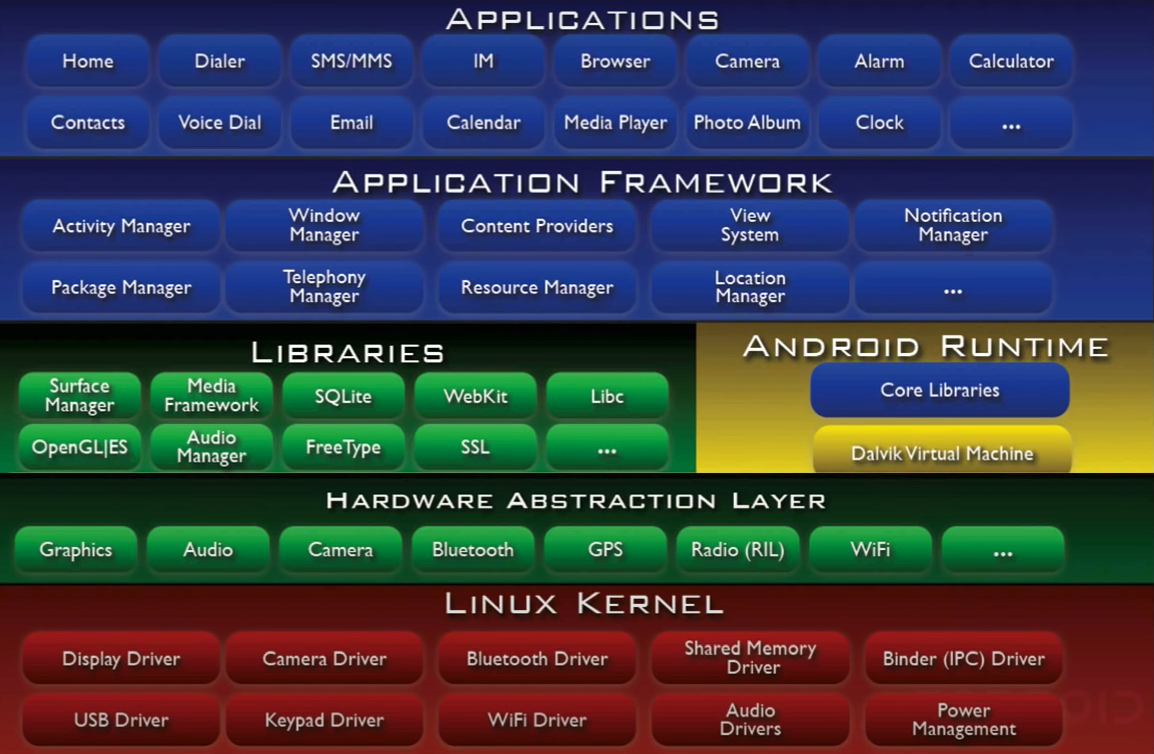
**Mobile Device/Platform Security**

1. **Android Platform Architecture**



**Android is an open software platform for mobile development. It has been developed through massive collaborative efforts and investments by many companies. The main company behind android development is Google. It is intended to be a complete stack that includes everything from the OS through middleware and up through applications.**

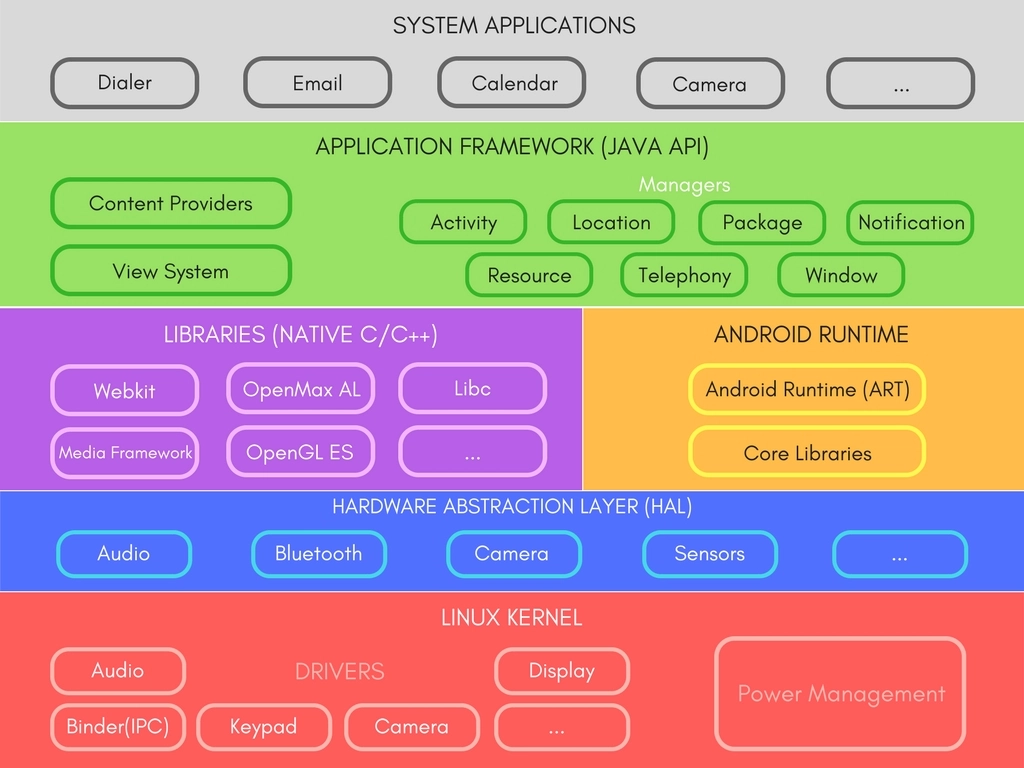
**\*Middleware is computer software that provides services to software applications beyond those available from the operating system**

1. **Layer**

**The following are the layers that compose the Android architecture as labeled in the picture above :**

* **Applications**
* **Application Framework**
* **Android Runtime and Core Libraries**
* **Hardware Abstraction Layer**
* **Linux Kernel**

**Today, with technological advancements, some layers are edited but not too much, following the picture below**

****

1. **Application**

**This is the layer that an end-user directly interact with. The term also refers to an APK file which stands for Android package.**

**\*APK file: a Zip archive containing app code, resources, and meta information. Android apps can be written in Kotlin, Java, and C++ and are run inside Virtual Machine. Some default application on Android platform are listed below:**

**Home: could be accessed by home button on android devices. It includes launcher icons of applications that end-users commonly use. You can start the arbitrary app by clicking/tapping on the launcher.**

**Contacts: provides a means to store and retrieve contacts. Contact information could be shared across other apps to enhance functionality.**

**Messages: send and receive SMS messages.**

**Browser: Android comes with a default browser for web browsing.**

**This layer is also referred to as user-level in contrast to the layers below that are mostly tuned for application development.**

1. **Application Framework**

**The entire feature-set of the Android OS is available to you through APIs written in the Java language. The framework exposes a safe and uniform means to utilize Android device resources.**

* **Activity Manager**

**Presenting an entry point to the app. The Android ActivityManager is responsible for predictable and consistent behavior during application transitions. It provides a slot for app creators to have their apps react when the Android OS performs global actions. Some examples of the way applications can react to these transitions include pausing activity in a game, stopping music playing during a phone call.**

* **Resource Manager**

**Providing access to non-code resources such as icons, audio and video files, animations, text files, and the like. It also ensures that the right resources are delivered to the end-users.**

* **Notification Manager**

**Enables all apps to display custom alerts in the status bar. It does this by giving users visual, audio or vibration signals or a combination of them when an event occurs.**

* **Content Provides**

**Enable apps to access data from other apps, such as the Contacts app, or to share their own data.**

* **View System**

**Use to build an app’s UI, including lists, grids, text boxes, buttons, and even an embeddable web browser**

1. **Android Runtime(ART)**

**Android currently uses Android Runtime (ART) to execute application code. ART is written to run multiple virtual machines on low-memory devices by executing DEX (Dalvik Executable) files, a bytecode format designed specially for Android that's optimized for minimal memory footprint.**

**Some of the major features of ART include the following:**

* **Ahead-of-time (AOT) and just-in-time (JIT) compilation**
* **Optimized garbage collection (GC)**
* **Better debugging support**

**Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features, that the Java API framework uses.**

1. **Core Libraries**

**Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++.**

**Although most android applications are written in Java, there are special scenarios where you may require to use C/C++ . That’s why Android provides a NDK (Native Development Kit) to access some of these native platform libraries directly from your native code.**

**Some of the core libraries that are presented in the Android operating system:**

* **Media Framework**

**Includes support for playing a variety of common media types, so that you can easily integrate audio, video and images into your applications.**

* **SQLite**

**Enables applications to have very fast native database functionality without the need for third party libraries.**

* **OpenGL**

**an API for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.**

* **SSl/TLS**

**Secure Socket Layer (now technically known as Transport Layer Security) for security.**

* **Libc**

**The core of Android contains libraries written in C and C++, which are low-level languages meant for embedded use that help in maximizing performance. Libc provides a means to expose low-level system functionalities such as Threads, Sockets, IO, and the like to these libraries.**

* **Webkit**

**Work with modern WebView APIs on Android 5 and above.It enables application developers to render web components in the view-system by using WebView.**

* **Surface Manager**

**Responsible for ensuring the smooth rendering of application screens. It does this by composing 2D and 3D graphics for rendering.**

* **SGL (Scalable Graphics Library)**

**A graphics library implemented in low-level code that efficiently renders graphics for the android platform.**

1. **Hardware Abstraction Layer (HAL)**

**A HAL defines a standard interface for hardware vendors to implement, which enables Android to be agnostic about lower-level driver implementations. Using a HAL allows you to implement functionality without affecting or modifying the higher level system. HAL implementations are packaged into modules and loaded by the Android system at the appropriate time. The Hal mediates interactions between the OS kernel layer & higher layers. Android apps rarely access the HAL directly, so you needn’t know all the details.**

1. **Linux Kernel**

**The foundation of the Android platform that enables all of Android's functionality. With regards to Android, the Kernel is responsible for many foundational functionalities including:**

* **Device Drivers**

**USB, Bluetooth, Wifi, DIsplay, Audio, Power, Flash memory, Binder**

* **Memory Management**

**As different applications run, the Kernel ensures the memory space they use doesn't conflict and overwrite each other. It also helps making sure no single app takes too much space.**

* **Process Management**

**Responsible for creating, pausing, stopping, shutting, or killing down processes. Enables various functionalities such as running multiple processes at the same time, communicating between processes, running processes in the background, and so on.**

1. **Android Sandbox Mechanism**
2. **What is Sandbox**

**In cybersecurity, a sandbox is an isolated environment on a network that mimics end-user operating environments. Sandboxes are used to safely execute suspicious code without risking harm to the host device or network.**

**Using a sandbox for advanced malware detection provides another layer of protection against new security threats—zero-day (previously unseen) malware and stealthy attacks, in particular. And what happens in the sandbox, stays in the sandbox—avoiding system failures and keeping software vulnerabilities from spreading.**

1. **Sandboxing mechanism in Android**

**The Android platform takes advantage of the Linux user-based protection to identify and isolate app resources. This isolates apps from each other and protects apps and the system from malicious apps. To do this, Android assigns a unique user ID (UID) to each Android application and runs it in its own process.**

**Android uses the UID to set up a kernel-level Application Sandbox. The kernel enforces security between apps and the system at the process level through standard Linux facilities such as user and group IDs that are assigned to apps. By default, apps can't interact with each other and have limited access to the OS. If app A tries to do something malicious, such as read application B's data or dial the phone without permission, it's prevented from doing so because it doesn't have the appropriate default user privileges. The sandbox is simple, auditable, and based on decades-old UNIX-style user separation of processes and file permissions.**

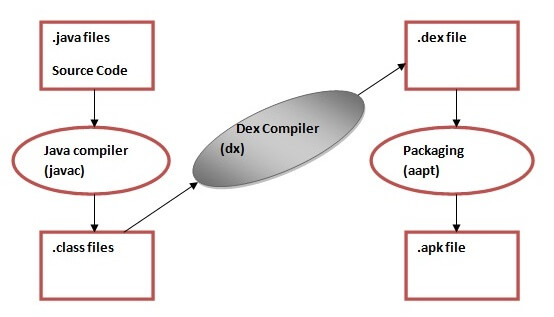
**Because the Application Sandbox is in the kernel, this security model extends to both native code and OS applications. All of the software above the kernel, such as OS libraries, application framework, application runtime, and all applications, run within the Application Sandbox. On some platforms, developers are constrained to a specific development framework, set of APIs, or language. On Android, there are no restrictions on how an application can be written that are required to enforce security; in this respect, native code is as sandboxed as interpreted code.**

1. **Dalvik VM, Android Runtime & How it operates**
2. **What is Dalvik Virtual Machine**

**The Dalvik Virtual Machine (DVM) is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for memory, battery life and performance. Dalvik uses less space, which means an uncompressed .dex file is smaller in size(few bytes) than compressed java archive file(.jar file). DVM uses JIT (Just-In-Time) compiler.**

1. **How DVM works**

**The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.**

****

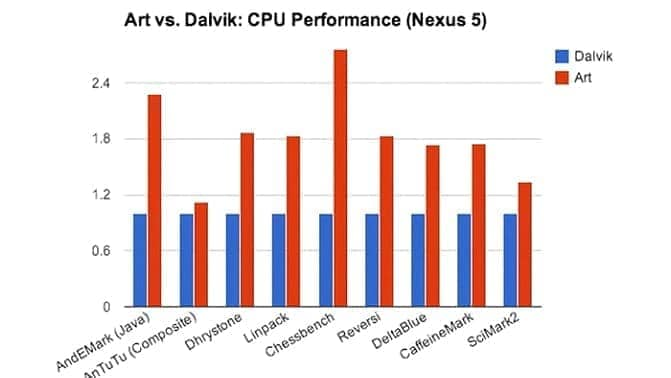
**In Android, code is written and compiled java source file(bytecode) on java compiler, but at that point it is recompiled once again using Dalvik compiler to Dalvik bytecode(dx tool converts java .class file into .dex format and .odex format) and this Dalvik bytecode is then executed on the Dalvik virtual machine.**

1. **What is Android Runtime (ART)**

**The successor of Dalvik is Android Runtime (ART), which uses the same bytecode and .dex files (but not .odex files), with the succession aiming at performance improvements transparent to the end users. ART performs the translation of the application's bytecode into native instructions that are later executed by the device's runtime environment.**

**The new runtime environment was included for the first time in Android 4.4 “KitKat” as a technology preview and replaced Dalvik entirely in later versions. Android 5.0 “Lollipop” is the first version in which ART is the only included runtime. ART uses AOT (Ahead-of-time) compiler.**

1. **How ART works**

****

**To maintain backward compatibility, ART uses the same input bytecode as Dalvik, supplied through standard .dex files as part of APK files, while the .odex files are replaced with Executable and Linkable Format (ELF) executables. Once an application is compiled by using ART's on-device dex2oat utility, it is run solely from the compiled ELF executable; as a result, ART eliminates various application execution overheads associated with Dalvik's interpretation and trace-based JIT compilation. As a downside, ART requires additional time for the compilation when an application is installed, and applications take up slightly larger amounts of secondary storage (which is usually flash memory) to store the compiled code.**

1. **Android Security Feature**

**Android has become one of the most popular smartphone operating systems in the world, 88 percent of all the smartphone users worldwide are using Android OS, and Google’s operating system has just turned 10.**

* **App sandbox**

**The Android platform takes advantage of the Linux user-based protection to identify and isolate app resources. To do this, Android assigns a unique user ID (UID) to each Android app and runs it in its own process. Android uses this UID to set up a kernel-level App Sandbox.**

* **App permissions**

**One feature that’s often overlooked, but which you should pay close attention to, is the app permission settings located in the App & Notifications menu. In this window, you’ll be able to see which apps have permission to access different phone functions. Pay extra attention to apps that have access to your microphone, camera, and biometric sensors, as these can be used to monitor your daily activities and private information.**

* **Lock screen preferences**

**In the “Security” menu of your device, there are various settings for managing your lock screen. App notifications, for instance, still make their way onto your lock screen, which means people can still see important messages, even if your phone is locked. To fix this, you can simply limit how much is shown on your lock screen.**

**Another important setting is Smart Lock, a feature that allows you to automatically lock a device based on its location. For example, if you’re carrying your phone, it can let you automatically keep your phone unlocked; but as soon as it leaves your hand, it locks itself immediately.**

* **Multi-factor authentication**

**Android uses the concept of user-authentication-gated cryptographic keys that requires cryptographic key storage and service provider and user authenticators.On devices with a fingerprint sensor, users can enroll one or more fingerprints and use those fingerprints to unlock the device and perform other tasks. The Gatekeeper subsystem performs device pattern/password authentication in a Trusted Execution Environment (TEE). Aside from accessing your device and apps with just a passcode, multi-factor authentication forces users to provide another set of identification like an SMS code, fingerprint, or facial recognition scan. Although this adds another step to your sign-in process, it does make it much more difficult for anyone to hijack your accounts. This feature can be found in the “Sign-in & security” options of your system’s settings.**

**Android 9 and higher includes Protected Confirmation, which gives users a way to formally confirm critical transactions, such as payments.**

* **Biometrics**

**Android 9 and higher includes a BiometricPrompt API that app developers can use to integrate biometric authentication into their apps in a device- and modality-agnostic fashion. Only strong biometrics can integrate with BiometricPrompt.**

* **Encryption**

**Once a device is encrypted, all user-created data is automatically encrypted before committing it to disk and all reads automatically decrypt data before returning it to the calling process. Encryption ensures that even if an unauthorized party tries to access the data, they won’t be able to read it.**

* **Find my device**

**If you lose your Android phone, anyone who picks it up — could be a harmless passerby, could be a hacker — can see what’s inside. Fortunately, Android has a “Find my device” feature that allows you to track, lock, and wipe data from your lost or stolen device.**

* **Keystore**

**Android offers a hardware-backed Keystore that provides key generation, import and export of asymmetric keys, import of raw symmetric keys, asymmetric encryption and decryption with appropriate padding modes, and more.**

* **Trusty Trusted Execution Environment (TEE)**

**Trusty is a secure Operating System (OS) that provides a Trusted Execution Environment (TEE) for Android. The Trusty OS runs on the same processor as the Android OS, but Trusty is isolated from the rest of the system by both hardware and software.**

* **Verified Boot**

**Verified Boot strives to ensure all executed code comes from a trusted source (usually device OEMs), rather than from an attacker or corruption. It establishes a full chain of trust, starting from a hardware-protected root of trust to the bootloader, to the boot partition and other verified partitions.**

1. **About Rooting and how to root a device**
2. **What is Rooting?**

**Rooting is the process of allowing users of the Android mobile operating system to attain privileged control (known as root access) over various Android subsystems. Rooting gives the ability (or permission) to alter or replace system applications and settings, run specialized applications ("apps") that require administrator-level permissions, or perform other operations that are otherwise inaccessible to a normal Android user. On some devices, rooting can also facilitate the complete removal and replacement of the device's operating system, usually with a more recent release of its current operating system.**

1. **Methods**

**When you root your device, you are flashing a custom ROM to replace the Android OS pre-installed on your device. Another term that we would be using frequently is the bootloader. The bootloader is a piece of software that boots up your phone’s OS, and it needs to be unlocked to root your phone. There are several methods to root your Android device. This is usually being done in one of two ways:**

* **Soft Root**

**This method relies on a privilege escalation vulnerability in the Linux kernel or an application running as root. Once the tool performing the rooting has obtained root permissions, it has unlimited access to the filesystem. This is usually performed by One Click rooting tools. One Click rooting tools are apps that are installed on the device and trigger the vulnerability upon launch.**

**Detail:**

[**https://www.digitaltrends.com/mobile/how-to-root-android/**](https://www.digitaltrends.com/mobile/how-to-root-android/)

* **Hard Root**

**The second way is hard rooting. Hard rooting relies on the ability to flash the firmware of the device. This effectively also allows full access to the filesystem. A hard root requires a device that has a bootloader that can be unlocked or a vulnerability in the bootloader.**

* **Different way**
* **One way that was previously used to persist root access via adb (Android Debug Bridge).**
* **Before SELinux was ported to Android, the most common way to persist root access was to drop a suid binary into the filesystem, which allowed everyone running it do things as root. In order for this not to become a security problem, usually, an app is used. Android mascot with deep roots illustrating root access. The app asks the user for permissions to run commands as root via SU (also referred to as Superuser app).**
* **Installing a custom ROM that provides root access by default is also a way to achieve root access.**

1. **Rooting with Magisk**

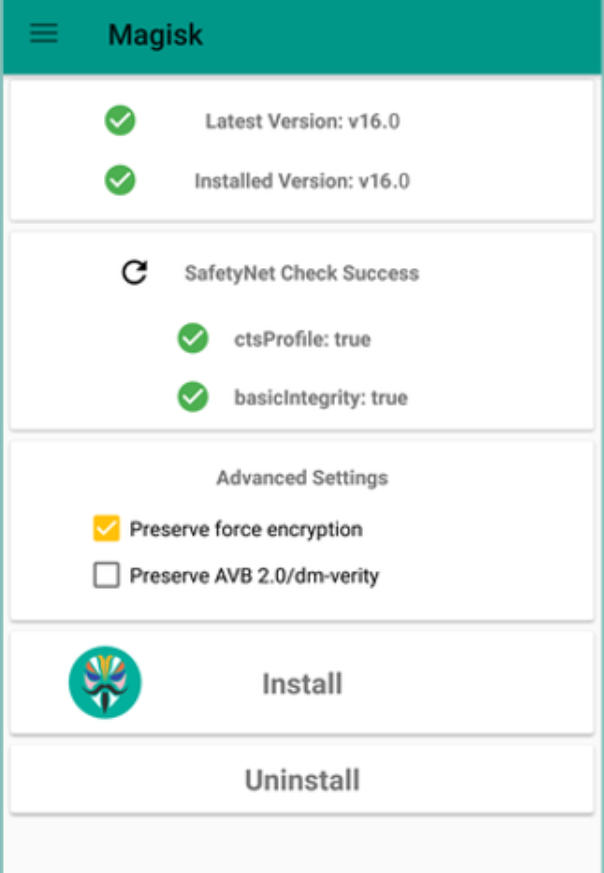
**Magisk has become popular in recent years. With Magisk (first developed by topjohnwu), you can have root and custom mods while still using services like Google Pay. It works by leaving the system partition untouched and modifying the boot partition. This is why it’s referred to as a “systemless” root method. ( supporting devices higher than Android 4.2).**

* **Rooting and Flashing definition**
* **Rooting refers to a process of attaining root access, in other word, administrative (superuser) permissions to an Android device.**
* **Flashing, to be specific, is flashing a ROM. ROM is a file containing executable instructions of an Android operation system and related apps. Flashing a ROM means installing a ROM on your phone so that the original Android OS can be changed and updated.**

**Source:** [**https://github.com/topjohnwu/Magisk**](https://github.com/topjohnwu/Magisk)

* **The first thing you’ll need is the Magisk zip file. This is the file we will be flashing in the next step. You can find the latest version of the zip at the** [**official XDA thread**](https://forum.xda-developers.com/t/magisk-the-magic-mask-for-android.3473445/)**. Download the latest zip and transfer it to your phone/tablet, or download it directly on your phone/tablet, OR An easy way to do this is to install the Magisk Manager app (found in Step 3), which will prompt you to download the latest zip that’s currently available.**
* **Next, we will go into recovery mode and flash the zip. Find out how to boot into recovery mode on your device ( described in this paper). Once you boot into your custom recovery, follow these steps:**
* **In recovery mode, select the Install button.**
* **Find the folder where you download the zip file.**
* **Select the zip file**
* **Swipe the slider to install Magisk**
* **Tap Reboot System**
* **The Magisk framework is now installed on your device. To manage it, you’ll need the Magisk Manager. You can download the latest Magisk Manager APK** [**here**](https://github.com/topjohnwu/Magisk/releases/) **and make sure you have “Unknown sources” enabled in the Settings. Install the app and open it up.**

**The Manager app allows you to adjust root settings and also install modules. Using the Manager app is an important part of getting the most out of this root method.**

****

* **The last step is to verify that everything is working properly. Open the newly installed Manager app. We want to see a bunch of green check marks in the app. This means you have successfully obtained root.**

1. **Bootloader**
2. **What is Bootloader?**

**In the simplest terms, a bootloader is a piece of software that runs every time your phone starts up. It tells the phone what programs to load in order to make your phone run. The bootloader starts up the Android operating system when you turn on the phone. This is a pretty important job, so it’s very important that nothing goes wrong with it. That’s why phones keep their bootloaders stored in special stable memory. Alternatively, the bootloader can start up recovery mode. When a phone is in recovery, it can execute large pieces of code that totally rewrite the Android operating system.**

1. **How to Lock/Unlock**

* **Unlocking the bootloader**

**To unlock the bootloader and enable partitions to be reflashed, run the fastboot flashing unlock command on the device. After setting, the unlock mode persists across reboots.**

**Devices should deny the fastboot flashing unlock command unless the get\_unlock\_ability is set to 1. If set to 0, the user needs to boot to the home screen, open the Settings > System > Developer options menu and enable the OEM unlocking option (which sets the unlock\_ability to 1). After setting, this mode persists across reboots and factory data resets.**

**When the fastboot flashing unlock command is sent, the device should prompt users to warn them that they might encounter problems with unofficial images. After the user acknowledges the warning, the device should perform a factory data reset to prevent unauthorized data access. The bootloader should reset the device even if it can't reformat it properly. Only after a reset can the persistent flag be set so that the device can be reflashed.**

* **Locking the bootloader**

**To lock the bootloader and reset the device, run the fastboot flashing lock command on the device. Devices intended for retail should be shipped in the locked state (with get\_unlock\_ability returning 0) to ensure that attackers can't compromise the device by installing a new system or boot image.**

1. **Android Recovery Mode**

**All Android phones come with built-in recovery mode that is separate from the original operating system. The recovery mode is used to access different features of the phone without accessing the phone’s OS. The main function of the recovery mode is to fix the phone while staying away from the faulty OS of the phone.**

**Note: In recovery mode, the phone’s touch screen will not work, you will have to use the hardware buttons to navigate. To move between options, press Volume up and down buttons to move up and down respectively. You can press the Power button to select any of the options.**

1. **How To Boot In**

**All you need to do is turn off the phone and turn it on by pressing specific keys (depending on the phone). On some phones you will enter Recovery mode directly and on others you may need to navigate through the different options to access Recovery mode.**

* **Samsung Galaxy Series**
* **Turn off the phone**
* **Now, press and hold Power+Home+Volume Up buttons..**
* **Keep holding until the device logo shows up and your phone restarts again, you should enter recovery mode.**
* **Nexus Series**

* **Turn off the phone**
* **Press and hold Power+Volume Up+Volume Down buttons.**
* **Keep holding until you see a menu with the Recovery mode option.**
* **Navigate to the Recovery mode option and press Power button.**

1. **Custom Recovery and Stock Recovery**

**The recovery comes with the phone by default is the stock recovery and the developer version of the recovery is called custom recovery. What is the difference between these recoveries ?**

* **Stock Recovery**

**You cannot do any modifications to the system files using the stock recovery and the options are also limited. The main purpose of the stock recovery is to delete all user data and files when the phone’s software shows error or the phone fails to boot. It can only be used to install the update provided by the manufacturer of the device.**

* **Stock Recovery can be used to take backups of the user data in most of the phones.**
* **Can do a complete reset of your phone which is also called hard reset.**
* **It can install update packages provided by the phone manufacturer after signature verification.**
* **It can be accessed by pressing a few keys together when the phone is booting.**
* **This can be controlled with the volume keys and power buttons in most of the phones. Touchscreen support has been added to the recent models.**
* **Custom Recovery**

**As the name indicates it is customized to have a lot of options. You can create your own recovery or just port it from other recoveries. The most widely used Custom Recoveries are Clockworkmod (CWM) , Team Win Recovery Project (TWRP), Cannibal Open Touch (COT), etc .**

* **Comes with a lot of features**
* **Can take a complete backup of all the partitions of your phone and is called Nandroid backup. This can be used to bring back your phone to life when something goes wrong during the update.**
* **Can install custom updates and ROMs easily with few clicks.**
* **It has options to do an advanced backup and restore facility.**
* **Can do pre-flash wipe. Which means formatting all the partitions of the android file system to install new ROM.**
* **Can be accessed through key combination while the phone is booting also can be accessed through some softwares with root access.**
* **Inputs can be given by pressing the buttons. Touch inputs are available in almost all the custom recoveries.**
* **The file system can be accessed in the recovery.**
* **ADB sideload is possible.**
* **Can be used to connect the phone’s SD card to the PC.**
* **Themes can be applied, to have a good look.**
* **Can skip signature verifications while installing updates.**
* **Can root and unroot your phone in advanced options and much more.**

1. **Android Debug Bridge (adb)**
2. **What is ADB?**

**ADB, Android Debug Bridge, is a command-line utility included with Google’s Android SDK that lets you communicate with an Android device. ADB can control your device over USB from a computer, copy files back and forth, install and uninstall apps, run shell commands, and more.**

**It is a client-server program that includes three components:**

* **Client**

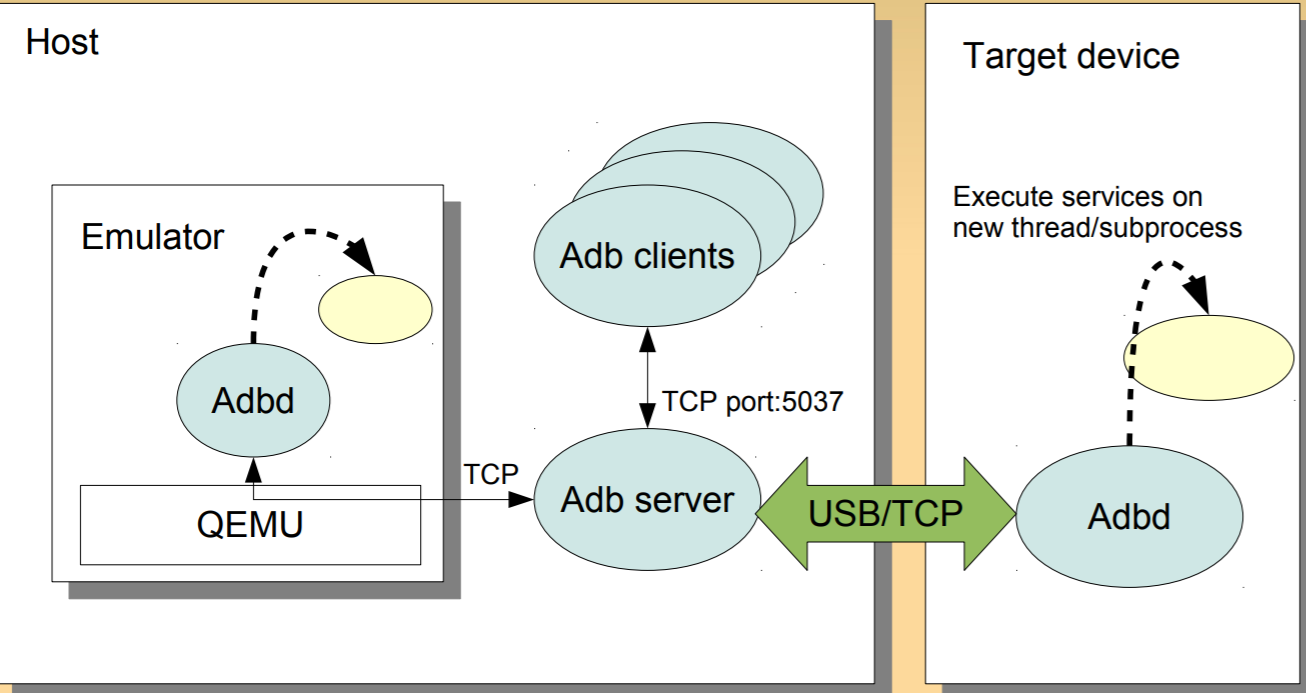
**Which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an adb command.**

* **Daemon (adbd)**

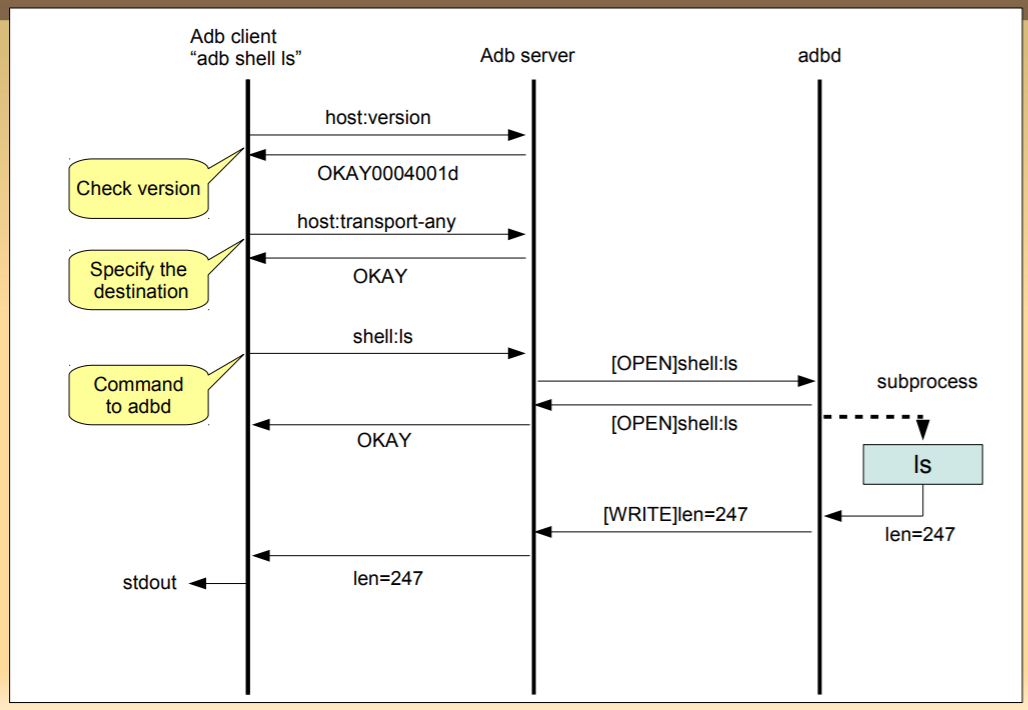
**Which runs commands on a device. The daemon runs as a background process on each device.**

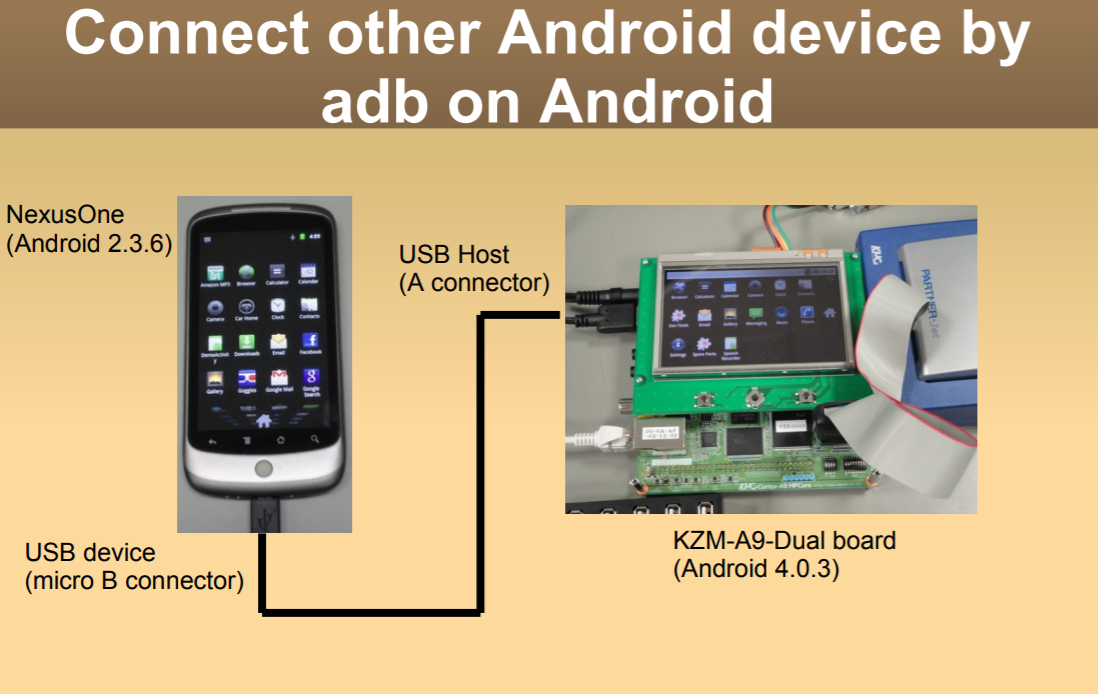
* **Server (adb start-server, adb kill-server)**

**Which manages communication between the client and the daemon. The server runs as a background process on your development machine.**

****

**How ADB works?**

****

****

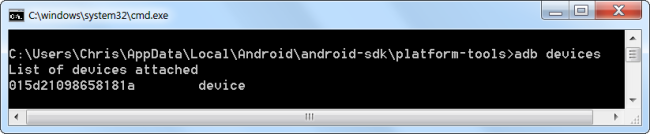
1. **Enable adb debugging on your device**

**To use adb with a device connected over USB, you must enable USB debugging in the device system settings, under Developer options.**

**On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options at the bottom.**

**On some devices, the Developer options screen might be located or named differently.**

**You can now connect your device with USB. You can verify that your device is connected by executing adb devices from the android\_sdk/platform-tools/ directory. If connected, you'll see the device name listed as a "device".**

****