ANDROID BASED MOBILE APPLICATION DEVELOPMENT and its SECURITY

Suhas Holla[#1], Mahima M Katti[#2]

[#] *Department of Information Science & Engg, R V College of Engineering Bangalore, India*

***Abstract*— In the advancing world of technology, Mobile applications are a rapidly growing segment of the global mobile market. Mobile applications are evolving at a meteor pace to give users a rich and fast user experience. In this paper, Android mobile platform for the mobile application development, layered approach and the details of security information for Android is discussed.**

Google released Android which is an open-source mobile phone operating system with Linux-based platform. It consists of the operating system, middleware, and user interface and application software. Certainly, Android is about to become the most widely used OS on mobile phones, but with Android comes a security vulnerability that few users take into account. On Android Market, where you can download thousands of applications for Android, anyone can upload their programs without having to submit them to careful security checks. This makes Android a prime target for computer criminals. In this paper, we discuss a layered approach for android application development where we can develop application which downloads data from the server. Also an Android Application Sandbox (AASandbox) which is able to perform both static and dynamic analysis on Android programs to automatically detect suspicious applications is also discussed.

***Keywords*— Android, application framework, android runtime, layered approach, AASandbox**

I. INTRODUCTION

Android is a new, next-gen mobile operating system that runs on the Linux Kernel. Android Mobile Application Development is based on Java language codes, as it allows developers to write codes in the Java language. These codes can control mobile devices via Google-enabled Java libraries. It is an important platform to develop mobile applications using the software stack provided in the Google Android SDK. Android mobile OS provides a flexible environment for Android Mobile Application Development as the developers can not only make use of Android Java Libraries but it is also possible to use normal Java IDEs. The software developers at Mobile Development India have expertise in developing applications based on Android Java Libraries and other important tools. Android Mobile Application Development can be used to create innovative and dynamic third party applications. Mobile Development India has worked extensively on projects ranging from gaming software, organizers, media players, picture editors to go-cart devices and more.

II. BACKGROUND STUDY

The platform was officially announced and the SDK tools were available in October 2008. Currently there is only one mobile phone that runs the Android OS, the G1 from T-Mobile. According to the official Android website (Android 2008) the platform is based into the four core features as shown in the Fig 1:

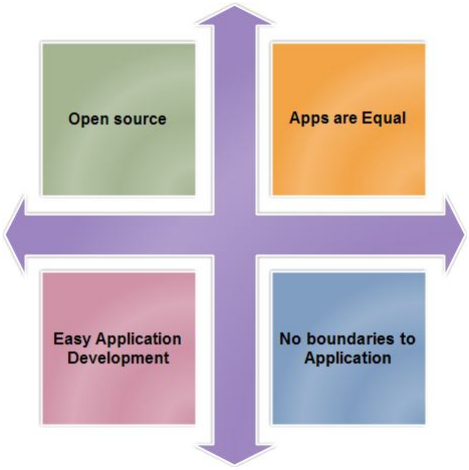

Fig. 1 Four core features of the android platform

A. Application Fundamentals

Android applications are written in Java programming language. However, it is important to remember that they are not executed using the standard Java Virtual Machine (JVM). Instead, Google has created a custom VM called Dalvik which is responsible for converting and executing Java byte code. All custom Java classes must be converted (this is done automatically but can also be done manually) into a Dalvik compatible instruction set before being executed into an Android operating system. Dalvik VM takes the generated Java class files and combines them into one or more Dalvik Executable (.dex) files. It reuses duplicate information from multiple class files, effectively reducing the space requirement (uncompressed) by half from a traditional .jar file. Dalvik was

created to support the nature of lightweight mobile operating systems require because of the limited hardware capabilities compared to conventional desktops or laptops.

B. Android Platform overview

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language [3]. Android based on Linux version 2.6. The system services such as security, memory management, process management are controlled by Linux. Fig 2 shows android architecture.

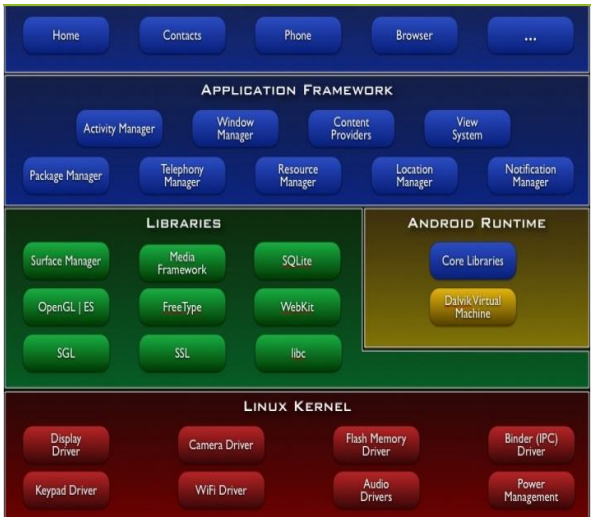

Fig. 2 Architecture of android [1]

C. Developing Android Applications

The Android SDK provides an extensive set of application programming interfaces (APIs) that is both modern and robust. Android handset core system services are exposed and accessible to all applications. When granted the appropriate permissions, Android applications can share data among one another and access shared resources on the system securely [5]. Android applications are written in Java programming language.

D. Application Framework

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs 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 may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

- A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification Manager that enables all applications to display custom alerts in the status bar
- An Activity Manager that manages the lifecycle of applications and provides a common navigation backstack.

E. Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language [5]. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

III. Layered Approach for Application Development

In this paper we suggest layered approach for android application development. This can be used for web based application development.

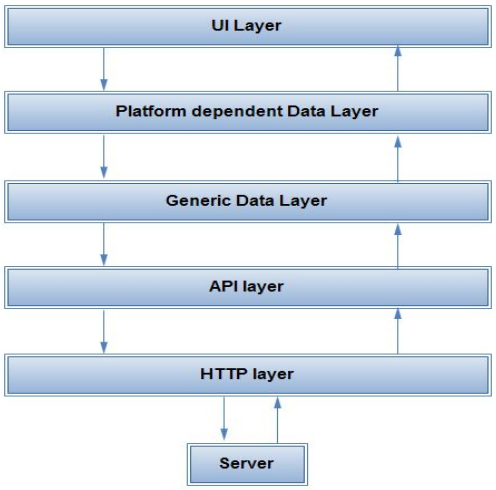

Fig. 3 Layered architecture

Figure 3 shows the layered approach for the android application development. The lowest level is HTTP layer which is responsible for sending HTTP get and post requests to the server and receiving the response. Next layer is API layer. This is for parsing the response from the server and formulating the query and passing it to the HTTP layer. The API layer gets the response string from the HTTP layer and parses the string. It also helps in extracting the necessary fields and passes it to the data layer. The Generic Data layer contains the components that include designing business layers and implementing functionalities like caching, exceptional management, logging and validation. Next is platform dependent data layer which takes the data from the API layer and use it. It stores the data in the platform dependent way. Some classes like Adapter, Listview etc store the data dependent on the platform. Last one the UI layer. This helps in showing the data to the user and manages user interactions. It has two components user interface components and user process components. User interface components provide a way for users to interact with the application. User process components synchronize and organize user interactions. UI layer is responsible for views in android. It has Views, buttons, layouts etc.

A. The application model

In Android's application model [1], an application is a package of components, each of which can be instantiated and run as necessary (possibly even by other applications). Components are of the following types [5]:

Activity components form the basis of the user interface; usually, each window of the application is controlled by some activity. **Service** components run in the background, and remain active even if windows are switched. Services can expose interfaces for communication with other applications. **Receiver** components react asynchronously to messages from other applications. **Provider** components store data relevant to the application, usually in a database. Such data can be shared across applications [3].

Consider, e.g., an online photo viewing application for an Android based phone. This application may have several components. There are activities for viewing the photos on the phone in the form of grid or list. There may be a service for downloading a photo in the background. There may be receivers for pausing a application when a call comes in, and for restarting the application when the call ends. The application should not affect the high priority functionality of the device like incoming call, incoming sms, battery low indication etc. Finally, there may be a provider for storing the photos and its details on the phone.

B. Component classes and methods

The Android SDK has a base class for each type of component (Activity, Service, Receiver, and Provider), with callback methods that are invoked at various points in the life cycle of the associated component. Each component has a life cycle. Each component of an application is defined by extending one of the base classes, and overriding the methods in that class. In particular:

- The Activity class has methods that are run when activity is created, or activity calls some other activity, or returns to the activity.
- The Service class has methods that are run when the service is started, or some component binds to this service or even combination of both.
- The Receiver class has a method that is run when a message is sent to this receiver.
- The Provider class has methods to delete, query and update the data stored by this provider.

C. Component classes and methods

The Google Android mobile phone platform is one of the most anticipated smartphone operating systems. Smart phones can be used in place of Computers/Laptops. As mobile devices attain increasing capabilities, there are many more opportunities for novel applications development. Recent development of mobile application development has reached a high demand on today's cellular market. Android defines a new component-based framework for developing mobile applications, where each application is comprised of different numbers and types of components. Activity components are the basis of the user interface; each screen presented to the user is a different Activity [6]. Service components provide background processing that continues even after its application loses focus. Content Provider components share information in relational database form. SQLite is embedded into android which supports relational database. For instance, the system includes an application with a Content Provider

devoted to sharing the user's address book upon which other applications can query. Finally, Broadcast Receiver components act as an asynchronous mailbox for messages from the system and other applications. As a whole, this application framework supports a flexible degree of collaboration between applications, where dependencies can be as simple or complex as a situation requires.

IV. ANDROID SECURITY FRAMEWORK

The Google Android mobile phone platform is one of the most anticipated smartphone operating systems. Smart phones can be used in place of Computers/Laptops. As mobile devices attain increasing capabilities, there are many more opportunities for novel applications development. Recent development of mobile application development has reached a high demand on today's cellular market. Android defines a new component-based framework for developing mobile applications, where each application is comprised of different numbers and types of components. Activity components are the basis of the user interface; each screen presented to the user is a different Activity [6]. Service components provide background processing that continues even after its application loses focus. Content Provider components share information in relational database form. SQLite is embedded into android which supports relational database. For instance, the system includes an application with a Content Provider devoted to sharing the user's address book upon which other applications can query. Finally, Broadcast Receiver components act as an asynchronous mailbox for messages from the system and other applications. As a whole, this application framework supports a flexible degree of collaboration between applications, where dependencies can be as simple or complex as a situation requires.

A. Android Application Sandbox

Sandboxes are often located within kernel space since access to critical parts of the OS can be realized [2]. The kernel is a very essential part of a system because it acts as bridge between hardware and software. One approach of sandbox systems is to monitor system and library calls including their arguments. This is often done through system call redirecting, also known as system call hijacking. System calls, short system calls, are function invocations made from user space into the kernel in order to request some services or resources from the operating system.

B. Static and Dynamic Analysis of Android Applications

Two common practices [4] for malware detection are Static Analysis and Dynamic analysis. Static analysis involves decompilation, decryption, pattern matching and system call analysis. In all these cases software is not being executed. Here, a common approach is filtering binaries by malicious patterns, called signatures. Another technique for malware detection is Dynamic analysis which involves running the system in controlled environment and monitoring its behavior. It involves monitoring file changes, network activity, processes and threads etc. A common approach to dynamic software analysis is Sandboxing. A sandbox can be defined as "an environment in which the actions of a process are restricted according to a security policy". In practice, this means that a sandbox is an instance of the target OS, which is isolated in a way that prevents malware from performing harmful actions. Since both techniques have certain disadvantages, Thomas Bl¨asing et al. [6] proposed a novel two-step analysis of Android applications, consists of a full-fledged kernel-space sandbox, and a fast static pre-check. AASandbox executes automatically, without any need for human interaction, and saves the logs of system calls and static analysis for further inspection. As an input, the AASandbox takes an Android application archive, which is packaged in a *.apk file and is therefore referred to as *APK*. Applications are written in Java and run in a Dalvik virtual machine. Application source code is first compiled to standard Java bytecode, and then optimized and converted to Dalvik executable format for being interpreted Dalvik VM. Byte code is then packaged together with other application resources, including UI layouts, localization and a manifest file which defines the structure of the application. The AASandbox first performs a static pre-check, followed by a full-blown dynamic analysis as shown in Fig 4.

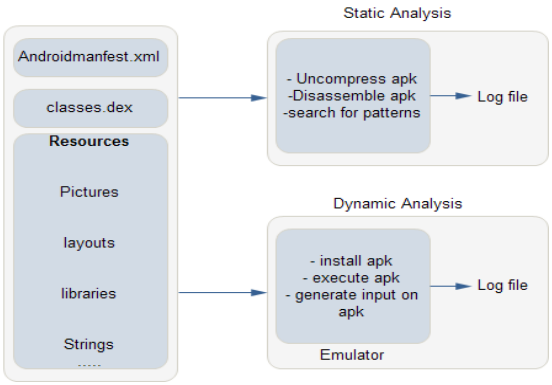

Fig. 4 Design of the Android Application Sandbox (AASandbox)

V. SECURITY ISSUES RELATED TO ANDROID PLATFORM

The integrity of the Android platform is maintained through a variety of security measures [3].

A. Applications as Operating System User

Each and every application is a user using the operating system. When an application is installed, the operating system

creates a new user profile associated with the application. Each application runs as a different user, with its own private files on the file system, a user ID, and a secure operating environment. The application executes in its own process with its own instance of the Dalvik VM and under its own user ID on the operating system.

B. Explicitly Defined Application Permissions

When an Android requires explicitly defined application permissions in the manifest file. To access shared resources on the system, Android applications register for the specific privileges they require. While developing the application required permissions should be specified in Android manifest file. For example if the phone vibration functionality is required then it must be specified in the android manifest file. While installing the application it shows the list of resources that the application is going to access. Some of these privileges enable the application to use phone functionality to make calls, access the network, and control the camera and other hardware sensors. Applications also require permission to access shared data containing private and personal information such as user preferences, user's location, and contact information. Applications might also enforce their own permissions by declaring them for other applications to use. The application can declare any number of different permission types, such as read-only or read-write permissions, for finer control over the application.

C. Limited Ad-Hoc Permissions

Content providers might want to provide some on-the-fly permissions to other applications for specific information they want to share openly. This is done using ad-hoc granting and revoking of access to specific resources using Uniform Resource Identifiers (URIs). URIs points to specific data assets on the system, such as MediaStore, Contacts, CallLog etc. Here is an example of a URI that provides the phone numbers of all contacts:

content://contacts/phones.

D. Application Signing for Trust Relationships

All Android applications packages are signed with a certificate, so users know that the application is authentic. The private key for the certificate is held by the developer. This helps establish a trust relationship between the developer and the user. It also allows the developer to control which applications can grant access to one another on the system. No certificate authority is necessary; self-signed certificates are acceptable.

VI. CONCLUSION

With the vigorous development through Android, mobile applications have been widely used on the various mobile devices. Android mobile applications are evolving at a meteor pace to give a rich and fast user experience. The maturity of the hardware and software platforms of mobile devices and the promotion of the Mobile Internet have brought a great opportunity to the migration of the web applications to mobile platforms. In case of the security, Static analysis scans the software for malicious patterns without installing it. Dynamic analysis executes the application in a fully isolated environment, i.e. sandbox, which intervenes and logs low-level interactions with the system for further analysis. Both the sandbox and the detection algorithms can be deployed in the cloud, providing a fast and distributed detection of suspicious software in a mobile software store akin to Google's Android Market. The ultimate goal is to protect the mobile applications from the malicious attributes and safeguard the interests of Android mobile users. With the mobile capabilities, the Internet connection capabilities and complete software platforms available, the future of mobile web application appear limitless.

VII. FUTURE WORK

The era of mobile web application has just started, and there is a long way for it to march. Development of mobile web application will be emphasized on following aspects:

1) More and more sensors will be added to mobile phones, so new APIs to use those capabilities will bring brand new applications to users.

2) Multimedia capabilities will be enhanced and engine will support more types of multimedia such as flash and svg .

3) The dedicated Integrated Development Environment (IDE) will be improved to accelerate the applications' development. Visualization programming and JavaScript debugging will be the most important functions of the IDE.

REFERENCES

[1] What is android? http://developer.android.com/guide/basics/what-is-android.html

[2] 3G Mobile Terminal Development Trend of the operating system [M/OL]. http://pda.c114.net/32/c4948.html, 2007

[3] Android Architecture 2010[R/OL]. http://www.cnmsdn.com/html/201003/1268713218ID2058_2 .html.

[4] Static detection of malicious code in executable programs by J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, Y. Lavoie, and N. Tawbi.

[5] Android Official Website (2008)—"Android | Official Website", <http://www.android.com/>.

[6] An Android Application Sandbox System for Suspicious Software Detection, by Thomas Bl¨asing, Leonid Batyuk, Aubrey-Derrick Schmidt, Seyit Ahmet Camtepe, and Sahin Albayrak

[7] www.blackhat.com