Volume 6, Issue 5, May 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Security Risk Associated with Android Applications

Manvindar Singh Chauhan*, Kulvinder Singh
Department of CSE, DIET Rishikesh,
Uttarakhand, India

Abstract— Android is the most popular mobile operating system. Its omnipresence leads to the fact that it is also the most popular target amongst malware developers and other computer criminals. Even after being the more secure operation system than any other Smartphone operating system, having very few restrictions for developer, and having no security scan over the apps being uploaded on its market increases the security risk for end users. In this paper we have reviewed android security model, security measures provided by Android and security issues in the android applications.

Keywords—Android security, Smartphone security, IMEI, SELinux, Dalvik Virtual Machine

I. INTRODUCTION

Android is the currently the most popular Smartphone in the market, leaving behind the iPhone, which initially fuelled the Smartphone market. The reason behind the popularity of the Android is, its source code is released by the Google under open source licences. Most of the Smartphone manufacturer companies which require a ready-made, low-cost, and customizable operating system go for Android. These companies ship their devices with a combination of open source and proprietary software required for accessing Google services. The open nature of Android had attracted a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices originally shipped with other operating systems.

The open nature of Android and its large user base have made it an attractive and profitable platform to attack. Common exploits and tool kits on the OS can be utilised across a wide number of devices, meaning that attackers can perform exploits en masse and reuse attack vectors. It is obvious why Android is a target, but why is it vulnerable? Google did take measures in the development of the Android kernel to build security measures in; the OS is sandboxed, preventing malicious processes from crossing between applications. Whilst this attempt to eliminate the concept of infection is admirable in some regards, it fails to address the issue of infection altogether.

II. ANDROID PLATFORM ARCHITECTURE

The Android architecture can be divided into four sections that are described below.

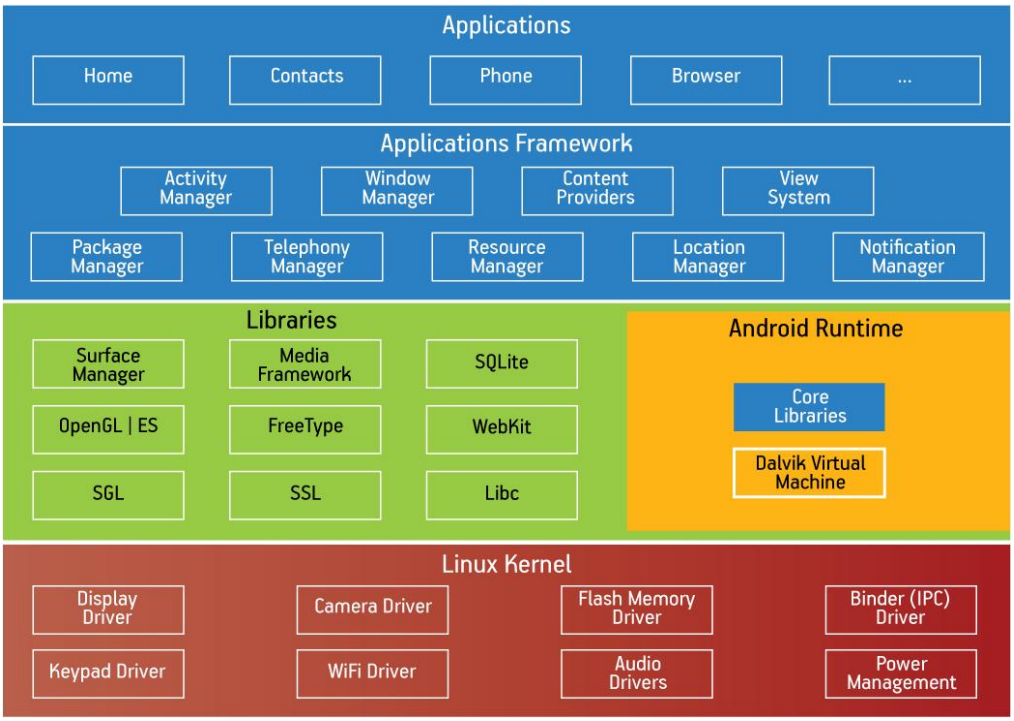

Figure 2.1: Android system architecture

A. Application Layer

This is the top most layer in the android architecture. All the applications are installed on this layer.

B. Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android framework includes the following key services:

- Activity manager: Controls all aspects of the application lifecycle and activity stack.
- Window manager: It is used to create views and layouts.
- Content providers: Allows applications to publish and share data with other applications.
- View system: An extensible set of views used to create application user interfaces.
- Package manager: Provides information about installed packages on device.
- Telephony manager: It is used to handle network connection settings.
- Resource manager: Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- Location manager: Provides location updates when user enters or leaves a specified geographic location.
- Notifications manager: Allows applications to display alerts and notifications to the user.

C. Libraries

The Android system provides some C/C++ libraries. Different components of the Android system can utilize these libraries. All these libraries are accessible with the help of the application framework [18][19]. Some of the important libraries of Android system are System C library, Media Libraries, Surface manager, libwebcore.

D. Android Run Time

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine(DVM) which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

E. Linux Kernel

This is the bottom most layer of the android architecture. It provides services like power management, memory management, security etc. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc.

III. SECURITY MECHANISMS IN ANDROID

The Linux kernel has features like user identifiers, pre-emptive multi-tasking, etc., that is used to enforce security between applications and system files. Unlike in desktop Linux systems where all application belonging to a user executes with same user id, in the Android system each application is assigned a unique identifier and separate instance of its own virtual machine so the application code runs in its own memory and process. For sharing data between application a “sharedUserId” feature used in the Android system.

Android systems have a permissions based mechanism to enforce security restrictions on applications. At installation time, a user has the chance to set flags to allow different permissions on applications. By default, a normal level of protection is granted to every application in a manifest file. Initially, the application is not able to access resources such as using the GPS, contact lists, writing to another application, accessing network services, etc. The installer package shows the lists of permissions where a user can allow or deny these permissions. When a user sets the permission then it can easily access the resource and it is not possible to revoke these permissions until the application is uninstalled.

Android security mechanisms also use the concept of sand-boxing to implement secure multiprocessing of applications. A reference monitor is used for inter-component communication (ICC) between applications. The application configuration is also written to the manifest file.

Digital certificates are used in application development by developers to sign their code. For this, a private key is used which creates trust between applications. Signing of applications does not deal with a certificate authority, and self-signed certificates are accepted identifying the author of the application. If a developer tries to install an unsigned application, then the system will not allow this.

The application also has public and private application components. Private application components are only accessible to other components, but not from other applications. Private component permission is set in the Manifest file by the developer. Public components are accessible to other applications but a developer has the choice to assign permissions to these components.

IV. EVALUATING ANDROID SECURITY

Our Android application study is focused on following points:

1. Exploring issues uncovered in previous studies and malware advisories,
2. Searching for general coding security failures, and
3. Exploring misuse/security failures in the use of Android framework.

Literature Survey

W. Enck, D. Octeau, P. McDaniel and S. Chaudhuri on their research 'a study of Android application security', introduced the ded decompiler, which generate android application source code directly from its installation image. They analysed 21 million lines of recovered code of Smartphone applications and concluded that low or no restriction of entry for application developers increased the security risk for end users. They also identified the leaks of user's personal / phone identifiers [1].

S. Powar, Dr. B. B. Meshram, on their research 'Android security framework', described android security framework and identified the increased exposure of open source Smartphone is increasing the security risk. Permission module to secure the phone is very basic. User has only two options at the time of application installation first allow all requested permissions and second deny requested permissions leads to stop installation [2].

S. Kaur and M. Kaur, on their research 'implementing security on Android application' identified the points to improve the security [3].

S. Smalley and R. Craig on their research 'Security Enhanced (SE) Android: Bringing Flexible MAC to Android', described how android software stack defines and enforces its own security model for apps through its application-layer permissions model. However, at its foundation, android depends upon the UNIX operating system kernel to shield the system from malicious or imperfect apps and to isolate apps from each other. At present, android leverages UNIX operating system discretionary access control (DAC) to enforce these guarantees, despite the notable shortcomings of DAC. In this paper, they motivate and describe their work to bring flexible mandatory access control (MAC) to Android by enabling the effective use of Security Enhanced Linux (SELinux) for kernel-level MAC and by developing a set of middleware MAC extensions to the Android permissions model [4].

P. Gilbert, W. Enck, L.P. Cox, B.G. Chun, J. Jung, A.N. Sheth and P. McDaniel on their research 'TaintDroid: An Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphone', reviled that Smartphone operating systems often fail to provide users with adequate control over and visibility into how third-party applications use their private data. They address these shortcomings with TaintDroid, system-wide dynamic taint tracking and analysis system capable of at the same time tracking multiple sources of private data. TaintDroid display real-time analysis by leveraging Android's virtualized execution environment and Monitoring private data to inform use of third-party applications for phone users and valuable input for Smartphone security service firms seeking to identify misbehaving applications [5].

M. Ongtang, S. McLaughlin, W. Enck and P. McDaniel on their study 'Semantically Rich Application-Centric Security in Android', proposed secure application interaction (Saint), an improved infrastructure that governs install-time permission assignment and their run-time use as dictated by application provider policy [6].

A.D. Schmidt, H. G. Schmidt, J. Clausen, A. Camtepe, S. Albayrak, K. Ali Yüksel and O. Kiraz on their study 'enhancing security of Linux-based android devices' present an analysis of security mechanism in Android Smartphone with a focus on Linux. The results are also applicable to any other Linux-based Smartphone; together with Android. They analysed android framework and the Linux kernel to check security functionalities. They surveyed well accepted security mechanisms and tools which could increase device security. Their second contribution focuses on malware detection techniques at the kernel level. They tested applicability of existing signature and intrusion detection methods in android platform. They focused in observation on the kernel, that is, identifying critical kernel event, log file, file system and network activity events, and making efficient mechanisms to monitor them in a resource restricted setting [7]. They presented a simple decision tree for deciding the suspiciousness of the application [7].

Client-side data logging performed by Android applications has not garnered much attention from a security standpoint. However, during Android application review, we often see sensitive user data like user names, passwords, and account numbers written to application logs. This information can be easily retrieved by an attacker if he is able to gain access to the device [8].

V. RESEARCH FINDING

Security is enforced by two basic methods. Firstly, each application run as a separate Linux process with their own user IDs. Therefore vulnerability in one application does not affect other applications. IPC mechanisms provided by Android need to be secured, by second enforcement mechanism. Android implements a reference monitor to mediate access to application components based on permission. If an application tries to access and other component, the end user must grant the appropriate permissions at installation time [9].

Phone identifiers, used as device fingerprints, are leaked through and sent to advertisement and analytics servers. Phone identifiers specifically the IMEI, are used to track individual users [1].

Android users need a way to determine if applications are leaking their personal information. They created a mapping between API calls and the permissions they must have to execute. AndroidLeaks is capable of analyzing 24,350 in 30 hours. AndroidLeaks drastically reduces the number of applications and the number of traces that a security auditor has to verify manually [10].

From a vulnerability perspective, it is found that many developers fail to take necessary security precautions. For example, sensitive information is frequently written to Android's centralized logs, as well as occasionally broadcast to unprotected IPC [1].

VI. CONCLUSION

Android Smartphones are rapidly becoming a dominant computing platform. Android has very few restrictions for developer, increases the security risk for end users. In this paper we have reviewed security issues in the Android based Smartphone. In this research, we have studied both dangerous functionality and vulnerabilities. There are several other factors which are responsible for the vulnerability in the android system like android OS fragmentation, lack in releasing security patches/OS updates by the vendors.

REFERENCES

[1] Enck W., Octeau D., McDaniel P. and Chaudhuri S., A Study of Android Application Security, The 20th SENIX conference on Security, 21-21, (2011).

[2] Powar S., Meshram B. B., Survey on Android Security Framework, International Journal of Engineering Research and Applications, 3(2), (2013).

[3] Kaur S. and Kaur M., Review Paper on Implementing Security on Android Application, Journal of Environmental Sciences, Computer Science and Engineering &Technology, 2(3), (2013).

[4] Smalley S. and Craig R., Security Enhanced (SE) Android: Bringing Flexible MAC to Android, www.internetsociety.org/sites/default/files/02_4.pdf . (2012).

[5] Enck W., Gilbert P., Chun B.G., Cox L.P., Jung J., McDaniel P. and Sheth A.N., TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, 9th USENIX Symposium on Operating Systems Design and Implementation. (2010).

[6] Ongtang M., McLaughlin S., Enck W. and McDaniel P., Semantically Rich Application-Centric Security in Android, Computer Security Applications Conference, 340–349 (2009).

[7] Schmidt A.D., Schmidt H.G., Clausen J., Camtepe A., Albayrak S. and Yuksel K. Ali and Kiraz O., Enhancing Security of Linux-based Android Devices, http://www.dailabor.de/fileadmin/files/publications/lk2008-android_security. pdf (2008).

[8] Naveen Rudrapp on Secure Coding for Android Applications. (2015).

[9] Enck W., Ongtang M., and McDaniel P., Understanding Android security, IEEE security Privacy, 7 (2009).

[10] Gibler C., Crussell J., Erickson J. and Chen H., Android Leaks: Automatically Detecting Potential Privacy Leaks In Android Applications on a Large Scale, 5th international conference on Trust and Trustworthy Computing, 291-307 (2012).

