Mobile Penetration Testing

Mobile Penetration Testing

Study Material

Quiz

Self Learning / Study Material

Mobile App Penetration Testing

Mobile App Penetration Testing

Chapter 1: Android Overview	Chapter 2: Android Architecture & OWASP Mobile Top 10
History and Features Of Android	Architecture Of Android
Android Application and API Level	Dalvik VM
Android IDE (Integrated Development Environment)	OWASP Mobile Top 10
	Android Rooting

Chapter 3: Android Application Sandboxing & Application Signing	Chapter 4: Authentication & Encryption
Application Sandboxing	Authentication (Cryptographic Key Storage, User Authenticators, Biometrics)
Application Signing (JAR signing, v2 signature scheme, v3 signature scheme)	Encryption (File Based Encryption, MetaData Encryption, Enabling Adiantum)

Chapter 5: GoogleBouncer & AndroidManifest.xml File	Chapter 6: Andriod Architecture & Rooting
GoogleBouncer (SecurityFeatures and Attacks(Delayed & Update) to bypass it)	Android Architecture
AndroidManifest.xml (Important File Contents present)	Android Rooting & Creating Emulator
Reverse Engineering	
Basic Command	

Chapter 7: IOS Architecture & Jailbreak	Chapter 8: Android&IOS Phone Penetration Testing
IOS Architectures	Basic Android Phone Penetration Testing's
Jailbreaking Exploit	Basic IOS Phone Penetration Testing's
IOS Data's Security	
IOS Network and Internet Security's	

History and Features Of Android

ANDROID OVERVIEW

Android is an operating system based on Linux and is an open source developed by Open Handset Alliance, which is led by Google and other companies. Android's first beta version was released in 2007 which was later made commercial in September 2008.

Android Application: -

Android applications are made in Java using Android Software Development Kit so a prequisite knowledge of Java is required.

Categories of Android applications include news, multimedia, games, music, food & drink, etc.

Android applications after getting developed one can sell them through Google Play Store, Opera Play Store, Amazon App Store etc.

ANDROID OVERVIEW

Features of using Android: -

1. GUI: Android uses a beautiful Graphical User Interface which is quite intuiting and attractive
2. Connectivity: Android's connectivity features include GSM, CDMA, 3G, LTE.
3. Messaging: Some of Android's messaging features includes SMS and MMS
4. Storage: Andoid's storage database that it uses is termed as SQLite.
5. Multi-Tasking: Multiple applications can run simultaneously so users can perform multiple tasks at the same time
6. Resizable Widgets: Widgets can be resized so users can view the content according to their comfort.
7. Wifi Direct: It lets apps directly pair over a high bandwidth peer to peer connection.

Android Application and API Level

ANDROID APPLICATION

Android applications are made in Java using Android Software Development Kit so a prequisite knowledge of Java is required.

Categories of Android applications include news, multimedia, games, music, food & drink, etc.

Android applications after getting developed one can sell them through Google Play Store, Opera Play Store, Amazon App Store etc.

API Level

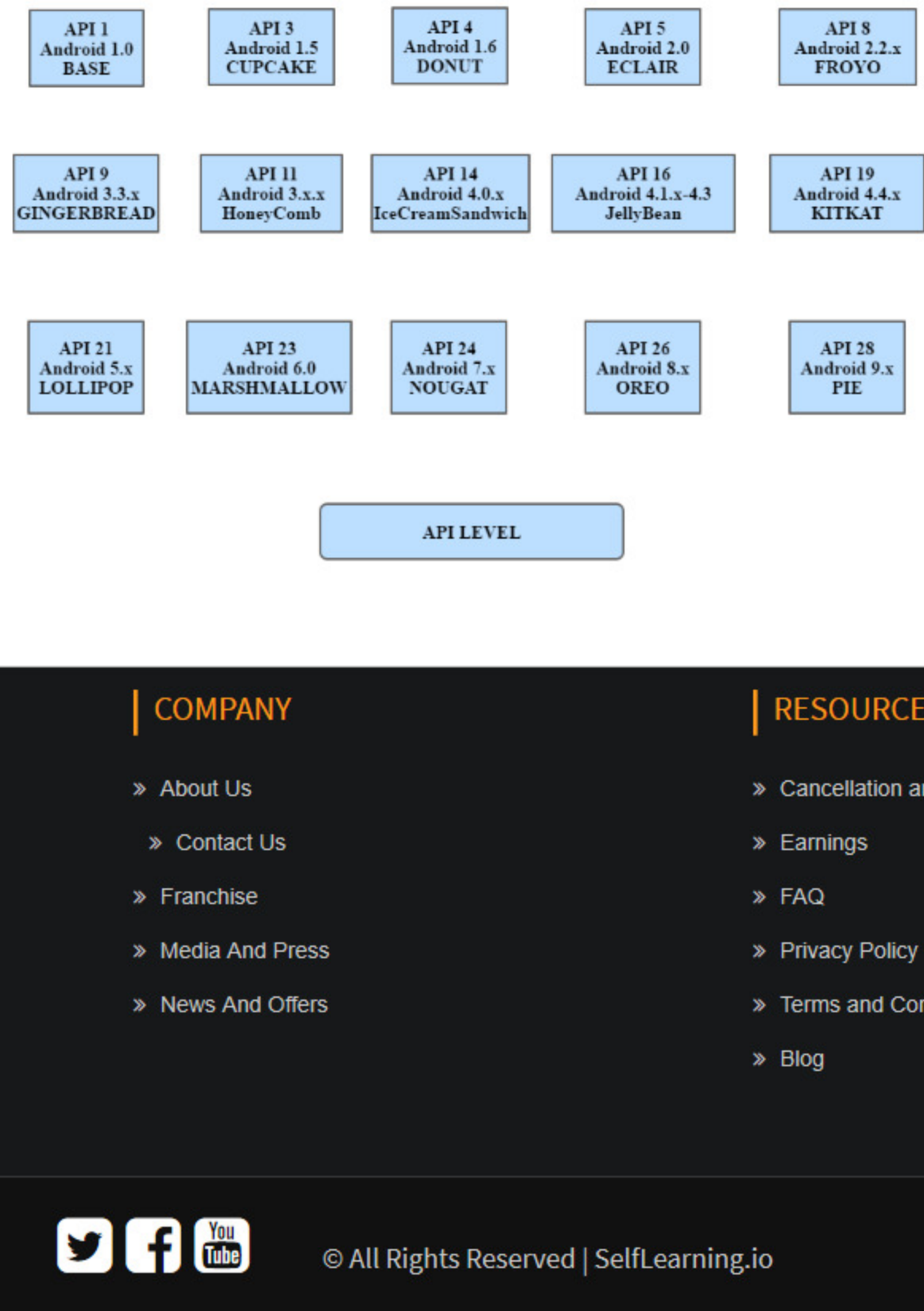
An API (Application Programming Interface) is a framework that makes the Android applications interact with the Android operating system. It consists of a core set of packages, XML elements and attributes for declaring a manifest file and accessing resources. It also consists of a set of permissions that an application is allowed and the permissions not allowed by the system.

An API Level is an integer value that describes the version of the Android platform being used.

API 1: Android 1.0: BASE

API 3: Android 1.5: CUPCAKE
API 4: Android 1.6: DONUT
API 5: Android 2.0: Eclair
API 8: Android 2.2.x: Froyo
API 9: Android 3.3.x: GingerBread
API 11: Android 3.x.x: HoneyComb
API 14: Android 4.0.x: IceCream Sandwich
API 16: Android 4.1.x-4.3: JellyBean
API 19: Android 4.4.x: KITKAT
API 21: Android 5.x: LOLLIPOP
API 23: 6.0: Marshmallow
API 24: 7.x: Nougat
API 26: 8.x: Oreo

API 28: 9.x: Pie



Android IDE (Integrated Development Environment)

IDE (Integrated Development Environment) is an environment to develop android applications.
Some of the most commonly used Android IDE's are: -

Android Studio

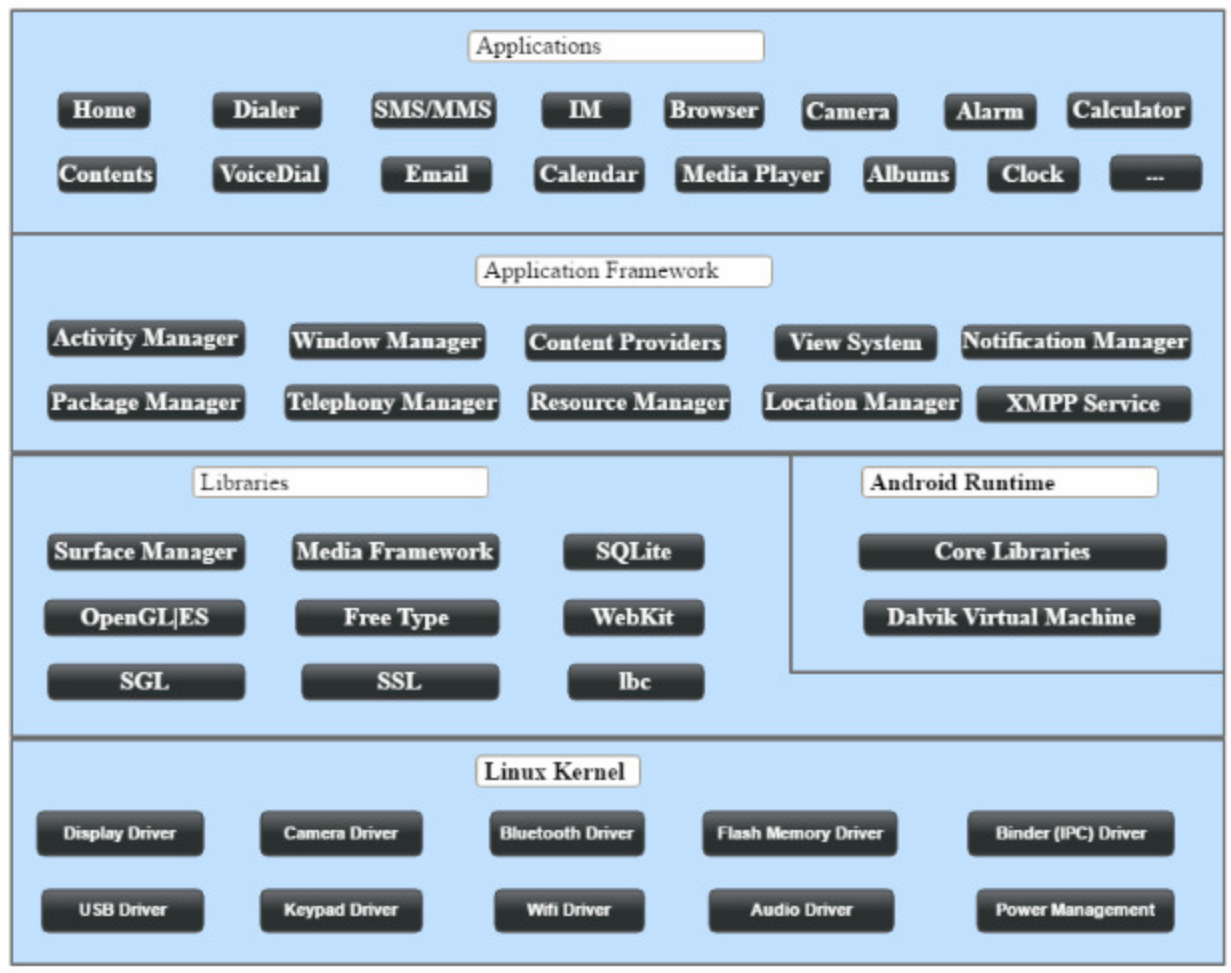
Visual Studio

Eclipse IDE

Visual Studio – Xamarin

Basic4Android

Architecture Of Android



Android Architecture consists of 4 main layers: Linux Kernel, Libraries, Application Framework, Applications

1.Linux Kernel: It consists of hardware drivers like Deploy driver, Camera driver, Flash Memory driver, Binder (IPC) driver, Keypad driver, Wifi driver, Audio driver, Power management

2.Libraries: The libraries section is divided into two types native libraries and Android Runtime libraries.

In native libraries it consists of Surface Manager, Media Framework, SQLite, Open GL|ES, Free Type, WebKit, SGL, SSL, lbc. Webkit describes the built in web browsing capabilities of Android applications.

In Android Runtime libraries consists of Core Library and Dalvik Virtual Machine. Dalvik VM makes use of it's own core Linux features for memory management and multi-threading for running it's own processes.

3. Application Framework: It consists of Activity Manager, Window Manager, Content Providers, View System, Package Manager, Telephony Manager,Resource Manager, Location Manager, Notification Manager

4. Applications: It is the top layer of Android architecture consisting of all the android applications like Home, Contacts, Phone, Browser, Games, Calculator, Alarm etc.

Dalvik VM

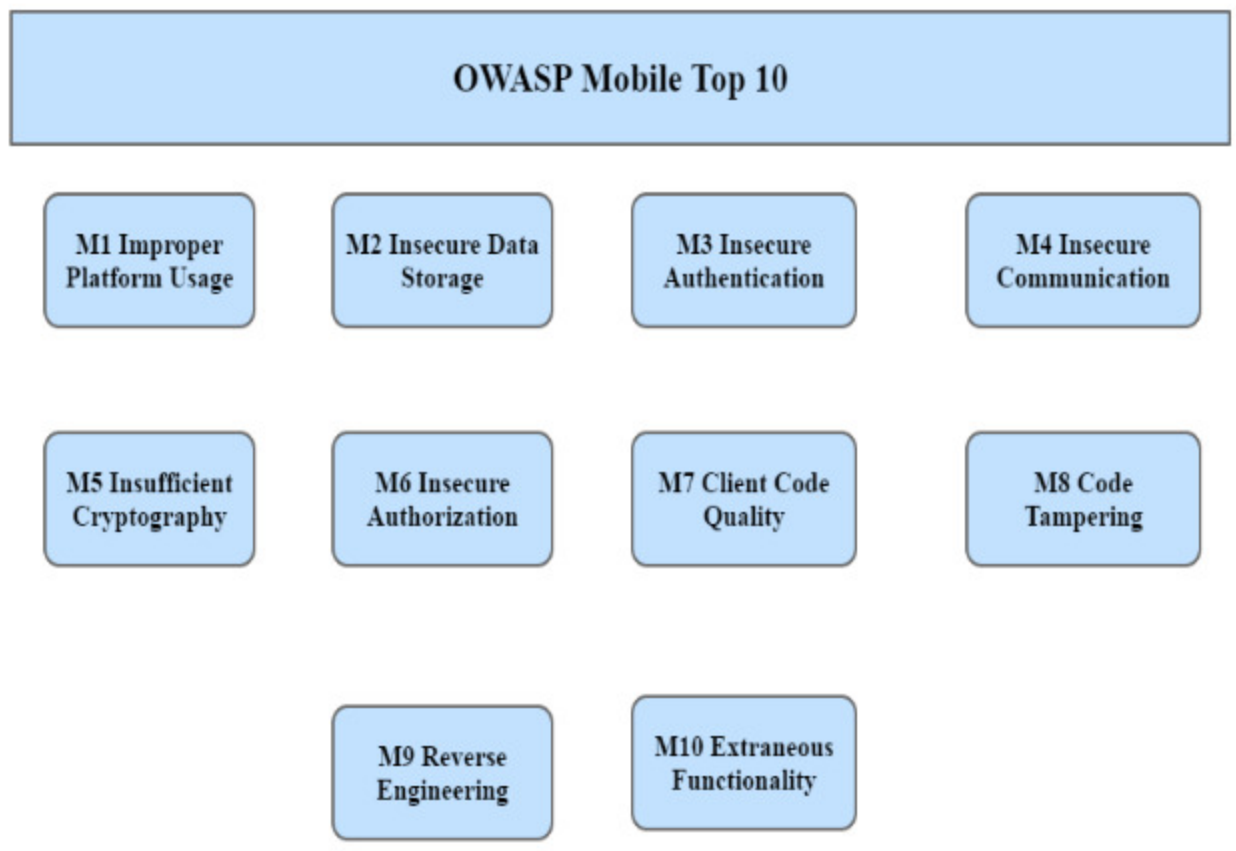
DVM is an optimised version of the JVM used for android devices. It has excellent memory management, and optimised battery features.

In DVM first the java source file is compiled into .class file using javac tool.

The .class file is further converted into .dex file using dx tool.

The Android Asset Packaging Tool (AAPT) handles the packaging process of android and creates .apk file

OWASP Mobile Top 10



Improper Platform Usage: -

It covers misuse of a platform failure or feature to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the KeyChain, or some other security control that is part of mobile operating system.

Insecure Data Storage: -

It covers insecure data storage and unintended data leakage.

Insecure Communication: -
It covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc

Insecure Authentication: -
This category captures notions of authenticating the end user or bad session management including failure to identify the user at all when should be required, failure to maintain user's identity when required and weakness in session management.

Insufficient Cryptography: -
The code applies to a sensitive information asset. However, the cryptography is insufficient in some way. To note that everything related to SSL/TLS goes to insecure communication and if the app fails to use cryptography at all when it is required that goes under Insecure Data Storage. This category is for issues where cryptography is attempted but it wasn't done correctly.

Insecure Authorization: -

It relates to any failures in authorization (eg authorization decisions in the client side, forced browsing, etc). It is distinct from authentication issues (eg, device enrolment, user identification, etc).

If the app does not authenticate users at all in a situation where it should (eg, granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.

Client Code Quality: -

This was the "Security Decisions via untrusted inputs", one of our lesser used categories. This would be the catch all for code level implementation problems in the mobile client. That's distinct from server side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code level mistakes where the solution is to rewrite some code that's running on the mobile.

Code Tampering: -

This category covers binary patching, local resouirce modification, method hooking, method swizzling, and dynamic memory modification.

Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct way of subverting the intended use of the software for personal or monetary gain.

Reverse Engineering: -
This category includes analysis of the final core binary to determine it's source code, libraries, algorithms, and other assets. Softwares such as IDA Pro, Hopper, otool, and other binary inspection tool gives the attacker insight into the inner workings of the application.

Android Rooting

Rooting allows all users to run admin privileged level commands which were not allowed in stock configuration. Rooting allows users to delete system files, remove pre-installed applications.

Few key terms to be known: -

Bootloader: Bootloader loads and starts bootloading tasks and processes.

Sideloading: It simply means transferring a media file to a mobile device via USB, Bluetooth or Wifi

Overcloaking: It simply means increasing the frequency of processor to the maximum level thereby increasing the performance of your android device.

Bloatware: It is a software using large amounts of unnecessary features, memory and RAM.

Flashing: It means installing something on your device

Bricking: It means breaking during flashing or other acts

Advantages of rooting: -

Complete control of the feel and look of the device

Complete control of the kernel

Complete control of applications allowing us to remove bloatware.

It allows us to install custom hardware or software increasing the features of the device.

Methods Of Android Rooting: -

We can root Android from terminal using adb (Android Debug Bridge) commands.
C:\AndroidExploit\bin> adb push exploit /data/local/tmp
C:\AndroidExploit\bin> adb shell
$ls -l
$./exploit root
C:\AndroidExploit\bin> adb shell
We should see uid = 0 (root)

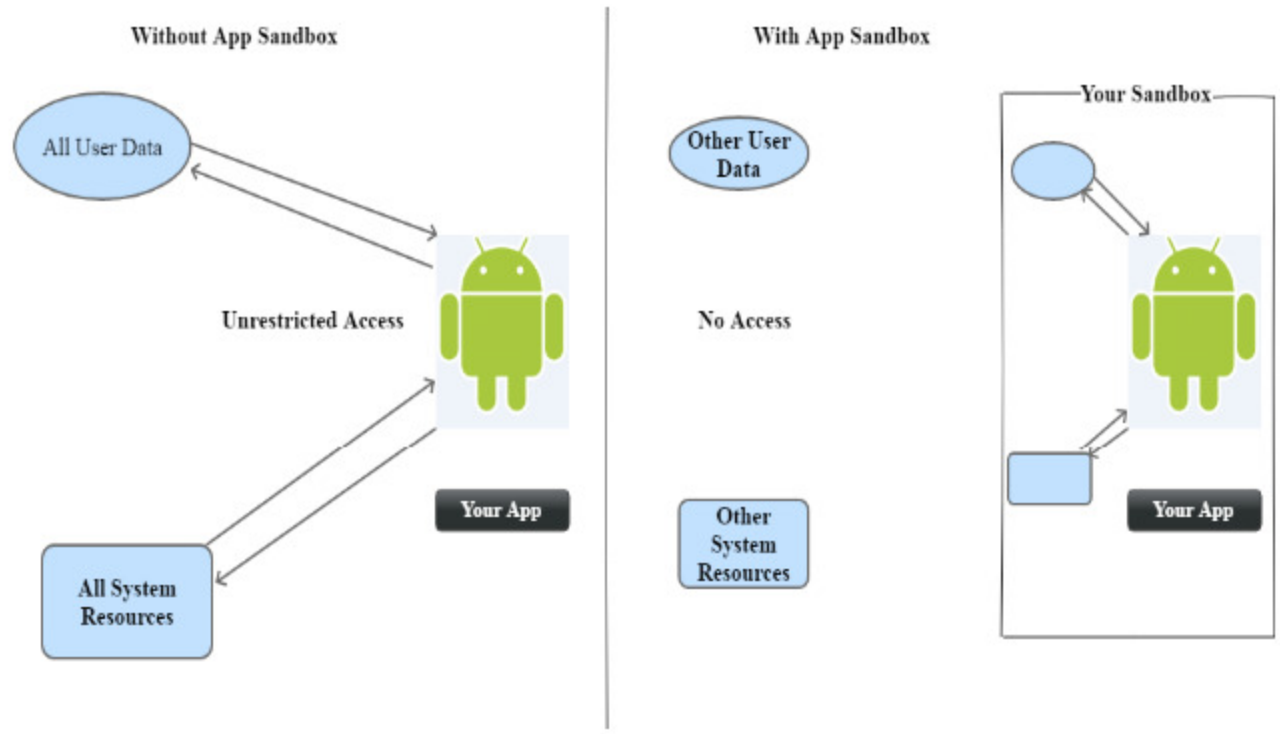
Another method of rooting called "systemless root" uses various techniques to root the Android device without changing the system partition. An example of it is Magisk.

Creating Emulator: -

An emulator can be created by installing the Santoku OS in the system. Santoku is a customized OS created from Ubuntu based OS specifically for testing of Android based applications.

Another way is to use Genymotion in the system in which Android OS can be created and then testing of Android applications can be done.

Application Sandboxing

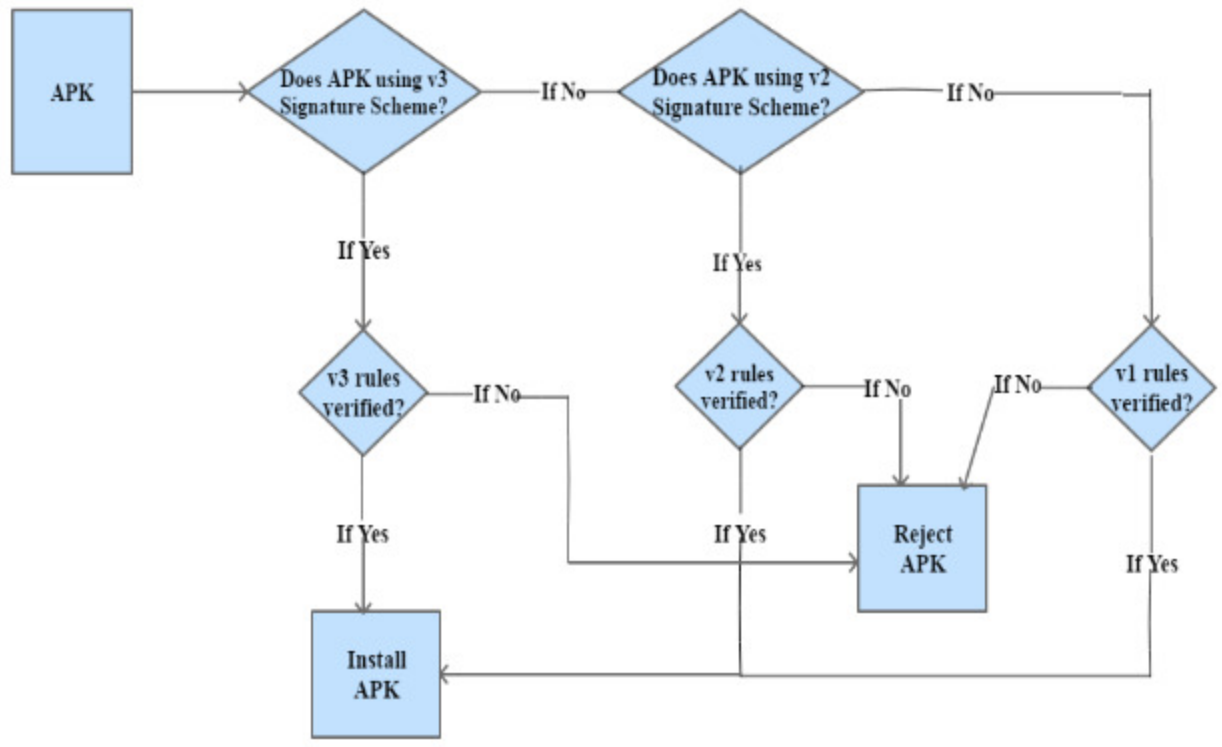


Android Sandboxing is a security feature that isolates android application data and code execution from other android applications. To perform sandboxing, the user assigns a unique ID (UID) to each application and runs it by it's own processes.

Android uses this UID to setup a kernel level application sandbox. The security benefit of using this application sandboxing is that if one application is trying to do something malicious of another application, the operating system will prevent it because the first application didnt get appropriate user privileges.

Since application sandbox is in the kernel so all the layers above the kernel like libraries, application framework, etc run within the android application sandboxing.

Application Signing (JAR signing, v2 signature scheme, v3 signature scheme)



Android Application Signing is the process that helps developers of the application to identify the legitimate author and hence update the APK. Applications that are attempted to be installed without being signed are either rejected by the package installer services such as Google Play, etc.Android application signing is the first step of placing an Android application in an android application sandbox. Different android applications have different User ID (UID). This prevents one application from gaining access to another application.When the Android application is installed in the android device, the package manager verifies the certificate if it is properly set or not. The certificate contains the public key used to match the key used to sign in any other APK.

Android applications can be self-signed or signed by a third party. Android application signing can be done by three methods JAR Signing, v2 Signature Scheme and v3 Signature Scheme.

JAR Signing (v1 Signature Scheme): -

It is based on signed JAR. It is not completely secure as they do not secure all of the Android APK such as zipped metadata. It offers a sizeable attack surface. The APK verifier must uncompress all compressed entries, consuming more time and memory.

To address these issues in Android 7.0, v2 Signature Scheme was introduced.

To address these issues in Android 7.0, v2 Signature Scheme was introduced.

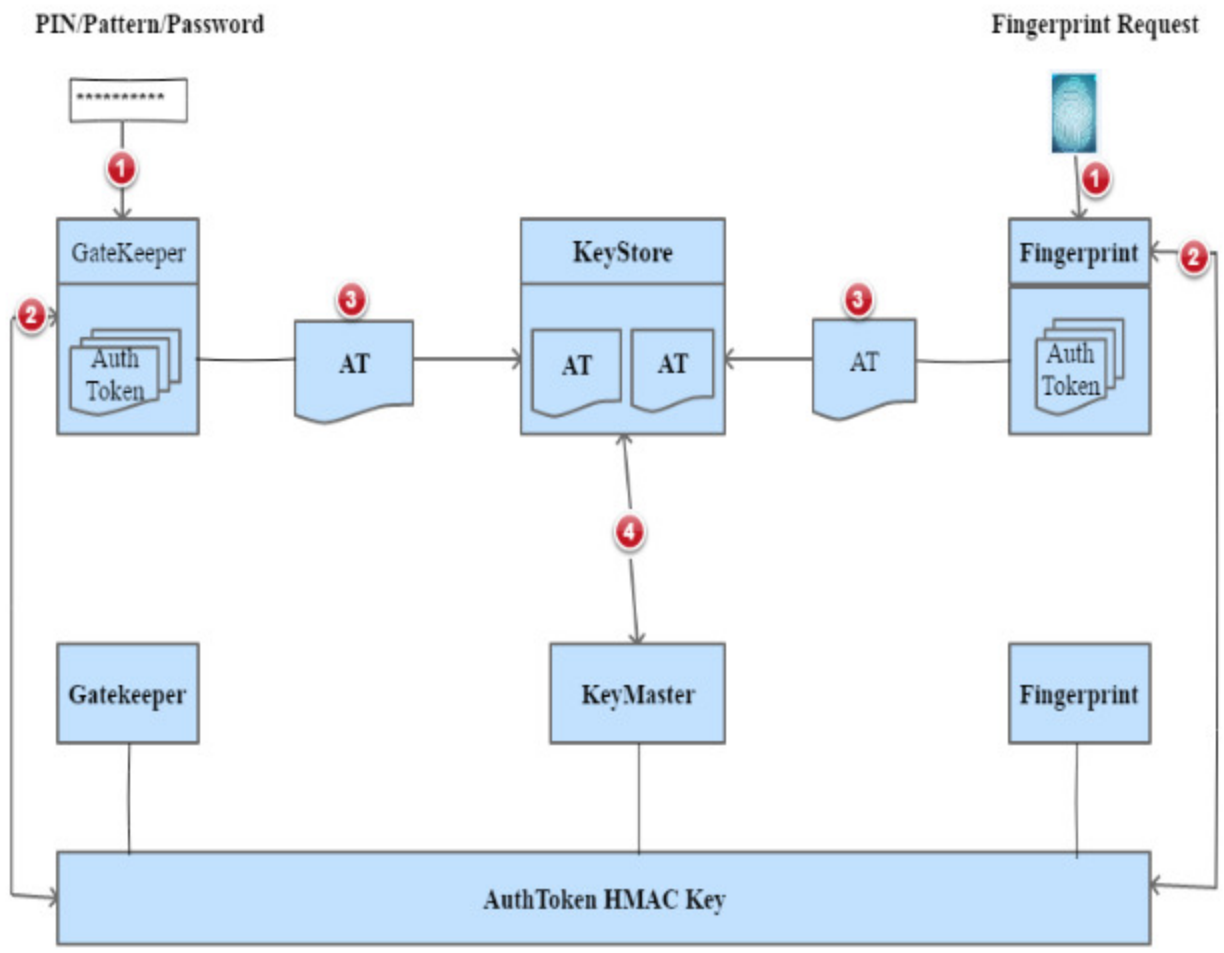
APK Signature Scheme v2: -

Android v2 Signature Scheme provides more security features maintains integrity in application signing. It was introduced when Android 7.0 was launched.

In v2 Signature Scheme it considers the whole file as a single file, also known as Binary Large Object (BLOB) and if it finds any zipped metadata not getting matched, it is invalidates the signing. Due to these features it increases the installation time of the APK.

APK Signature Scheme v3: -
It is supported by Android 9. It enables the feature of Android Key Rotation, which gives the Android apps the ability to change their signing key as part of their update. V3 scheme also adds information about the SDK versions and a proof of rotation present in the signing block.

Authentication (Cryptographic Key Storage, User Authenticators, Biometrics)



Android supports the concept of user authentication generated cryptographic keys consisting of Cryptographic Key Storage and User authenticators.

Cryptographic Key Storage: -
Android stores cryptographic keys using features such as hardware-backed Keystore and Keymaster

User Authenticators: -
Android uses Gatekeeper for password/pin/pattern authentication and Fingerprint for fingerprint authentication.

Biometrics:-

Android versions 9 and higher supports BiometricPrompt API that allows developers to integrate Biometric authentication into their applications in a device. Biometric Based Unlock security are mostly measured on the basis of False Accept Rate (FAR). FAR is a metric that mistakenly accepts randomly chosen incorrect inputs.

Encryption (File Based Encryption, MetaData Encryption, Enabling Adiantum)

Trusty Trusted Execution Environment (TEE)

Trusty is a secure Android OS having the same processor that Android uses but is isolated from the rest of the system in terms of hardware and software. The isolation feature of this OS prevents malicious apps from getting installed and hence increases the security strength.

The Trusty repositories can be downloaded and installed from Android Open Source Project (AOSP)

Encryption

It is the process of encoding all user data on an Android device using symmetric encryption keys. The encryption maintained in Android are of two types File Based Encryption and Full Stack Based Encryption.

File Based Encryption: -
In this type of encryption, different types of files can be encrypted with different keys and unlocked independently. Android 7.0 and higher versions supports this encryption. It also supports Direct Boot which allows the devices to boot directly to the lock screen.

MetaData Encryption: -

When File Based Encryption is used, information such as file data, design layout, permissions are not encrypted. These information are called filesystem metadata.

It was introduced in Android 9.0, metadata encryption encrypts file based encryption information as well as metadata information.

In Metadata Encryption, a single key is present at boot time which encrypts metadata information. The key is protected by KeyMaster which is again protected by verified boottime.

Enabling Adiantum: -

Adiantum is an encryption mechanism in Android OS that is used on those Android devices whose CPU lacks AES instructions.

Due to lack of AES instructions, providing Adiantum encryption on those devices provides lesser overhead.

GoogleBouncer (SecurityFeatures and Attacks(Delayed & Update) to bypass it)

Google introduced a new feature called Bouncer to keep malicious apps from Google Play Store. Bouncer automatically scans malicious apps in Play Store and developer accounts for finding something suspicious in them.

It does scanning with the help of reputation engine and cloud infrastructure.

But as every other security feature has some flaws in it, Bouncer too has a critical which is that it can be fingerprinted. Bouncer uses QUME software that can emulate hardware platforms. Bouncer only does dynamic analysis.

Due to all these security loopholes in Bouncer many of the attacks can bypass Google Play Store's security check. Attacks such as Delayed attack, Update attack, etc are used to bypass Play Store's security check.

Delayed Attack: In this attack the application contains malicious payloads that looks legitimate when running in Bouncer. After it gets into the user's device, it is only then that the payloads gets activated at runtime.

Update Attack: In this attack, no malicious payload need to be included in Bouncer's detection but can download contents from it's remote command & control center (C&C) to upload stolen data or receive further commands.

AndroidManifest.xml (Important File Contents present)

AndroidManifest.xml is a file in android which contains essential information about android app to the android build, android OS and many more details.
The Manifest.xml file declares the following: -

It contains android package's name, it's build tools and location of code entities.

The components of the app which includes activities, services, content providers, broadcast receivers, etc

The permissions that the app needs in order to access protected parts of system or apps.

The hardware and software features the app requires which affects which devices can install the app from Google.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.androidapp.basicElements"
    android:versionCode="1"
    android:versionName="1.0"
    <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".BasicElements"
        android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN />
        <category android:name="android.intent.category.LAUNCHER />
    </intent-filter>
    </activity>

    </application>
    <uses-sdk android:minSDKVersion="2" />
</manifest>
```

Reverse Engineering

Reverse Engineering is the process of converting the apk file into reusable to perform static analysis of the code or during run time using dynamic analysis or a combination of both.
For this we use tools such as adb, dex2jar, JD-GUI, apktool.

ADB (Android Debugging Bridge) is a versatile command line tool used for installing and debugging of android apps.

Dex2jar is used to convert .dex file into .java jar files.

JD-GUI is used to open up the .jar file.

Apktool is a reverse engineering 3rd party tool to decode resources to original form and then rebuild them after making some modifications.

Basic Command

To install Android apk we use adb: -
santoku@santoku# adb install appname.apk

To convert apk file to jar we use dex2jar: -
santoku@santoku# dex2jar appname.apk

To get Android Manifest.xml file and smali code we use apktool: -
santoku@santoku# apktool d appname.apk

To see the logs we use logcat: -
santoku@santoku# adb logcat

To go to the shell we use again adb: -
santoku@santoku# adb devices
santoku@santoku# adb shell

To unzip an Android apk we use: -
santoku@santoku# unzip appname.apk

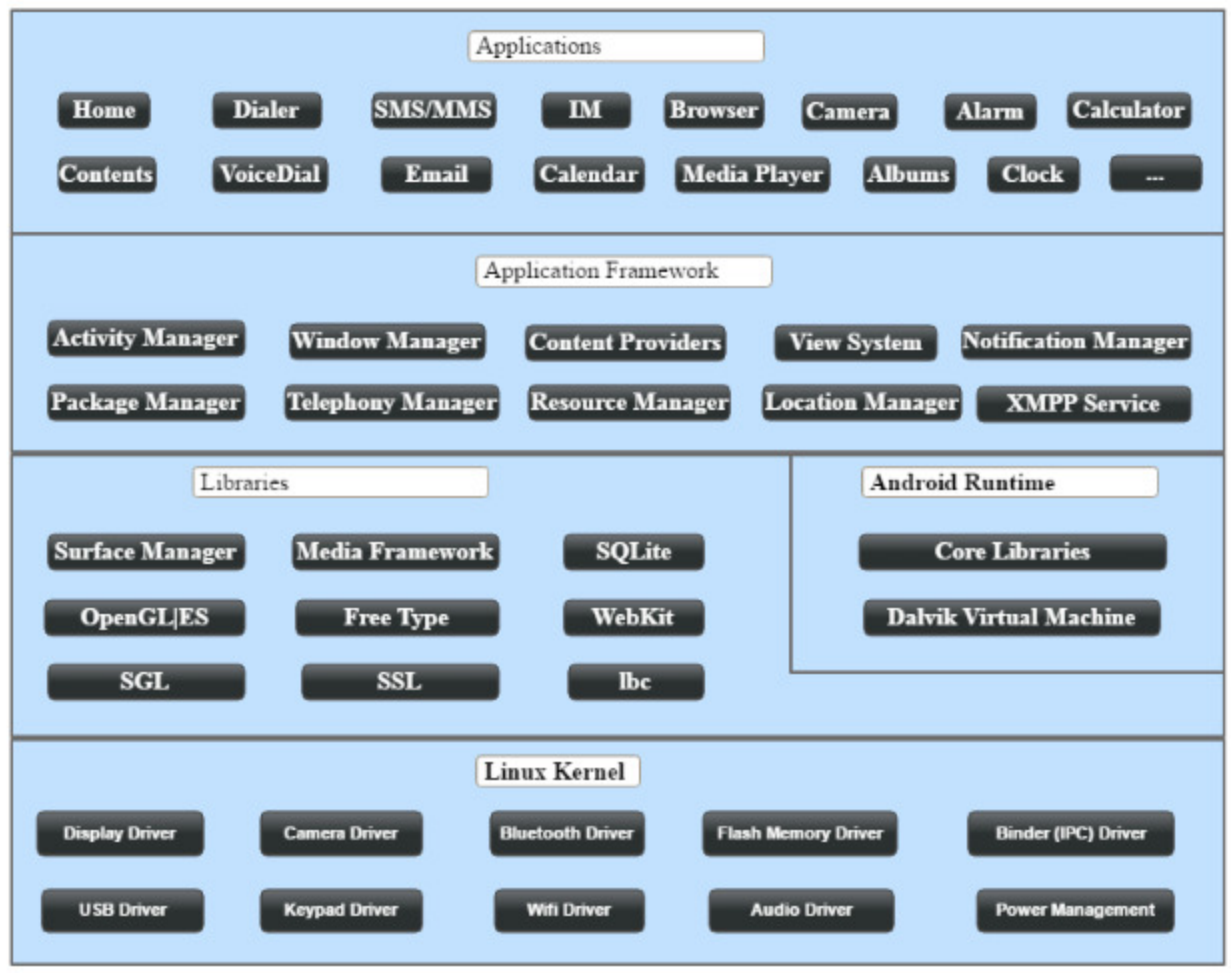
To know the type of file we use the file command: -
santoku@santoku# file filename

To dump the database we use sqlite command: -
santoku@santoku# sqlite3 filename

To access information from outside the app we can use the following command: -
santoku@santoku# adb shell am start filename/.directoryname
Am: It is Activity Manager tool
Start: It is used to launch an activity

To find strings used we can use the grep command: -
santoku@santoku# grep -lr 'content://'*

Android Architecture



Android Architecture consists of 4 main layers: Linux Kernel, Libraries, Application Framework, Applications.

1. Linux Kernel: It consists of hardware drivers like Deploy driver, Camera driver, Flash Memory driver, Binder (IPC) driver, Keypad driver, Wifi driver, Audio driver, Power management

2. Libraries: The libraries section is divided into two types native libraries and Android Runtime libraries.In native libraries it consists of Surface Manager, Media Framework, SQLite, Open GL|ES, Free Type, WebKit, SGL, SSL, lbc. Webkit describes the built in web browsing capabilities of Android applications.

In Android Runtime libraries consists of Core Library and Dalvik Virtual Machine. It makes use of it's own core Linux features for memory management and multi-threading for running it's own processes.

3. Application Framework: It consists of Activity Manager, Window Manager, Content Providers, View System, Package Manager, Telephony Manager, Resource Manager, Location Manager, Notification Manager

4. **Applications:** It is the top layer of Android architecture consisting of all the android applications like Home, Contacts, Phone, Browser, Games, Calculator, Alarm etc.

Android Rooting & Creating Emulator

Android Rooting :-

Rooting allows all users to run admin privileged level commands which were not allowed in stock configuration. Rooting allows users to delete system files, remove pre-installed applications.
Few key terms to be known: -

Bootloader: Bootloader loads and starts bootloading tasks and processes
Sideloading: It simply means transferring a media file to a mobile device via USB, Bluetooth or Wifi
Overcloaking: It simply means increasing the frequency of processor to the maximum level thereby increasing the performance of your android device.
Bloatware: It is a software using large amounts of unnecessary features, memory and RAM.
Flashing: It means installing something on your device
Bricking: It means breaking during flashing or other acts

Advantages of rooting: -
Complete control of the feel and look of the device
Complete control of the kernel
Complete control of applications allowing us to remove bloatware.
It allows us to install custom hardware or software increasing the features of the device.

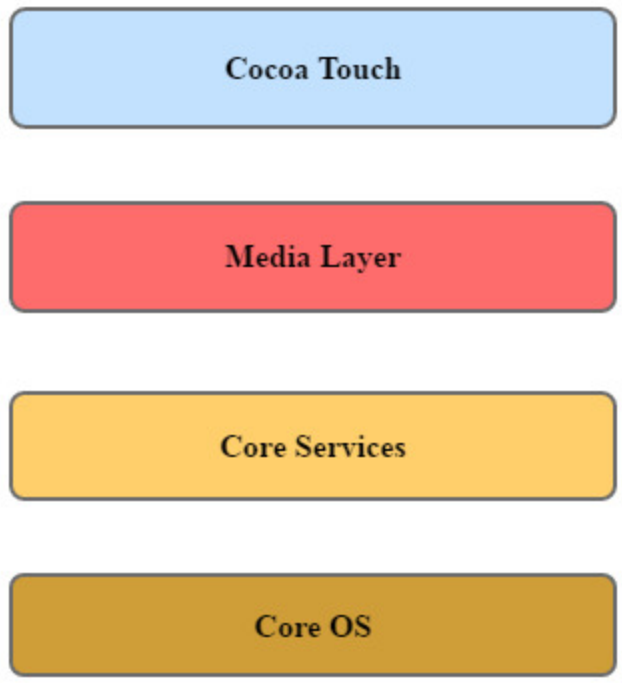
Methods Of Android Rooting: -

We can root Android from terminal using adb (Android Debug Bridge) commands.
C:\AndroidExploit\bin> adb push exploit /data/local/tmp
C:\AndroidExploit\bin> adb shell
$ls -l
$./exploit root
C:\AndroidExploit\bin> adb shell
We should see uid = 0 (root)

Another method of rooting called "systemless root" uses various techniques to root the
android device without changing the system partition. An example of it is Magisk.

Creating Emulator: -
An emulator can be created by installing the Santoku OS in the system. Santoku is a customized OS created from Ubuntu based OS specifically for testing of Android based applications.
Another way is to use Genymotion in the system in which Android OS can be created and then testing of Android applications can be done.

IOS Architectures



Cocoa Touch: It is at the highest level of iOS layers. It consists of StoryBoards, Documents, Gesturing, MultiTasking, Notifications, UIKit Framework. It provides an abstraction layer of iOS.

Media Layer: It defines the entire multimedia architecture within Apple powered mobile devices and frameworks. It includes Graphic Technologies, Audio Technologies, Video Technologies, AirPlay.

Core Services Layer: It provides important services to apps but have no direct bearing on application's user interface. It consists if iCloud, In-App purchases, SQLite, Core Data, Core Location.

Core OS Layer: It provides low level services related to hardwares and networks. It includes BlueTooth, External Accesories, Accelerator Framework. The services are mostly based on Kernel and Device Driver's layer.

Jailbreaking Exploit

Jailbreaking can be defined as a process of installing a modified set of kernel patches that allows users to run third party applications not signed by OS vendor.

It provides root level access of the operating system and permits downloading of third party applications, themes, extensions on an iOS devices.

It removes sandbox instructions, enabling malicious applications to get access to restricted mobile resources and information. Types of jailbreaking: Tethered, Semi-Tethered and Untethered

Types Of jailbreaking exploits: -

1. Userland Exploit: It allows user-level access but does not allow iboot-level access.

2. iBoot Exploit: An iBoot jailbreak allows user-level and iboot-level access.

3. Bootrom Exploit: It allows user-level access and iboot-level access.

Previous Next

Study Material

Quiz

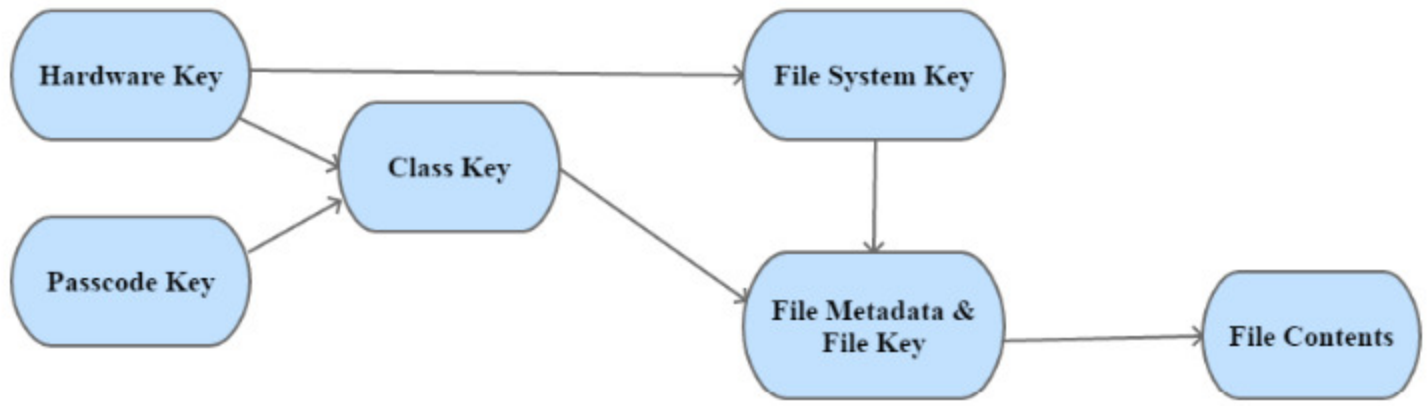
Mobile App Penetration Testing

Chapter 1: Android Overview

History and Features Of Android

Android Application and API

IOS Data's Security



iOS Data Security

Hardware Security Features: -

It has a dedicated AES-256 crypto-engine built into Directed Memory Access (DMA) path between the flash storage and main system memory making encryption more efficient. Each device has a userID(UID) and groupID(GID) which are 256 bit keys. They are created and stored in application processor directly, no hardware or software can access them directly.

File Data Protection: -

It protects the file data by constructing and managing a hierarchy of keys in conjuction with hardware encryption engine which will use per-file key to encrypt the file. The per-file key is wrapped with corresponding class keys. The AES engine decrypts the file contents read from the flash storage.

Passcodes: -

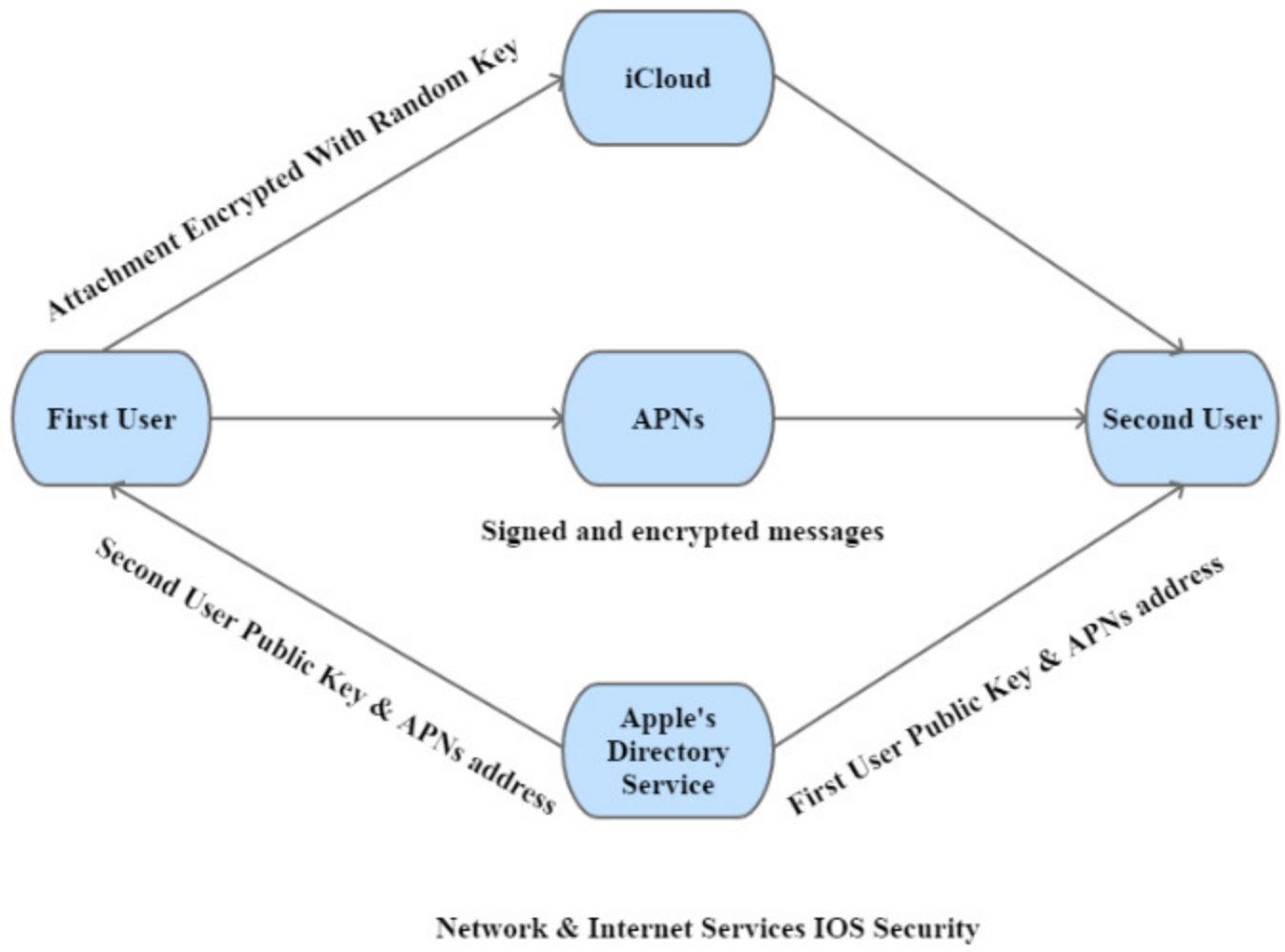
It is an important element to iOS security, by setting up Passcode, it is automatically enabled by iOS. It is used for generating encryption keys hence stronger your passcode stronger your encryption keys are generated.

KeyChain Security: -

IOS keychain item contains metadata such as creation/modification of time stamps and access group of keychain. All these contents are encrypted using AES 128 in Galois/CounterMode(GCM).

Previous Next

IOS Network and Internet Security's



Network & Internet Services IOS Security

SSL/TLS: -

IOS provides APIs such as CFNetwork and SecureTransport so that developers can maintain a secure SSL, TLS networking session through details of implementation not open to public.

Airdrop Security: -

It allows users to share files on their iOS device. When a user enables AirDrop, a 2048 bit RSA identity and it's hash is created and stored on the device. When AirDrop is open, a signal is emitted through Bluetooth Low Energy such that nearby devices that also have AirDrop turned on can receive it. After the sender chooses to whom he/she wants to send, a TLS connection is created between the sender and the receiver, with iCloud identity certificates being exchanged. After the receiver accepts the files to transfer, the transmission begins.

FaceTime Security: -

It is used to do video calls on iOS devices with others using FaceTime. It establishes end to end connection between users using a Session Initiation Protocol. The contents of communication are encrypted and only the sender and receiver can decrypt them.

ICloud Security: -

Icloud stores contacts, photos, calendar, and other documents and synchronizes them on all of user's iOS devices. IOS keychain synchronizes all user's passwords on different iOS devices.

Files stored on iCloud are broken down into blocks and encrypted using AES-128 and key is derived from SHA-256 hash of it's block contents.

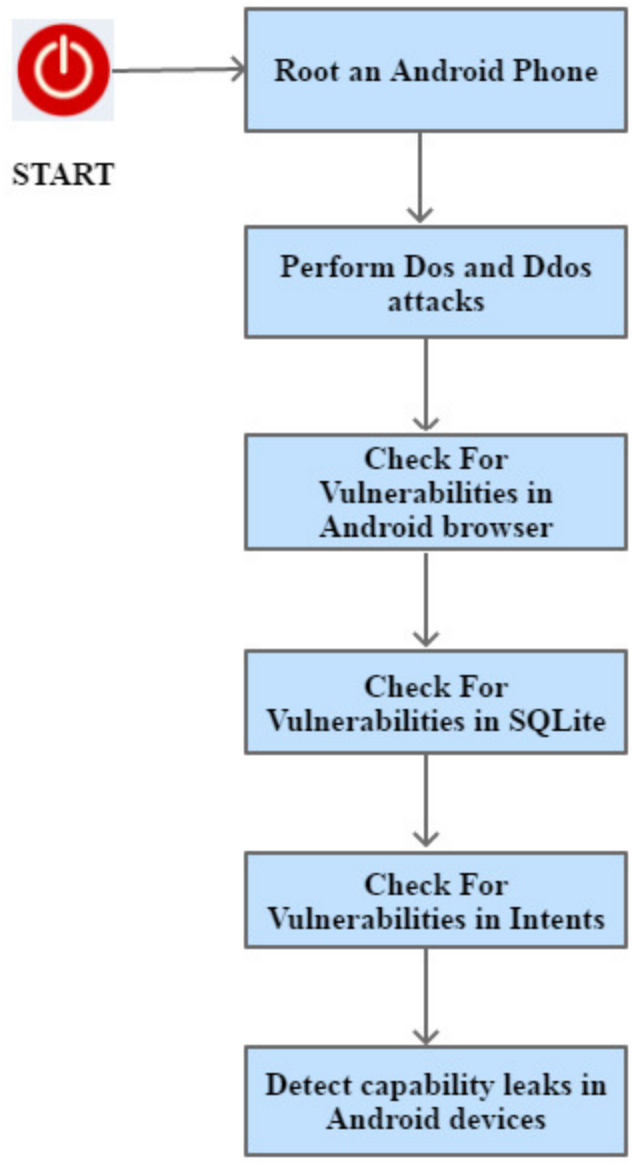
Continuity and Handoff: -

Continuity and Handoff feature has been introduced in iPhone to be syncronized with iOS and OS X devices. With Continuity a user's MAC/iPAD that share the same Wifi network associated with his iPhone and can make and receive calls as user's iPhone. The audio received from iPhone associated through the iPAD/MAC using encryption established through APNs.

Handoff is similar to Continuity, two devices establishes a Bluetooth Low Energy 4.0 connection through APNs. Then each generates a 256 bit AES key. The key exchange are used to encrypt and authenticate messages sent through Bluetooth in GCM.

Previous Next

Basic Android Phone Penetration Testing's



Try to root an Android phone and gain administrative access using tools such as SuperOneClick, Superboot, One Click Root, etc

Using tool such as Andosid, LOIC we can perform Dos and Ddos attacks on Android phone.

Check whether cross-application-scripting-error is present in Android browser which allows hackers to easily hack the Android device and try to break down the web browser's sandbox using infected javascript code.

Check whether password is stored in email as plain text using SQLite database and also check whether Skype on Android uses unencrypted SQLite database to store contacts, profile information, and instant message logs.

Try to exploit Android Intents to obtain user's private information.

We can use ComDroid to detect application's communication vulnerabilities.

We can use tool Woodpecker to detect capability leaks in Android devices.

Previous

Study Material

Quiz

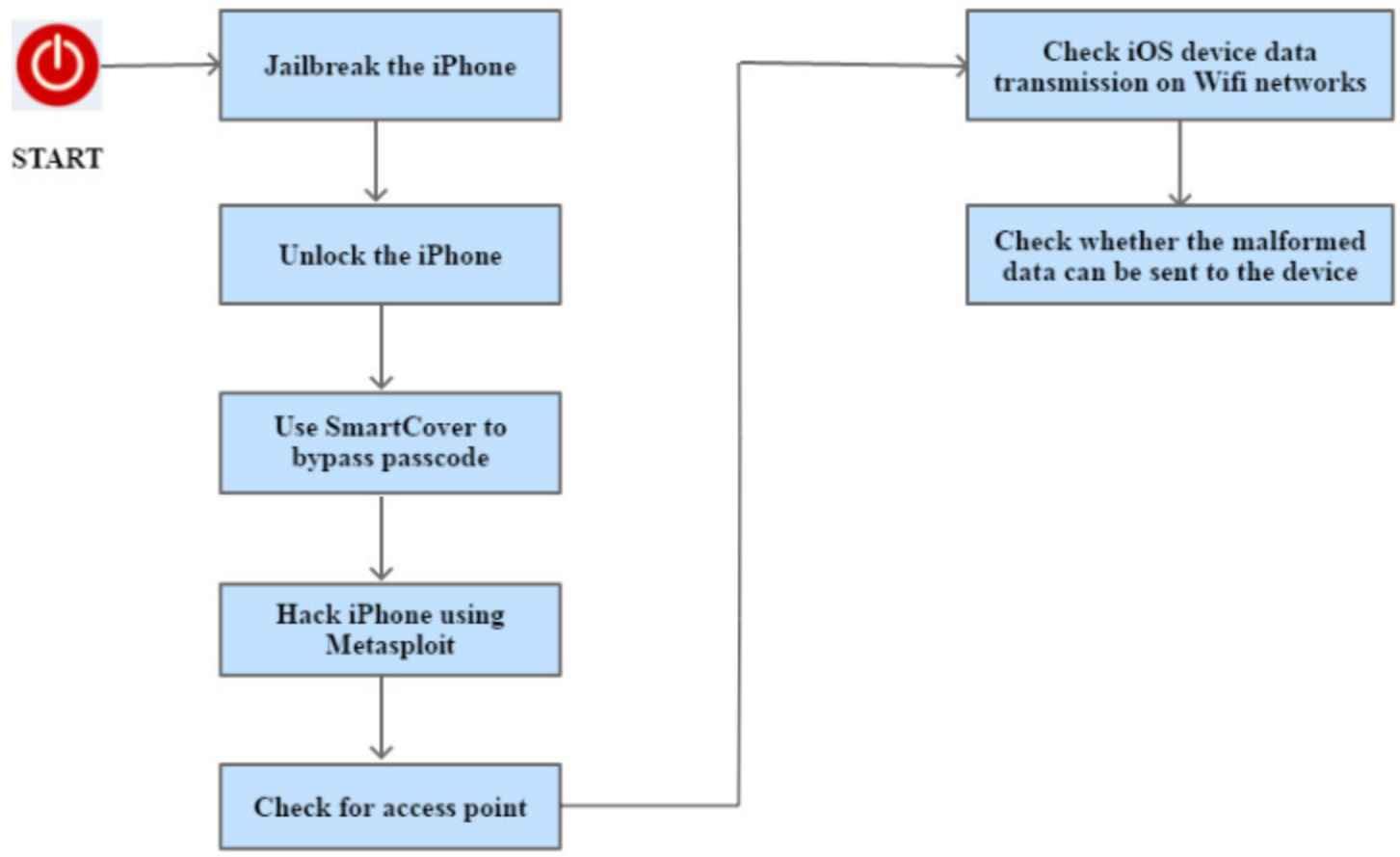
Mobile App Penetration Testing

Chapter 1: Android Overview

History and Features Of Android

Android Application and API

Basic IOS Phone Penetration Testing's



Try to Jailbreak the iPhone using tools such as Pangu, evasion7, Redsnow, Absinthe, Snowbreeze, Pwnagetool, etc.

Unlock the iPhone sim using tools such as iPhoneSimFree and anySIM

Hold the power button of an iOS operating device till the power off message appears. Close the smart cover till the screen shuts and open the smart cover after few seconds. Press the cancel button to bypass password code security.

Using Metasploit tool to exploit iPhone vulnerabilities. Send malicious code as payload to gain access to the device

Setup Access Point with same name and encryption type

Perform man-in-the-middle/SSL Stripping attack by intercepting wireless parameters of iOS device on Wifi network. Send malicious packets on Wi-fi network using Cain & Abel tool

Use social engineering techniques such as sending emails, SMS the user to trick him to open links that contain malicious web pages.