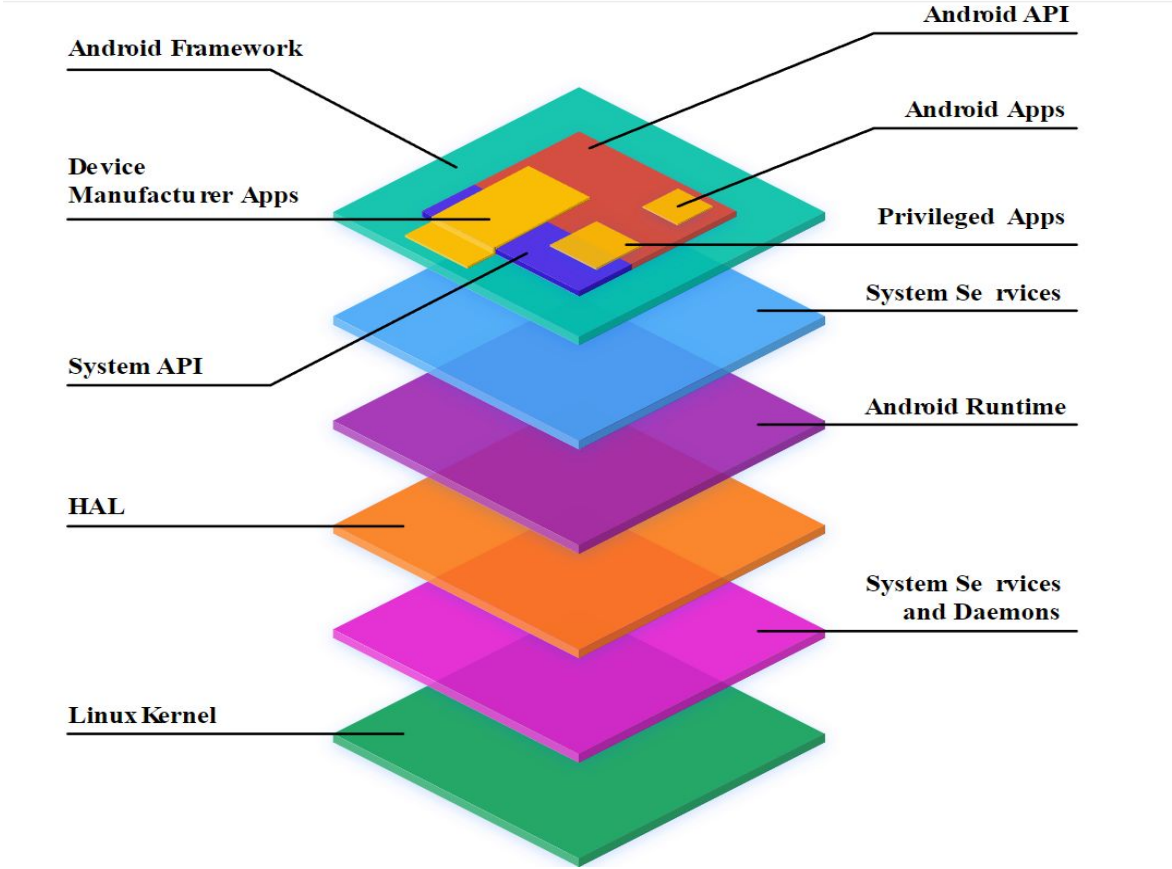
Android Open System Platform(AOSP)

AOSP Testing

Overview of AOSP

- The *Android Open System Platform (AOSP)* is publicly available and modifiable Android source code. Anyone can download and modify AOSP for their device. AOSP provides a complete and fully functional implementation of the Android mobile platform.
- There are two levels of compatibility for devices implementing AOSP:
    - AOSP compatibility and
    - Android compatibility.
- An *AOSP-compatible device* must confirm to the list of requirements in the Compatibility Definition Document (CDD).
- An *Android-compatible device* must conform to the list of requirements in the CDD and Vendor Software Requirements (VSR) and tests such as those in the Vendor Test Suite (VTS) and Compatibility Test Suite (CTS).

AOSP Architecture



Android App:

An app created solely using the Android API. Google Play Store is widely used to find and download Android apps, though there are many other alternatives.

Privileged app:

An app created using a combination of the Android and system APIs. These apps must be pre-installed as privileged apps on a device.

Device manufacturer app:

An app created using a combination of the Android API, system API, and direct access to the Android framework implementation. Because a device manufacturer might directly access unstable APIs within the Android framework, these apps must be preinstalled on the device and can be updated only when the device's system software is updated.

System API:

The System API represents Android APIs available only to partners and OEMs(original equipment manufacturers) for inclusion in bundled applications. These APIs are marked as @SystemApi in the source code.

Android API:

The Android API is the publicly available API for third-party Android app developers. API allows you to add specific functionalities to your application.APIs are the set of protocols and tools used for building software applications.

Android framework:

A group of Java classes, interfaces, and other precompiled code upon which apps are built. Portions of the framework are publicly accessible through the use of the Android API. Other portions of the framework are available only to OEMs through the use of the system APIs. Android framework code runs inside an app's process.

System services:

System services are modular, focused components such as SystemServer, SurfaceFlinger, and MediaService. Functionality exposed by Android framework API communicates with system services to access the underlying hardware.

Android Runtime (ART):

A Java runtime environment provided by AOSP. ART performs the translation of the app's bytecode into processor-specific instructions that are executed by the device's runtime environment.

Hardware Abstraction Layer (HAL):

A HAL is an abstraction layer with a standard interface for hardware vendors to implement. HALs allow Android to be agnostic about lower-level driver implementations. Using a HAL lets you implement functionality without affecting or modifying the higher level system.

Native daemons and libraries:

Native daemons in this layer include init, healthd, logd, and storaged. These daemons interact directly with the kernel or other interfaces and don't depend on a userspace-based HAL implementation. Native libraries in this layer include libc, liblog, libutils, libbinder, and libselinux. These Native libraries interact directly with the kernel or other interfaces and don't depend on a userspace-based HAL implementation.

Kernel:

The kernel is the central part of any operating system and talks to the underlying hardware on a device. Where possible, the AOSP kernel is split into hardware-agnostic modules and vendor-specific modules.