Android Malicious Behavior Detection Based on Sensitive API Monitoring

Quan QIAN¹, Jing CAI, Rui ZHANG

School of Computer Engineering & Science, Shanghai 200444, China
qqian@shu.edu.cn

Abstract. A two-step model combining static permission filtering and sensitive API monitoring has been proposed for android malicious behavior analysis. Permission filtering matrix is used to determine whether an application has potential risks. And sensitive API monitoring, based on the reverse engineering, can monitor those sensitive APIs such as sending SMS, accessing user location, device ID, phone number, etc. From experiments, it shows that our proposed method is feasible and effective for monitoring this kind of malicious behavior.

Keywords: Android Security, privacy leakage, permissions filtering, malicious behavior monitoring

1 Introduction

With the popularity and rapid development of Android OS, its security issues are also increasingly prominent. For instance, the security report from NetQin Company shows that they detected more than 65,227 new malware in 2012, a 263% increase over 2011[1,2]. The main purpose of this paper is to analyze the Android applications accurately and comprehensively based on combining static and dynamic method to reveal the malicious behaviors of applications leaking user’s privacy data. Privacy leakage mentioned in this paper refers to Android applications using sensitive permissions granted by user during the installation to collect user’s privacy data, including user’s device ID IMEI, phone number, user contacts, call records, location information, etc., and send user’s privacy data via SMS or network.

2 Android malicious behavior analysis

Generally speaking, malware analysis mainly includes static and dynamic approach. Static analysis is a kind of method based on program’s source code[3]. Dynamic analysis refers to the tracking and monitoring its run-time behavior through running the program[4]. It is more accurate for capturing the actual malicious program

¹This work is partially supported by Shanghai Municipal Natural Science Foundation (13ZR1416100) and Key Principle Project of Shanghai Educational Committee (J50103).
behavior. We present a combination of static and dynamic security analysis model that enable the analysis of malicious behavior more comprehensively and accurately. Fig.1 shows the whole steps.

![Diagram](image)

**Fig.1.** Malicious behavior analysis framework for android App

### 2.1 Permission Filtering Module

Some permission may not exist risks by itself, but if there are some permission combined there may exist a security risk. For Android permissions, there are four different security levels. Those are **Normal**, **Dangerous**, **Signature** and **SignatureOrSystem**. Here, we mainly concern the Dangerous level which has great potential risks for leaking user privacy data. Fig.2 shows the procedure of permission filtering module.

![Diagram](image2)

**Fig.2.** The procedure of permissions filtering module
2.2 Dynamic Monitoring Module

This module is to monitor the call information of sensitive APIs in APK. We implement dynamic real-time monitoring by inserting monitoring code to the decompiled APK.

Smali and baksmali are an assembler and disassembler respectively for the dex format used by the DVM[5]. In this paper in order to avoid the differences between JVM and DVM, we try to directly rewrite Dalvik bytecode, insert the monitoring Smali bytecode into the decompiled Smali files. The procedure of dynamic monitoring module is shown in Fig.3.

A. Smali Bytecode

Smali is an Intermediate Representation(IR) for Dalvik Bytecode. Smali code is a kind register based language which can shield the source code level differences. For instance, malware sometimes use source code obfuscation to avoid detection. But in Smali code, the core sensitive APIs are inevitably exposed. So, we can monitor these sensitive APIs to track the behavior of those suspicious programs.

B. Smali bytecode library for sensitive APIs

The Smali bytecode library stores sensitive APIs and their corresponding Smali bytecode. The main function of the library is to locate the detailed position of sensitive APIs in Smali files after the target APK was decompiled.

C. Monitoring bytecode library for Sensitive APIs

The monitoring bytecode library is to store the sensitive APIs calling information when the APK is running. For different APIs, monitoring information to be recorded is different. Such as SMS sending text messages, we need to record the message recipients as well as the content of the message. The unique part of each API is its input and output. According to API's function prototypes and register naming principles in Smali syntax, we can obtain the Smali register number of each API parameters.
3 Experiments

To illustrate Android users facing the growing threat of information leakage, we choose 642 popular applications to conduct experiments in the Permissions filtering module. We handled 642 APK samples by permissions filtering module and found that almost 26% apps have security risks for leakage user's sensitive data.

The security policy violated by most of those 642 apps is about IMEI permission combinations, namely, the most common information leaked is IMEI. The reason may be the IMEI can determine phone type and device parameters, and can provide accurate user identity information for developers and advertisers. The next is phone Number, Contacts, and Location. If these sensitive information is used illegally, it will possibly bring huge losses to users.

To verify effectiveness and feasibility of dynamic monitoring module, we did experiments on the Android emulator in Windows7. We chose an APK, SendSMS_example.apk, that will automatically send text message in the background. The prototype system successfully detected the leakage behavior.

4 Conclusions

A two-step malicious Android application detection method was proposed in this paper. First of all, we use permission combination matrix to discover those potential risk applications. And then those suspicious applications are further sent into the dynamic monitoring module to track the call information of the sensitive APIs while it is running. As a conclusion, it shows some advantages of our approach:

1. Using Smali bytecode, it is based on intermediate language, which shows some advantages over Java source code method and it possesses anti-obfuscation to a certain degree.
2. The method is simple, just insert some monitoring Smali bytecode, and the increasing performance impacting can be ignored.
3. This method can be used in a wide scale, which can deploy remotely and provide monitoring service automatically.

References