A monitoring method of sensitive calls based on the Android platform software behavior

Cheng Sun, Sujuan Qin

State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

sun_cheng@bupt.edu.cn, qsujuan@bupt.edu.cn

Keywords: Android security, API Monitor, Xposed

Abstract. In recent years, due to the Android system, good openness and portability features, Android mobile phone system was able to end the rapid popularization and rapid development, providing a major service on the Android platform based on the Android platform for mobile software that is Android APP. Meanwhile, the open-source Android system, making the Android system itself is vulnerable to security issues; other Android system itself permission mechanism, also led to the abuse of privileges and authority Android APP inconsistent application problems. To solve these problems, this paper design a new framework based Xposed Android APP sensitive call behavior monitoring system for domestic several famous non-famous APP test, compared to the previous test mode awakened analysis of effectiveness, accuracy, and software performance assessment, the results show that this solution can provide more efficient, convenient, and accurate monitoring of the sensitive situation of APP API calls.

1. Introduction

With the rapid development of mobile Internet, the emergence of an increasingly rich Internet services, bring convenience, but abuse flooding APP permissions to public life. Ordinary users of Android phones are often unable to detect the use of APP in the background without the user's permission which were operating illegally. Specifically, users download and run a stand-alone game APP, Normally, such an APP would call some audio API interface, but some illegal APP will take the initiative to apply for networking, viewing contacts, location information, etc. private information permission. When the users use the APP, the private information of the phone will be leaked eventually through the background, and users themselves do not detect[3].

Paper[1] proposes a dynamic test under the environment by injecting HOOK way to hacker mode interference Android system uptime, obtain APP sensitive interfaces during operation call situation. By this way HOOK technology, injected So library functions can be directly in the underlying Android running on the target system in the process, when the process is responsible for APP application provides functionality will be noticed and get the address of the ioctl function, ioctl function Binder is responsible for the transmission of data communications, including calls for the descriptor function and code, intercepted ioctl function to deal with the object and remove the corresponding descriptor and code, the final output by Log function Android SDK development kit android.util.Log in Log into logcat, complete the monitoring. This process, by monitoring the process responsible for providing functions to complete the functionality of the API to provide surveillance, have the following questions: 1. HOOK injected into the process to be run So stored files, since the monitor is not released after each Operation will waste a lot of memory to run too long after the process will lead to the collapse of even a system crash; 2. complex permission request, the underlying need to obtain root privileges before they can run linux So file; 3 each time only for one injection process When API monitor incomplete case sensitive API APP call different process occurs;

4 for matching descriptor and the code is to get in C / C ++, due out Java layer, where the matching data need to Android source code Get , the process is cumbersome[4].

Aiming at the above problems[5], propose appropriate solutions. Experimental results show that this scheme has many advantages: improve the integrity of API monitoring, Reducing the difficulty of obtaining matching data, simply need root privileges, improve the stability and efficiency of the program.

Organizational structure of this paper are as follows: Section II describes and analyzes the Xposed Framework and its implementation principle; Section III for specific requirements process running Android APP API monitoring, design and implementation monitoring module; Section IV for several well-known domestic non-barrier Android APP famous test experiments, analysis of experimental data; Section V final summary presentation and related work.

2. Design and Implementation of monitoring system based on Xposed framework

2.1 Monitoring System.

As mentioned earlier in paper[2], Xposed framework will load XposedBridge.jar package all running APP process, now do not need to run APP violent disturbances[6][7], but the process needs to run APP, APP call API interface to obtain information, here to make the following design:

1. hook XposedBridge.jar package provided () function, before Zygote process starts, complete function HOOK process needs to be monitored.

2. Examples of XC_MethodHook class that implements beforeHookedMethod and afterHookedMethod method, these two functions are being HOOK before the function call and the call after the execution, where you can add log output in the corresponding function in after.

3. monitored maintain API library, where all required monitoring functions save descriptor, it can be implemented by HashMap.

4. Log log processing module, automatic output more readable log.

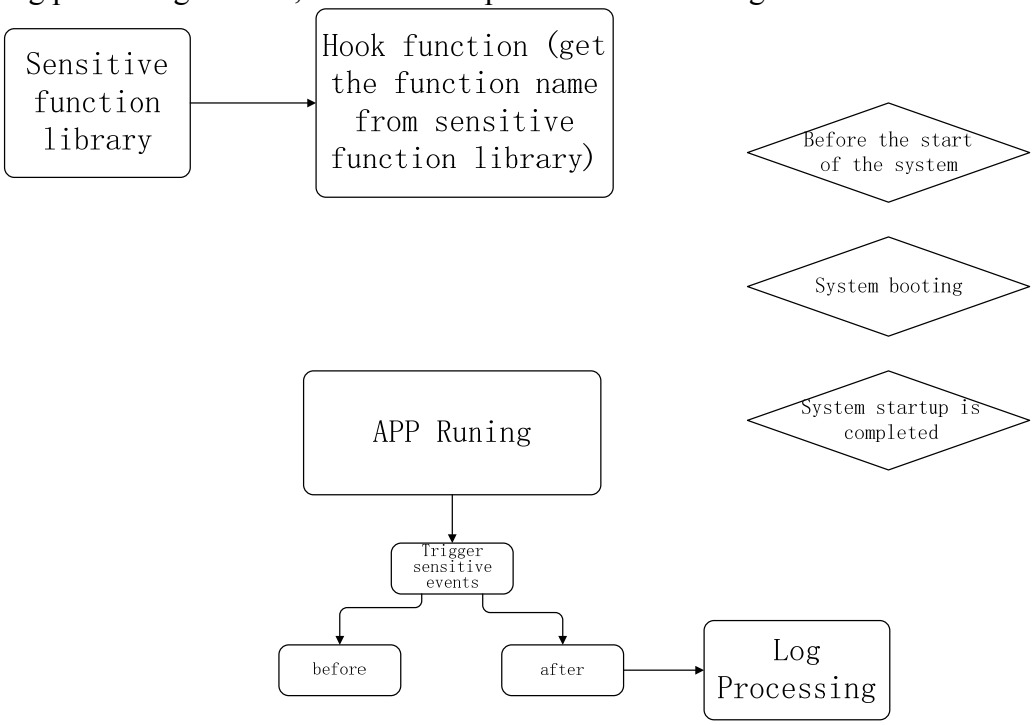

Figure I. Design flow

2.2 System key algorithms and code implementation.

First, you need to start before the Android system to complete the setup function Hook operation, although Xposed help function injection, but we need to define their own specific function here () function processed through hookAll, custom initZygote () function to achieve this operation.

Code: void initZygote () {

hookAll (XSmsManager.getInstances ());

```
    // SMS Manager
    hookAll (XActivityManager.getInstances ());
    // Activity Manager
}
```

hookAll () custom function, passing parameters of type enum type, enumerate various methods need to be monitored.

Here a SMS Manager, for example, include its class diagram.

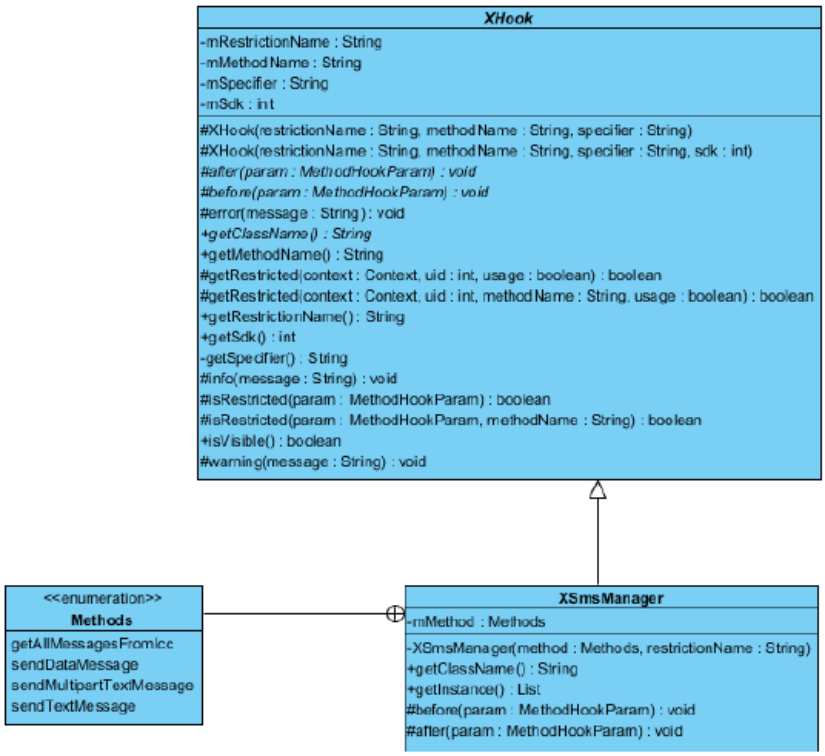

Figure II. Injection principle SmsManager

3. Assessment and analysis of experimental data

3.1 Purpose and methods

Experimentally using the following diagram flowchart, the environment is already installed Android4.0 mobile devices ZTE 930HD, by way of the following figure, were tested micro-channel, high moral map, Caculitor Calculator Get the resulting three APP testing methods described herein log log, comparative data log log [1] test methods were a comparison, final conclusion ---- proposed new test method described herein is more efficient, more comprehensive, more convenient and easier to maintain.

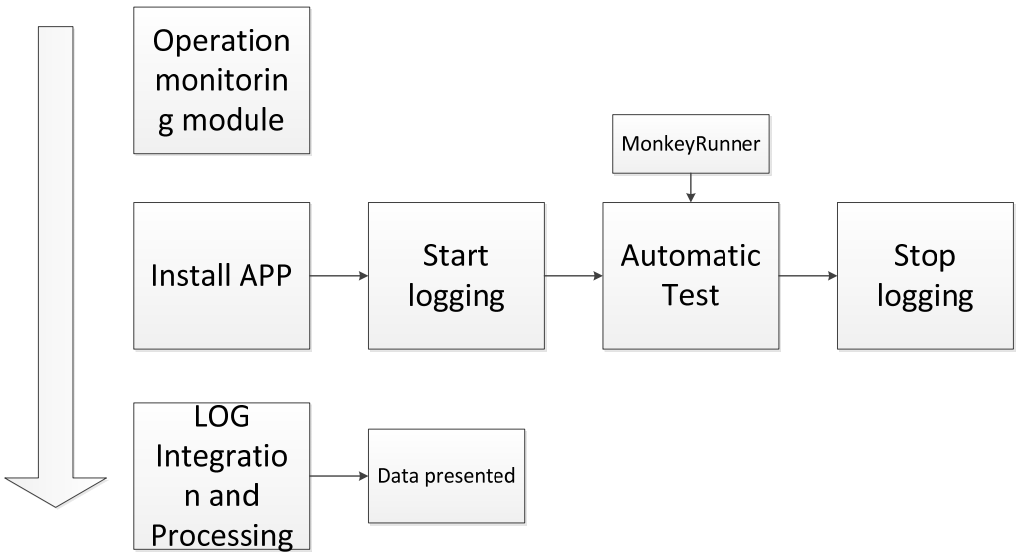

Figure III. Experimental flow chart

3.2 Data Analysis

Micro-channel running Android version during background monitoring has been carried out to obtain the following data:

Runtime:loadLibrary
Configuration.MCC(null;null)
ActivityManager: getRecentTasks(null;null)
ActivityManager: getRunningTasks(null;null)
TelephonyManager:getDeviceId(null;null)
ConnectivityManager: Connectivity.getAllNetworkInfo(null;null)
NetworkInfo: NetworkInfo.isConnected(null;null)
WifiManager: WiFi.getConnectionInfo(null;null)
LocationManager:requestLocation(null;null)
IoBridge:connect(sdk.open.talk.gepush.com/121.52.236.26:5224;null)
TelephonyManager:getSimOperator(null;null)
PackageManager:getPackagesForUid(10033;null)

The data above, there are function names, function names and some parameters (already except weight); by calling the library, we learned that the actual use of micro-channel permissions:

android.permission.ACCESS_COARSE_LOCATION
android.permission.READ_PHONE_STATE
android.permission.READ_CONTACTS
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_WIFI_STATE
android.permission.BLUETOOTH
android.permission.BROADCAST_SMS
android.permission.CAMERA

By comparison with the micro-channel real application permissions, obtained without the use of multi-application permissions are:

android.permission.INTERNET"
android.permission.MODIFY_AUDIO_SETTINGS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.RECORD_AUDIO
android.permission.READ_SMS
android.permission.VIBRATE
android.permission.WAKE_LOCK
android.permission.WRITE_EXTERNAL_STORAGE

android.permission.WRITE_SETTINGS
com.android.launcher.permission.INSTALL_SHORTCUT
android.permission.BROADCAST_STICKY
com.tencent.mm.location.permission.SEND_VIEW
com.android.launcher.permission.INSTALL_SHORTCUT

Through the data, the micro-channel App actual usage rights is far less than for authority, while at the same time, our monitoring tool can also monitor the actual real state API function calls, function calls can be seen in the main Java function calls, so monitor the performance of more than [1] in a way, it is the operation and maintenance aspects of the library to consider, where monitoring of Java layer function, more than paper[1] the C-level function match.

4. Summary and further work

Through the development of the new module Xposed framework for Android APP to monitor sensitive API interface libraries establish sensitive, to generate a complete log. Get the complete APP sensitive behavior in the course of running this way and recorded. Also continue to expand the library sensitive, making it possible to monitor the data more and more information. API calls of APP can be well monitored in the run-time.

Acknowledgment

This work is supported by NSFC (Grant Nos. 61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

References

[1]. Pang Zhou, RESEARCH AND IMPLEMENTATION OF DYNAMIC TESTING SYSTEM FOR APPLICATIONS ON ANDROID SYSTEM(master's degree, Beijing University of Posts and Telecommunications, China 2015).p.37

[2]. Lutkowski, Skulimowski, Poryzała. An optical data transmission system for blind persons by means of a mobile phone with the Android operating system. International Journal of Adaptive Control and Signal Processing. Lodz, Poland, 2014-12-1, p.1504-1513.

[3]. Gunadi, Hendra,Tiu, Alwen. Efficient runtime monitoring with metric temporal logic: A case study in the android operating system. 19th International Symposium on Formal Methods. Australia,2014-5-16,p. 296-311

[4]. Park, Jiyeon,Kim, Bongjae,Kim, Sung-Ryul, Yoon, Jin Hyun,Cho, Yookun.Performance analysis of security enforcement on Android operating system. 2011 ACM Research in Applied Computation Symposium. Korea,2011-12-5,p.282-286

[5]. Dipert, Brian. Fundamentals of the android operating system. Electronic Products. United States,2012-4-1,p.356-361

[6]. De La Cruz Briyith,Cuellar Ricardo,Rojas Ermin, Molina Valentín,Robles Horderlin Vrangel. Appropriation of Bluetooth technology and the Android operating system for solving problems in biomedical engineering. Case study: ECG signal transmission system to Android mobile system via Bluetooth. 13th Conferencia Iberoamericana en Sistemas, Cibernetica e Informatica, CISCI. Colombia,2014-6-13,p. 136-141

[7]. Moon,Sung Wook,Kim Myeong,Ho Jun Cha,Nam Ju,Kim Dong. Implementation of smartphone environment remote control and monitoring system for Android operating system-based robot platform. 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence,2011-12-26,p.211-214