ISSN No. 0976-5697

Volume 6, No. 1, Jan-Feb 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Extension to Android Security Framework

R M. Sherkar
Mtech [Student], Computer Science and Engg.
Government College Of Engineering Amravati
Amravati, India

Prof. R V. Mante
Assistant Professor, Computer Science and Engg.
Government College Of Engineering Amravati
Amravati, India

Dr. P N. Chatur
Head of Dept., Computer Science and Engg.
Government College Of Engineering Amravati
Amravati, India

Abstract: Android is an open source mobile operating system developed by Google popular mobile-device platform developed by Google. It allows the application to share their data and code with another application. However, these sharing can be tightly controlled by permission given in the manifest file of Android. Overall, user can't predict what the application can do with their data. Hence, this assignment describes the framework which is helpful in providing security to data that's present on android devices. It provides not only internal security but also externally by using AES encryption algorithm so that unauthorized party can't read the user's personal/private data.

Keywords: isolation; AES encryption; context; taint; android-platform.

I. INTRODUCTION

In this new era, the use of smartphone has been increasing very rapidly and android operating system which is an open source platform has become very popular. As given by [ASEC], in Q3 2011, 52.5% of all devices sold were Android devices, followed by Symbian (16.9%) and Apple's iOS (15.0%), according to Gartner analysis.

Now-a-days the use of smartphone in private and corporate sector has been increasing very rapidly because of this security of data is greatly essential. Smartphone can be used for doing net banking, shopping, money transfer, sharing corporate files. Because of these factors android based smartphones become a very attractive target for malicious and unauthorized users. Up till now android operating system security model is successful in preventing the attacks from malware. Also an extension to this is an anti-theft concept which is described in further sections.

A. An Overview to Android:

From the above discussion, android operating system becomes popular hence pre-installed on all smartphones which is currently being sold out in order to meet different requirement than personal PCs and server related operating system that to in security and functionality.

Now coming to its structure, Android operating system is responsible for implementing a complete set of software i.e. the stack of software for running mobile applications. Bottom most layer is Linux kernel layer having networking, power management, memory management and device drivers. Next part of Android is having some local libraries for database management, graphics and the functionality of web browser that can be shrieked out through the interfaces employed in Java. Furthermore, Android entails of basic Java archives, and a computer-generated machine for running source bytecode called as ".dex" which is derived from JVML bytecode. Upper layer of this is the application framework level, which acts as an intellectual machine for applications. Finally, the last layer i.e. top most layer contains code for applications, which is prepared in Java with an interface given by an SDK [1].

Now moving towards Android's application model, concentrating on how applications are prepared and run on mobile device in detail.

a. ***Activities:*** It provides interface so that user can interact with the application in a convenient way. For example, consider "hike" application in which user can do messages through one activity while can provide attachments including photos, files, contacts, locations, etc. through a different activity. Though, these activities can work in an organized manner in order to provide organized structure and each one is executing in its own address space. Moreover, another app such as camera which can start activity in order to provide or share pictures to users.

b. ***Service:*** It is one of the component of Android app model which can run in the background without interrupting with others and can perform work for remote procedures. It doesn't provide any interface for user to interact because it is running in the background of app. For example, a service can play music in the background whereas user is working on another application.

c. ***Content Providers:*** This one is responsible for storing and sharing application's data. You can store data in the file system which is accessed by app. With the help of it, user can query and modify data. For example, on "whatsapp" application allow user to store or update contact address book which is not present in the contacts but received message from it on the application.

d. ***Broadcast Receiver:*** Broadcast receiver is responsible for displaying messages whenever events of applications starts up such as when system boots up, battery charging, etc.

Above three components [2], of Android application model excluding content provider component are activated by a message called as Intent which is responsible for linking applications. Android mobile applications using this intent for doing both intra-application and inter-application communication. In general terms, Intents are nothing but the messages that do communications between the components of Android application model.

For the components like activities and services, an intent outlines the event for performance. For example, to "send" a file or "display" something. For broadcast receivers, the intent provides the statement that is being broadcasted. There are two types of Intent:

a. **Explicit intents** state the component name in order to start it. Explicit intent is used to start a component in the particular app or service you want to start. For example, start a novel activity in answer to a user action or start a service to play a song in the background.

b. **Implicit intents** do not label a precise component, however in its place state an overall act to accomplish, which lets a module from different app to hold it. For example, suppose user wants to provide a location on a map, he/she can use an implicit intent to demand that a different skilled app display an identified location on a map.

II. EXISTING SECURITY ON ANDROID

Android is an open source Linux-based operating system which is automated using Java besides this executed in its own address space. Android pools operating system skins like multi-tasking of processes, memory management, Unix user identifiers (UIDs) for each of its procedure in implementation plus file authorizations through the type-safe features of Java and its application programmable interface libraries. However, the resultant security outline is like a multi-handled server. Dissimilar to a personal computer operating system where all user applications shares the same UID given by Linux Kernel level, applications on Android are discretely subdivided from each other. Applications on Android platform are having different UID with distinct permission set provided by Android. Each and every process is restricted to tamper with other files or data plus sharing of such files or data can be done explicitly with the help of programmer/user [3].

The Android permission based model is a straightforward way for providing security in order to access various resources or data that are available on Android. Even though permission given to applications on Android are classified to distinct security stages like Regular, Unsafe, Signature and SOS (Signature-Or-System), however, consignment of these security stages to various data or resources is depend on developer's ability and their own understanding. Because of this, Android security faces several security related problem from malicious application and from some of the legitimate applications. Android platform allow user to download application from Google Playstore and install it. While doing so, user will get a dialog box contains list of permissions, which he/she has to accept all to install the app successfully or can deny it for cancelling installation process. Basically, there are a digit of security issues in this: 1) The user has to accept all permissions and grant them in order to carry on installation process successfully, 2) once the installation and granting of permission is done; there is no way for revoking the permissions that are already granted at the installation time 3) in between there is no way of restricting the application for accessing the data or resources, 4) the permissions that are granted at installation time can be reverted back by uninstalling it [4].

Technically speaking, Android combines two levels of enforcement [5], [6]: at the Linux kernel level and the application framework level. At the Linux kernel level Android is a multi-process system. During installation, an application is assigned with a unique Linux user identifier (UID) and a group identifier (GID). Thus, in the Android OS each application is executed as a different user process within its own isolated address space. All files in the memory of a device are also subject to Linux access control. On a Linux, file access permissions are set for three types of users: the owner of the file, the users who are in the same group with the owner of the file and all other users. For each type a tuple of read, write and execute (r-w-x) permissions is assigned. In Android, by default, the files in the user's home directory can be read, written and executed by the owner and the users from the same group as the owner. All other users cannot work with these files. So as different applications by default have different user identifiers files created by one application cannot be accessed by another.

At the application framework level, Android provides access control through the inter-component communication (ICC) reference monitor. The reference monitor provides mandatory access control (MAC) enforcement on how applications access the components. In the simplest form, protected features are assigned with unique security labels—permissions. Protected features may include protected application components and system services (e.g., Bluetooth). To make the use of protected features, the developer of an application must declare the required permissions in its package manifest file: AndroidManifest.xml [7].

A. Present Security Apps:

Android operating provides internal security to data present on the device having it but rather many people wish to download and install various security apps for extra protection. However, Google Playstore consist of number of application including these types of apps. All such type of applications are generated at Application level framework and each has their own pros and cons. Some of these apps are given below.

a. File cover
b. APP Lock
c. Smart App protector
d. Gallery private
e. Gallery pro lock
f. Free data vault

a. Advantages :

a) ***Safety:*** When you start using such apps you will feel protected getting that your files or data are properly protected and secure from unauthorized party plus no one can see what you don't want to show them.

b) ***Easy To Use:*** These applications are easy to handle and can install easily because of familiar GUI.

c) ***Free of cost:*** Near about 1000+ apps are present at Google Playstore however many of them are freely available that can be easily downloaded.

b. Disadvantages:

a) ***Locks:*** Once you start using these applications you will have to wait for a while in order to view the content of files/folder.

b) ***Passwords:*** By using password mechanism, every time user has to type it in order to unlock the folder/file.

III. RELATED REVIEW

First, This paragraph describes the previous work in this field. Following sections provides research in order to enhance the security of the Android operating system environment.

A. Extension to Android security:

Number of solutions are given by the researchers in order to improve the security of Android platform. We have giving some of these that are related to our proposed work. As everyone know that, while installing application on Android platform, user has to grant all permissions that are requested in the manifest file. The platform supports all-or-nothing approach i.e. user has to grant all permissions for installation or else deny the installation. Again, user cannot revoke the permissions given at installation time while running particular app. To solve this problem, some solutions are given.

Apex [8], while installing the applications, it provides such a mechanism that user can change the permissions to an application at installation time, meaning that user has a provision to change permission given in the Android manifest file. Semantically Rich Application Centric Security in Android [9], provides the framework that gives the enhancement to existing Android security. It provides installation time rules that adjust the assignment of the permissions in order to protect their API. Also it controls how applications interacts with each other. Crepe [10], provides the mechanism so that user can create their own special rules i.e. policies that can control the granting of permissions in the manifest file automatically during the installation time.

According to some research, [11], [12] concentration is given on private data only. MockDroid [11], is a framework which provides mock data i.e. false data when applications are trying to access the data without having that access through the unbound network. Also, according to TISSA [12], when the app is trying to read the data, it first send request to content provider and checks the current security settings related to the app. If the reading operation is allowed then and then only app can access the data in other cases it is denied. Taindroid [13], the system assigns taint i.e. label to each and every predefined data and controls the access of the app. When app is trying to access the data through unbound network connections at that time it notifies the user about this happening and label the application name.

AppFence [14], it shows the shadow data when the app trying to access the data unauthorizingly and block the access of that application. Context plays a very vital to enhance the security of Android, in [8], [10], context provides security rules at run-time. The framework given in [15], [16], shows the usage of context in order to limit access to data.

B. Security Profile:

According to Moses [7], it provides isolation by keeping application and data related to work separated from recreational app and personal/private data. Within the same device, such environment can run in its own address space. Meaning that, application and data belongs to entertainment are not able to access the data related to corporate sector. Here, security environments are associated with one or more context that determined activation/deactivation of security profile (environment).

Context are nothing but the Boolean expression that is defined over any information obtained from smartphone's logical or physical sensors like GPS, mobile data, Wi-Fi. When this value becomes true, then security profile associated with one or more context becomes activated. But it may happen that one or more profile becomes activated at the same time. To solve this conflict each security profile assigns with a priority. Profile having higher priority is activated first than a profile having lower priority. If profile is having same priority at this moment, the profile which is activated first remains in a working condition.

User can dynamically switched between these profiles with the help of MOSES GUI. Each profile is associated with the password so that no one can tamper with the data.

IV. PROPOSED WORK

As given in the MOSES [7], separate environments are given so that unauthorized user can't tamper the data. But all this research provides internal security to the data but what about the external security when the cell phone get stolen. Now-a-days, smartphone has no value but the data which is present on the device has great importance. Now question arises how to secure or get the data back when the phone get stolen. One secure way is to apply pattern/pin locks in such a way that unauthorized party can't crack it. This one is the more general way to provide security. More enhanced way is given in following diagram.

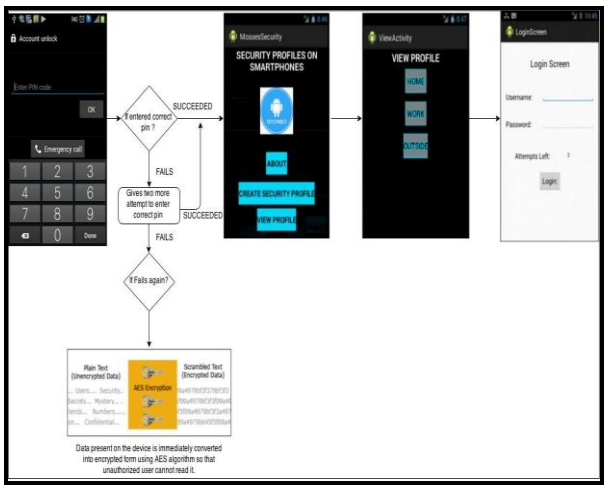

Figure 1. Proposed Architecture

Each security profile is associated with owner of the profile and can be protected with password. Additionally system also supports remote management of profile that is handled by enterprise administrator and protected with the password given by corporate world so that user cannot

tamper with data even if working from home. The main objective is to protect data not to save phone because now-a-days the price of smartphone is decreasing very rapidly. Also daily backup is send to authorized mail-id which is in encrypted form so that no one can tamer with the data. For encryption purpose AES algorithm is used. It is a cryptographic algorithm, used to protect data electronically present on device. In particular, it is an iterative, symmetric-key block cipher that can use keys of 128, 192, and 256 bits, and do encryption and decryption of data in the blocks of size 128 bits (16 bytes). In contrast to public-key ciphers, which is using a pair of keys, the symmetric-key ciphers uses same key to encrypt and decrypt data. Encrypted data returned by block ciphers have the same number of bits that of the input data. Iterative ciphers in AES uses a loop structure that recurrently performs permutations and substitutions of the input data provided by user. The total storage size for operating system is given by following equation.

$$total\_size = size(OS) + size(executing\_app) + size(executing\_appdata) \quad (1)$$

Where, *size(OS)* is the total size required by the operating of handset, *size(executing_app)* is the size required for executing the particular application and $size(executing\_app's data)$ is the size required for storing application's data.

V. CONCLUSION

From the above sections, it is cleared that the given framework allow user to do their official work while seating at home plus it also provides security environment in which app and data related to corporate world can't be tampered by apps related to third party. When user enter wrong login credential at that time all data converted into encrypted from which can be decrypted by authorized user.

VI. ACKNOWLEDGMENT

We have taken hard work in this assignment. Nevertheless, it cannot be possible without the generous support and aid of many dignities belongs to our institution. We would like to show my sincere gratitude to all of them who directly or indirectly supports us throughout.

VII. REFERENCES

[1] A. P. Fuchs, A. Chaudhuri, and J. S. Foster. SCanDroid: Automated security certification of Android applications, Technical report, University of Maryland, 2009.

[2] Erika Chin and et. al., "Analyzing Inter-Application Communication in Android," University of California, Berkeley, CA, USA, June-July 2011.

[3] E. Konstantinou and S. Wolthusen. Metamorphic virus: Analysis and detection. Technical report, Information Security Group at Royal Holloway, University of London, 2009.

[4] Muneer Ahmad Dar and Javed Parvez, "A Novel Strategy to Enhance the Android Security Framework," International Journal of Computer Applications (Volume 91-No.8), April 2014.

[5] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," IEEE Security and Privacy, vol. 7, no. 1, pp. 50-57, Jan./Feb. 2009.

[6] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google Android: A Comprehensive Security Assessment," IEEE Security and Privacy, vol. 8, no. 2, pp. 35-44, Mar./Apr. 2010.

[7] Yury Zhauniarovich and et. al., "MOSES: Supporting and Enforcing Security Profiles on Smartphones," IEEE Transactions on Dependable and Secure Computing, (Volume 11-No. 3), May-June 2014.

[8] M. Nauman, S. Khan, and X. Zhang, "Apex: Extending Android Permission Model and Enforcement with User-Defined Runtime Constraints," Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS '10), pp. 328-332, 2010.

[9] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically Rich Application-Centric Security in Android," Proc. Ann. Computer Security Applications Conf. (ACSAC '09), pp. 73-82, 2009.

[10] M. Conti, B. Crispo, E. Fernandes, and Y. Zhauniarovich, "CRePE: A System for Enforcing Fine-Grained Context-Related Policies on Android," IEEE Trans. Information Forensics and Security, vol. 7, no. 5, pp. 1426-1438, Oct. 2012.

[11] A.R. Beresford, A. Rice, and N. Skehin, "MockDroid: Trading Privacy for Application Functionality on Smartphones," Proc. 12th Workshop Mobile Computing Systems and Applications (HotMobile'11), pp. 49-54, 2011.

[12] Y. Zhou, X. Zhang, X. Jiang, and V. Freeh, "Taming Information Stealing Smartphone Applications (on Android)," Proc. Fourth Int'l Conf. Trust and Trustworthy Computing (TRUST '11), pp. 93-107, 2011.

[13] W. Enck, P. Gilbert, B.-G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth, "Taintdroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," Proc. Ninth USENIX Conf. Operating Systems Design and Implementation (OSDI '10), pp. 1-6, 2010.

[14] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These Aren't the Droids You're Looking for': Retroffiting Android to Protect Data from Imperious Applications," Proc. 18th ACM Conf. Computer and Comm. Security (CCS '11), pp. 639-652, 2011.

[15] D. Feth and A. Pretschner, "Flexible Data-Driven Security for Android," Proc. IEEE Sixth Int'l Conf. Software Security and Reliability (SERE '12), pp. 41-50, 2012.

[16] D. Feth and C. Jung, "Context-Aware, Data-Driven Policy Enforcement for Smart Mobile Devices in Business Environments," Proc. Int'l Conf. Security and Privacy in Mobile Information and Comm. Systems (MobiSec '12), pp. 69-80, 2012.