A Novel Strategy to Enhance the Android Security Framework

Muneer Ahmad Dar
Scientist-B,
National Institute of Electronics & Information Technology (NIELIT),
Srinagar India

Javed Parvez
Assistant Professor,
Department of Computer Science,
University of Kashmir
Srinagar India

ABSTRACT
With the widespread use of the Smartphone, the security of data stored in a Smartphone has reached to an utmost importance to all of us. Installation of every Android app asks for some critical permission to access our critical files and we have to accept the permissions in order to install that application. We propose a novel approach of enhanced security framework which can be integrated with the existing Android Security Framework to make Android more secure and to keep track of the files accessed by any of the vulnerable apps downloaded from different sources on the web. The proposed enhanced security framework enhances the security of Android File System by restricting the apps whose behavior matches with the malware. A novel approach to secure the data on Smartphone's using cryptographic Algorithms is also discussed in this paper.

Keywords
Smartphone, Android app, Security Framework, File System, Malware, Cryptographic Algorithms

1. INTRODUCTION
Smartphone have assumed an increasingly vital role in the lives of Information and Communication Technology (ICT) users. Today, a constantly increasing number of consumers use Smartphone for a broad variety of tasks and purposes, ranging from social networking to instant messaging, from mobile banking to Global Positioning Satellite (GPS) based navigation. Smartphone serve as special tools for organizing the users' daily lives through productivity applications such as calendars, memos or calculators. The apps downloaded from Google Play like True Caller, Viber, Whatsupp, IndiaLive and billions of other apps have changed the life of a typical android user with 1.2 billion people worldwide using mobile apps at the end of 2012. This number is forecast to reach 4.4 billion users by the end of 2017[21]. Analyst estimates for downloads of apps in 2013 range from 56 to 82 billion. In 2017, there could be 200 billion downloads [21]. With these many number of users using the apps downloaded from the Google play, securing the use of these apps is of paramount importance to researchers.

Trusted apps are available in Google's market which is self-signed by the developers but Malware has even appeared in Google's market. Two examples are DroidDream[7, 8] and Droid Dream Light [9]. Both these apps were found on the Android market in early 2011 and both applications steal personal data and are very much like traditional Trojans seen on the desktop.

It is all too common to hear about these bad apps that steal and alter our valuable data and can make our Smartphone non functional, and those discussions always end with one thing -- someone says you need to read an app's permissions before you install it.

Fig 1: Permission Screen for App Installation

In this paper we propose an enhanced security framework which will restrict the access of vulnerable apps by checking the behavior of these apps. In section 2 we identify the inbuilt security features of Android and Its limitations. In section 3 we give advantages and disadvantages of security apps available on Google Play. In section 4 we then have a detailed description of the proposed security Android Framework which will protect our data on our personal Smartphone. Finally we draw our conclusions in section 5.

2. EXISTING SECURITY OF ANDROID

Android is a Linux-based mobile operating system programmed with Java and implemented with its own security framework. Android combines OS features like efficient shared memory management, preemptive multi-tasking of processes, Unix user identifiers (UIDs) for each of its programs in execution and file permissions with the type-safe features of Java language and its well known API library. The resulting security framework is much more like a multi-user server than the sandbox found on the J2ME platforms. Unlike a desktop computer operating system where a user's applications all run under the same UID, Android applications are individually partitioned from each other. Android applications run in distinct processes under distinct UIDs each with different set of permissions. Programs have no permission to read or write each other's files/data or code, and sharing data/files between applications must be done explicitly by the programmer. The Android GUI environment has some novel and distinct security features that help support this isolation of processes [7].

The permission based model is the basic mechanism for securing access to various files or resources in Android. Although the app permissions are categorized to different protection levels such as Normal, Dangerous, Signature and Signature-Or-System, the assignment of these protection levels of various resources is left to the developer's will and his/her own understanding. This feature of Android Security framework may lead to attacks by malicious software and a number of vulnerabilities in the security framework of Android. When an application is downloaded and installed by the user on Android, the Android framework prompts the user to accept a list of required permissions, the user may grant all of the permissions in order to install the application successfully or deny the permissions to cancel the installation. Practically, there are a number of security issues in such a framework: 1) The user must accept and grant all of the required permissions in order to install the application successfully, 2) once the app is installed and permissions are granted; there is no mechanism for restricting an application to revoke the permissions already granted 3) there is no way of restricting access to the resources based on dynamic constraints as the permission model is based on install-time check only, 4) granted permissions can only be revoked by uninstalling the application.

At the time of installation, the user is presented with a dialog box listing all permissions requested by the app to get successfully installed. These permission requests are defined in an XML File called AndroidManifest.xml, which is shipped with every Android app.

However, this security framework has a few drawbacks [3]:

• All or No Permission:

A user cannot grant single permissions, while rejecting others in order to install the app. Among the list of permissions an app might request a suspicious permission among the other legitimate permissions, will still be able to confirm the installation.

• Often, the users of the app cannot judge the appropriateness and legitimacy of permissions for the app in question. In some cases it may be well understood, for example when a chess game app requests the privilege to reboot the Smartphone or to send SMS messages. In many cases, however, users will simply not be able to understand the appropriateness of the permission.

• Functionality, which is supposed to be possible only given the appropriate permissions, can still be achieved with less number of permissions or even with none at all.

3. EXISTING SECURITY APPS

The Google store contains numerous apps that cater to the need of protecting files in Android phones. All these apps are developed at application layer and have their own pros and cons. Given below are a few apps that are widely used by Android phone users.

- File Cover
- Smart App Protector
- Gallery Private
- APP Lock
- Gallery Lock Pro
- Free Data Vault

3.1 Advantages

3.1.1 Safety

Once you begin using these apps you will feel safe knowing that your files/folders are properly protected and no one can see what you do not want them to see.

3.1.2 Easy To Use

These apps are so easy to use that anyone can install them.

3.1.3 Free of cost

Most of the apps mentioned above are free of cost and can be easily downloaded from Google store.

3.2 Disadvantages

3.2.1 Locks

Once you begin using these apps you will not be able to access your files as quickly as you used to. Locked files take time to unlock.

3.2.2 Passwords

When one uses a password to protect files one must make sure that it is something that one can remember. This ensures access of file whenever required.

Other than the security Apps downloaded from the Google Store which can protect our files from the unauthorized access by simply encrypting them and making them password protected, we have other Antivirus Apps which are listed below:

- AhnLab Mobile Security.
- AVG Antivirus FREE for Android.
- Avira Free Android Security.
- F-Secure Mobile Security.
- BullGuard Mobile Security.
- EndUser Protection.

3.3 Advantages

3.3.1 On Demand Scan.

3.3.2 Anti Theft.

3.3.3 On Access Scan

.

3.4 Disadvantages

3.4.1 No Firewall.

3.4.2 No Privacy Advisor.

3.4.3 No Web Protection.

4. PROPOSED SECURITY FRAMEWORK

The objective is to provide security against the Apps which are installed by the end user and is given all the permissions at the time of installation. This enhanced security has the desirable property of not disturbing a regular user in any noticeable way. In fact, the user need not even be aware that the Security API has been applied. We have to prevent the modification and access of data from mobile phones by other external malicious applications unknowingly. We propose an API which will enhance the security of existing Android Framework by addressing the following limitations of Android Security Framework.

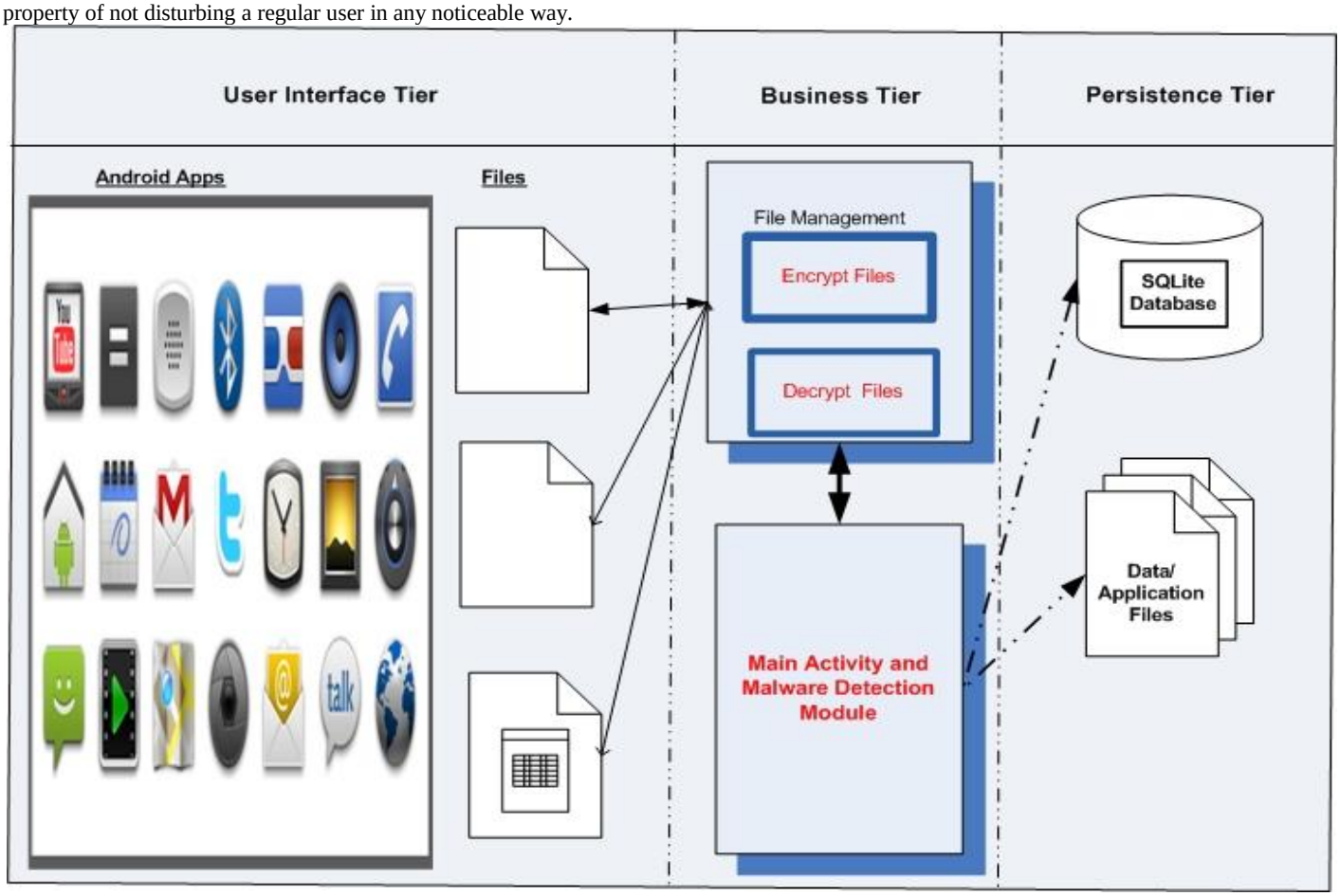

Fig 2: System Architecture

- Data/Files are not stored in encrypted format within media.
- The lack of strong security control of user´s private information that permits malware to access the information stored in the device.
- Configurable firewalls are not integrated into Smartphone operating systems.

4.1. File Encryption

The first step in our proposed security API is implemented by adapting an encryption technique utilizing Advanced Encryption Algorithm (AES) and applying it to all the personal files in the Smartphone. All the files which are important for the cell phone owner should be encrypted. Nobody can break the algorithm password as AES is considered to be the strongest cryptographic algorithm among the existing ones. When any app attempts to access any encrypted file it will notify the security API. If the security API finds any vulnerability it will restrict the access mechanism of the app and will deny it access to the file. If it fails to find any vulnerability It will allow the app to access the file by decrypting the file.

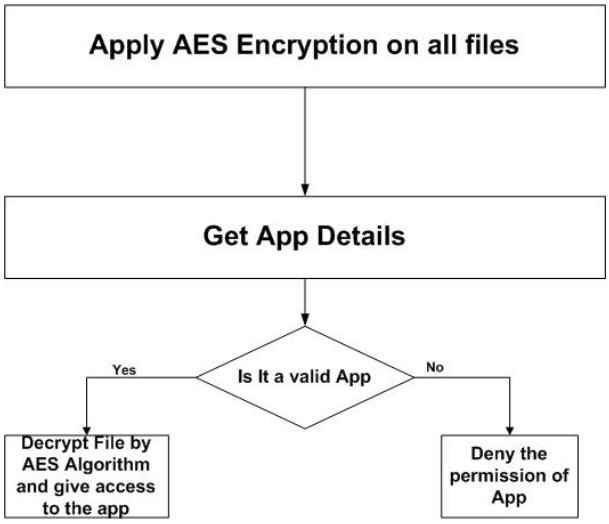

Fig 3: System Flow

4.2. Malware Detection

File operations offered by the proposed Security API should aid in the detection of potentially malicious Apps whose behavior matches that of Malware. Malware recognition is usually achieved by signature matching, heuristic analysis, or comparing hash-values. We provide details in the following sections about how our proposed API can provide for these malware recognition techniques.

4.2.1. Signature Matching

Our Proposed API shall provide a method to conduct pattern matching using regular expressions. That method will only return true or false for any given pattern described by a regular expression passed to the method. Currently one or multiple byte sequences are found in common signatures which can be linked in various relations, such as logical (AND, OR) or regarding location in a target file. Such relations can be implemented through regular expressions which will ensure user and system data privacy, while still allowing signature-based detection. Different pattern matching algorithms can also be used, which can improve pattern matching performance or compatibility with the use of existing vendor signature databases.

4.2.2. Heuristics

Apart from pattern or hash matching we also propose to use heuristic techniques to detect the malicious applications. The major advantage is that potentially malicious behavior can also expose currently unknown malware, and is often unaffected by minor alterations which would circumvent pattern or hash matching. Such heuristics are mostly based on sequential execution of specific library, API, or system calls, and may contain also certain passed arguments. Heuristic techniques can be divided into static and dynamic heuristics [6]. While the former is easier to implement, the latter yields better results.

4.2.3. Hashes

Malware can also be detected using hash functions. Desktop malware is usually spread in a highly decentralized manner via exploitation of software vulnerabilities, and often the same malware may be spread by many people for whom it has been "personalized". This leads to hashes being a less useful approach. On mobile platforms, however, with difficult app vulnerability exploitation and with centralized software distribution, hash-based malware detection gains value.

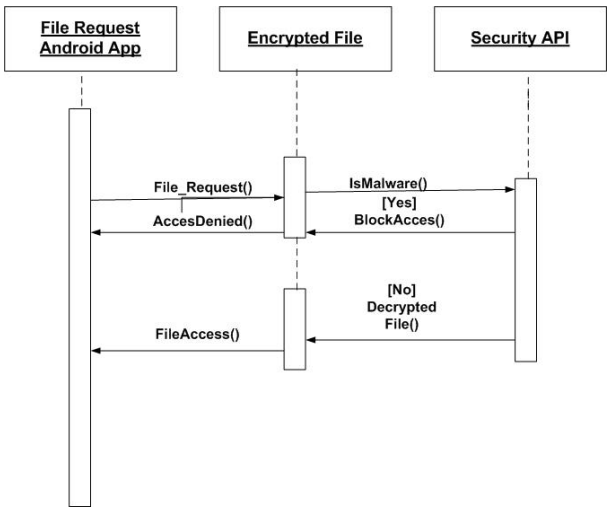

Fig 4: Sequence Diagram of the System

4.3. Advantages of the Proposed System

4.3.1. There is no need for a user to create passwords for files and to remember those passwords. The Security API will automatically encrypt and decrypt files.

4.3.2. If a novice user has installed any malicious app and has given all the permissions at install time, this Security API will restrict the app from accessing any of the files.

4.3.3. Our proposed system uses the latest techniques to detect Malware which includes signature matching, Heuristics and Hash functions.

4.3.4. This enhanced security has the desirable property of not disturbing a regular user in any noticeable way. In fact, the user need not even be aware that the Security API has been applied.

5. CONCLUSION

With the current security architecture, many Smartphone operating systems are vulnerable to attacks because the Smartphone user is instrumental in deciding which applications will be installed on the phone. It is not easy for a user to judge applications by their description. The Android framework is one platform that expects the user to be security conscious and implicitly assumes applications developers are not malicious. Because of this, a user may unknowingly install software that poses a security threat or is not efficient enough to handle the user's privacy issues. Our aim is to provide a system to free the user from making decisions as to which applications to install and to provide protection to the user's personal files and data from any malicious apps downloaded from Google store. Thus, our proposed Security API enables users to install the apps and if the built-in security of Android is not able to prevent the unauthorized access of critical data, then this enhanced security framework will provide necessary safeguards.

6. REFERENCES

[1] Rafael Fedler, Marcel Kulicke and Julian Sch¨utte 2013 IEEE 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE)

[2] Android OS Security:Risks and Limitations A Practical Evaluation Rafael Fedler, Christian Banse, Christoph KrauSS, and Volker Fusenig 5/2012

[3] National Security Agency, "SELinux," January 2009. http://www.nsa.gov/research/selinux/.

[4] Ruben Jonathan Garcia Vargas "Security Controls for Android"2012 IEEE Fourth International Conference on Computational Aspects of Social Networks (CASoN) 215

[5] H. Jonsson, "Text Mining of Personal Communication," IEEE 2010.

[6] Shabtai A., Fledel Y., Glezer C., "Google Android: A comprehensive Security Assessment," IEEE Security and Privacy, 2010.

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

[8] V. Svajcer, "Aftermath of the droid dream Android market malware attack," http:// nakedsecurity.sophos.com/2011/ 03/ 03/ droiddream-android-market-malware-attack- aftermath/

[9] L. Dignan, "Malware sneaks by google's android market gatekeepers again,"http://www.zdnet.com/blog/ security/malware-sneaks-bygoogles-android-market-gatekeepers -again/ 8696 .

[10] "New droiddream variant found on android phones," http:// www.fsecure.com/ weblog/ archives/ 00002170.html.

[11] W. Enck, M. Ontang and P. McDanial, "Understanding Android Security," IEEE Security & Privacy Magazine, 7(1), 10-17, 2009.

[12] Wook Shin, Sanghoon Kwak, Shinsaku Kiyomoto,Kazuhide Fukushima, and Toshiaki Tanaka,"A Small but Non-negligible Flaw in the Android Permission Scheme", 2010 IEEE International Symposium on Policies for Distributed Systems and Networks.

[13] A. Shabtai, et al. "Google Android: A Comprehensive Security Assessment," IEEE Security and Privacy Magazine.

[14] A. Shabtai, Y. Fledel, Y. Elovici, "Securing Android-Powered Mobile Devices Using SELinux", IEEE Security and Privacy, http://doi.ieeecomputersociety.org/10.1109/MSP.2009.144.

[15] http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/e.

[16] Harunobu Agematsu, Junya Kani, Kohei Nasaka, Hideaki Kawabata "A proposal to realize the provision of secure Android applications" IEEE 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing.

[17] App Store Review Guidelines - App Store Resource Center: http://developer.apple.com/jp/appstore/guidelines.html

[18] Hideaki Kawabata, Takamasa Isohara, Keisuke Takemori, Ayumu Kubota:"Threat of Script abuse Android Permissions and Static Analysis", IPSJ SIG technical reports, 2011-CSEC-53-3, pp.1-6, 2011.5 (in Japanese).

[19] W. Shin, S. Kwak, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A small but non-negligible flaw in the android permission scheme," Policies for Distributed Systems and Networks, IEEE International Workshop, pp. 107-110, 2010.

[20] W. Shin, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A formal model to analyze the permission authorization and enforcement in the android framework," Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust, pp. 944– 951, 2010.

[21] W. Shin, S. Kiyomoto, K. Fukushima, and T. Tanaka, "Towards formal analysis of the permission-based security model for android," In Proceedings of the 2009 Fifth International Conference on Wireless and Mobile Communications, ICWMC '09, Washington, D.C., U.S.A., pp. 87–92, 2009.

[22] Android Open Source Project. Android Security Overview, October 2012. http://source.android.com/tech/security/.

[23] AV-Test. Test Report: Anti-Malware solutions for Android. Technical report, March 2012. http://www.avtest.org/fileadmin/pdf/avtest 2012-02 android antimalware report english.pdf.

[24] D. Bilar. Opcodes as predictor for malware. International Journal of Electronic Security and Digital Forensics, 1:156–168, 2007. Available at: http://inderscience.metapress.com/content/N760240L16832162.

[25] T. Bl¨asing, L. Batyuk, A.-D. Schmidt, S. Camtepe, and S. Albayrak. An android application sandbox system for suspicious software detection. In 5th International Conference on Malicious and Unwanted Software (MALWARE), pages 55–62, Oct. 2010.

[26] J. Burns. Exploratory Android surgery (talk slides). In Black Hat Technical Security Conference USA, May 2009. Available at: https://www.blackhat.com/html/bh-usa-09/bhusa-09-archives.html.

[27] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In Proceedings of the 12th USENIX Security Symposium, SSYM'03, Berkeley, CA, USA, 2003. USENIX Association.

[28] W. Enck, M. Ongtang, and P. McDaniel. Understanding android security. IEEE Security & Privacy, 7(1):50–57, Jan.– Feb. 2009.

[29] R. Fedler, C. Banse, C. Krauß, and V. Fusenig. Android OS security: Risks and limitations. Technical report, Fraunhofer AISEC, May 2012.

[30] R. Fedler, J. Sch¨utte, and M. Kulicke. On the effectiveness of malware protection on Android. Technical report, Fraunhofer AISEC, April 2013.

[31] Fraunhofer AISEC. App-Ray. http://app-ray.de/.