DOI: 10.3969/j. issn. 1673-5188. 2016. S0. 003
http://www.cnki.net/kcms/detail/34.1294.TN.20160603.1100.002.html, published online June 3, 2016

Design and Implementation of Privacy Impact Assessment for Android Mobile Devices

CHEN Kuan-Lin and YANG Chung-Huang
(Dept. Software Engineering and Management, National Kaohsiung Normal University, Kaohsiung, Taiwan 802, China)

Abstract

There are a lot of personal information stored in our smartphones, for instance, contacts, messages, photos, banking credentials and social network access. Therefore, ensuring personal data safety is a critical research and practical issue. The objective of this paper is to evaluate the influence of personal data security and decrease the privacy risks in the Android system. We apply the concept of privacy impact assessment (PIA) to design a system, which identifies permission requirements of apps, detects the potential activities from the logger and analyses the configuration settings. The system provides a user-friendly interface for users to get in-depth knowledge of the impact of privacy risk, and it could run on Android devices without USB teleport and network connection to avoid other problems. Our research finds that many apps announce numerous unnecessary permissions, and the application installing confirmation dialog does not show all requirement permissions when apps are installed first time.

Keywords

privacy impact assessment; privacy risk; personal information; Android permission; configuration settings

1 Introduction

The sales of smartphones reached 1.2 billion units in 2014 [1]. According to the data from International Data Corporation (IDC), the worldwide smartphone market grew 13% year over year in 2015 Q2 [2]. Particularly, Android dominated the market with an 82.8% share in 2015 Q2 [2], leaving its competitors iOS, Windows mobile OS and Blackberry far behind.

Smartphones have become widespread because of a wide range of connectivity options such as Wi-Fi, GPS, Bluetooth and near field communication (NFC). However, ubiquitous internet connectivity and availability of personal information such as contacts, messages, photos, banking credentials and social network access has attracted the attention of malware developers towards the mobile devices and Android.

Internet security threat reports say that there are too many apps containing malware. Symantec has analysed about 6.3 million apps in 2014, and there are more than one million apps that are classified as malware which included 46 new families of Android malware [3]. In addition, there are approximately 2.3 million suspect apps. Technically, they are not malware, but they display undesirable behaviour, such as bombarding the user with advertising.

In order to avoid malicious apps from the official Google Play, Google introduced a security service named Bouncer [4], which can quietly and automatically scan apps. Any found malicious apps or malware that may be detrimental to users, damage the system or tries to steal privacy information, will be removed from Google Play.

Although Google had done a good job of keeping malware out of the store, the mobile threat report published by Lookout Mobile Security in 2014 showed that Android mobile devices encountered 75% more malware than that in 2013 [5]. Therefore, it is necessary to find more detail information about system and apps to avoid using malicious apps and protect personal or privacy information from being stolen.

In this paper, we propose a privacy impact assessment (PIA) system on Android mobile devices. The proposed framework evaluates the Android security risks based on permission request patterns of applications and configuration settings by users, which aims to minimise privacy risks. We also scan the log messages by a logcat command in Android shell, which helps us know what potential activities are running.

The rest of the paper is organized as follows. The second chapter introduces related work. The third chapter is the literature review about background information. The fourth chapter describes the system architecture and assessment rules. The fifth chapter demonstrates the design and implementation of

This work was supported in part by the Ministry of Science and Technology of Taiwan, China under Grant No. MOST 102-2221-E-017-003-MY3.

PIA. The sixth chapter is the practical test results including system performance. The seventh chapter is the conclusion and future work.

2 Related Works

2.1 Risk Assessment for Permissions

Yang Wang et al. did a quantitative security risk assessment for Android permissions and applications called DroidRisk [6]. Its objective is to improve the efficiency of Android permission system. They used two data sets with 27,274 benign apps from Google Play and 1260 Android malware samples, extracted the name, category, and requested permissions of each app by a crawler, and found the most significant differences between benign apps and malware.

The results demonstrate that malware are likely to request more permissions than benign apps. Malware also request more dangerous permissions that can change the settings or use money-related services than benign apps. Yang Wang et al. also computed the risk levels for all Android permissions. **Table 1** shows the top 20 permissions with highest risk levels [6].

2.2 Android Custom Permissions

Custom permissions are simply permissions declared by third-party applications. They are often used to protect different application components for services and content providers. For example, if Alice wants to share service between her own Android apps, the intent-filter in app A can be used for pending request, and then app B could use the intent to call the correspond service. However, in this case any apps can use app A's service if they know the service's action name in the intent-filter. Therefore, developers define their own custom permissions to protect their application components for data sharing. Any other apps cannot access a component unless the custom permission is requested and granted.

However, there are some security issues with custom permission. It might leak user data such as online browsing history, user's in-app purchases and fake messages inserted via its app [7]. The vulnerability is talking about the custom permission's registered strategy. Custom permissions are always defined as "signature" protection-level in order to check whether the apps is signed with the same key or not, but it may be damaged by a malicious app which defines the same permission name with "normal" protection-level during "Race" [8]. If the malicious app is installed on an Android device before the benign app, the same permission name will be registered using a "first one wins" strategy. This scenario allows all third-party apps to access the component and the sharing data [9].

▼Table 1. Top 20 permissions with highest risk levels

Ranking	Permission Name
1	WRITE_APN_SETTINGS
2	RECEIVE_WAP_PUSH
3	WRITE_SMS
4	INSTALL_PACKAGES
5	READ_SMS
6	RECEIVE_SMS
7	SEND_SMS
8	DELETE_PACKAGES
9	BROADCAST_PACKAGE_REMOVED
10	RECEIVE_MMS
11	CHANGE_WIFI_STATE
12	WRITE_CONTACTS
13	DISABLE_KEYGUARD
14	KILL_BACKGROUND_PROCESS
15	READ_LOGS
16	CALL_PHONE
17	MOUNT_UNMOUNT_FILESYSTEMS
18	PROCESS_OUTGOING_CALLS
19	SET_WALLPAPER_HINTS
20	EXPAND_STATUS_BAR

3 Literature Review

3.1 Privacy Impact Assessment

A PIA is a process for evaluating a proposal in terms of its impact upon privacy, which helps an agency identify the potential effects [10]. PIA enables an organisation to systematically and thoroughly analyse how a particular project or system will affect the privacy of the individuals involved [11]. PIA aims to minimise privacy risks. With it, we can identify and record risks at an early stage via analysing how the purposed uses of personal information and technology will work in practice.

3.2 Android Permission Framework

Android apps can only access their own files by default. In order to interact with the system and other applications, such apps request additional permissions that are granted at the installed time and cannot be changed [8].

Android provides a permission-based security model in the application framework. Developers must declare the permissions required using the <uses-permissions> tag in AndroidManifest.xml [12]. Android permissions are divided into four protection-levels, with different potential risks as discussed [13]:

1) Normal: A lower-risk permission that gives requesting applications access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the

user's explicit approval.

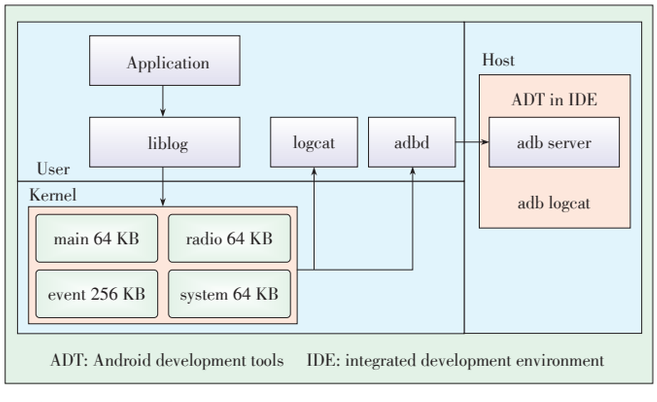
2) Dangerous: A higher-risk permission that gives a requesting application access to private user data or control over the device, which may cause negative impact on the user. Because this type of permission introduces potential risks, the system may not automatically grant it to the requesting application. A user must accept the installation of dangerous permissions at the install time.
3) Signature: A permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
4) SignatureOrSystem: A permission that the system grants only to applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission.

3.3 Android Logger

Logging is an essential component of any Linux operating systems, including embedded ones. Either post-mortem or real-time analysis of a system's logs for errors or warnings is vital to isolate fatal errors [14]. Though Android's kernel still maintains its own Linux-based kernel-logging mechanism, it also uses another logging subsystem, colloquially referred to as the logger. This driver acts as the support for the logcat, dmesg, dumpsys, dumpstate and burgeport command. One program logcat displays a continuously updated list of system and application debug messages [15]. It provides four separate log buffers, depending on the type of information: main, radio, event and system [16]. **Fig. 1** shows the flow of log event and components that assist the logger.

3.4 Android Intent

The primary method for inter-component communication, both within and between applications, is via intents [17]. It can be used with startActivity to launch an Activity, broadcastIntent to send it to any interested BroadcastReceiver components, and startService or bindService to communicate with a background Service [18].



▲Figure 1. Android log system.

3.5 Static and Dynamic Analysis

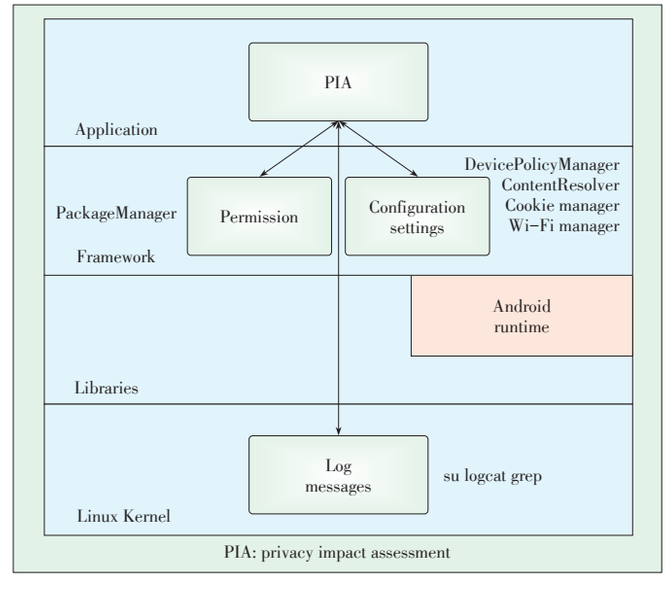
Static analysis works by disassembling and decompiling without actually running application. This does not affect the device. The methods for static analysis are quick, but they fail against the encrypted, polymorphic and code transformed malware [12]. Dynamic analysis methods execute an app in a protected environment, emulating all the resources and features. Therefore, they inspect its interaction for identifying malicious activities.

3.6 CIS Security Benchmark

The Security Benchmark [19] was proposed by the Center for Internet Security (CIS) and is a standard for security configuration of operating systems, including Linux, Windows, iOS and Android. This document defines the situation where operating system configuration settings should be in order for the system to be more secure. System administrators or users can set an Android operating system configuration based on this document in order to heighten the security of Android mobile devices.

4 System Architecture

A PIA system is designed for Android mobile devices, which supports the versions of Android above five. The PIA system can be installed on Android platforms because it is implemented into an Android application package (APK), and it does not need any additional condition such as network connection or USB teleport, which avoids other privacy risks. The system is composes of three parts as shown in **Fig. 2**. It per-



▲Figure 2. PIA system architecture.

forms with static method and dynamic methods at the same time.

4.1 Identifying Permission Requirements

The PIA system invokes the package manager to parse each app's permissions which should be declared in the AndroidManifest.xml file by developers (**Fig. 3**), and then calculates the privacy impact assessment score automatically referring to the protection-level [13] defined by Google and the report of the top 20 permissions with highest risk levels [6]. This main purpose of this method is to detect whether the app is using excessive permissions for dangerous requirements.

In general, the privacy impact assessment score of permissions for different categories are listed in **Table 2**. But, if the permission is in the top 20 with highest risk levels, the privacy impact score must be raised up to 10, because it can cause more huge damage to the system or the user and the app may have horrible potential motivation that declares the permission with highest risk levels. A custom permission defined by developers aims to share data or components with the developers' applications, so its privacy impact score is 8. The custom permission still has some potential privacy risks discussed in section 2 even if it can generally protect other apps to access data

4.2 Analysing Configuration Settings

According to the configuration of the CIS Security Benchmark document [19] (**Table 3**), the PIA system verifies the items and configuration type by ContentResolver, device policy, cookie and Wi-Fi manager, in order to improve and repair configuration settings of the Android mobile device. If the system configuration does not pass the test, its privacy impact score is eight point, in contrast, its privacy impact score is zero point. The main purpose of this method is to suggest users what system configurations they should adjust and then actually enhance the security of the device with international safety standards.

```
//getPermission
try {
    PackageInfo pinfo =
      getPackageManager().
        getPackageInfo
          (pkg, PackageManager.GET_PERMISSIONS);
    group = pinfo.requestedPermissions;
}
```

▲Figure 3. Getting requested permissions by package manager.

▼Table 2. Privacy impact score

Permission type	Privacy impact score	Top 20 with highest risk level
Normal	2	10
Dangerous	8	10
Signature	5	–
SignatureOrSystem	5	–
Custom	8	–

4.3 Detecting Potential Activities

In order to mine characters related to the Android intent, the PIA system uses su and logcat command with grep command together in the Android shell, which aims to track the messages stored in the log buffer. This method can scan potential events or harmful activities when apps installed in the device are running. **Fig. 4** shows the source code of the system that filters a specific keyword in the Android shell.

4.4 PIA Score Formula

The system finally computes the total privacy impact scores referring to Tables 2 and 3, according to the following formulae:

1) The App's PIA score: The sum of all permission scores is divided by the quantity of permission.

▼Table 3. Verifing items

Verifing item	Type	Pass/fail
Android version	System	0/8
Set auto-lock time	System	0/8
Third-party apps	System	0/8
Set screen lock	System	0/8
Encrypt phone	System	0/8
Disable Wi-Fi	Device	0/8
Disable camera	Device	0/8
Browser cookie settings	Browser	0/8

```
//su
Process p = Runtime.getRuntime().exec("su");
DataOutputStream pp =
  new DataOutputStream(p.getOutputStream());

//logcat
pp.writeBytes("logcat -v time -d |grep '"
  + package_name
    + "' |grep 'android.intent.action.'\n");

BufferedReader bufferedReader =
  new BufferedReader
    (new InputStreamReader(p.getInputStream()));
pp.writeBytes("exit\n");
pp.flush();

StringBuilder log = new StringBuilder();
String line;
while ((line=bufferedReader.readLine())!=null)
{
    log.append(line);
    log.append("\n");
}
```

▲Figure 4. The source code of the system that filters a specific keyword in the Android shell.

2) The Android system's final PIA score: The sum of all App scores plus the sum of the configuration scores is divided by the quantity of App plus the quantity of the configuration item.

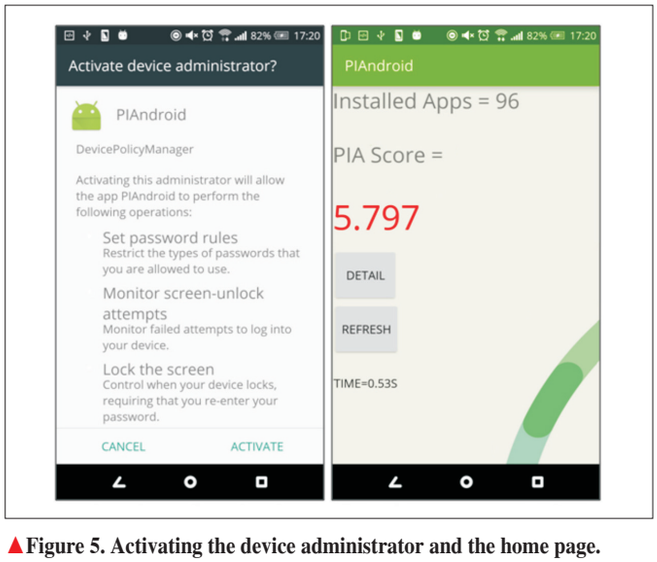
As a result, the user is able to get in-depth knowledge of the impact of privacy risks on each Android mobile device.

5 System Implementation

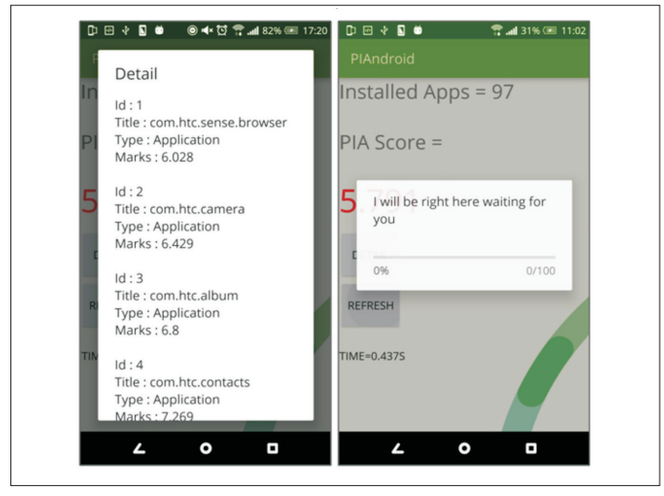
After installing an app on an Android mobile device, the user can begin to use it. When the app is executed, the screen in **Fig. 5** will appear. At this time, the user must activate the device administrator for the system because the device policy manager will be used in the setting page in order to repair system configurations. Then the user can observe the Android system's privacy impact assessment result in the home page. When the user clicks the "detail" button, the page will show the detail information that including each app's score and system configuration score. If the user click "refresh" button, the system will calculator the PIA score again in the background thread by AsyncTask (**Fig. 6**).

The "permission" page displays all apps that have been installed in the mobile device shown (**Fig. 7**). If the user selects any apps in the list, he can enter the interface shown in **Fig. 8** and read all permissions that the apps request, including normal, dangerous, signature, system and custom permissions. The user can also check the introduction about permission's protection-level information by click "introduce" button.

In the "intent" page, the user can click the "monitor" button to start scanning log messages from the log buffer by su, logcat and grep commands. The screen displays the record related to the Android intent or Apps' activities. For example, when the Google drive uploads a photo from the user's device, the "android.intent.action.SEND" log message will appear in the log buffer, and then the system shows "sending data" on the screen. However, if the user does not have super user privilege, which means he has not rooted his device, the screen will display "sorry" message as shown in **Fig. 9**.



▲Figure 5. Activating the device administrator and the home page.



▲Figure 6. Detail information and refreshing.

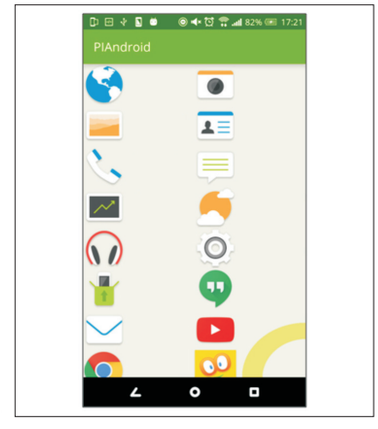
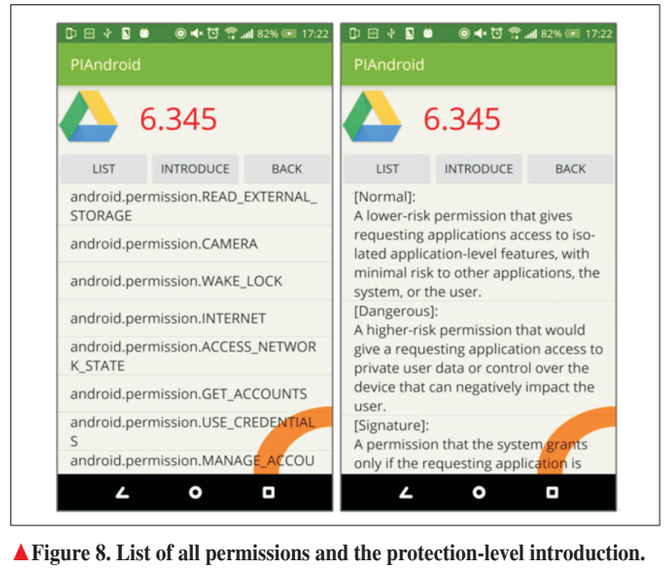
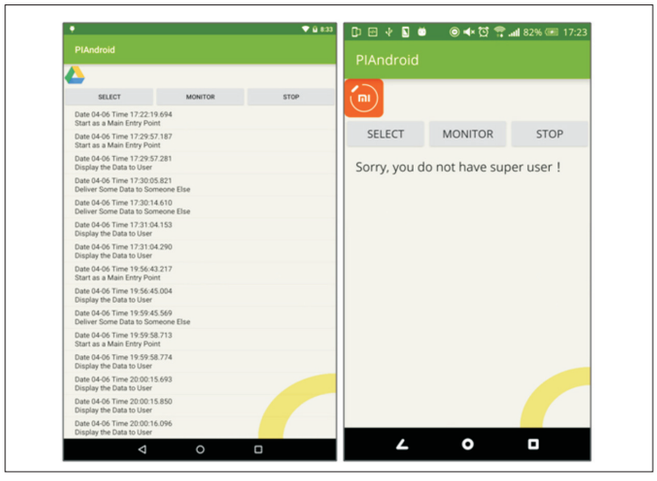


Figure 7. ▶ List of all applications.



▲Figure 8. List of all permissions and the protection-level introduction.



▲Figure 9. Display of log messages.

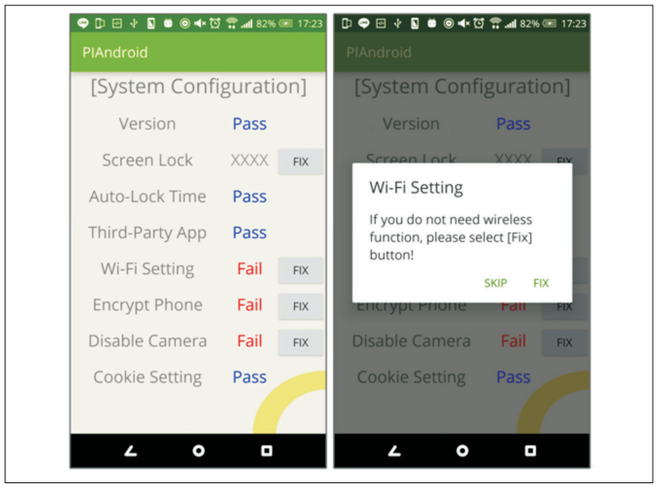
In the "setting" page, the user sees the interface shown in **Fig. 10**. The items shown on the left column are the test items, while the middle column shows the test results. The "FIX" button shown on the right column appears when the device does not pass the test.

After clicking the "FIX" button, the user chooses to fix the setting or skip this item, if the user chooses fixing, repair will be done automatically or the interface jump to the settings page.

6 Practical Tests

In our PIA system, the application installing confirmation dialog did not show all requirement permissions such as "Signature", "SignatureOrSystem" and custom permissions when apps were installed first time,. Many apps announce numerous unnecessary permissions, for example, Mi Fit is used to connect with Mi band to set, track and follow the user's health and fitness data, but this app announces the "camera" permission and "read external storage" permission.

The system performance was assessed, including the execution time, CPU and memory used. We marked the passing time of the system that calculated the PIA score when it ran on the device first time, and used the CPU monitor to record CPU and memory used. The test environment and results are shown in **Table 4** and **Fig. 11**.



▲Figure 10. Detecting and repairing system configurations.

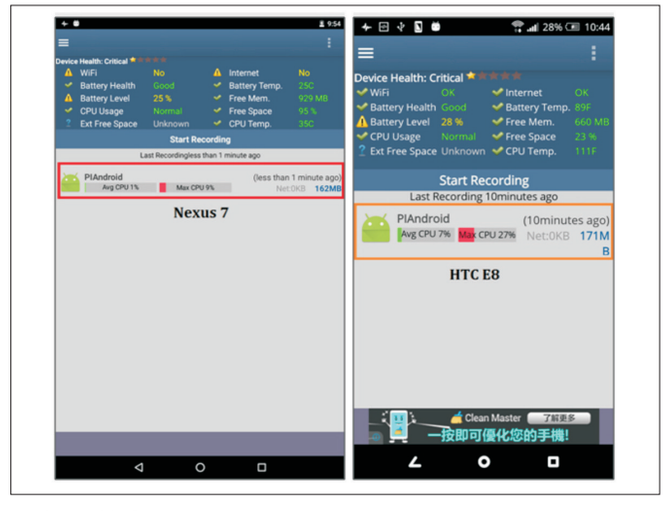
7 Conclusions and Future Work

The privacy risk of personal information is a serious issue to mobile device users when they use various apps for work or leisure in their daily life. In this paper, a privacy impact assessment framework for Android mobile device offers the management of privacy risks, including apps' permission, app execution information and unsuitable system configurations. After scanning is completed and results are obtained, the final report can be used for information security verification, when an app's score on the report is too high, the user can choose not to install the app to make his device more reliable. While a failed item on the setting page needs to be corrected, the user can click "FIX" button to make corrections. The purpose of this system is to reduce, eliminate and minimize privacy risks to the user or vulnerabilities. It improves the security of mobile devices.

The research contribution of this study lies in developing a

▼Table 4. System performance

Device	Nexus 7	HTC E8
Version	6.0.1	6.0.1
Quantity of apps	30	97
Average time (s)	0.755	0.552
Max CPU used	9%	27%
Memory used (MB)	162	171



▲Figure 11. CPU and memory used.

remediation system for Android platforms. Users can run the system on their devices by installing this app, and the system does not need any additional conditions such as internet connection or standard USB teleport, which especially avoids other risks. However, this study plans to break through the restriction of the shell command, because it is too dangerous to use su command for Android system. This may cause more disadvantage; after gaining root privilege, an app has access to the entire system and to low-level hardware. Malicious apps can abuse su to allow themselves irremovable, bypass Android's security measures, and infect smartphones system [20].

References

[1] Gartner. (2015, March 3). *Gartner Says Smartphone Sales Surpassed One Billion Units in 2014* [Online]. Available: http://www.gartner.com/newsroom/id/2996817
[2] International Data Corporation. (2015, August). *Smartphone OS Market Share, 2015 Q2* [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[3] P. Wood, B. Nahorney, K. Chandrasekar, et al., "Internet security threat report," Symantec, Mountain View, USA, Tech. Rep. vol. 20, Apr. 2015.
[4] N. J. Percoco and S. Schulte, "Adventures in bouncerland: failures of automated malware detection within mobile application markets," in *Black Hat*, Las Vegas, USA, Jul. 2012.
[5] Lookout Mobile Security, "2014 mobile threat report," Lookout, San Francisco, USA, Tech. Rep, 2014.
[6] Y. Wang, J. Zheng, C. Sun, and S. Mukkamala, "Quantitative security risk assessment of android permissions and applications," in *27th International Conference on Data and Applications Security and Privacy XXVII*, Newark, USA, Jul. 2013, pp. 226–241. doi: 10.1007/978-3-642-39256-6_15.
[7] TrendLabs Security Intelligence Blog. (2014, March 20). *Android Custom Permissions Leak User Data* [Online]. Available: http://blog.trendmicro.com/trendlabs-security-intelligence/android-custom-permissions-leak-user-data/
[8] P. Walvekar. (2014, April 16). *Race Conditions on Android Custom Permissions* [Online]. Available: https://datatheorem.github.io/2014/04/16/custom-permissions/
[9] N. Elenkov, *Android Security Internals: An In-Depth Guide to Android's Security Architecture*. San Francisco, USA: O'Reilly Media, 2014.
[10] *Privacy Impact Assessment Handbook*, Office of the Privacy Commissioner, Wellington, New Zealand, 2007, pp. 5, 21–27.
[11] Information Commissioner's Office, "Conducting privacy impact assessments code of practice," ICO, Scotland, UK, 2014.
[12] P. Faruki, A. Bharmal, V. Laxmi, et al., "Android security: a survey of issues, malware penetration and defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 998–1022, Dec. 2015. doi: 10.1109/COMST.2014.2386139.
[13] Developer Console. (2016). <*permission*> [Online]. Available: http://developer.android.com/intl/zh-tw/guide/topics/manifest/permission-element.html
[14] K. Yaghmour, *Embedded Android*. San Francisco, USA: O'Reilly Media, 2013.
[15] A. Hoog, *Android Forensics: Investigation, Analysis, and Mobile Security for Google Android*. Waltham, USA: Elsevier, 2011.
[16] J. J. Drake, P. O. Fora, Z. Lanier, et al., *Android Hacker's Handbook*. Indianapolis, USA: John Wiley & Sons, 2014.
[17] W. Klieber, L. Flynn, A. Bhosale, L. Jia, and L. Bauer, "Android taint flow analysis for app sets," in *3rd ACM SIGPLAN International Workshop on the State of the Art in Java Program Analysis*, New York, USA, Jun. 2014, pp. 1–6. doi: 10.1145/2614628.2614633.
[18] Developer Console. (2016). *Intent* [Online]. Available: http://developer.android.com/intl/zh-tw/reference/android/content/Intent.html
[19] Center for Internet Security. (2012, October 1). *CIS Google Android 4 Benchmark v1.0.0*. [Online]. Available: https://benchmarks.cisecurity.org/downloads/show-single/index.cfm?file=android4.100
[20] Y. Shao, X. Luo, and C. Qian, "RootGuard: protecting rooted android phones," *IEEE Computer Society*, vol. 47, no. 6, pp. 32–40, Jun. 2014. doi: 10.1109/MC.2014.163.

Manuscript received: 2016-04-21

Biographies

CHEN Kuan-Lin (kuanlin81625@outlook.com) received the BS degree from National Pingtung University of Education in 2014. He is currently a master student at Department of Software Engineering and Management, National Kaohsiung Normal University. He is actively involved in mobile platform security.

YANG Chung-Huang (chyang@nknu.edu.tw) has a PhD degree in computer engineering from the University of Louisiana at Lafayette in 1990. He is currently professor at the National Kaohsiung Normal University, distinguished professor at the Xi'an University of Posts and Telecommunications, and guest professor at the Xidian University. Previously, he was a software engineer at the RSA Data Security, Inc. (Redwood City, USA) in 1991, a postdoctoral fellow at the NTT Network Information Systems Laboratories (Yokosuka, Japan) in 1991-1993, and a project manager of the Information Security and Cryptology Project at the Telecommunication Laboratories, Chunghwa Telecom (Taiwan) in 1995-1997. For more details, please refer to http://security.nknu.edu.tw/.