International Journal of Engineering Innovation & Research
Volume 4, Issue 6, ISSN: 2277 – 5668

Personal Information Security on Android

Sunitha Sooda
Computer Science and Engineering,
HKBK College of Engineering, Bangalore
Email:sunithajagadeesht@gmail.com

Archana Hombalimath
Computer Science and Engineering,
HKBK College of Engineering, Bangalore
Email: vharchana@yahoo.co.in

Abstract **– Mobile phones which use Android or other smartphone operating systems, contains richer applications than normal phones. These mobile phones enable users to leave more things to phones in order to deal with deal with. Ain this way the personal information becomes more and more on mobile phones. Backup is an important technique to keep the information safe. Users could only synchronize their contacts, messages and so on to the Internet and easily synchronous back all the information whenever it is necessary. Some application settings such as activations, game scores and so on are also important to users. The lost of personal information caused by mobile phone updates, replacement and lost becomes not uncommon in recent days. It is very important to propose a better application settings integration and management scheme. This paper gives a solution on Android platform after a study to Android security mechanisms and system components.**

Keywords **– Android Platform, Personal Information Security, Synchronization, Personal Information, Security.**

I. INTRODUCTION

Developed by the OHA (Open Hand-set Alliance), a consortium of 80 hardware, software, and telecom companies devoted to advancing open standards for mobile devices led by Google, Android is a widely anticipated Linux-based open source operating system for mobile devices which provides a base operating system, an application middleware layer, a Java software development kit (SDK), and a collection of system applications [1]. The Android OS can be used as an operating system for cell phones, net books and tablets, TV and other devices. The operating system, programmed with Java and enhanced with its own security mechanisms tuned for a mobile environment, quickly became popular amongst the developer community for its open source nature and adoption by telecommunications providers world-wide.

Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 150,000 apps available for Android. Android Market is the online app store run by Google, though apps can also be downloaded from third-party sites [2]. Developers write primarily in the Java language, controlling the device via Google-developed Java libraries. The Android framework includes an operating system, middleware and key applications. Mobile devices incorporate integrated access services which increases the probability of inflicted by various types of malware.

This paper analyse the problem on personal information security on mobile platform and work out a solution to the lost and recovery of personal information after an explanation of security mechanism on Android platform.

II. THREATS TO PERSONAL INFORMATION ON ANDROID

Designed for open, programmable, networked devices, Android is vulnerable to typical smart phone attacks. Such attacks can make the phone partially or fully unusable, cause unwanted SMS/MMS (short message service/multimedia messaging service) billing, expose private information, or even infect every name in a user's contacts. Attack vectors include cellular networks, Bluetooth, the Internet (via Wi-Fi, General Packet Radio Service/Enhanced Data Rates for Global Evolution, or 3G network access), USB, and other peripherals. Smart phones are likely to become a fertile ground for various types of malware. Another major factor attracting hackers is that smart phones are often carried for business purposes and are likely to have sensitive information. They also provide remote access to a company's most sensitive data, which can lead to data leakage if their phones are hacked into. The challenge in ensuring personal information security on smart phones is becoming similar to that confronting the PC [3].

Google released most of the Android code under the Apache License, a free software and open source license, and keeps the reviewed issues list publicly open for anyone to see and comment. The risks to Android are signicant, mainly because it is an open source software stack, whose source code can be more easily accessed, manipulated, and exploited by hackers, operated in a heterogenic mobile environment [4].

Nevertheless, Android is built on Linux kernel which is known for safety and has its own security mechanisms.

However, when a new malware emerges, by the time the countermeasure comes out, users would suffer substantial losses. But if users sync their personal information online, the losses would be reduced. Though synchronization, which is "partially" supported by Android, is not able to prevent private information exposing, it has always been one of the most important means to keep the information safe in recent days.

III. SCHEME PROPOSITION

Android could sync users' contacts, emails, system settings, etc. to Google's server, so users will be never worry about losing contacts or emails. If you want to sync other non-Google developed application settings to the Internet, you can only expect the developers to add the

synchronizing feature to each application. This is why we say synchronization is "partially" supported by Android. It is impossible to develop just one application to sync all application settings because of Android's permissions mechanism, which will be explained below. Besides, it not only burdens the developers, but also is not convenient to the users who should start every application and sync. However, with the help of Content Provider, which will be introduced below, and changing train of our thought, the problem above could be easily solved.

A. Content Provider

Android allows you to expose data sources (or data providers) through a REST (Representational State Transfer) -like abstraction called a content provider which stores and retrieves data and make it accessible to all applications. A SQLite database on an Android device is an example of a data source that you can encapsulate into a content provider. Content Providers are the only way to share data across applications; there's no common storage area that all Android packages can access [5].

Strictly speaking, though, the content providers' responsibilities comprise more of an encapsulation mechanism than a data-access mechanism. So, content-provider abstraction is required only if you want to share data externally or between applications.

Android ships with a number of content providers for common data types (audio, video, images, personal contact information, and so on). You can see some of them listed in the android provider package. You can query these providers for the data they contain (although, for some, you must acquire the proper permission to read the data) [6].

However, Android does not provide a content provider which specially stores data such as application settings. Though it could be developed easily, if it is not supported by Google, it will never be widely used.

B. Android Security Mechanisms [7]

The foundation of the Android software stack is the Linux kernel, which is used for its device drivers, memory management, process management, and networking. Usually, developers will not be directly programming to this layer. The next level up contains the Android native libraries, which are written in C/C++. Incorporating these

libraries, used by various system components in the upper layers, in Android applications is achieved via Java native interfaces. The next level is the Android runtime, comprising the Dalvik virtual machine and the core libraries. Dalvik runs .dex (Dalvik-executable) files that are designed to be more compact and memory- efficient than Java class files. The core libraries are written in Java and provide a substantial subset of the Java 5 SE packages (e.g., standard collections, I/O, networking, utilities) as well as some Android-specific libraries. The application framework layer, written fully in Java, includes Google-provided tools as well as proprietary extensions or services. The topmost application layer provides applications such as a phone, Web browser, email client, and more. Each application in Android is packaged in an .apk (Android package) archive for installation. The .apk is similar to a standard Java jar file in that it holds all code and non-code resources (e.g., images, manifest) for the application.

As every other currently existing operating system, Android offers several security mechanisms to render it a fairly secure system. In following, we will take a deeper look into these mechanisms which are incorporated into the Android framework.

1) User IDs

At install time, Android gives each package a distinct Linux user ID. Thus, two different packages' code can't run in the same process, since they need to run as different Linux users.. In a way, this creates a sandbox and prevents applications from interfering with one another. On a different device, the same package may have a different UID; what matters is that each package has a distinct UID on a given device.

Developers can use the shared User Id attribute in the Android Manifest . xml's manifest tag of each package to have them assigned the same user ID.

2) File Access

Files in Android are subject to the Linux permissions mechanism. Any data stored by an application will be assigned that application's user ID, and not normally accessible to other packages.

3) Application Permissions

A central design point of the Android security architecture is that no application, by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or e-mails), reading or writing another application's files, performing network access, keeping the device awake, etc. To make use of protected features of the device, developers must include in their AndroidManifest.xml one or more <uses-permission> tags declaring the permissions that your application needs.

At install time, permissions requested by the application are granted to it by the package installer, based on checks against the signatures of the applications declaring those permissions and interaction with the user. No checks with the user done while an application is running.

C. ASIMS

ASIMS (Application Settings Integration and Management Scheme) is the solution this paper presents to enhance the security of personal information on Android. Android operating system consists of several layers: the Linux kernel, the Android runtime, the C/C++ libraries, the application framework and the applications. We could add an application, which stores other applications' settings and syncs them to the Internet, to the applications layer. Concrete realization method of ASIMS is as follows:

1) Defining column names for the SQLite database

It is arguably the most widely deployed SQL database engine in the world. Besides Android, SQLite can be found in the Apple iPhone, Symbian phones, Mozilla Firefox, Skype, PHP, Adobe AIR, MacOS X, Solaris, and many other places.

The column names are defined as blow: APPLICATION_ID This column allows the application to distinguish different applications settings and enable the application to prevent malwares from modifying, or deleting other applications' settings.

APPLICATION_NAME This column will be displayed in the user interface to offer users convenience to recognize each application.

CREATED_DATE This column records the first time an application's setting being stored in the database.

MODIFIED_DATE This column stores the last time this record being modified.

SETTINGS This column stores the settings of an application. It must be limited to a proper size to prevent some applications storing too large data into the database, which would slow down the whole application.

2) Implementing the Content Provider

Implementing the application involves extending the ContentProvider class and overriding onCreate() to create the database and then implement the query, insert, update, delete, and getType methods. A query method requires the set of columns it needs to return. The insert method in a content provider is responsible for inserting a record into the underlying database and then returning a URI that points to the newly created record. The update method is responsible for updating a record and then returns the number of rows updated in the process. The delete method is responsible for deleting a record and then returns the number of rows deleted in the process.

3) Enhancing the un installation system

When a user delete an application, the settings of this application will remain in the database. As a result, the database would become bigger and bigger, slower and slower. To avoid this, we can adding a new feature which will prompt users to delete the record associated with the application which is being uninstalled to the un installation system.

4) Adding the synchronization feature

We can use Google account to authenticate users. When the application is synchronizing the data to the Internet, the application sends a synchronization request to the storage server first. Then the storage server transfers the user's identification to Google server to verify whether the user is legitimate. After this, the storage server return the verification result to the application. According to the result, the application choose whether or not to synchronize. Figure 1 shows how this works.

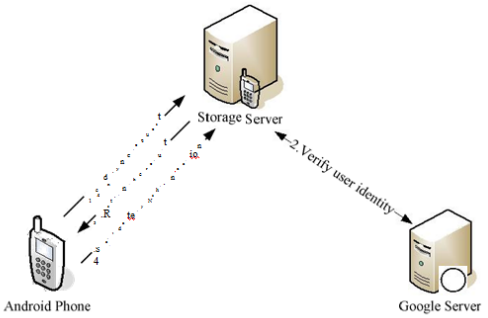

Fig.1. The synchronization operation.

5) Developing the UI (user interface)

Users could select which application to sync. The Sequence Diagram is as below (Figure 2):

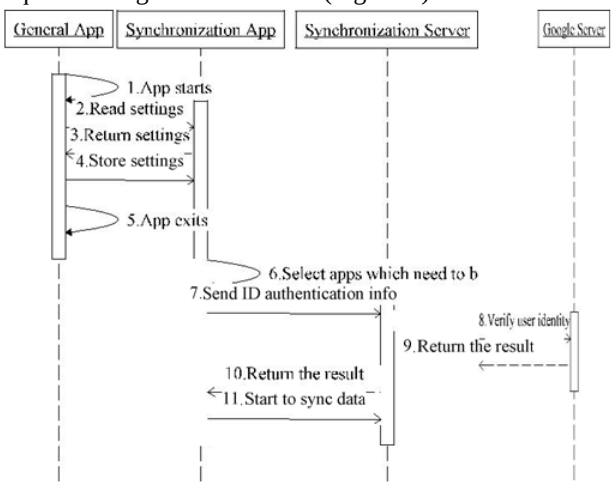

Fig.2. The sequence diagram of ASIMS.

IV. COMPARISON AN DISCUSSION

Android's build-in synchronization system consist of the back up service and offcial apps' (like gmail, google calendar and etc.) synchronization feature. Android's data backup (ADB) has become build-in application data and settings backup solution from the release of Android 2.2. It copys users' persistent application data to remote "cloud" storage,in order to provide a restore point for the application data and settings. During a backup operation, Android's Backup Manager queries the application for backup data, then hands it to a backup transport, which then delivers the data to the cloud storage. During a restore operation, the Backup Manager retrieves the backup data from the backup transport and returns it to the application so the application can restore the data to the device.

If an application wants to request a backup operation, it must call the method dataChanged() in the application. This means it would be impossible to sync all application settings just in one application. Therefore, when a user resets their device or upgrades to a new device, he must start every application and perform the backup operation one by one. It would be very unconvenient for the users.

Table I: The Difference between ASIMS and ADB

	ASIMS	**ADB**
Belongs to Which	Applications	Application Framework
Supports OS Version	Every	Android 2.2+
Storage Server	Any	Google's
Centralized Management	Yes	No
Key Needed	No	Yes

Table 1 shows the difference between ASIMS and Android's data backup. From the table, we can see that ASIMS provides a more friendly solution. ASIMS proposes a solution which is more like what the sms, contacts and emails backup scheme does. A user would

just open one application and click a single button and the whole system would be kept in sync. It not only reduce the burden of developers, but also is more acceptable for users.

V. CONCLUSION

Driven by an analysis of Android security mechanisms and content provider, this paper presents a new Android-based application settings integration and management scheme. It enhances the security of personal information on Android by syncing a user's personal information to the Internet and syncing back if necessary to prevent the lost of personal information. Different from Android's bulid-in synchronization system, ASIMS exposes an interface, which allows the other applications to store settings in one place, and allows users to select which application to sync.

REFERENCES

[1] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security", IEEE Security & Privacy, vol. 7, no. 1, pp. 50–57, 2009.
[2] Andrew Kameka , "Android has 150k apps, 350k daily activations, and more notes from Eric Schmidt's MWC keynote", Androinica.com, March 2011.
[3] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici and Shlomi Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms", arXiv:0912.5101v1, 2009, pp. 1-2.
[4] SHABTAI, A., FLEDEL, Y., KANONOV, U., ELOVICI, Y., DOLEV, S., AND GLEZER, C., "Google Android: A Comprehensive Security Assessment", IEEE Security & Privacy, vol. 9, no. 2, 2010, pp. 35 - 44.
[5] Sayed Hashimi, Satya Komatineni, and Dave MacLean, Pro Android 2, Apress, USA, 2010, pp. 76-77.
[6] Google Inc., "Content Providers", android.com, Apr 2011.
[7] Google Inc., "Security and Permissions", android.com, Apr 2011.

AUTHOR'S PROFILE

Sunitha Sooda
received the B.E. degree in information technology engineering from SPMU, Sri Padmavathi Mahila Visva Vidyalayam, Tirupati. in 2010 and M. Tech. degree in electronics from Visvesvaraya Technological University, Belagavi, Karnataka in 2014. Currently working as Asst. professor in HKBKCE, Bangalore.

Archana Hombalimath
received the B.E. degree in electronics & communication engineering from Visvesvaraya Technological University, Belagavi, Karnataka in 2008 and M. Tech. degree in electronics from Visvesvaraya Technological University, Belagavi, Karnataka In 2014.Currently working as Asst. professor in HKBKCE, Bangalore.