International Journal of Computer and Internet Security.
ISSN 0974-2247 Volume 3, Number 1 (2011), pp. 17-24
© International Research Publication House
http://www.irphouse.com

Analysis of Framework and Security Issues in Android

Vishal Dahiya and Khyati Rami

Indis Institute of Technology and Engineering, Ahmedabad, India

Abstract

Android is an application execution environment for mobile devices. It includes an operating system, application framework, and core applications. This article attempts to unmask the complexity of Android security and note some possible development pitfalls that occur when defining an application's security. We conclude by attempting to draw some lessons.

Keywords: android, security, dalvik.

Introduction

Mobile Internet is the wireless internet services that can be accessed using handled devices such as mobile phones. Mobile Internet can be classified as limited and unlimited based on the service subscribers have to pay on download packet basis for the internet service where as in unlimited mobile internet services subscribers will receive unlimited access to news, entertainment, email etc for one month of subscription fee.

Android is software stack for mobile device that includes an operating system, middleware and key applications. It is a mobile platform that is complete, open and free. The third party developers can create applications, which are written in java programming language based on Linux Kernel, using Android SDK, JDK 5 or 6 and Ellipse IDE Version 3.2 or any latest version of Ellipse IDE, with the rich set of Google Android API(Application Programming Interface).

Framework of Android

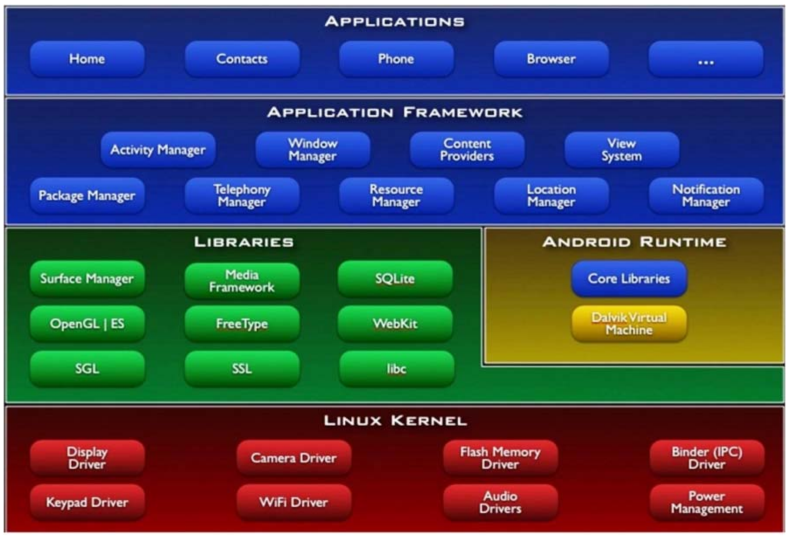

Figure 1: Diagram of Android Framework.

The Linux Kernel

The Kernel is based on Linux and figures as the Operating System. It is Responsible for Device Drivers, Memory Management.

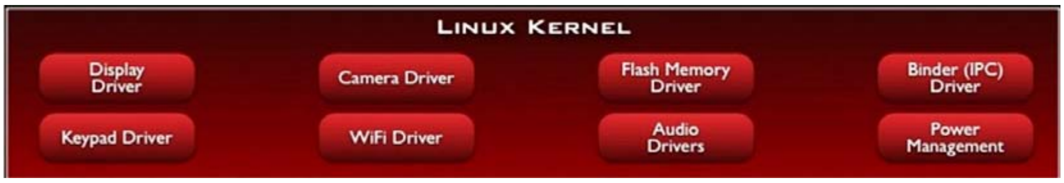

Figure 2: Linux Kernel.

Process Management and networking. The Linux Kernel is independent of the hardware platform, which gives the whole Android platform great flexibility.

The Android Run Time

The Android runtime Consist of two Components. First a set of core libraries which provides most of the functionality available in the core libraries of the Java Programming Language. Second the virtual machine Dalvik which operates like a translator between the applications which runs on the android is written in Java. As the Operating System is not able to understand this Programming language directly, the Java Programs will be received and translated by the virtual machine Dalvik. The translated code can then be executed by the Operating System. A very important notice is that applications will be encapsulated in Dalvik. For every program an own virtual machine is available even if some programs are running parallel. The Advantage is that the different programs do not affect each other, so a program error for example can lead to a crash of the program but not of the whole system.

The System Libraries
The available libraries are all written in C/C++. They will be called trough a Java interface and its capabilities are exposed to the developer through the Android application Framework.

Libraries Included are
System C Libraries: Derived implementation of the standard C Library, tuned for embedded Linux based Devices

Media Libraries: M.L Based on Packet Video's Open Core; the libraries supports playback and recording of many popular audio and video formats, as well as static image files, including MPEG-4, H.264, AAC, JPG & PNG.

Surface Manager: Surface Manger manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple application.

SGL: SGL ("Scalable Graphics Library") Underlying 2D graphic engine.

SQLite: This is powerful and lightweight relational database engine available to all applications.

The Application Framework
The Application framework is used to implement a standard structure of an application for this specific Operating System. Developers have full access to same framework API's used by the core applications. The application architecture is designed to simplify the reuse of components: Any application can publish its capabilities and any other application can use of those capabilities. There is a bored range of functionality provided for the developer, including

- Views
- Content Providers
- Resource Manager
- Notification Manger
- Activity Manger

Views: A rich and extensible set of view that can be used to build an application, including lists, grids, textbox, buttons and even an embeddable web Browser

Content Providers: That Enable application to access data from other applications (Such as Contacts), or share their own Data

Resource Manager: It is providing access to non-code resource such as localized string, graphics and layouts.

Notification Manager: That enables all applications to display custom alerts in the status bar.

Activity Manager: That Manages the Lifecycle of applications and provides a common navigation back stack.

Applications

All Applications are written in the Java Programming language. Android will ship with a set of core applications including: Email Client, SMS Program, Calendar, Maps, Browser, Contacts.

Android Application Components

Android defines four component types

- *Activity* components define an application's user interface. Typically, an application developer defines one activity per"screen." Activities start each other, possibly passing and returning values. Only one activity on the system has keyboard and processing focus at a time; all others are suspended.
- *Service* components perform background processing. When an activity needs to perform some operation that must continue after the user interface disappears (such as download a file or play music), it commonly starts a service specifically designed for that action. The developer can also use services as application-specific daemons, possibly starting on boot. Services often define an interface for Remote Procedure Call (RPC) that other system components can use to send commands and retrieve data, as well as register callbacks.
- *Content provider* components store and share data using a relational database interface. Each content provider has an associated "authority" describing the content it contains. Other components use the authority name as a handle to perform SQL queries (such as SELECT, INSERT, or DELETE) to read and write content. Although content providers typically store values in database records, data retrieval is implementation- specific—for example, files are also shared through content provider interfaces.
- *Broadcast receiver* components act as mailboxes for messages from other applications. Commonly, application code broadcasts messages to an implicit destination. Broadcast receivers thus subscribe to such destinations to receive the messages sent to it. Application code can also address a broadcast receiver explicitly by including the namespace assigned to its containing application.

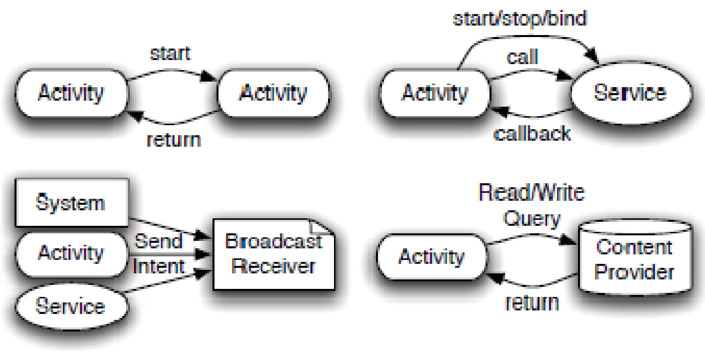

Figure 3: Android Application Components.

Previous related work

There have been several studies done on the topic of security of the Android system, but few of them focus on the formal aspect of the permission enforcing framework. Enck et al. developed an application evaluation tool, Kirin [3]. They represent the Android security policy with the notion of access matrix and test security policy invariants of an application at the time of installation. Chaudhuri proposed a typed language [4] to specify applications and reason about data flow. The type checking result of an application code makes it clear whether the application can preserve the secrecy and integrity of local data or not. Enck et al.'s automated tool and Chaudhuri's language-based approach differ from ours in that they evaluate applications in order to exclude malicious applications.

We have focused on the behavioral aspect of the framework rather than specification of application logic, and have attempted to confirm if it correctly authorizes permissions in accordance with the given requirements. Android permission scheme was inspired by the standard Role-Based Access Control model [1]. The permission authorization process involves the comparison of signing certificates. Moreover, the notion of access is introduced to model the permission protected interactions between application components.[7]

Security Framework in android

Table 1 is showing the security mechanism incorporated in android.

Table 1: Security mechanism in Android.

Mechanism	Description	Security issue
Linux mechanisms	Each application is associated with a different user ID (or UID). The application's directory is only available to the application.	Prevents one application from disturbing another Prevents one application from accessing another's files

Environmental features	Each process is running in its own address space. Type safety enforces variable content to adhere to a specific format, both in compiling time and runtime. Smart phones use SIM cards to authenticate and authorize user identity.	Prevents privilege escalation, information disclosure, and denial of service Prevents buffer overflows and stack smashing
Android-specific mechanisms	Each application declares which permission it requires at install time. Each component in an application (such as an activity or service) has a visibility level that regulates access to it from other applications	Limits application abilities to perform malicious behavior Prevents one application from disturbing another, or accessing private components or APIs

Next section is discussing the major issues related to the security of android.

Security issues in Android

Android has two basic parts of security enforcement [6]. First, applications run as Linux users and thus are separated from each other. This way, a security hole in one application does not affect other applications. However, there is also a concept of inter-process communication (IPC) between different applications, or more precisely, between the Android components of the applications such as activities and services [6]. The Java-based Android middleware implements a reference monitor to mediate access to application components based upon permission labels defined for the component to be accessed. Any application requires an appropriate permission label before it can access a component (mostly, but not necessarily, of another application). A number of features further refine Android’s security model. One example is the concept of shared user IDs, i.e., different applications can share the same user ID if they are signed by the same developer certificate. Another refinement are protected APIs: Several security-critical system resources can be accessed directly rather than using components.

It is providing good security levels in some aspects but it also compromise with security at other levels such as

- Android core libraries are open and user can write own application that can embed into a security problem malware, trojen horse or other virus. User can circulate very easily with this framework.
- Self signed certificates can be easily hacked by hacker or we can say easy targets.
- Permission based security can be adversely affect a user because GPS can allow any user to be tracked very easily if permission granted without user knowledge.
- It is unencrypted browser; so network information can easily propagated so the

personal mail of the user can be hacked.

- Vulnerability increases with the sharing of information on the network.
- Incompatibility with many antivirus increases the risk of using android OS.
- Once your system becomes vulnerable it is very easy to get effected by Denial of service effect(DoS).
- Abuse of costly services and functions (such as sending SMS/ MMS messages, making phone calls, or redirecting phone calls to high-rate numbers) by maliciously using the permissions granted by the owner at installation.
- Malicious activity against a network or network device (for example, sending spam, infecting other devices, sniffing, or scanning) by maliciously using the permissions granted by the owner at installation.
- Receiving spam, SMS/MMS messages, or emails.
- Pushing advertisements to the browser application when browsing the Internet.
- Loss of hardware components.
- Causing a malfunction in hardware components.
- An attacker can maliciously inject code via a Web browser. WebKit, Android’s open source Web engine, has a history of such vulnerabilities. Some recent attacks on it include a buffer overflow in an outdated native library and an explicit cross-site scripting (XSS) vulnerability. Both attacks let the attacker run malicious code on the device, with abilities and privileges assigned to the browser application.
- Injecting malicious applications via Bluetooth is, however, not likely to occur because several protection mechanisms exist:
  - A device can set the Bluetooth connection as not discoverable
  - If the Bluetooth connection is set as discoverable, it’s only for two minutes
  - The owner needs to accept the connection and the owner needs to manually install the file.

Above discussed issues are existing in android framework. These issues need to be addressed by the Goggle android framework developers.

Discussion

Each layer requires some strategies to be enforced for security. Use virtual private network for communication which require username and password, gives more secured access. Antivirus compatibility can be increased so that no vulnerable files or malware or Trojan horse effect can be applied on to the application. Emails and SMS can be encrypted so that no one can access the private data of the user. Android should incorporate a mechanism that can prevent or contain potential damage stemming from an attack on the Linux-kernel layer, such as the SELinux access control mechanism. Several vulnerabilities and bugs have already exploited these pathways to gain and maintain root permission on the device. Second, the platform needs better protection for hardening the Android permission mechanism or for detecting misuse of granted permissions. We recommend the remote management, VPN, and login solutions to provide telecom operators with a competitive edge when targeting corporate customers.

Conclusion

Android is providing better security as compared to iphone or other application but lacks in security at certain aspects. During permission security analysis two application shares the same UID (User Identification) which is not secure. If the phone user accidentally gives permission to the malware then it will be a problem. Open nature and self signed certificates are providing less security. Android security system should follow an architecture which is secured in every aspect.

Android's security framework is based on the label-oriented ICC Mediation. Partially out of necessity and partially for convenience, the Google developers who designed Android incorporated several refinements to the basic security model, some of which have subtle side effects and make its overall security difficult to understand.

References

[1] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," ACM Trans. Inf. Syst. Secur., vol. 4, no. 3, pp. 224–274, 2001.

[2] Google, Inc., "Security and permissions," http://code.google.com/android/devel/security.html (Retreived 2010-04-15).

[3] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in The 16th ACM conference on Computer and Communications Security, 2009, pp. 235–245.

[4] A. Chaudhuri, "Language-based security on Android," in The 4th Workshop on Programming Languages and Analysis for Security, 2009, pp. 1–7.

[5] W. Shin, S. Kwak, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A small but non-negligible flaw in the Android permission scheme," in IEEE International Symposium on Policies for Distributed Systems and Network, 2010.

[6] I. Krstic and S.L. Garfinkel, "Bitfrost: The One Laptop per Child Security Model," Proc. Symp. Usable Privacy and Security, ACM Press, 2007, pp. 132–142.

[7] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," IEEE Security & Privacy, vol. 7, no. 1, 2009, pp. 50–57.