Course Outline

Android Internals

Course Summary

Description

The Android Internals course trains the student on the internals of the latest Android OS.
Android Internals focuses on the common Android architecture, vendor modules and hardware abstractions using HIDL and AIDL for HAL

Objectives

At the end of this course, students will be able to:

- Build and run Android Cuttlefish VM, a configurable virtual Android device representing the current state of the AOSP development
- vnc to cuttlefish VM in the cloud
- Build and update the Android kernel; understand what is the Generic Kernel Image (GKI)
- Navigate the Android source code and correlate running processes to their original source code
- Open intermediate files to understand data marshalling for binder transaction
- Use intermediate files for method signature lookup
- Understand Android blueprint files
- Understand dynamic partitions - super.img
- Understand the purpose of the Generic System Image (GSI) and system_ext image
- Understand & customize Android, Linux and SELinux security policy & enforcements
- Understand how to create an OTA and what is A/B Updates
- Understand binder architecture - binder, hwbinder, vndbinder and their corresponding service managers
- Understand boot - init, zygote, system_server
- Use debug tools, dumpsys, strace, gdb, line2addr
- Use reverse engineers tools: dex2jar, jd-gui
- Create a vendor HIDL Service, a native daemon (introduced in Android 8, still in-use)
- Create a vendor HIDL Native client
- Create a vendor Java SDK Library (introduced in Android 10)
- Create a persistent app with control UI
- Create a vendor HIDL Java client
- Upgrade an HIDL to AIDL (hidl2aidl) (introduced in Android 11)
- Create a vendor AIDL service
- Create a vendor AIDL Java client

Topics

- Cuttlefish, Android Emulator and a physical device
- Building Android Auto, Android Phone, Android TV
- developer.android.com & source.android.com
- The device folder
- The hmm command
- The Discretionary Access Control (DAC) security in Linux
- The Mandatory Access Control (MAC) - SELinux
- Process Status (ps) and Private Application Files in /data/data
- The Soong Build System
- Generic System Image (GSI)
- Android Dynamic Partitions - super.img
- A/B System Updated and “fastbootd”
- The Android Linux Kernel
- The Generic Kernel Image (GKI)
- binder, vndbinder, hwbinder kernel drivers
- Understanding HIDL Android Modules
- Understanding AIDL for HAL Android Modules
- Vendor Interface Object (VINTF)
- Booting Android - the init process and .rc files
- Java Native Interface (JNI)
- Vendor Java SDK Library
- Dalvik, ART, JIT and AOT
- Signature, normal and dangerous permissions
- Platform certificates and signing
- Activity Lifecycle and intents
- SELinux public policy, macros, avc denials and audit2allow
- The Shared System Image (SSI)
- Android Pony EXpress Files - APEX

Course Summary (cont'd)

Audience

This Android Internals course is for developers who want to dig deeper than the standard Android SDK. It is for those who want to hack the system a bit in order to add system services and hardware support for non-standard components or port Android to completely new boards.

Prerequisites

To take this course, you should have a firm understanding of either Java, C++ or both.
You should be able to answer most of the following questions:

- What is the difference between a class and an object?
- What is the difference between static and non-static field?
- What is a header file?
- Be able to read and understand Linux shell scripts and be familiar with basic Linux terminal commands
- Familiar with following commands: ls, ps, cp, mv, pwd, cat, chmod, chown, mount, and similar
- Be familiar with users and groups in Linux and r/w/x permissions

Duration

Five days

Course Outline

Course Outline

I. Session #1 - 4 hrs:

A. Introduction
B. Reviewing the downloaded source code
C. developer.android.com - frameworks/base/core
D. source.android.com - frameworks/base/services
E. The device folder
F. The hmm command
G. Cuttlefish, Android Emulator and a physical device
H. Lab:
   1. Starting a Cuttlefish build
   2. Examining the build's output
   3. acloud setup / create
   4. adb shell
   5. cloud cuttlefish vnc

II. Session #2 - 4 hrs:

A. Boot process
   1. Verified boot - dm-verity
   2. init
   3. zygote
   4. system server
B. Security in Android:
   1. User IDs and the Discretionary Access Control (DAC)
   2. Mandatory Access Control (MAC) - SELinux
   3. Android Permissions
   4. Process Status (ps) and private application files in /data/data
C. Lab:
   1. Find ps & top commands' source codes
   2. Build configurations, Images and Partitions
   3. Android Make Build System
   4. AndroidProducts.mk and COMMON_LUNCH_CHOICES
   5. PRODUCT_MAKEFILES and makefile Inheritance
   6. Generic System Image (GSI)
   7. Generic Kernel Image (GKI)
   8. Android Shared System Image (SSI)
   9. Android dynamic partitions, super.img, A/B System updates and fastbootd
D. Lab:
   1. Download the Android Common Kernel; build and replace Cuttlefish' GKI & Kernel Objects

III. Session #3 - 4 hrs:

A. Discussion topics:
   1. Binder
   2. Understanding HIDL packages
B. Lab: HIDL HAL:
   1. Defining an hidl_interface module
   2. Reviewing the generated Binder Proxy (BpHwCpu) and Binder Stub/Native (BnHwCpu) intermediate build file
   3. Vibrator Service Source code walkthrough

IV. Session #4 - 4 hrs:

A. Discussion topics:
   1. Vendor Interface (VINTF)
   2. Init & rc files
B. Lab: hw service implementation:
   1. Extending and implementing hidl hal
   2. HAL implementation - kernel open, read & write
   3. hwservicemanager registration
   4. "Vendor:true" - vendor and product partition availability
   5. Adding a Vendor Interface (VINTF) object
   6. Adding a framework compatibility matrix file
   7. Location Manager Service code walkthrough

V. Session #5 - 4 hrs:

A. Lab: Vendor SELinux Policies
   1. Adding a new hal label in file_contexts
   2. Running dmesg to see avc denied errors
   3. te_macros - understand public sepolicy macros
   4. audit2allow
   5. system properties and property_contexts
   6. Allow rules for hwbinder and the hwservice_manager
B. Lab: Debug & test hw service:
   1. Creating a native test binary that acts as an HIDL native client
   2. Enable access permission to the kernel driver - DAC & MAC
   3. Understand tombstone file - addr2line
   4. strace
   5. gdb

Course Summary (cont'd)

VI. Session #6 - 4 hrs:

A. Discussion topics:
   1. JNI
   2. Vendor Java SDK Library
   3. ART Compilation
B. Lab: cpp HIDL client, exposed via JNI
   1. JNI_OnLoad
   2. cc_library_shared - adding
   3. Java_sdk_library
   4. If finished early, start Session #7

VII. Session #7 - 4 hrs:

A. Discussion topics:
   1. protectionLevel permissions
   2. Platform certificate
   3. Activity lifecycle
   4. Intent filters
   5. "Exported" activity
   6. Shared System Image
   7. dex2Jar
   8. jd-gui
B. Lab: Vendor platform persistent app
   1. The Application class
   2. BroadcastReceiver
   3. SELinux allow rules for a platform app
C. Lab: Adding a control activity (UI):
   1. Adding an activity source code and layout
D. Lab: Adding a Java HIDL Client
   1. Adding HIDL Java lib and instantiating a Java HIDL Proxy class
   2. Adding UI support for the new call
   3. /system_ext partition (SSI)

VIII. Session #8 - 4 hrs:

A. Android 11 introduces the ability to use AIDL for HALs in Android. This makes it possible to implement parts of Android without HIDL.
B. Lab Vendor AIDL for HAL:
   1. hidl2aidl - convert the legacy hidl to aidl files
   2. aidl_interface - create a new module
C. Lab: Adding a vendor service daemon:
   1. /dev/vndbinder - vendor binder communication
   2. Defining an .rc & .xml file for aidl interface
   3. Adding vintf for aidl
   4. aidl stable version and upgrading an aidl
   5. Adding SELinux allow rules for aidl service
   6. Implementing aidl generated methods
   7. Registering an aidl service with the vndservicemanager ("service list")
D. Lab: Modify the vendor platform app:
   1. Adding an aidl Java Proxy
   2. Adding SELinux rules

IX. Session #9 - 4 hrs:

A. Extra topics
B. APEX
   1. APEX File Format
   2. The apexd (APEX Daemon)
   3. Building an APEX
   4. APEX signing and update
C. Android Automotive
   1. Vehicle System Isolation
   2. Car 3rd Party API
   3. Vehicle Properties
   4. CarService
   5. CarSensorManager
   6. IVehicle.hal - OBD2 Live Sensor Data
   7. Android Auto Security
D. Open Q&A