A Grey List-Based Privacy Protection for Android

Dapeng Wang, Guang Jin, Jiaming He, Xianliang Jiang and Zhijun Xie
College of Information Science and Engineering, Ningbo University, Ningbo, China
Email: wdpeng24@gmail.com, {jinguang, hejiaming, xiezhijun}@nbu.edu.cn, jiangxianliang@foxmail.com

***Abstract*—The openness of Android OS and the easy development of applications have attracted a large number of enthusiasts to join in. Due to the deficiencies of permission mechanism of Android OS, an Android privacy protection system called Protectedroid is presented in this paper which is a model based on grey list and enables the design of security strategies more easily. Protectedroid adds an enforcing privacy protection module between App and private data. When an application is added to the grey list and intends to access sensitive data and resources, Protectedroid will intercept data access request and return false objects for the grey applications. Experimental results show that: Protectedroid can effectively throttle the privacy leakage and ensure the effectiveness and stability of the entire system.**

***Index Terms*—Android, Protectedroid, privacy, permission mechanism, grey list**

I. Introduction

Mobile APP era has brought us not only the colorful lifestyles, but thrilling privacy security issues. When the smart phones have been indispensable in our daily lives, more and more users have become accustomed to using a mobile phone to store their sensitive data and process their social and commercial information [1][2]. Meanwhile a large amount of leisure time is being pouring into the small icons on the smart phones. All these make phones store and record a lot of users' personal data (e.g., location information, bank accounts, and emails).

For example, 87% of the 90K Android apps in the Google Play [3] that some researcher randomly sampled required permission for Internet access [4]. It seems that the installed third-party applications can access the sensitive data and Internet service easily, and the personal data always contain tremendous latency commercial value. Moreover, open application management mode of Android makes the Internet flood with many malicious third-party applications. So data privacy threat seems much more obvious. Some researchers surveyed 872 Android applications and found that more than a third of them requested privacy-related permissions [5]. TaintDroid [6] showed that, among 30 popular third-party Android applications, there were 68 possible abuse of private information. So Android treats all applications as potentially treat [7]. Here we call all these applications untrusted applications. In the context of Android "All-Or-None" permission granted mode, the installed applications have get the set of permissions they required, and no longer need the authorization from the user while access user's private data protected by these permissions. As for the close relation between user's privacy data and the third-party applications, protecting private data has been the priority of the research in terminal security [8].

II. Background

Developers can visit the phone hardware (e.g., GPS and cameras), personal data and almost all mobile resources via the huge number of API. Android platform employs permission mechanism to protect these sensitive data [9]. If applications require access to privacy and security-related resources, each one must declare upfront what permissions it requires in the Android Manifest file (as show in Fig. 1), and the user will decide whether to grant permissions or not during installation.

```
Code 1 An example of AndroidManifest.xml.
...
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.android.camera"
        android:versionCode="1"
        android:versionName="1">
    <original-package android:name="com.android.camera" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus"
android:required="false" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.SET_WALLPAPER" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_SMS" />
…
</manifest>
```

Figure 1. An example of AndroidManifest.xml.

The reality is that some developers are malice or lack of capacity to develop intact applications. Thus leaving vulnerabilities in the codes or requesting more permissions than they need actually, failing to follow the principle of least privilege. We call this type of applications permission-hungry applications. Some researchers have developed a gadget Stowaway [7], with the help of this gadget, they valued 940 applications downloaded from Google Play and found that nearly one-third of them were permission-hungry application. Permissions directly determine the application's function and terminal's security. But the Android platform native permission mechanism cannot make users feel safe to install colorful third-party applications unless users refuse to install them. This undermines user’s pursuit of the various functions of the applications. Coarse-grained permission model ("All-Or-None") simplifies the installation process, enhances the user experience, but it increases the possibility of leak of privacy, curbing user’s pursuit of application functions. Some researchers have proposed a new mechanism of distribution based on fine-grained mechanism -- users can dynamically adjust the status of granted permission according to their needs. This mechanism could well protect private data, but this requires the user's configuration all by their own, which requires a high level of the user’s professional abilities. It needs users to have knowledge of system’s permission mechanism, how to provide service, and data access mechanism and make a judgment of whether the application is safe or not according to the introduction. In the real world, the smart phone is no longer an expensive object and the general public can afford it. Farmers or workers have a phone run with a smart operating system is no longer something strange in our life, thus the ability we mentioned above cannot be met.

Some researchers performed two usability studies: an Internet survey of 308 Android users, and a laboratory study wherein they interviewed and observed 25 Android users [10]. Study participants displayed low attention and comprehension rates: both the Internet survey and laboratory study found that 17% of participants paid attention to permissions during installation, and only 3% of Internet survey respondents could correctly answer all three permission comprehension questions. This indicates that current Android permission warnings do not help most users make correct security decisions [10]. This greatly reduces the user's experience, and can not work well for some people to protect their privacy. The current problem is when installing an application, users do not want to face complex professional choices, and also do not want to make their private data exposed. But unlike traditional computing environments, in pervasive scenarios mobile users are likely to become the system administrators for their devices; they’ll manage their own security rather than rely on external security management services [11]. Clearly, the current mechanism of the Android platform cannot meet users’ needs for privacy security and functions of applications.

For untrusted applications, researchers have proposed a number of highly efficient and reliable solutions. The permission solution TISSA [12] implements lightweight access constraints in Android framework layer, which means the user can dynamically make appropriate constraint strategy for the permissions list. Ultimately, the user experience and privacy protection has been balanced on the basis of the normal operation of the application, but this requires the user have a certain professional knowledge. In fact, there are only a small number of the users who have such knowledge. A similar finding is MockDroid [13], which implements a virtualization mechanism [14]. The mechanism returns false results to misdirect the applications to ensure the safety. According to the advantages and defects of the mechanisms mentioned above, the paper designed a private data protection scheme based on roles user set for the application, thus meeting user’s demands of the colorful applications and the security of their private data.

III. PROTECTEDROID MODEL DESIGN

There are two privacy controls for Android smart phones that authorize users to run permission-hungry applications while protecting personal data from being exfiltrated:

- Covertly substituting shadow data in place of data that the user wants to keep private[15];
- Blocking network transmissions that contain data the user made available to the application for on-device use only [15].

However, users who have the consciousness of protecting personal data will make a initial judgment on the role of application and there are many criterions for the user’s convenience to make the judgment such as the list of permissions not in conformity with application’s functional performance, the application from an unknown source, the app reviews not good and the application’s description being missing. In order to effectively protect personal data and allow users to experience a variety of APPs, this paper proposed a scheme called Protectedroid which has balanced the user experience, private data security and effectiveness of the system. In this paper, the grey list means the application whose recognition is not limited and reliability and trustworthiness from Android market is not sure and the source is not clear, yet the user demand and intend to try it. Before and after installation user can revise the role of application at his or her will. When the application intends to access personal data, Protectedroid will determines whether real services and data are returned for the application according to its role, thus ensuring the security of personal data.

Firstly, we explain the way how android content provider component manage the structured set of data, and the way other applications access these resources. This section will help us get the following Protectedroid architecture and the details of implementation easily.

Content providers [16] are one of the primary building blocks of Android applications, providing content to applications. They encapsulate data, provide mechanisms for defining data security and provide it to applications through the single content resolver interface.

An application accesses the data from a content provider with a content resolver client object. This object has methods that call identically-named methods in the provider object, an instance of one of the concrete subclasses of content provider. The content resolver methods provide the basic CRUD [17] (create, retrieve, update, and delete) functions of persistent storage. The content resolver object in the client application's process and the content provider object in the application that owns the provider automatically handle inter-process communication. Content provider also acts as an abstraction layer between its repository of data and the external appearance of data as tables.

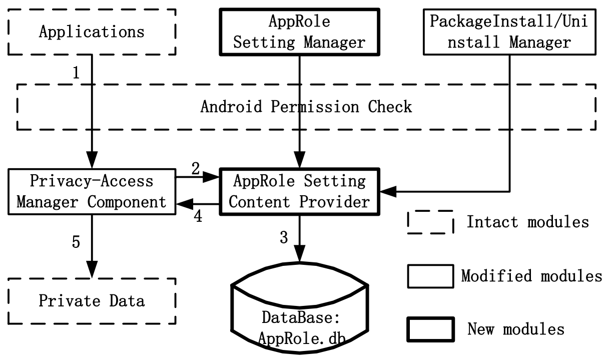


Figure 2. The Protectedroid architecture.

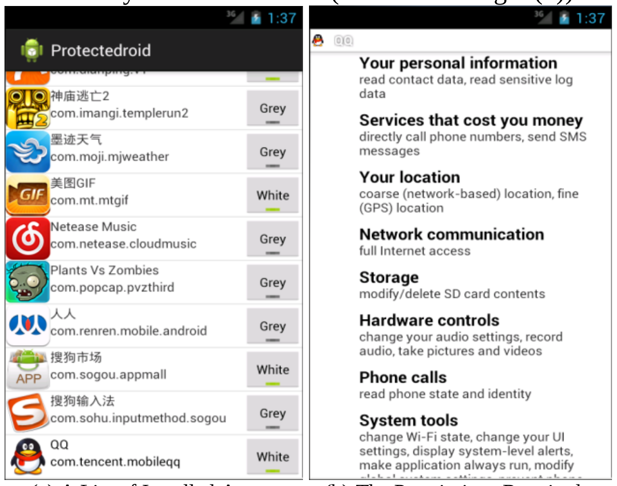
The Protectedroid architecture is shown in Fig. 2. It is mainly made up of App Role Setting Manager, App Role Setting Content Provider and DataBase(AppRole.db). App Role Setting Content Provider is a module for managing the local database AppRole.db, and providing CRUD interfaces for other applications; App Role Setting Manager provides friendly interactive interface between user and smart phone, and allows user to reset the role of applications and calls interfaces provided by App Role Setting Content Provider to store the new role data in the database. To finish the function of Protectedroid, the lightweight modification will be made to the original modules of Privacy-Access Manager Component and Package Install / Uninstall Manager. The kinds of private data in the Android system are various, such as user location, phone status, contacts, SMS Messages and User account information, etc. Privacy-Access Manager Component is just an abstract module, it acts as Manager Components and content Provider Components in Android system to manage a variety of data and resources. Just because of Privacy-Access Manager Component's entity diversity, Protectedroid has a good expandability. Package Install / Uninstall Manager is in charge of the installation process of APP and extract information from the application Package and provides applying and granting permission interface. To realize Protectedroid model, Package Install / Uninstall Manager also need to provide users with the interactive interface for application role setting, allowing user to make their own smart terminal security policy timely.

IV. Implementation

We have implemented the Protecedroid system based on Android version 4.0.1_r1 and successfully run it on emulator. Currently, we choose to protect three types of private data: contacts, call log and messages. Models can be easily expanded to provide additional protection of data privacy. In the following, we explain in more details about the system implementation.

A. App Role Setting Manager

App Role Setting Manager is an independent and complete Android application, and has the same digital signature certificate with App Role Setting Content Provider. It contains only two Activities, one Activity shows the installed third-party applications and the states of the current role with Toggle Button which is also the interface used by user to reset the role of applications (as shown in Fig. 3(a)). The results will be upgraded through App Role Setting Content Provider to the database. The other Activity will present the permissions, which are applied by these applications, in order to allow user conveniently to make decision (as shown in Fig. 3(b)).



1:37

Protectedroid

神庙逃亡2 com.imangi.templerun2	Grey
墨迹天气 com.moji.mjweather	Grey
美图GIF com.mt.mtgif	White
Netease Music com.netease.cloudmusic	Grey
Plants Vs Zombies com.popcap.pvzthird	Grey
人人 com.renren.mobile.android	Grey
搜狗市场 com.sogou.appmall	White
搜狗输入法 com.sohu.inputmethod.sogou	Grey
QQ com.tencent.mobileqq	White

1:37

QQ

Your personal information
read contact data, read sensitive log data

Services that cost you money
directly call phone numbers, send SMS messages

Your location
coarse (network-based) location, fine (GPS) location

Network communication
full Internet access

Storage
modify/delete SD card contents

Hardware controls
change your audio settings, record audio, take pictures and videos

Phone calls
read phone state and identity

System tools
change Wi-Fi state, change your UI settings, display system-level alerts, make application always run, modify

(a) A List of Installed Apps (b) The Permissions Required

Figure 3. APP Role Setting Manager.

B. Privacy-Access Manager Component

This Manager Component shows itself in the native Android system as contacts provider, calendar provider, telephony manager, location manager, etc. These modules are in the Android system application framework layer, and work hard at managing access to the data and resources. Protectedroid needs to lightly modify these native modules, adding application role deciding to strengthen the privacy access control. Here takes contacts content provider as an example to illustrate the original Android private data access mechanism and Protectedroid mechanism operating process when Protectedroid is realized, as shown in Fig. 4.

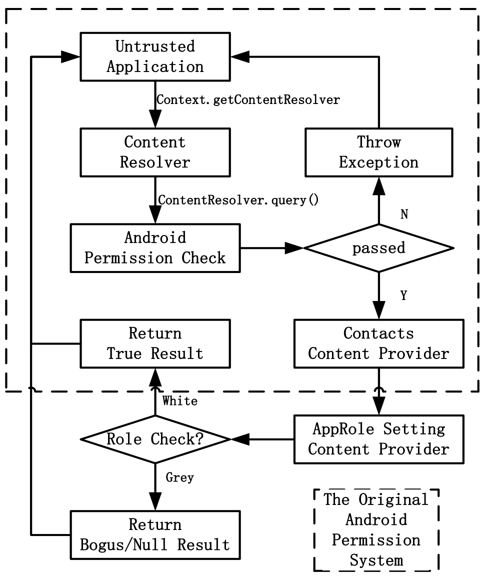


Figure 4. Example: Protecting Contacts.

When an installed third-party application intends to visit the user contacts, method getContentResolver() is called in the Activity firstly. This method then calls ApplicationContext.getContentResolver() and return mContentResolver object. Secondly, Content Resolver class provides the same four CRUD methods defined in Content Provider class. mContentResolver object is responsible for processing the data access requests. Android uses a lightweight Binder-based inter-process communication model (ICC) as the main form of communication between the components [18]. Content Resolver gets Content Provider component through matching with Uri and this process must be through the middle permission checking process. After the checking, a Content Provider proxy object can be obtained. From Caller's perspective, proxy object is the same with the local object. The former can call the same methods and variables of the latter. Now the access request can be executed by the Contacts Provider proxy object. In the original Android system Contacts Provider will immediately process the request and return the results, the dotted line in Fig. 4 encloses those components that also exist in the original Android. After adding Protectedroid mechanism (outside the dotted area), the mechanism requires the checking of application role when Contacts Provider receives the data access request. First, the forcing privacy protection module gets Calling UID through Android inter-process communication mechanism and caller's package name can be got by method PackageManager.getPackagesForUid (Calling UID). According to the package name, role queries issued to App Role Setting Content Provider. If the requestor is set in the white list, the request will be processed normally and results are returned. If not, the requestor is set in the grey list, and then null or false objects will be returned. So Protectedroid can eliminate the leak of the private data completely.

C. *App Role Setting Content Provider*

Content providers are the standard interface that connects data in one process with code running in another process. A content provider presents data to external applications as one or more tables that are similar to the tables found in a relational database. A row represents an instance of some type of data the provider collects, and each column in the row represents an individual piece of data collected for instance.

When a request is made via a Content Resolver, the system inspects the authority of the given URI and passes the request to the content provider registered with the authority. The content provider can interpret the rest of the URI however it wants. The UriMatcher class is helpful for parsing URIs. Visit URI shaped like content:// com.android.provider.authprovider/authorities. Content URIs include the symbolic name of the entire provider (com.android.provider.authprovider) and a name that points to a table (authorities). When you call a client method to access a table in a provider, the content URI for the table is one of the arguments. App Role Setting Content Provider inherited content provider class and implemented the parent query (), update (), insert (), delete () methods to provide other applications operating local database interfaces. AppRole.db maintains a local database for users to install and update third-party application role. Privacy management module (such as ContactsProvider, CalendarProvider, etc.) in the detection of the data access request, it will first interact with App Role Setting Content Provider, and check the requester's role. The role result determines the authenticity of the private data returned to the requestor. The type of private data determines the returned result. Considering the user experience and system overhead, the complex algorithm is not necessary. Contacts, for example, null object returned to requestor seems reasonable and safe enough to personal data.

D. *Package Install/Uninstall Manager*

Package Install / Uninstall Manager are the module which process APK installation and uninstallation. An Android application is stored in an APK file (i.e., a file named by {Application Name}.apk). We must install the APK on our Android phone in order to run it. There are three different ways to install APK files:

- The easiest way is simply to download the application from the Android Market or from a web site directly to the SD card mounted in the phone, then click on the APK file to install it;
- The second way is to download the APK file to your computer, mount our phone’s SD card in the computer (or connect the phone with the SD inserted in it to the computer via USB cable) and copy the APK file to the SD card, then insert the SD card in the phone and install the APK file from the SD card using an Application Installer or File Manager that you download from the Google Play;
- The third way is to install the Android SDK on the computer, then connect the phone via USB cable to the computer and use the Android Debug Bridge (contained in the SDK) to install the APK on the phone[19].

And in some cases, there is no installation Activities during installation. No installation Activities generate that there is no UI for user to set app’s role timely, and in this case, Protectedriod will always storage applications’ role as grey list by default, and provides dynamic role settings (App Role Setting Manager) to compensate for this defect.

One dialog will be provided for users to set the applications’ role in the first time when the installation process has installation Activities (as shown in Fig. 5(b)).

In addition to systematic applications cannot be uninstalled, others are handled by Package Install/ Uninstall Manager, and deleting process involves removing the app role data from the database (AppRole. db).

V. EVALUATION

The application’s role depends entirely on the user's judgment. The following paper will show the results of the user experience, effectiveness and performance of Protectedroid from the user's perspective.

A. *User experience*

In fact, no one wish that the operating system running on the smart phone has a poor user experience, and this is one reason why the phone is called portable equipment or smart phone.

There is no doubt that Android has an excellent user experience, and we all know that too many dialogs and decisions will make users become boring. In this paper, Protectedroid have studied to provide a simple and friendly user interface while provide a privacy protection mechanism. As shown in Fig. 5(b), the only dialog is appeared in application's installation process, and the only thing user must to do is just a simple decision. And initializing null object demands much less system expense and fewer system resources. So, Protectedroid guarantees the users experience fully.

B. *Effectiveness*

1) Robustness

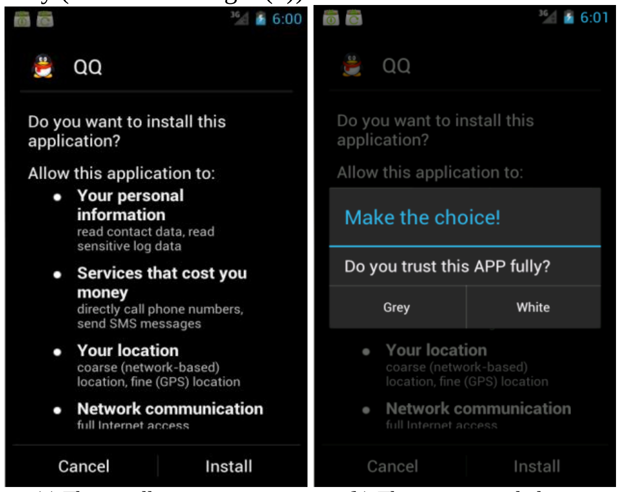
We can see that App Role Setting Content Provider is small but the most primary building blocks of Protectedroid, providing the real applications' role to Privacy-Access Manager Component. But Protectedroid will be killed when the overall system is running low on memory, how to assure App Role Setting Content Provider would be killed by the systems or some other system manager applications?

Android has enhanced the robustness and error handling of content provider. It has a set of perfect mechanism to ensure that there is one content provider instance alive on the system. while Privacy-Access Manager Component want to query one application's role data and the App Role Setting Content Provider has been killed by system, Android will create one new content provider (App Role Setting Content Provider) instance and put content provider entity into memory. This mechanism can indemnify content provider instance against the unpredictable destroy, and insure the robustness of Protectedroid.

2) Viability

As a normal user, he or she has an initial judgment on the role of the application before it is installed while the judgment may not be necessarily correct. During the installation, the Protectedroid will offer recommendations to user to set all unknown applications into the grey list. After the applications are used for a short period, their roles can be corrected in App Role Setting Manager combined with the further understanding of the applications, thus reducing the possibility of the wrong judgment. But, if a credible application is set to grey list, will this application still work well?

In this paper, the most popular chatting tool QQ is used to do the experiment. QQ is a Communication app developed by Tencent Technology (Shenzhen) Company Ltd[20]. It was released on Aug 15, 2013 for Android 2.1 and up. The data from Tencent Technology Company showed that QQ has 800 million monthly active users, of which 550 million use mobile phone for access [21], there are over 10000000 users has already installed QQ APK on their android device. Meanwhile Tencent is currently one of the largest Internet service providers, and also one of the largest Internet service subscribers. QQ has a very high recognition and credibility in the APP market. First, QQ applies to the system for permissions: Your personal information, Services that cost your money, your location and Network communication, etc. (as shown in Fig. 5(a)). These authorities directly control the user's private data. If the user does not understand QQ's role during the installation, the suggestion is to be set in Grey (as shown in Fig. 5(b)).


6:00

QQ

Do you want to install this application?

Allow this application to:

- **Your personal information**
  read contact data, read sensitive log data
- **Services that cost you money**
  directly call phone numbers, send SMS messages
- **Your location**
  coarse (network-based) location, fine (GPS) location
- **Network communication**
  full Internet access

Cancel | Install

6:01

QQ

Do you want to install this application?

Allow this application to:

Make the choice!

Do you trust this APP fully?

Grey | White

- Your location
  coarse (network-based) location, fine (GPS) location
- Network communication
  full Internet access

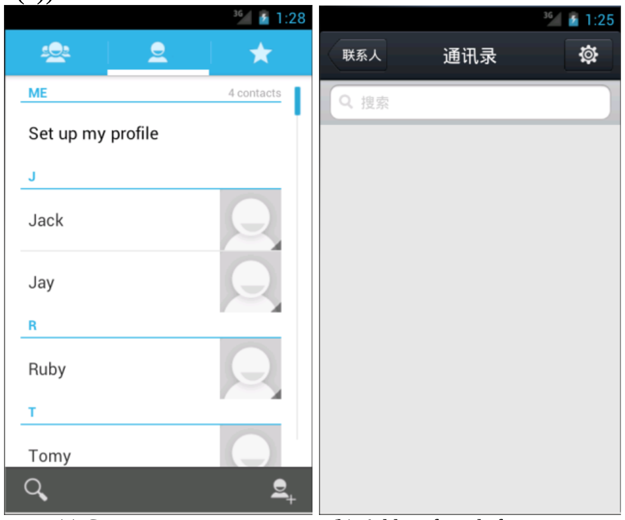
Cancel | Install

(a) The installing activity (b) The prompting dialog

Figure 5. The installation process.

QQ gets the permission "Your personal information" when we chose and clicked the "install", and it seems that the installed QQ could read contact and other sensitive log data optionally. Nevertheless, under the protection of forcing protecting module in the Protectedroid, QQ cannot provide user the interface add new friends from contacts normally, the contacts data access request was handled by the enforcing privacy protection module and null object would returned as the access result while QQ was set as grey list(as shown in Fig. 6(a)(b)).

In addition to this effect, all other functions could work well, we can chat with others, read news and track friends' trends activity successfully (as shown in Fig. 6(c)).


1:28

ME 4 contacts

Set up my profile

J

Jack

Jay

R

Ruby

T

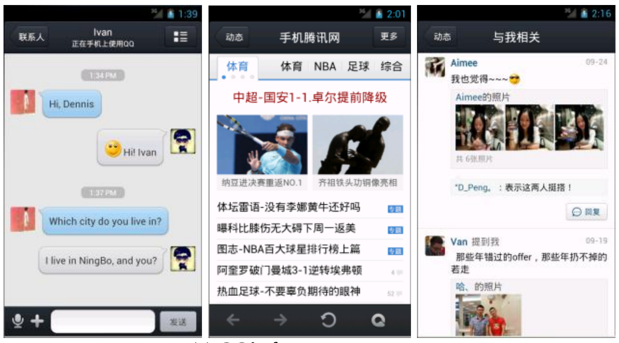
Tomy

1:25

联系人 通讯录

搜索

(a) Contacts activity (b) Adding friends from contacts



Hi, Dennis

Hi! Ivan

Which city do you live in?

I live in NingBo, and you?

中超-国安1-1.卓尔提前降级

纳豆进决赛重返NO.1

齐祖铁头功铜像亮相

体坛雷语-没有李娜黄牛还好吗

曝科比膝伤无大碍下周一返美

图志-NBA百大球星排行榜上篇

阿奎罗破门曼城3-1逆转埃弗顿

热血足球-不要辜负期待的眼神

Aimee 09-24
我也觉得~~~

Aimee的照片

共 6张照片

*D_Peng. ：表示这两人挺搭！

Van 提到我 09-19
那些年错过的offer，那些年扔不掉的若走

哈、的照片

(c) QQ’s function activities

Figure 6. Protecting contacts.

C. Performance

Protectedroid makes some lightweight modify to some modules (ContactsProvider, TelephoneyProvider, etc.) in the Android system application framework layer, and recompiles the system image file. To measure the performance overhead, we use CaffeineMark 3.0 to measure the performance of Android app on the original system and the modified system.

The CaffeineMark is a series of tests that measure the speed of Java programs running in various hardware and software configurations. CaffeineMark scores roughly correlate with the number of Java instructions executed per second, and do not depend significantly on the amount of memory in the system or on the speed of a computers disk drives or internet connection.

The CaffeineMark 3.0 on Android uses seven tests to measure various aspects of Java virtual machine (VM) performance. Each test runs for approximately the same length of time. The score for each test is proportional to the number of times the test was executed divided by the time taken to execute the test. The following is a brief description of what each test does:

- Sieve: The classic sieve of eratosthenes finds prime numbers;
- Loop: The loop test uses sorting and sequence generation as to measure compiler optimization of loops;
- Logic: Tests the speed with which the virtual machine executes decision-making instructions;
- String: Tests string concatenation and search;
- Float: Simulates a 3D rotation of objects around a point;
- Method: The Method test executes recursive function calls to see how well the VM handles method calls;
- Overall: The overall CaffeineMark score is the geometric mean of the individual scores [22].

From the Fig. 6, the geometric average score(Overall) tell us that there is no observable performance overhead, which is expected as our system only do the lightweight modifications to the framework and the original system was not affected.

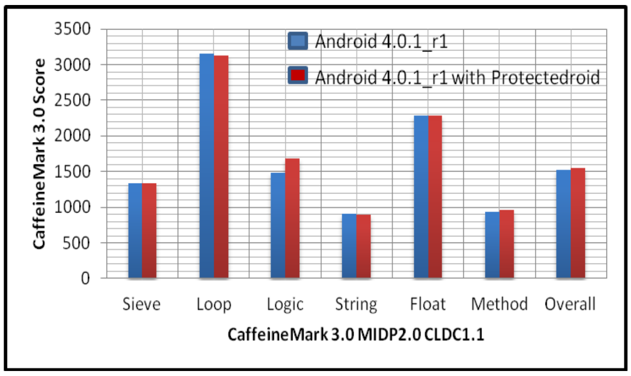


Figure 7. The result of CaffeineMark 3.0.

Overall, Protectedroid is able to balance the user experience and protection of private data on the basis of guaranteeing the stability and effectiveness of the system, meeting the necessity and the goal of the design we set before.

VI. Conclusion

This paper implements a user-friendly strategy based on grey list to protect personal data from applications whose credibility degree is not high but its function is required by users. This has made up for the users’ lack of competence while making fine-grained security policy. Different users’ competence determines the different judgment criteria. But Protectedroid is flexible to accept users with different competence. In order to improve the ease of developing security policies, there are just two roles of applications: grey and white. Privacy-Access Manager Component will return null or false objects to requester while the grey list applications request privacy and security-related data. Therefore, the underlying handling process is coarse-grained and not intelligent enough. The Protectedroid can protect personal data well but is likely to reduce some applications’ special functions. For example, when a navigation application is set as grey list by the user, it cannot return an accurate route. Based on this circumstance, the next goal is to design an intelligent service who can return null or other better false objects intelligently depending the context of the application [23], and to protect the privacy or security of personal data on the basis of re-enhance the user experience, and to make various applications enrich the lives of people.

Acknowledgment

This research was supported in part by Zhejiang Provincial Technology Innovation Team (2010R50009), Natural Science Foundation of Zhejiang Province (LY12F02013), Ningbo Natural Science Foundation (2012A610014), Ningbo Municipal Technology Innovation Team (2011B81002), and The Graduate Teaching Innovation on Ningbo University (2011004).

References

[1] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy, Privilege Escalation Attacks on Android, In ISC, 2011.

[2] B. Gu, X. Li, G. Li, A. C. Champion, Z. Chen, F. Qin, and D. Xuan, D2Taint: Differentiated and Dynamic Information Flow Tracking on Smartphones for Numerous Data Sources, Technical Report, 2012.
[3] Google Play, https://play.google.com/store/apps/, 2013.
[4] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, Networkprofiler: Towards Automatic Fingerprinting of Android Apps, In IEEE International Conference on Computer Communications (INFOCOM), 2013.
[5] A. P. Felt, H. J. Wang, A. Moshchuck, S. Hanna, and E. Chin, Permission Re-Delegation: Attacks and Defenses, In Usenix Security Symposium, 2011.
[6] W. Enck, P. Gilbert, C. Byung-gon, L. P. Cox, J. Jung,P. McDaniel, et al, TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, In Usenix Symposium on Operating Systems Design and Implementation (OSDI), 2010.
[7] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, Android Permissions Demystified, In ACM Conference on Computer and Communications Security, 2011.
[8] M. L. Polla, F. Martinelli, and D. Sgandurra, A Survey on Security for Mobile Devices, IEEE Communications Surveys & Tutorials, 2012, 15(1):446-471.
[9] A. P. Felt, S. Egelman, M. Finifter, D. Akhawe, and D. Wagner, How to Ask for Permission, In Usenix conference on Hot Topics in Security, 2012.
[10] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, Android Permissions: User Attention, Comprehension, and Behavior, In Symposium on Usable Privacy and Security (SOUPS), 2012.
[11] A. Toninelli, R. Montanari, O. Lassila, and D. Khushraj, What's on Users' Minds? Toward a Usable Smart Phone Security Model, IEEE Pervasive Computing, 2009, 8(2): 32–39.
[12] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, Taming Information-Stealing Smartphone Applications (on Android), In International Conference on Trust and Trustworthy Computing (TRUST), 2011.
[13] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, MockDroid: Trading Privacy for Application Functionality on Smartphones, In International Workshop on Mobile Computing System and Applications, 2011.
[14] T. R. Qiu, Y. Y. Li, and X. M. Bai, The Difference Degree of Condition Attributes and Its Application in the Reduction of Attributes, Journal of Computers, 2012, 7(5):1087-1091.
[15] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, These aren't the Droids You're Looking For: Retrofitting Android to Protect Data from Imperious Applications, In ACM conference on Computer and communications security, 2011.
[16] Android Developers, http://developer.android.com/guide/topics/providers/content-providers.html, 2013.
[17] Y. B. Guo, L. K. Zhang, F. R. Lin, and X. M. Li, A Solution for Privacy-Preserving Data Manipulation and Query on NoSQL Database, Journal of Computers, 2013, 8(6): 1427-1432.
[18] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, and A. R. Sadeghi, XManDroid: A New Android Evolution to Mitigate Privilege Escalation Attacks, Technische Universität Darmstadt, Technical Report TR-2011-04, 2011.
[19] How to Install APKs, Research supported by the US Department of Education, Grant Number H327A100014, http://vbraille.cs.washington.edu/doc/.
[20] Tencent, http://www.tencent.com/zh-cn/index.shtml, 2013.
[21] 199IT, http://www.199it.com/archives/109851.html/, 2013.
[22] http://www.benchmarkhq.ru/cm30/, 2013.
[23] H. J. Yang, C. Y. Wang, and N. Du, High Level Synthesis Using Learning Automata Genetic Algorithm, Journal of Computers, 2012, 7(10):2534-2541.

Dapeng Wang is currently a MS candidate in the Faculty of Information Science and Engineering of Ningbo University. His research interests mainly focus on privacy security on smartphone(Android). Previously, he received his BS degree from Changshu Institute of Technology in 2011.

Guang Jin received his MS and PhD degree in Computer Science from Zhejiang University in China. Since 2001, He has been with the College of Information Science and Engineering of Ningbo University in China. He is currently a full professor and his research interests focus on network security and wireless networking. He has published over 30 papers in conferences and journals including IEEE Communications Letters. He also published two books on networking. Especially his new book, the Wireless Networking Technique, has been used as the textbook in over 30 universities or colleges in China.

Jiaming He received his PhD degree from Zhejiang University in China. He has been with the College of Information Science and Engineering of Ningbo University in China. He is currently a full professor and his research interests focus on communications technology and mobile Internet technology.

Xianliang Jiang is a PhD candidate in the College of Computer Science and Technology at Zhejiang University. His research interests are in ad hoc routing protocol, mobile computing and networks security. Previously, he received his BS degree from University of Science and Technology of China in 2009 and MS degree from Ningbo University in 2012.

Zhijun Xie received his PhD degree from Renmin University of China. He has been with the College of Information Science and Engineering of Ningbo University in China. He is currently an associate professor and his research interests focus on wireless communications and networking.