



**BACHELOR OF COMPUTER
APPLICATIONS
SEMESTER 4**

**DCA2203
SYSTEM SOFTWARE**

Unit 12

Android Operating System

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction	-	-	3
1.1	Learning Objectives	-	-	
2	Android Operating System	1	1	4-6
3	Android Architecture	1	2	7-10
3.1	Linux Kernel	-	-	
4	Android Architecture Libraries	-	3	11-14
5	Android Architecture Application Framework	-	4	15-16
6	Applications	-	5	17-18
7	Security and Permission	-	6	19-23
8	Summary	-	-	24
9	Glossary	-	-	25
10	Terminal Questions	-	-	25-26
10.1	Answers	-	-	
11	References	-	-	27

1. INTRODUCTION

In the previous unit, we learned Universal Plug and Play (UPnP) and briefed that UPnP architecture offers pervasive peer-to-peer network connectivity of PCs of all form factors, intelligent appliances, and wireless devices. Additionally, the investigation was completed, and its technical capabilities and advantages. The Android operating system, its technologies, and its widely used applications for mobile devices are all covered in this topic.

Android Operating System has become the most common Operating System (OS) used mainly in the Mobile devices. It is the software to run a mobile phone without the proprietary barriers, and it offers numerous advantages. The intent of Android technology is to accelerate innovation in mobile devices and offer consumers a richer, less expensive, and better mobile experience.

1.1 Learning Objectives:

After studying this unit, the learners should be able to:

- ❖ *Explain about the android OS*
- ❖ *List and describe various layers of Android architecture*
- ❖ *Discuss about the Linux kernel used for Android*
- ❖ *Describe about the libraries of android architecture*
- ❖ *Discuss about the application framework of android architecture*
- ❖ *Determine how application file built for android OS*
- ❖ *Discuss about the security aspects of android technology*

2. ANDROID OPERATING SYSTEM

Nowadays, it's easy to find mobile devices with various types of technologies and Mobile Operating Systems (OS) running on them. The types of Mobile OS that you may find are Symbian, iPhone, RIM's BlackBerry, Windows mobile, Linux, Palm webOS, Android, etc.

Although the mobile operating system has its own merits and demerits, Android is the first truly open and comprehensive platform for mobile devices. It is the software to run a mobile phone but without proprietary obstacles.

Android is a software stack for mobile devices that includes an operating system, middleware, and key applications. The Android Software Development Kit (SDK) provides the tools and Application Programming Interfaces (APIs) necessary to begin developing applications on the Android platform using the Java programming language. Android Inc was founded in Palo Alto of, California, U.S., by Andy Rubin, Rich miner, Nick sears, and Chris White in 2003. Later, Android Inc. was acquired by Google in 2005. After the original release, there have been several updates in the original version of Android. Android is now developed by Open Handset Alliance (OHA), a group of 84 technology and mobile companies that have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.

You must be wondering why we need to go for Android compared to other OSs. This is true because it offers a straightforward and effective SDK, and getting it doesn't involve paying any licensing, distribution, or development expenses. With Android, you can develop over many platforms like Linux, Mac OS, and windows. You can also get excellent documentation with the support of an active developer community. Android is built on the open Linux Kernel. Furthermore, it utilizes a custom virtual machine designed to optimize memory and hardware resources in a mobile environment. Android is open source; hence, it can be extended to incorporate new cutting-edge technologies as they emerge. The incorporate Java programming for the development of mobile device applications with the support of open-source libraries like SQLite, WebKit, OpenGL. You can import a 3rd party Java library to enhance the applications. The platform will continue to evolve as the developer community works together to build innovative mobile applications. Table 12.1 shows list of Android versions.

Name	Version number(s)
Apple pie	0.9
Banana bread	1.1
Cupcake	1.5
Donut	1.6
Eclair	2
	2.0.1
	2.1
Froyo	2.2 - 2.2.3
Gingerbread	2.3 - 2.3.2
	2.3.3 - 2.3.7
Honeycomb	3
	3.1
	3.2 - 3.2.6
Ice Cream Sandwich	4.0 - 4.0.2
	4.0.3 - 4.0.4
Jelly Bean	4.1 - 4.1.2
	4.2 - 4.2.2
	4.3 - 4.3.1
KitKat	4.4 - 4.4.4
	4.4W - 4.4W.2
Lollipop	5.0 - 5.0.2
	5.1 - 5.1.1
Marshmallow	6.0 - 6.0.1
Nougat	7
	7.1 - 7.1.2
Oreo	8
	8.1
Pie	9
Android 10	10
Android 11	11
Android 12	12

Table 12.1 List of android versions

History of Android:

It's interesting to learn about the evolution of Android. These include Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, and Lollipop. The code names for Android now span from A to J. Let's take a chronological look at the history of the Android.

- 1) In October 2003, Andy Rubin established Android Incorporation in Palo Alto, California, in the United States.

- 2) Google acquired android Incorporation on August 17, 2005. Since that time, it has been a subsidiary of Google Inc.
- 3) Andy Rubin, Rich Miner, Chris White, and Nick Sears are some of Android Incorporation's most important workers.

Android SDK Feature

The feature of Android SDK is that one can use it as a development kit for Global System for Mobile Communications, originally Groupe Special Mobile (GSM), Enhanced Data rates for GSM Evolution (EDGE), and 3G networks, WiFi, Bluetooth technologies etc. It provides rich libraries with media, SQLite, WebKit, SSL (Secure Sockets Layer) etc. WebKit is an open-source web browser engine. SQLite is a software library that implements a self-contained, serverless, transactional SQL database engine with zero-configuration. SSL is the standard security technology for establishing an encrypted link between a web server and a browser. It also provides hardware control with an accelerometer, compass, microphone, camera, GPS, touch screen, power, etc. You can also be supported with location-based service and map (e.g., Google API).

SELF ASSESSMENT QUESTIONS – 1

1. Android utilizes a custom virtual machine that was designed to optimize “_____” and _____ resources in a mobile environment.
2. With Android, you can go for development over many platforms like Linux, Mac OS, windows. (True/False)
3. Android is built on the open “_____” Kernel.

3. ANDROID ARCHITECTURE

The Android platform is built as a stack with various layers running on top of each other, lower-level layers providing services to upper-level services. Figure 12.1 outlines the current (layered) Android Architecture.



Fig 12.1: Android Architecture

Linux Kernel is based on version 2.6 and on top of its libraries are developed in C/C++ which expose the capabilities through application framework which are introduced by the applications. The modified Linux kernel operates as the Hardware Abstraction Layer (HAL) and provides device driver, memory management, process management, as well as networking functionalities, respectively. The library layer is interfaced through Java (which deviates from the traditional Linux design). It is in this layer that the Android specific libc (Bionic) is located. The surface manager handles the user interface (UI) windows. The Android runtime layer holds the Dalvik Virtual Machine (DVM) and the core libraries. Most of the functionalities available in Android are provided via the core libraries.

Also running as processes within the Linux kernel is the application (app) runtime. Each app runs in its own instance of the Android runtime, and the core of each instance is a Dalvik Virtual Machine (VM). The Dalvik VM is a mobile-optimized virtual machine, specifically designed to run fast on the devices that Android targets. Also present at this layer, and in each app's runtime, are the Android core libraries, such as the Android class libraries, I/O, and other similar things.

One of the critical components added by Android is Android run time. Dalvik Virtual Machine is part of Android run time optimized for mobile devices. Dalvik Virtual Machine Android run time also contains a core library which provides a substantial subset of Java 5 standard edition.

In the application framework layer, you can find the code compiled for and running on Dalvik VMs that provides services to multiple applications. Running at this level are entities such as the Package Manager, responsible for managing applications on the phone, and the Activity Manager, which is responsible for loading Activities and managing the Activity stack. The application framework houses the Application Program Interface API interface. In this layer, the activity manager governs the application life cycle. The content providers enable applications to either access data from other applications or to share their own data. The resource manager provides access to non-code resources (such as graphics), while the notification manager enables applications to display custom alerts. Anyone can write code that runs within the application framework; a good example is an application that shares out information to other applications as a Content Provider.

Finally, applications (like Home, dialer, SMS-MMS etc.) run at the top layer. This includes applications that a developer writes, or Google and other Android developers do. This layer is on top of the application framework, and they are the built-in as well as the user applications. It must be pointed out that a user application can replace a built-in application, and that each Android application runs in its own process space, within its own DVM instance. Typically, applications running at this layer include one or more of four different types of components. They are activities, Broadcast Receivers, Services, and Content Providers.

3.1 Linux Kernel

At the very bottom of Android architecture is the Linux kernel. Android relies on Linux version 2.6 for core system services. This Linux kernel is responsible for most of the things that are usually delegated to the operating system kernel and includes android's memory management programs, security settings, power management software and several drivers for hardware, file system access, networking and inter-process-communication. The kernel also acts as an abstract layer between the hardware and the rest of the software stack. This is the layer where all the device-specific hardware drivers will run viz. display driver, camera driver, Bluetooth driver, shared memory driver, binder (IPC) driver, USB driver, keypad driver, Wi-Fi driver and audio drivers.

Let us take a look at the different features of Linux Kernel:

- Open source: Software is open source and available for everyone to use, copy, and modify.
- Monolithic: It handles all hardware and driver operations
- Multitasking: It permits the simultaneous execution of numerous tasks.
- Modular: Loadable kernel modules can be added and removed at runtime.

Let us see the different functions that Linux Kernel perform:

- Functions
 - Device Drivers
 - Memory Management
 - Process Management

Let us understand the concept of Device Drivers:

The operating system and hardware components can communicate with one another using drivers.

Different manufacturers' hardware components must meet Android requirements in order to function with a device.

The different types of drivers are – USB driver, Bluetooth driver, Wi-fi driver, Display driver, Audio Driver, Power Manager, Flash Memory and Binder

Let us see how Linux Kernel performs memory management:

- The Linux Kernel makes sure there are no memory space conflicts between apps that are running simultaneously.
- All apps on the device are given enough memory to work, thanks to the Linux Kernel.
- No application uses more memory than is necessary thanks to the Linux kernel, which preserves memory.

Let us see how Linux Kernel manages different processes:

- A process can be started, stopped, paused, or killed by Linux Kernel.
- The Linux kernel makes sure that no other process will clash with the space assigned for one.
- The Linux kernel supports background processes as well as multitasking processes.
- After a process is finished, the Linux kernel frees up RAM.

SELF ASSESSMENT QUESTIONS – 2

4. In Android, the modified Linux kernel operates as the _____, and provides device driver, memory management, process management, as well as networking functionalities, respectively.
5. The Android runtime layer holds the “_____” “OBJ” and the “_____”.
6. Which of the following governs the application life cycle in application framework layer?
 - a) activity manager
 - b) window manager
 - c) package manager
 - d) resource manager

4. ANDROID ARCHITECTURE LIBRARIES

On top of the kernel are the libraries. Android includes a set of C/C++ libraries written as modules of code that are compiled down to native machine code. It controls the device to handle different kinds of data efficiently. These libraries tell the device how to handle different kinds of data and are exposed to Android developers via Android application framework. It provides some of the common services that are available for applications and various components of the Android system. It exposes its capabilities through Android application framework.

Libraries in Android architecture include the Surface Manager, Media Framework, WebKit, SQLite, OpenGL/ES, FreeType, SGL, SSL, Libc. These native libraries run as processes within the underlying Linux kernel.

Libc: Libc is a standard C library tuned for embedded devices.

SSL: It is a Secure Sockets Layer cryptographic protocol for secure internet communications.

SGL: SGL is a scalable graphic library or also called clear graphic library that gives an underlined 2D graphic engine. In Android graphics platform, you can combine 3D and 2D graphics in the same application.

OpenGL/ES: OpenGL/ES is a 3D library. OpenGL supports the CD graphics which are based on OpenGL1.0, so these libraries either use hardware 3D accelerator if they are available or use the highly optimized software. They have a software implementation that is hardware accelerable if the device has a 3D chip on it.

FreeType: Freetype is used for bitmap and vector font rendering. FreeType is a free, high quality and portable font engine.

WebKit: It is the open-source browser engine, used as a core of Android's browser. It acts as the web rendering engine that powers the default browser. It is the same browser that powers Safari from Apple and it renders well on small screens and on mobile devices.

SQLite: It is the basic datastore technology for the Android platform and is a very lightweight relational database engine that manages access to display subsystem. It is used as the core of most of its data storage.

Media Framework: The Media Framework was provided by Packet Video, one of the members of the open handset alliance and that contains the entire codex that make up the core of the media experiences. So, in there one will find all the audio and video codes that one need to build a rich media experience like IMPEG4, H.264, MP3, and AAC.

Surface Manager: The surface manager is responsible for composing different drawing surfaces on to the screen or you can say that it is responsible for graphics on the device's screen. It manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications. Surface manager seamlessly monitors the 2D and 3D graphics.

LibWebCore: LibWebCore is a modern web browser engine that gives us embeddable web view.

Media Libraries: It supports playback and recording of many popular audio and video formats, as well as static image files. Media libraries are based on PackerVideosOpenCORE. So, these libraries support playback as well as recording of many popular audio and video formats like MPEG4, H264, MP3 etc.

Android Runtime

The Android runtime layer also includes a set of core java libraries and DVM (Dalvik Virtual Machine) located in same layer. The Android runtime layer includes a set of base libraries that are required for java libraries. Every android application runs in its own process, with its own instance in the Dalvik Virtual Machine.

Core Libraries: Core Libraries are written in the Java programming language. The core library contains all the collection classes, utilities, I/O, all the utilities and tools that you use.

Dalvik Virtual Machine: Android based systems utilize their own virtual machine (VM), which is known as the Dalvik Virtual Machine (DVM). It is an extremely low memory based virtual machine, which was designed especially for Android to run on embedded systems

and work well in low power situations. It is also tuned to the CPU attributes. Applications for Android are developed in Java and executed in a virtual machine, called Dalvik VM. The DVM uses special bytecode, hence native Java bytecode cannot directly be executed on Android systems. The Dalvik VM creates a special file format (.DEX) that is created through build time post processing. Conversion between Java classes and DEX format is done by included “dx” tool. The Android community provides a tool (dx) that allows converting Java class files into Dalvik executables (dex). Dalvik executable or dex format is optimized for Android. The DVM implementation is highly optimized in order to perform as efficiently and as effectively as possible on mobile devices that are normally equipped with a rather slow (single) CPU, limited memory resources, no OS swap space, and limited battery capacity. The DVM has been implemented in a way that allows a device to execute multiple VM’s in a rather efficient manner. Every application runs in its own process if you have ten different applications, making ten different processes total. It also must be pointed out that the DVM relies on the modified Linux kernel for any potential threading and low-level memory management functionalities.

The Bionic Library

Compared to Linux, Androids incorporate its own C library known as Bionic library. The Bionic library is not compatible with Linux glibc. Compared to glibc, the Bionic library has a smaller memory footprint.

To illustrate, the Bionic library contains a special thread implementation that (i) optimizes the memory consumption of each thread and (ii) reduces the startup time of a new thread.

Android provides run-time access to kernel primitives. Hence, user-space components can dynamically alter the kernel behaviour. Only processes/ threads that do have the appropriate permissions are allowed to modify these settings.

Security is maintained by assigning a unique user ID (UID) and group ID (GID) pair to each application. As mobile devices are normally intended to be used by a single user only (compared to most Linux systems), the UNIX/Linux /etc/passwd and /etc/group settings have been removed. In addition (to boost security), /etc/services was replaced by a list of services (maintained inside the executable itself). To summarize, the Android C library is especially suited to operate under the limited CPU and memory conditions common to the

target Android platforms. Further, special security provisions were designed and implemented to ensure the integrity of the system.

SELF ASSESSMENT QUESTIONS - 3

7. In the Linux _____ layer all the device-specific hardware drivers will run.
8. Which of the following is a cryptographic protocol for secure internet communications?
 - a) Scalable Graphic Library (SGL)
 - b) Secure Sockets Layer (SSL)
 - c) FreeType
 - d) SQLite
9. Android based systems utilize their own virtual machine (VM) known as the "_____".



5. ANDROID ARCHITECTURE APPLICATION FRAMEWORK

In Android Architecture, Application framework is a set of basic tools with which a developer can build much more complex tools. It has several components and is used by all applications. Application Framework is written in Java language. It comes with mobile devices like Contacts or SMS box, or applications written by Google and any Android developer.

The developers can access all framework APIs and manage phone's basic functions like resource allocation, switching between processes or programs, telephone applications, and keeping track of the phone's physical location. The architecture is well designed to simplify the reuse of components. Application frame enables the reuse of various components. Libraries expose these capabilities to the application framework, enabling reuse of those components.

In the application framework, we have many components. Those components are Activity manager, Window manager, Content Providers, View System, Notification Manager, Package Manager, Telephony Manager, Resource Manager, Location Manager, etc.

Activity Manager: This is the component that actually manages the life cycle of the applications. It also maintains a common back stack so that applications that are running in different processes can have a smoothly integrated navigation experience.

Window Manager: The window manager manages Windows. It is mostly a java programming language abstraction on top of lower-level services provided by the surface manager.

Content providers: The work of Content provider in application framework is to enable applications access data from other applications. Since every application is running in its own classes, one application does not have access to the process area of another application but if they want to exchange data among each other, then they can achieve it through content providers. It allows applications to share their data with other applications.

View System: View System contains buttons and lists and all the building blocks of the User Interface (UI). It also handles things like event dispatching, and layout drawing.

Notification manager: Notification manager helps us to display custom alerts on status bar. This notification manager is primarily important in the case of broadcast receivers. Broadcast receiver helps us to display various times based alerts on statements.


Package manager: It is responsible for keeping track of what applications you have and what capabilities each of your applications have. Package manager essentially holds information about various applications loaded in the system. When any applications need information about other applications then package manager provides such information.

Telephony manager: It handles information about telephony. The telephony manager contains the APIs we use to build the phone application.

Resource manager: Resource manager gives us access to non-code resources such as localized strings or various others resources in the form of PNG, BMP as well as XML. Therefore, resource manager helps us to get access to those non-code resources. The resource manager stores local Strings, bitmaps, and layout file descriptions.

Location managers: It handles information about the location.

SELF ASSESSMENT QUESTIONS – 4

10. Which of the following in application framework enables applications to access data from other applications?
 - a) Activity manager
 - b) Window manager
 - c) Content provider
 - d) Resource manager
11. Which of the following handles things like event dispatching, layout drawing?
 - a) Activity manager
 - b) View System
 - c) Content provider
 - d) Resource manager
12. “_____”  gives us access to non-code resources such as localized strings or various other resources in the form of PNG, BMP as well as XML.

6. APPLICATIONS

At the top of Android Architecture is the application layer where you can find all the applications that are used by the final user. Android incorporates a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. By installing different applications, the user can turn his mobile phone into a unique, optimized and smart mobile phone. Underlying all applications is a set of services and systems along with a right 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. All applications are written using the Java programming language.

An application in an Android is a dot APK (.apk) file. When we compile and build our source code, then dot APK file is generated, which contains the whole application package. During the build process, Android projects are compiled and packaged into a .apk file, repository for all binary applications. It contains all the information necessary to run the applications on a device or emulator, such as compiled .dex files (.class files converted to Dalvik byte code), a binary version of the AndroidManifest.xml file, compiled resources (resources.arsc) and uncompiled resource files for the application.

The general process for a building .apk file is outlined below:

1. The Android Asset Packaging Tool (AAPT) takes your application resource files, such as the AndroidManifest.xml file and the XML files for activities, and compiles them. R.java is also produced so you can reference your resources from your Java code.
2. The aidl tool converts any .aidl interfaces that are into Java interfaces. Java code, including the R.java and .aidl files, are compiled by the Java compiler and .class files are output.
3. The dex tool converts the .class files to Dalvik byte code. Any 3rd party libraries and .class files you have included in your project are also converted into .dex files so they can be packaged into the final .apk file.
4. All non-compiled resources (such as images), compiled resources, and the .dex files are sent to the apkbuilder tool to be packaged into an .apk file.
5. Once the .apk file is built, it must be signed with either a debug or release key before it can be installed to a device.

6. Finally, if the application is being signed in release mode, you must align the .apk with the zipalign tool. Aligning the final .apk decreases memory usage when the application is running on a device.

Finally, when we get the .apk file, it is this APK file that is installed on mobile devices.

Applications of Android include a wide range of mobile applications. It includes:

- the ability for anyone to customize the Google Android platform.
- customizing a mobile phone using the Google Android platform
- features like weather details, opening screen, live RSS feeds, and even the icons on the opening screen to be customized.
- entertainment functionalities by Google Android can offer online real-time multiplayer games.

SELF ASSESSMENT QUESTIONS - 5

13. Java source code file passes through the Java compiler to get dot class files.
(True/False)
14. Android projects are compiled and packaged into an "_____" file, The repository for all binary applications
15. Which of the following tools converts the .class files to Dalvik byte code?
 - a) dex tool
 - b) DVM
 - c) Kernel
 - d) Zipalign

7. SECURITY AND PERMISSION

Android based smartphones are nowadays used for private and business applications, such as online banking or to access corporate networks. This makes them a very valuable target to be attacked to access important and secret data or use it for one's benefit. Mobile devices contain Contact Information, Location Data, Credentials, and Private Details. One can use Trojan software to access various information stored in devices. Espionage, communication eavesdropping, blackmailing, botnet formation, collecting valid email addresses for spam mailing and recording of sensitive information such as login credentials or credit card data are just some of the classical applications of trojan software. Even if substantial or large-scale attacks have so far been unsuccessful, they are growing more complex and effective. Thus, security is of paramount importance for both private and corporate users whoever is using devices with Android OS.

File System and User/Group Permissions

Android uses Linux as its underlying kernel and much of the Android security model is like Linux security model. In Linux, security is performed as per the concept of users and groups. Each user in a Linux system is assigned a user ID (UID) to differentiate one user from another. In addition, users can be added to groups and each group has a group ID (GID) to differentiate one group from another. A user can be a member of multiple groups, and each group can have multiple members.

Permissions are assigned to each resource on a Linux system, with a resource typically being a file. Each resource has a defined owner, which is the UID of the user that has primary responsibility for the file and can alter the permissions on it. Each resource also has a defined group, which is the GID of users with a set of permissions over and above that of the world, which is the group of all users on the system.

Similar to Unix/Linux operating systems, basic access control in Android is implemented through a three-class permission model. It distinguishes between the owner of a file system resource, the owner's group, and others. For these three entities, distinct permissions can be set to read, write, or execute. This system provides a means of controlling access to files and resources.

Example: only a file's owner may write (alter) a document, while members of the owner's group may read it, and others may not even view it. However, for mobile operating systems, this is not sufficient. Finer permission distinction is needed, as an open app market is not a strongly monitored and trustworthy software source. With the traditional approach, any app executed under the device owner's user ID could access any other app's data. Hence, the Android kernel assigns each app to its own user ID on installation. This ensures that an app can only access its own files, the temporary directory and nothing else – system resources are available through API calls.

When an Android package is installed, a new user ID (one that is not currently in use on the device) is created and the new app runs under that UID. In addition, all data stored by that application is assigned the same UID, whether a file, database, or other resource.

The Linux permissions on resources for that app are set to allow full permission by the associated UID and no permissions otherwise. Note that this UID is unique to the device; there is no guarantee (or even expectation) that the same UID will be used for the same application on different devices. Linux prevents applications that have different UIDs from accessing data, or otherwise accessing the process or memory of other applications, thus providing the basis for the separation between applications on the Android platform. This concept is known as the separation of concerns. Each app is well separated from others by default. Android introduces the capability for software components to run under the same user IDs and as part of the same processes.

Android Permission Basics

The Android permission model is based on the central construct of a permission. A permission is something that is granted to applications and required by APIs (or certain functions within code) to run. A simple example is the Android INTERNET permission; all networking APIs require a calling app to have this permission to execute the requested action of allowing network communications. Opening an outgoing network connection is something that not all applications need to do, and an activity that could cause the user to incur additional charges if they pay for the amount of data their device sends or receives (plus, this is the typical way an app would send sensitive data off the device to be exploited). So, opening a network connection is restricted by INTERNET permission.

Android applications request permission for . installation. All the permissions that an app requires are specified in the AndroidManifest.xml file. As per the app's manifest file, Android permissions are sometimes called Manifest permissions. The purpose of this manifest file is to present essential information about the Android system and describe various components of the application. It declares the permissions application must have to interact with other applications, so this is another new concept of permission. If your application wants to have access to the contact database, then you must ask for this permission in AndroidManifest.xml.

While the Android system specifies a great number of system permissions that are required to access certain portions of the default APIs, there is nothing to prevent an app from using custom permissions. The app defines a new permission and then runs with that permission. It provides an API (or methods) that require a calling process to have that permission, or both. The permissions system is a powerful construct meant to be used and extended by app developers outside the core Android developers.

An app needs to specify which permissions it requires in its manifest. The question then remains: how are these permissions granted? This is all handled when the app is installed. At that time, its AndroidManifest.xml file is read, and the list of all permissions it needs to run is parsed. The installer then generates a list of these permissions and prompts the user to review and approve them. This is an all-or-nothing process; either the user can choose to accept the list of permissions that the app claims it needs, and the app will then install, or the user can choose to reject the list, and the app's installation will fail. Once an app is installed, no further interaction with the user will occur to inform them that certain permission is being exercised or to confirm that the user still wants to allow the app to execute an action requiring a permission check. To confirm each app's permissions with the user only at install time, this design choice is a key design decision for Android's permission model.

Now is a good time to think about how and where permissions are enforced. While writing an app that only calls other APIs and services, you need to determine which permissions are required by the APIs and services your app calls and specify them in your manifest. You are the consumer of such services and need to ask permission. If, however, you include methods that other applications will call or other types of inter-process communication endpoints,

you are sitting on the other side of the table as a provider of services; you are the one who needs to grant permission. As a result, you'll need to add code to your app that verifies the permissions of any other apps calling you.

Android Permission Model

The permissions formally requested by an app at installation time through its manifest file do not have to match those that made use of it. Hence, it is completely impossible for a user to determine what activities an app performs with the permissions it has requested. There are several examples of actions which can be performed without proper permissions or without any permission at all.

Some of these actions are particularly valuable for malware purposes:

- **RECEIVE_BOOT_COMPLETED:** This is permission enabling an app to start at boot time. Malicious applications like malware may start automatically and run unnoticed.
- **START_ON_INSTALL:** this action enables an app to start up automatically right after installation.

These are the two permissions that may not be formally requested and presented to the user as permission.

The two types of Dangerous permissions required by the user are:

Install time permission:

- This is the authorization that the user must grant to install an application.

Run time permission:

- When the corresponding functionality is required, the user must provide these rights. Access to location and contact details, for instance

Next is Inter-Process Communication (IPC):

Using this technique, we may apply security rules based on how an application interacts with the other end. The following classes help the IPC process.

- IPC
 - Intent
 - Binder
 - Messenger

Let us understand what Intent is and what are its types:

It is the preferred method of data transfer for interacting with participants in events or broadcast listeners. It comes in two varieties.

- Intent
 - Explicit intent
 - Implicit intent

The two types of intents:

- Explicit intent
 - Designed to be received by an explicit component
 - It ensures that only application B receives data submitted from application A.
- Implicit intent
 - It specifies an action that needs to be performed
 - Used when a task must be assigned to a third-party program because it cannot be completed by the original application.

SELF ASSESSMENT QUESTIONS – 6

16. The Android permission model is based on the central construct of a _____.
17. All the permissions that an app requires are specified in the “_____”file.
18. The permissions formally requested by an app at installation time through its manifest file do not have to match those made use of. (True/False)

8. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- Android is developed by Open Handset Alliance (OHA), a group of 84 technology and mobile companies that have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.
- The feature of Android SDK is that one can use it as a development kit for GSM, EDGE, 3G networks, WiFi, Bluetooth technologies, etc.
- The Android platform is built as a stack with various layers running on top of each other, with lower-level layers providing services to upper-level services.
- Linux Kernel is based on version 2.6 and on top of its libraries are developed in C/C++ which expose the capabilities through application framework which are introduced by the applications.
- Android includes a set of C/C++ libraries written as modules of code that are compiled down to native machine code. It controls the device to handle different kinds of data efficiently.
- Every android application runs in its own process, with its instance in the Dalvik Virtual Machine.
- Compared to glibc, the Bionic library has a smaller memory footprint.
- Application framework is a set of basic tools with which a developer can build much more complex tools. It has several components and is used by all applications.
- An application in an Android is a dot APK file. When we compile and build our source code, a dot APK file is generated, containing the whole application package.
- Android uses Linux as its underlying kernel, and much of the Android security model is like as Linux security model.
- The Android permission model is based on the central construct of permission.

9. GLOSSARY

- **Applications:** Self-contained software designed for a mobile device and performing specific tasks for mobile users
- **Framework:** A platform for developing software applications, It is a foundation on which software developers can build programs for a specific platform.
- **kernel:** Is the central part of an operating system, manages the operations of the computer and the hardware, most notably memory and CPU time
- **Middleware:** Software that lies between an operating system and the applications running on it enables communication and data management for distributed applications.
- **OpenGL/ES:** OpenGL/ES is a 3D library, supports hardware 3D accelerator design.
- **Software Development Kit:** Collection of software development tools in one installable package, facilitating the creation of applications by having a compiler, debugger and perhaps a software framework.

10. TERMINAL QUESTIONS

Short Answer Questions

1. Write the features of Android SDK.
2. Draw the Android Architecture and briefly explain about each layer.
3. Briefly explain about the libraries of android architecture.
4. Briefly explain about the Application framework of android architecture.
5. How is an.apk files built? Write the steps.
6. What are manifest permissions? Explain.

10.1 Answers

Self-Assessment Questions

1. Memory, hardware
2. True
3. Linux
4. Hardware Abstraction Layer (HAL)
5. Dalvik Virtual Machine (DVM), core libraries

6. a) activity manager
7. Kernel
8. b) Secure Sockets Layer (SSL)
9. Dalvik Virtual Machine (DVM)
10. c) Content provider
11. b) View System
12. Resource manager
13. True
14. .apk
15. a) dex tool
16. permission
17. AndroidManifest.xml
18. True

Short Answer Questions

1. Features of Android SDK are that you can use it as a development kit for GSM, EDGE, and 3G networks, WiFi, Bluetooth technologies etc. (Refer to section 12.2 for more details)
2. Android architecture consists of Linux kernel, Libraries, Application Framework and Applications. (Refer to section 12.3 for more details)
3. In Android architecture, libraries are a set of C/C++ code written as modules and include the Surface Manager, Media Framework, WebKit, SQLite, OpenGL/ES etc. (Refer to section 12.5 for more details)
4. Application framework is a set of basic tools with which a developer can build much more complex tools. (Refer to section 12.6 for more details)
5. (Refer to section 12.7 for more details)
6. Android permissions, as per the app's manifest file, are sometimes called Manifest permissions. (Refer to section 12.8 for more details)

13. REFERENCES:

- Fedler, R., Banse, C., Christoph, K., & Fusenig, V. (2012). *Android OS Security: Risks and Limitations A Practical Evaluation*. Fraunhofer Research Institution AISEC.
- Six, J. (2011). *Application Security for the Android Platform*. O'Reilly.
- *framework*. (n.d.). Retrieved July 26, 2012, from developer.android:
<http://developer.android.com/guide/faq/framework.html>
- *Developers Android*. (n.d.). Retrieved 08 21, 2012, from Building and Running:
<http://developer.android.com/tools/building/index.html>

