RESEARCH REPORT:

1. What is Android OS?

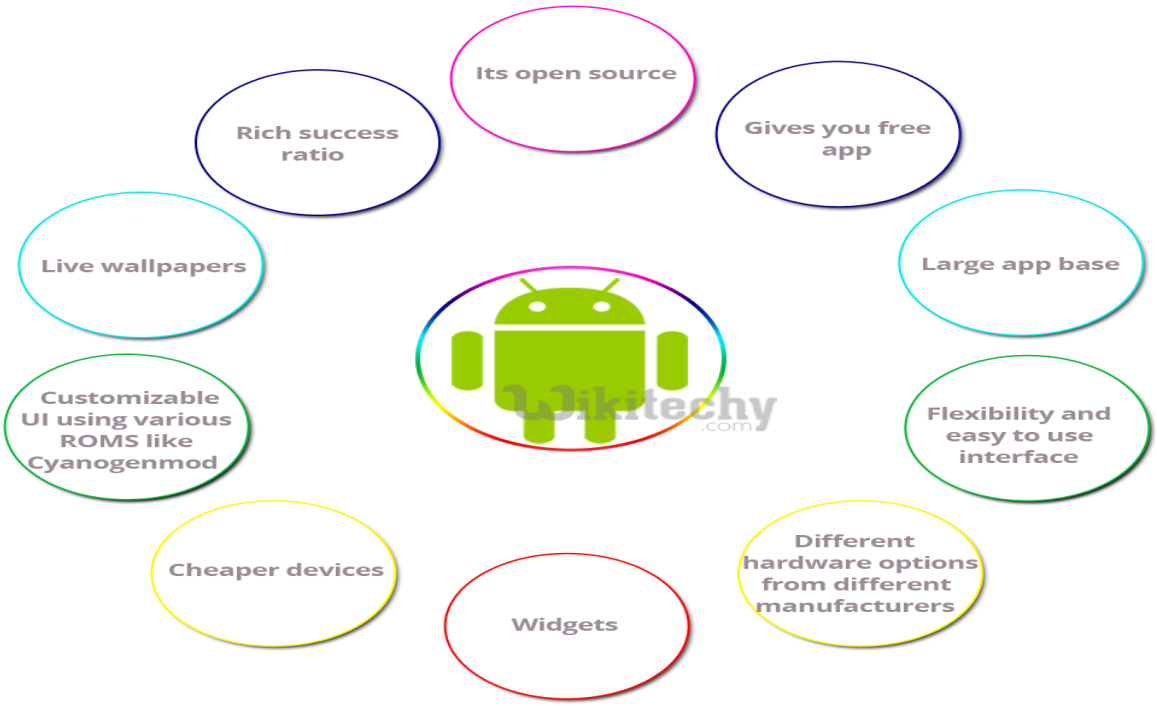
The Android OS is an open source operating system primarily used in mobile devices. Written primarily in Java and based on the Linux operating system, it was initially developed by Android Inc. and was eventually purchased by Google in 2005. The Android operating system is symbolized by a green colored Android Robot logo.

Based on the modified version of the Linux kernel version 2.6, the Android code was released by Google under the Apache license which is also a free software and open source license.

The Android OS consists of numerous Java applications and Java core libraries running under the Java-based object oriented application framework and the Dalvik Virtual Machine (VM). Dalvik is integral for the Android to run in mobile devices as these systems are constrained in terms of processor speed and memory.

As for multimedia support, the Android OS can back 2D and 3D graphics, common audio and video formats. It may also support multi-touch input (depending on device) and carries in its browser Google Chrome's V8 JavaScript runtime.

➢ Why Android?



➢ FEATURES:

1. Open Source: Android is built on an open-source platform, allowing developers to customize the operating system according to their needs.

2. User Interface: Android OS basic screen provides a beautiful and intuitive user interface.

3. Connectivity: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

4. Multitasking: Android supports multitasking, allowing users to run multiple apps simultaneously. Users can switch between apps seamlessly and even use split-screen mode to view two apps side by side.

5. Google Assistant: Android devices come with Google Assistant, a virtual assistant that helps users perform various tasks, answer questions, and control smart devices using voice commands.

6. Resizable widgets: Widgets are resizable, so users can expand them to show more content or shrink them to save space.

7. Integration with Google Services: Android seamlessly integrates with various Google services, including Gmail, Google Drive, Google Photos, and Google Maps. This tight integration enables users to access their data across different devices and services effortlessly.

8. Web Browser: Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.

9. GCM: Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.

10. Security: Android incorporates multiple layers of security features to protect users' data and privacy. These include built-in malware detection, app sandboxing, secure boot, and regular security updates.

11. Device Compatibility: Android is designed to run on a wide range of devices, including smartphones, tablets, smart watches, TVs, and automotive systems. This versatility ensures that users can choose from a diverse selection of devices that meet their specific needs.

➢ Android Security Model:

The Android Security Model is designed to protect the operating system and user data from unauthorized access. It's based on a combination of hardware and software security measures that are designed to keep the Android device secure. The Android security model is designed to protect against a variety of threats, including malware, data theft, and unauthorized access.

The Android Security Architecture consists of four layers of security that work together to protect the device and the user's data:

- **Application Sandbox:** The Application Sandbox is the first layer of security in Android. It's designed to separate the application code and data from the rest of the system. This means that each application runs in its own sandbox and can only access its own data and resources. The Application Sandbox is enforced by the Linux kernel, which prevents applications from accessing the resources of other applications.

- **Security Enhancements for Android (SEAndroid):** SEAndroid is a set of security enhancements that were added to Android to enhance its security. SEAndroid adds mandatory access control (MAC) to Android, which allows administrators to define policies that determine which applications and processes can access specific resources.

- **Android Framework:** The Android Framework is the third layer of security in Android. It provides a set of APIs that developers can use to build secure applications. The Android Framework includes features like permissions, which allow users to control which applications can access their data and resources.

- **Hardware Security:** The fourth and final layer of security in Android is hardware security. This includes things like Trusted Execution Environments (TEEs), which provide a secure environment for executing code and storing data. Hardware security is built into the device itself and is designed to protect against attacks that target the hardware components of the device.

2. What is IOS?

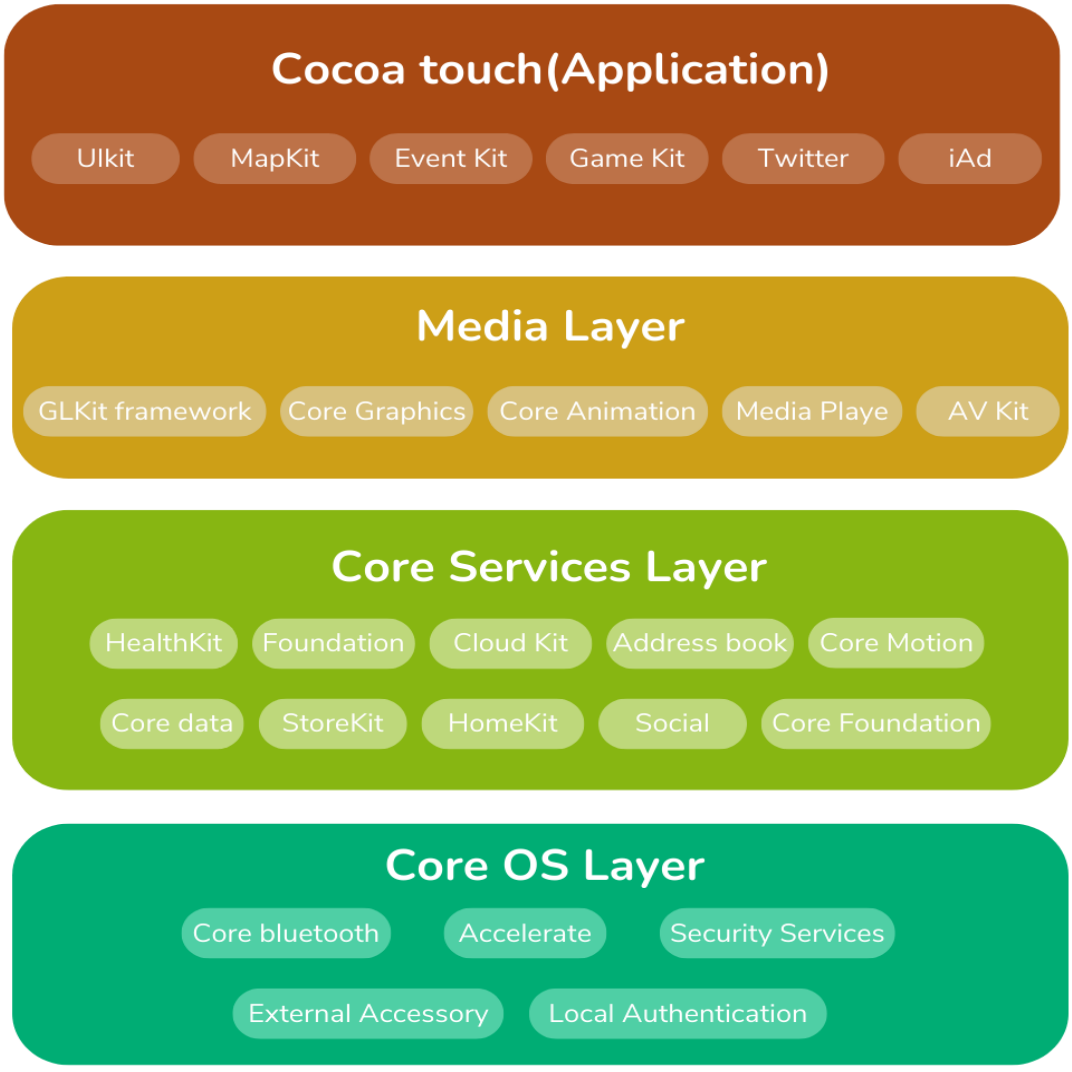
IOS is a mobile operating system for Apple-manufactured devices. iOS runs on the iPhone, iPad, iPod Touch and Apple TV.

iOS is best known for serving as the underlying software that allows iPhone users to interact with their phones using gestures such as swiping, tapping and pinching.

These finger actions are typically performed on multi-touch capacitive touch screen displays, which provide fast response and accept inputs from multiple fingers

iOS is derived from Mac OS X and is a Unix-like OS. There are four abstraction layers within iOS:

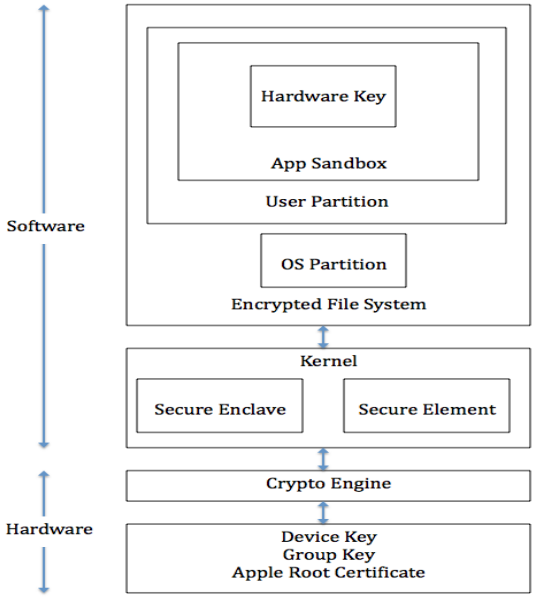
- **Core OS Layer:** Provides low-level features as well as frameworks for security and interaction with external hardware
- **Core Services Layer:** Provides services required by upper layers
- **Media Layer:** Provides the necessary technologies for graphics, audio and video.
- **Coca Touch Layer:** Where frameworks are located, which are often used when creating an application



iOS comes with a lot of default apps, including an email client, a Safari Web browser, a portable media player (iPod) and the phone app.

➢ iOS Security Architecture:

iOS is one of the most popular operating system available today. One of the most commonly mentioned preeminent features of this operating systems is the security.



- As can be seen from the image above, each app will be run in its own sandbox.
- Including a data security layer (Data protection class).
- The sandbox will linger in the user's partition. Data in this region will be subject to iOS's encryption algorithms.
- This partition is part of the file system next to the operating system partition. In addition, the hardware and firmware security of iOS is also very focused.
- As with the use of secure enclaves and secure elements in the kernel, crypto engines – the only things that can access the system's encryption keys.
- Along with that, the deepest of the system will be the device keys and group keys will talk about at the end with Apple's root certificate (Apple Root Certificate).

Hardware security:

- As mentioned above, each iOS device will have 2 AES-256-bit encryption keys built-in.
- The device ID (UID) and the device group ID (GID). These two key IDs will be attached to Application Processor (AP) and Secure Enclave Processor (SEP) right from the manufacturing process.
- These keys can only be read from the crypto engine.
- The GID is a shared value between processors of the same device class, used to prevent tampering with firmware files and to prevent other encryption tasks from accessing user information.
- Meanwhile, the UID is a unique value for each device, used to protect the system key in encrypting system files at the device level.
- UID values were not recorded during the manufacturing process, so Apple could not recover. (That's as Apple says).

Secure Boot:

Secure Boot is a fundamental component of the device's security architecture, ensuring that only trusted software is loaded during the boot process. Here's how Secure Boot works in iOS:

BOOT ROM(Read-Only Memory):

- The Secure Boot process begins with the Boot ROM, a piece of immutable firmware embedded in the device's hardware.
- The Boot ROM contains the first-stage bootloader code, which is cryptographically signed by Apple and verified by the device's hardware during the boot process.

LOW-LEVEL BOOTLOADER(LLB):

- After verifying the Boot ROM, the device proceeds to load the next stage of the boot process, known as the Low-Level Bootloader (LLB).
- The LLB is responsible for loading and verifying the integrity of the next boot stage, iBoot

iBOOT:

- iBoot is the second-stage bootloader in the iOS boot process.
- It verifies the integrity and authenticity of the kernel, the core of the operating system.

- iBoot ensures that the kernel has been signed by Apple using cryptographic keys that are securely stored in the device's hardware.
- If the kernel's signature is valid, iBoot proceeds to load the kernel into memory and execute it.

By employing a secure boot process that begins with immutable firmware in the Boot ROM and extends through subsequent boot stages, iOS ensures that only trusted and verified software components are loaded and executed on the device. This helps prevent unauthorized or malicious code from compromising the security and integrity of the iOS operating system

Code Signing:

In iOS, code signing is a security mechanism that ensures the authenticity and integrity of apps before they're allowed to run on a device.

- Digital Signature
- App verification
- App store
- Device Restrictions

Code signing helps protect users from malware and ensures that only trusted apps can run on iOS devices, enhancing the platform's security.

Encryption and Data Protection:

- Encryption and data protection are fundamental elements of the platform's security architecture, safeguarding user data against unauthorized access.
- iOS employs hardware-based encryption to encrypt data stored on the device, including emails, messages, photos, and app data.
- Each iOS device has a unique encryption key tied to its hardware, ensuring that even if someone gains physical access to the device, they cannot access the data without the encryption key.
- Additionally, iOS provides developers with the Data Protection API, enabling them to encrypt sensitive user data within their apps using the device's hardware encryption capabilities.
- This layered approach to encryption and data protection helps mitigate the risk of data breaches and unauthorized access, enhancing the overall security of iOS devices and the privacy of their users.

Sandbox:

Sandbox is a core feature of iOS security. It ensures that applications will run with mobile permissions and that only very few applications can run with root. An application will reside in one container, which stipulates that it only has access to the file it owns, as well as certain APIs. Access to resources such as files, network sockets, IPC or shared memory will be controlled by the sandbox**.**

Other general mechanism:

- iOS also uses address space layout randomization (ASLR) and eXecute Never(XN) to limit code execution attacks.
- ASLR will generate random addresses for executable files, data, heap and stack each time. Because libraries need access from multiple processes, the address will be randomly generated at device startup.
- This will make the address of processes very difficult to guess, limiting execution attacks as well as return-to-lib attacks.
- XN is a mechanism to mark an area on memory as impossible to execute.
- In iOS, the heap and stack of user processes will be highlighted. Pages labeled with writable cannot be executable at the same time. This helps limit the execution of machine code on the heap and stack.

3. Explain the Mobile Pentesting process and Tools?

Mobile pen-testing, short for mobile penetration testing, is the process of assessing the security of mobile applications and devices to identify vulnerabilities that could be exploited by attackers.

The process typically involves several steps:

1. Reconnaissance: Gathering information about the target mobile application or device, including its architecture, features, and potential attack surfaces.

2. Static Analysis: Analyzing the source code of the mobile application to identify security flaws, such as hardcoded credentials, insecure data storage, or improper input validation.

3. Dynamic Analysis: Inspecting an application's code in a running state. This is a more practical way of scanning, as it provides a real-time view into an application's performance.

4. Reverse Engineering: Decompiling or disassembling the mobile application to understand its inner workings, including the use of proprietary APIs, encryption algorithms, or obfuscated code.

5. Exploitation: This involves attempting to exploit the identified vulnerabilities to gain unauthorized access to the app, the device, or the data stored on the device.

6. Reporting: This involves documenting the findings and providing recommendations for remediation.

➢ Benefits of mobile penetration testing:

1. **Identifies and fixes security vulnerabilities:** Mobile penetration testing can help to identify and fix security vulnerabilities that could be exploited by malicious actors.
2. **Meets compliance requirements:** Mobile penetration testing can help to meet compliance requirements such as PCI DSS, HIPAA, and SOX.
3. **Improves the overall security of mobile applications:** By fixing vulnerabilities, mobile penetration testing can help to improve the overall security of mobile applications. This can help to protect sensitive data, prevent unauthorized access, and disrupt the functionality of the app.

➢ Types of Mobile Penetration Testing:

1. **White box testing:** This type of testing involves providing the penetration tester with complete access to the source code of the app.
2. **Black box testing:** This type of testing involves providing the penetration tester with no access to the source code of the app.
3. **Gray box testing:** This type of testing involves providing the penetration tester with limited access to the source code of the app.

➢ Tools used for Mobile Penetration testing:

1. Mobile Security Framework(MobSF): An open-source framework for static and dynamic analysis of Android and iOS applications. It provides various features for identifying security vulnerabilities.
2. AndroBugs Framework: Designed for Android applications.
3. Drozer: Security Testing Framework.
4. QARK (Quick Android Review Kit)
5. Apktool.

6. JD-GUI.
7. Burp Suite.
8. OWASP ZAP (Zed Attack Proxy)
9. Wireshark.
10. Frida.
11. Mobexler.
12. Needle.
13. Xcode Instruments.
14. MobSF(Mobile Security Framework) CLI.
15. SSL Kill Switch 2: A tool for bypassing SSL certificate validation, useful for intercepting and analyzing encrypted HTTPS traffic generated by mobile applications.

These tools, when used appropriately and in combination with each other, can help testers identify security vulnerabilities in mobile applications and devices, ultimately enhancing their overall security.

REFERENCES:

- ✓ https://www.immuniweb.com/resources/what-is-mobile-penetration-testing/
- ✓ https://www.techopedia.com/definition/25206/ios
- ✓ https://www.tutorialspoint.com/android/android_overview.htm
- ✓ https://itzone.com.vn/en/article/ios-security-architecture/
- ✓ https://redfoxsec.com/blog/ios-architecture/