Figure 5 is the screenshot just when the user is pushing the button; while the application calls the API and gets the phone number, the security manager detects it and shows a warning pop-up to the user. We believe that this kind of warning is very useful information for the user, because basically ordinary users cannot understand application behaviors (which applications retrieve phone numbers, when and why, etc.).

V. CONCLUSION

In this paper, we propose an application development and management system, or ADMS for short. The ADMS provides secure Android applications to the users through the collaboration of Android phone (the security manager checks all the security-related events issued by applications), the application developer (it is required to attach ADMS APIs to the application), and the market manager (applications without ADMS APIs are removed from the market). The ADMS is an integrated structure that manages procedures, operations, and techniques, and in such a sense, the ADMS is a “management system” that is based on the idea of ISMS (Information Security Management System”). Currently the ADMS is in the proposal stage and we will later implement the knowledge database and establish procedures to detect malicious application behaviors. Thereafter we will demonstrate effectiveness of the ADMS and evaluate the detection accuracy.

REFERENCES

public String getLine1Number() {

    try {
        SecurityManager sm = new SecurityManager(); //notification code
        sm.notification(); //notification code

        return getSubscriberInfo().getLine1Number();
    } catch (RemoteException ex) {
        return null;
    } catch (NullPointerException ex) {
        // This could happen before phone restarts due to crashing
        return null;
    }
}

Figure 3. Sample code of ADMS API.

Figure 4. Security Manager
Figure 5. Warning to the user from Security Manager