Android Internals Syllabus

Android OS Layer

- Android historical overview
- The Android Open Source Project
- Google mobile services
- Platform Architecture (Kernel, HAL, ART, NDK, Java and Kotlin API Framework, System Apps)
- Android Core - Virtual Machines: JVM, DVM & ART
- Android OS boot process explained with focus on Zygote
- Android System Server
- Android System Services (exploring, activating and creating)
- Communication with System Services through managers
- Exploring application framework
- Processes & Threads
- Android Memory Management
- Android File System Overview
- Security: Device, Kernel and App
- Android System permissions (signature|privileged)
- Android Platforms (Wear, TV, Android for Cars)
- Android releases primary key features
- AndroidX support library and Android Jetpack

Android Application from inside

- App Components (Activity, Service, Broadcast Receiver & Provider)
- Intents and Intent Filter - communication between processes
- **App Manifest File**
- App Security & Permissions

Command line tools

- Android SDK tools
- **Android Debug Bridge(ADB) - In details**
- Basic Oprations (Record tscreen, capturing screen shots, , Copying files etc)
- ADB shell (am - Activity Manager,pm - Package Manager, dumpsys, keyeevnt, etc)
- Actviting the camera app and all other predefined published system components
- Activing in app components (Activity, Service, Broadcast Receiver) through intents
- **Exercise** - Sending an SMS, Navigating and more with ADB
- **Exercise** - Obtaining APK, extracting Manifest and sending actions

App vulnerabilities aspects

- Exploring apps hard-coded values (like strings) and detecting malware

- Preform APK reverse engineering to project and extract files (Both Kotlin and Native cpp code)
- Common Encoding (Base64, Hex) and Encryprion (XOR, RC$, AES) methods.
- Code scrambling with D8 & Pro-Guard
- Bonus - Dynamic Code Loading

Android Native Development Kit (NDK) and JNI

- What is in NDK? And Why?
- What is JNI and how it works
- Shared code libraries
- Using NDK and JNI full Android System undue example
- Hands on native calculator Android app
- Using NDK Manually (without Android studio)
- NDK advantages and disadvantages
- More on JNIEnv
- NDK's Stable APIs

AOSP Walk-through

- AOSP System Requirements (Hardware & Software).
- AOSP Installation Quick Guide (downloading and configuring AOSP source code).
- Building and Running AOSP (build procedures, emulators).
- Cuttlefish Virtual Device (Google's virtual Android emulator).
- Android Studio for Platform (ASfP).
- AOSP Main Folders (Bionic, system folders, Compatibility Definition Document (CDD), Compatibility Test Suite (CTS)).
- Google Mobile Services Test Suite (GTS).
- System SELinux Policies (SELinux policy definitions, structure, and compilation process).
- HAL Interface Definition Language (HIDL) (overview, purpose, deprecation in favor of AIDL).
- AIDL-based HAL(improvements in IPC mechanisms).
- SELinux Policy Compilation (Building and verifying SELinux policies into binaries).

Android Boot process in details

- Bootloader (initializes hardware, verifies OS, loads Linux kernel and ramdisk).
- Linux Kernel (hardware detection, mounts ramdisk, launches /init).
- Init Process (mount points setup, SELinux policy, core daemons).
- Android System Properties (global key-value configurations via getprop).
- Zygote Process (ART VM initialization, JNI registration, Java runtime start).

- System Server (essential system services: bootstrap, core, other).
- Application Startup (Zygote forks apps upon Activity Manager requests).
- System Server Services (bootstrap, core, and conditional feature-based initialization).
- System UI Components (status bar, navigation bar, split-screen, notification shade lifecycle).
- Activity Manager Service (AMS) (manages lifecycle and state of apps, services, receivers, tasks, processes).
- Window Manager Service (WMS) (manages window states, hierarchies, session communications).
- Keyguard Service (handles device locking/unlocking via SystemUI and Binder communication).

Hands-On AOSP - Apps, Services and Security

- Dumpsys Tool (system services diagnostic and debugging via adb).
- Adding Custom Apps to AOSP (integration of APK or source code with Soong build system).
- Custom System Services (AIDL-based) (creating, registering, and exposing new system services via Binder).
- SELinux Context and Rules (type-enforcement, labels, permissions enforcement via TE and context files).
- SELinux Enforcement Modes (permissive vs enforcing modes, auditing via audit2allow, changing modes).
- Linux Capabilities (fine-grained privilege management, especially CAP_SYS_ADMIN implications).
- Verified Boot (integrity checking, cryptographic chain-of-trust from bootloader to OS).
- Android Permission Model (Linux UID, DAC/GID mappings, runtime permissions, SELinux integration).
- DAC vs SELinux Checks (differences in permission enforcement via user/group versus SELinux policies).
- Process Management & Security Labels (ps -AZ, context labels, memory stats, kernel-level debugging).

Android in-kernel changes with focus on Binder

- Linux Kernel Android API changes (Binder, Ashmem, Pmem/ION, Wakelock, Early Suspend, Alarm, Low Memory Killer, Logger, Real — time requirements)
- Inter Process Communication (IPC) in Android OS
- Binder as the base for IPC in android
- Security with Binder Tokens (Incl Examples from Framework)
- Binder in details (Proxy, Stub, Parcel, binder objects and transactions with native code)
- Binding to Framework service using adb

- What is AIDL?
- Building a AIDL Binder based Service and Client by Example
- ashmem (Anonymous Shared Memory) vs Linux Shared memory
- ashmem in details on (MemoryFile, FileDescripor, read/write, pin/unpin)
- ashmem and pmem
- Memory pressure and Legacy LMK driver
- Android low memory killer daemon
- Wakelocks kernel add-on and it framework PowerManager implementation
- Logger and Android logging system

Course Schedule:

Day 1

9:00 - 16:00
Android OS Overview - Theory

Day 2

9:00 - 16:00
Android Application from inside - Theory
Android Debug Bridge - Theory + Practice

Day 3

9:00 - 12:30
App vulnerabilities aspects - Practice
Android Native Development Kit (NDK) and JNI- Theory + Practice

13:30 - 16:00
AOSP Walk-through - Theory
Android Boot process in details - Theory

Day 4

9:00 - 16:00
Advanced Android System Debugging and Security Internals - Theory + Practice

Day 5

9:00 - 16:00
Android in kernel changes with focus on Binder - Theory + Practice

Pre-Requirements

All Participants Must have Basic Linux OS knowledge (File System structure, using command line tools, basic memory management knowledge etc.)