Android Platform and Security

Overview

- Summarize the characteristics of the Android platform
- Summarize the security design features of the Android platform and apps
- Note significant recent changes to security and privacy

Android Basics

- Popular alternative to tightly controlled iOS
- Lower entry level cost than iOS devices
- Developers compete with hardware features
  - Disparity makes maintenance difficult
- Fragmentation of OS Versions and patches causes vulnerabilities that aren't patched
  - This is rapidly improving

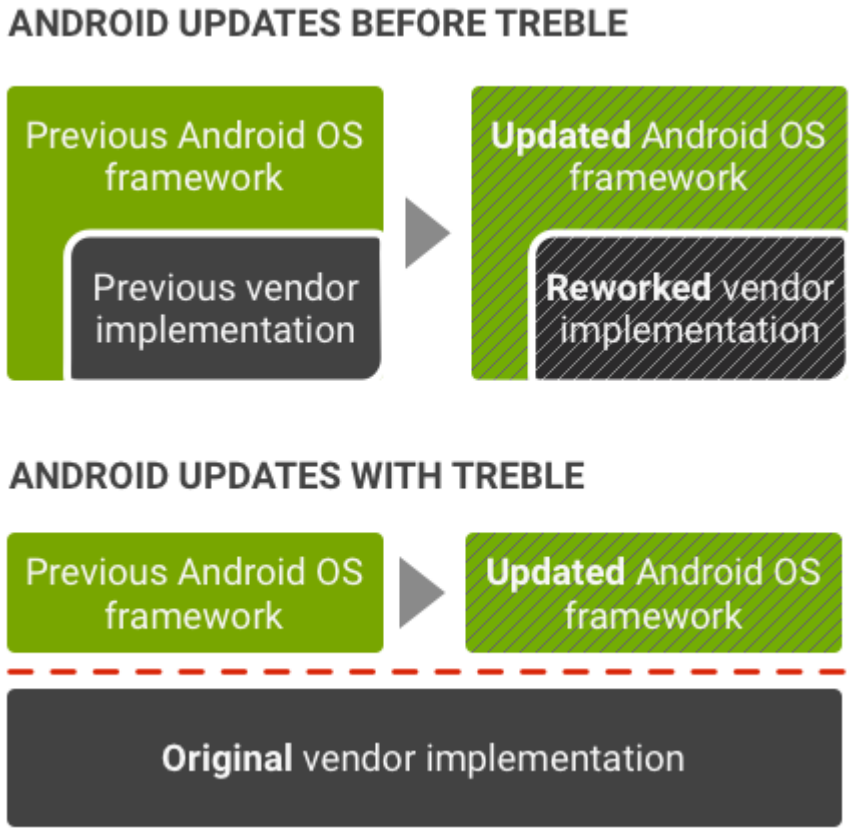
Android Supported Versions					
Version	Release date	API level	Last Update	U.S. Market Share as of 12/23	World Market Share as of 12/23
15	TBA	35	N/A	N/A	
14	October 4, 2023	34	January 2024	20.3	0
13	August 15, 2022	33		36.2	35.4
12	August 15, 2022, March 7, 2022	31,32		15.4	17.7
11	September 8, 2020	30		8.7	17.1
10	September 3, 2019	29	February 2023	5.1	9
9	August 6, 2018	28	January 2022	6.7	6.2
8	August 21, 2017, December 5, 2017	26,27	October 20321	0.9	2.9
7	October 4, 2016, August 21, 2017	24,25	October 2019	1.2	1.4
Legend: Old version	Older version, still maintained		Latest version	Latest preview version	

Source: https://en.wikipedia.org/wiki/Android_(operating_system)

current: https://gs.statcounter.com/android-version-market-share

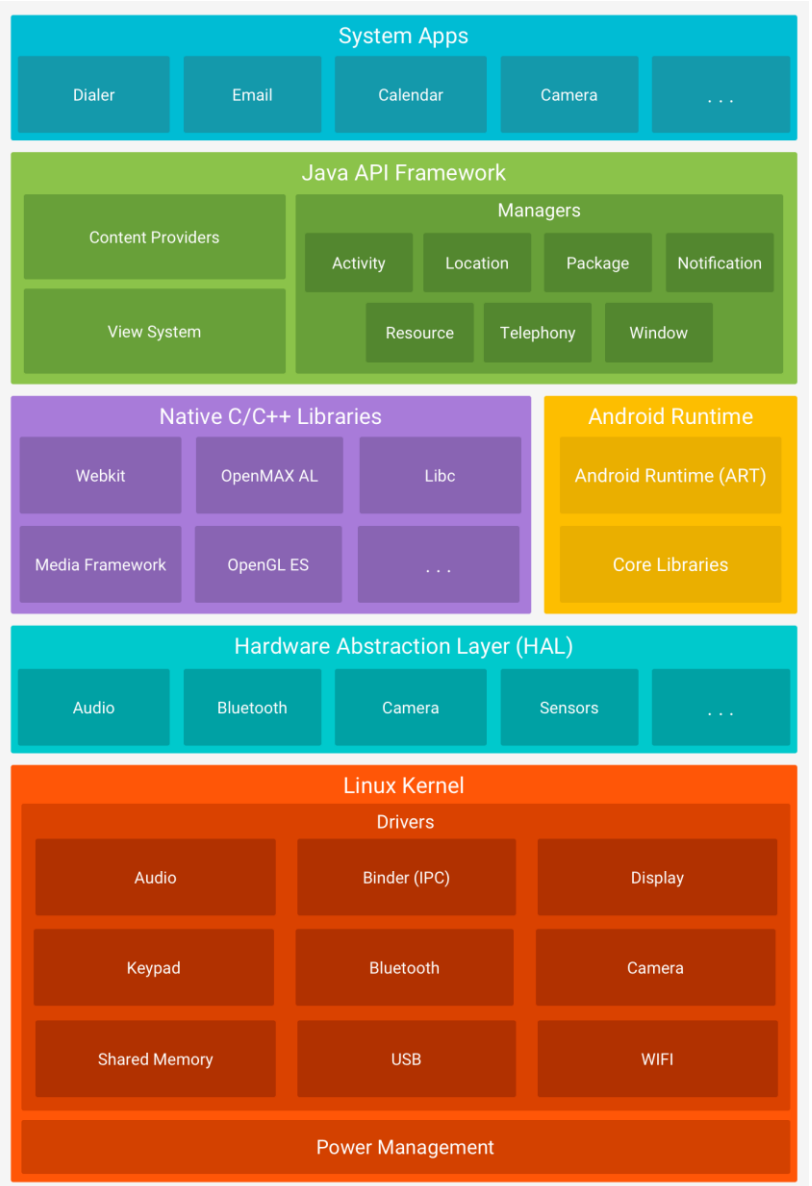
Addressed Fragmentation with Project Treble

- Treble is the implementation of a Hardware Abstraction Layer (HAL) and HAL Interface Description Language (HIDL) for Android
- OEM developers write their hardware layer to support the hardware using the HIDL conventions specified by Google, and Google provides the HAL that acts as a pass-through for the hardware
- The Android platform does not need to be modified for custom OEM
- The end user is able to replace the OS with new updates without risk of breaking device functionality related to hardware.



Android Architecture Overview

https://developer.android.com/guide/platform



Security Updates

- Google issues monthly security update bulletins
- Some issues may be MO installed software
- Each manufacturer and MO must test the updates before they are released
- The result is slow or no updates for some devices
- Combined with multiple versions of Android, this creates fragmentation
- Current security bulletins at: https://source.android.com/security/bulletin

Developer Requirements to Address Fragmentation

- Starting in 2019, the API level will advance each year. Within one year following each Android API release, new apps and app updates will need to target the corresponding API level or higher.
- Apps that are not updated are not affected, but stop running on newer APIs

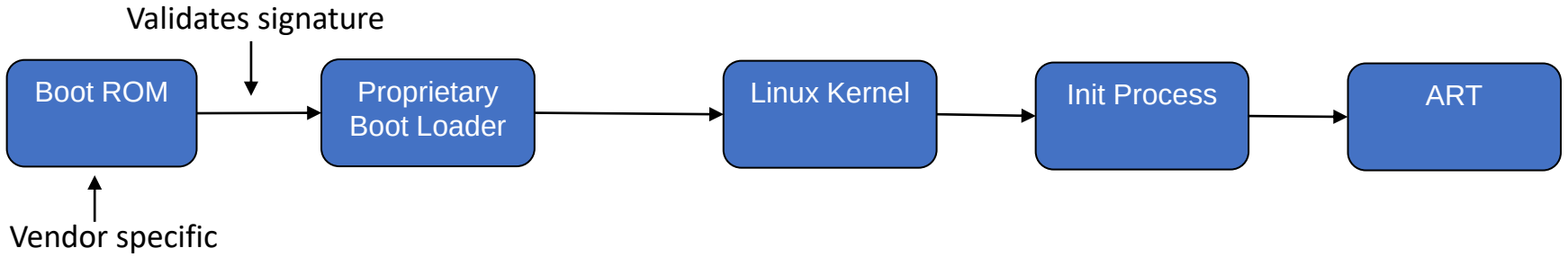
Android Architecture

- Processors – Older was primarily ARM (Advanced RISC Machine) processors
  - Newer versions support x86 architecture
  - 64 bit version starting with Android 5
- Kernel – Forked from Linux 2.6.30
  - Newer versions based on Linux 4 and 5
- OS Platform – Linux based
  - Does not include many familiar Linux tools
- Executable Architecture –
  - System code uses Executable and Linkable Format(ELF)
  - Apps use Android Runtime (ART) - ahead of time compiles Dalvik Executable (DEX) code
- File System – Linux ext4 internally, VFAT or FAT32 for SD cards

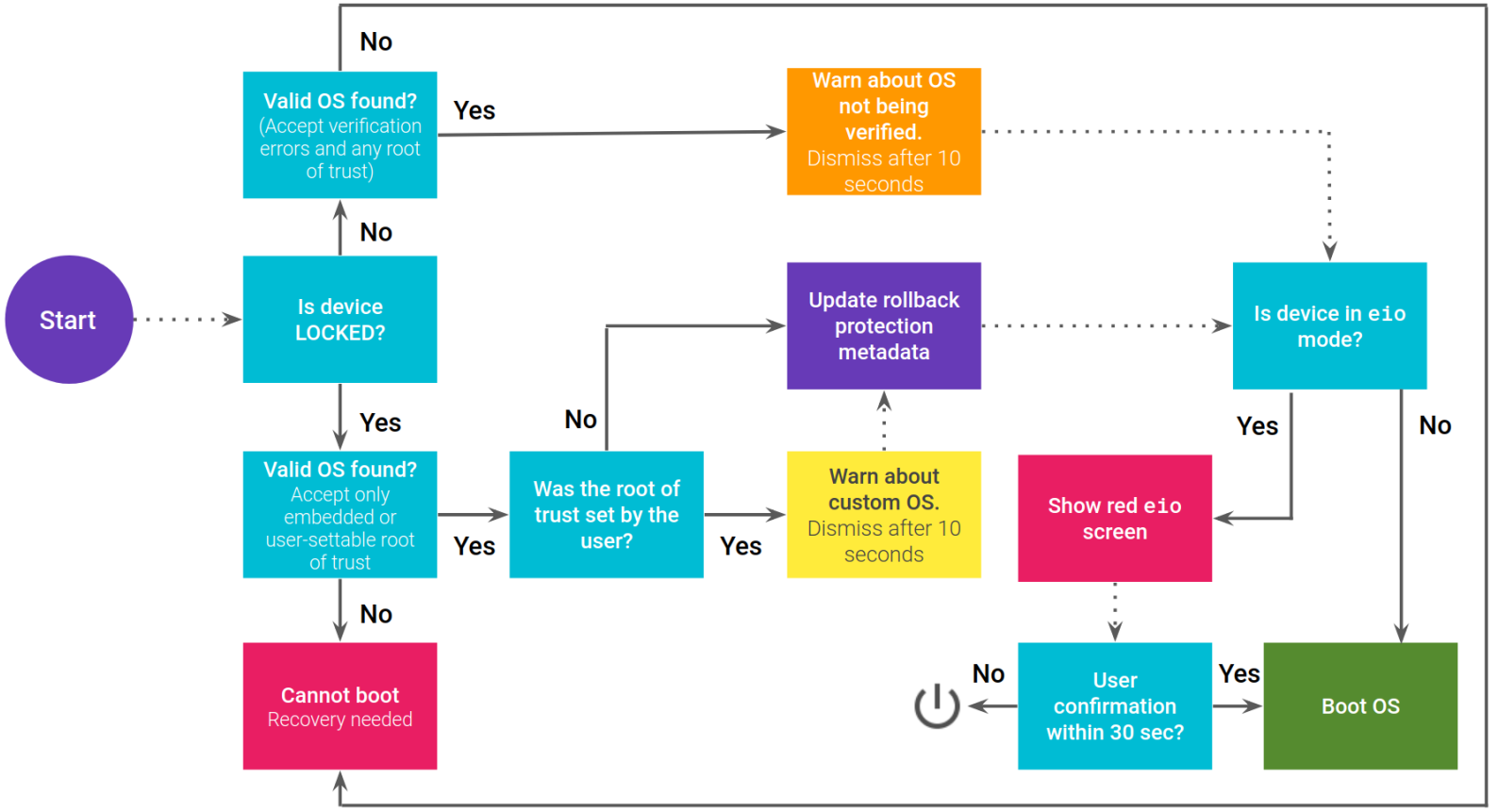
Android Security Design

Android Security

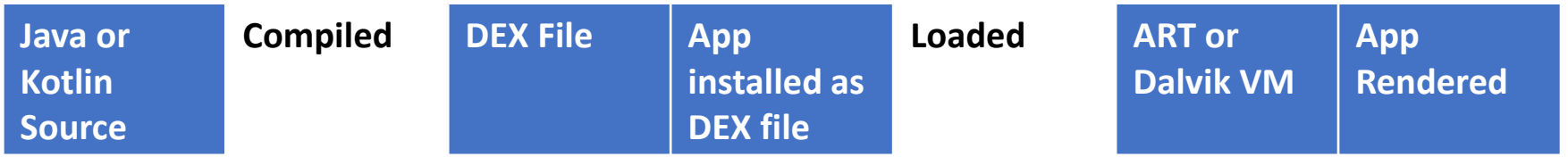
- Starts with Verified Boot Process (Android Verified Boot in version 8)
- Boot ROM is vendor specific – OEM is trusted source that is verified
- Each stage validates authenticity and integrity of software loaded
- Details at https://source.android.com/security



Android Verified Boot



Android App Execution



Android Runtime (ART)

Process	Code
Distribute App	DEX bytecode in APK
Install App	DEX to OAT Compiler
Run App	Native Code

Prior to ART, DEX code was installed. Installation and boot times were faster, but app launch times were slower.

Android Sandbox

- Linux UIDs and GIDs are used at the kernel level to restrict apps to their own sandbox
- As of Android 9, all apps run in their own SELinux sandbox
- Interaction with other apps using Android components like Intents
- Multiple apps from the same developer can cheat the sandbox architecture by using the same app signature key for multiple apps

Android Account Isolation

- Unix-like directory and rights structure used
- Each application receives a unique UID and GID at installation
  - Limited permissions within a designated application directory
- UIDs are based on app signature

```
root@android:/ # ps
USER PID PPID VSIZE RSS WCHAN NAME
system 118 32 140356 35008 ffffffff
corn.android.systernui
app_1 142 32 139732 37416 ffffffff
corn.android.launcher
app_9 157 32 141144 28132 ffffffff
android.process.media
app_8 201 32 137924 26376 ffffffff
corn.android.deskclock
app_20 232 32 136060 25424 ffffffff
corn.android.music
app_3 251 32 158532 27332 ffffffff
corn.android.rums
app_9 311 32 162768 30172 ffffffff
corn.android.gallery
app_15 335 32 155120 37108 ffffffff
corn.android.browser
app_24 365 32 149928 35500 ffffffff
corn.android.camera
```

Android Encryption

- File-based encryption allows different files to be encrypted with different keys that can be unlocked independently. Devices that support file-based encryption can also support Direct Boot, which allows encrypted devices to boot straight to the lock screen, thus enabling quick access to important device features like accessibility services and alarms.
- Metadata encryption - a single key present at boot time encrypts whatever content is not encrypted by FBE, such as directory layouts, file sizes, permissions, and creation/modification times. This key is protected by Keymaster, which in turn is protected by verified boot.
- Current Algorithm is 256-bit AES-XTS

Android Data Execution Protection and ASLR

- Protection against buffer overflows
- Java language includes string management and automatic bounds checking implemented in ART to mitigate memory misuse
- DEP prevents certain memory areas from being executed
- Address Space Layout Randomization – app starting address is randomized

Android Applications

Google Play Protect

- Play Store is digital distribution platform for Android applications
- Developers must register and agree to app evaluations
- Primarily uses automated scans and machine learning to check both apps and developers
- Play protect also tests community sourced apps and crawls other app stores
- Android device is scanned daily, and malicious apps automatically removed with user notification

Android Code Signing

- Unlike Apple, apps are not signed with a vendor CA signature
- Android developers sign their own applications with registered developer keys
  - Developers provide name, address and phone number and 2fa authentication
- Intended to simplify app updates:
  - Apps signed with the same key run with the same user ID on the platform
  - Developer can tell Android to replace local files if signed by the same key

Android Application Architecture

Activity

- This is a user interface whereby a user can enter data or interact with the application in some other way.

Service

- A service performs operations in the background: for example, playing music.

Content providers

- These provide information to third-party applications.
- A content provider can be seen as an interface that processes data in one process and feeds it to another independent process.

Broadcast receivers

- These respond to systemwide notifications such as “battery low” or “microphone unplugged.” The OS normally initiates these notifications or broadcasts, but trusted applications can also issue broadcasts

Android Intents

- Android Intents allow for Inter-Process Communication (IPC)
  - Between components in the same or different applications
  - Usually a request for an action to be performed
  - Can contain “extras” to pass information
- Intents can be reviewed statically or dynamically with adb or Drozer
- Developers can leave intents exposed
  - Facebook example of misuse - https://seclists.org/bugtraq/2013/Jan/27

Android Install-time Permissions

Install-time permissions

- Install-time permissions give an app limited access to restricted data or let it perform restricted actions that minimally affect the system or other apps. When install-time permissions are declared in an app, the Play Store presents an install-time permission notice to the user when they view an app's details page. The system automatically grants your app the permissions when the user installs your app.
- Android includes several sub-types of install-time permissions including:

  Normal permissions

  - These permissions allow access to data and actions that extend beyond your app's sandbox but present very little risk to the user's privacy and the operation of other apps.

  Signature permissions

  - The system grants a signature permission to an app only when the app is signed by the same certificate as the app or the OS that defines the permission.

Android Runtime Permissions

- Runtime permissions, also known as **dangerous permissions**, give an app additional access to restricted data or let it perform restricted actions that more substantially affect the system and other apps
- Runtime permissions must be requested by an app before it can access the restricted data or perform restricted actions
- Classified in permission groups
  - Calendar, Camera, Contacts, Location, Microphone, Phone, Sensors, SMS, Storage
- When an app requests a runtime permission, the system presents a runtime permission prompt for the group that the user can approve or deny once, when using the app or always

Example Permissions from Androidmanifest.xml

<uses-permission
android:name="android.permission.INTERNET">
</uses-permission>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION">
</uses-permission>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE">
</uses-permission>
<uses-permission
android:name="android.permission.READ:__CONTACTS">
</uses-permission>
<uses-permission

Example Android Normal Permissions

- ACCESS_LOCATION_EXTRA_COMMANDS
- ACCESS_NETWORK_STATE
- ACCESS_NOTIFICATION_POLICY
- ACCESS_WIFI_STATE
- BLUETOOTH
- BLUETOOTH_ADMIN
- BROADCAST_STICKY
- CHANGE_NETWORK_STATE
- CHANGE_WIFI_MULTICAST_STATE
- CHANGE_WIFI_STATE
- DISABLE_KEYGUARD
- RECEIVE_BOOT_COMPLETED
- REORDER_TASKS
- EXPAND_STATUS_BAR
- GET_PACKAGE_SIZE
- INSTALL_SHORTCUT
- INTERNET
- KILL_BACKGROUND_PROCESSES
- MODIFY_AUDIO_SETTINGS
- NFC
- READ_SYNC_SETTINGS
- READ_SYNC_STATS
- REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
- REQUEST_INSTALL_PACKAGES
- SET_ALARM
- SET_TIME_ZONE
- SET_WALLPAPER
- SET_WALLPAPER_HINTS
- TRANSMIT_IR
- UNINSTALL_SHORTCUT
- USE_FINGERPRINT
- VIBRATE
- WAKE_LOCK
- WRITE_SYNC_SETTINGS

Android Runtime Permissions

- READ_CALENDAR
- WRITE_CALENDAR
- CAMERA
- READ_CONTACTS
- WRITE_CONTACTS
- GET_ACCOUNTS
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- RECORD_AUDIO
- READ_PHONE_STATE
- READ_PHONE_NUMBERS
- CALL_PHONE
- ANSWER_PHONE_CALLS
- RECEIVE_MMS
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE
- READ_CALL_LOG
- WRITE_CALL_LOG
- ADD_VOICEMAIL
- USE_SIP
- PROCESS_OUTGOING_CALLS
- BODY_SENSORS
- SEND_SMS
- RECEIVE_SMS
- READ_SMS
- RECEIVE_WAP_PUSH

Risk - Excessive Application Permissions

- Some apps ask for permission to use many services that don't seem appropriate for app
  - **Example:** A flashlight app that wants to retrieve list of running apps and modify or delete contents of USB storage
  - Might be bad coding, not malicious intent
- User often has little understanding of why permission is required.
- It's suspicious when refusal of permissions has no effect on execution of application

Key Security and Privacy Updates to Android 14

- Hardware-assisted AddressSanitizer (HWASan) helps prevent bugs from making it into Android releases. It detects potential buffer overflow conditions
- With apps that share location data with third-parties, the system runtime permission dialog now includes a clickable section that highlights the app's data-sharing practices, including information such as why an app may decide to share data with third parties.
- Android 12 introduced an option to disable 2G support at the modem level, which protects users from the inherent security risk from 2G's obsolete security model. Android 14 enables this security feature in Android Enterprise, so IT admins can restrict the ability of a managed device to downgrade to 2G connectivity
- Added support to reject null-ciphered cellular connections, ensuring that circuit-switched voice and SMS traffic is always encrypted and protected from passive over-the-air interception

Source: https://source.android.com/docs/security/enhancements#android-14