Android – linux OS ; Java & Kotlin ; smartphones & tables ; OHA led by Google

2007 – Android SDK
2008 – Android 1.0 (Alpha)
2012 – Android 4.1 JellyBean
2014 – Android 5.0 Lollipop
2020 – Android 11.0
2024 – Android 15.0

©Operating System – manage resource ; interface blw hardware & software ; process mgmt ; memory mgmt ; nlw mgmt ; enforcing security

The stack of layers –

1) Linux Kernel – abstraction ; drivers for hardware instruction ; OS functions
wiFi, Bluetooth, USB, audio, display etc.

2) Libraries – native ; handle different types of data ; c/c++
media framework, webkit, libc etc.

→Android Run Time :

Java Source code → Java Compiler → Java Byte code → Dex Compiler → Dalvik Byte code
(.class) many → JVM → (.dex) one → DVM

→Dalvik Byte code : optimized ; low mem, low processing env ; .class → .dex

* Application Sandboxing happens at Kernel level for both native & OS applications

3) Application Framework – Activity Manager, Content Provider, Notification Manager etc.

4) Applications – Home, Contacts, Phone, Browser etc.

Android Application Components –

1) Activity – UI ; user interaction
2) Service – background processing
3) Content Provider – comm blw OS & app
4) Broadcast Receiver – data & dbms operations

} Alarm Clock Example

AndroidManifest.xml
- root of project source set
- Android build tools ; OS ; 4P
- app components ; permissions ; hardware/software features

Resources – additional files ; static conten ; bitmaps ; layout ; strings ; menu ; values

MainActivity.java : <intent-filter> ; MAIN action & LAUNCHER category

Intent – a messaging object you can use to ~~receive~~ request an action from another app component
startActivity ; broadcastIntent ; startService or bindService

Action – mandatory
Category – optional (B, A, 4, H, L)
Data
<intent-filter>
comp name, extras, flags

Explicit Intent – specify app to satisfy intent ; start component in own app ; class name known ; download a file in bg.

Implicit Intent – general action to perform ; class name unknown ; start component in another app ; location on google maps.

Intent Resolution – figuring out which activity handles the intent.

EXPLICIT

```
Intent intent = new Intent (getApplicationContext(),
                            Act2.class)
(this, hello.class)     ("unique name")
```

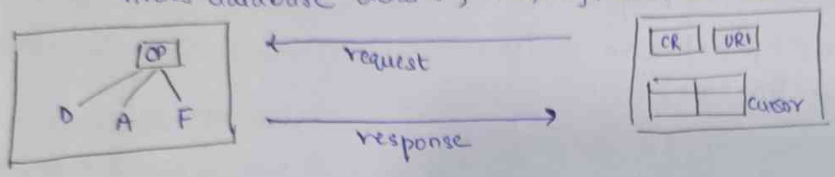
IMPLICIT

```
Intent intent = new Intent();
intent.setAction = (Intent.ACTION_DIAL);
                    VIEW, CALL
```

(*) Service – bg processes ; no UI ; IPC ; long-running
- Started (startService() / stopService stopSelf)
- Bound (bindService() / unbind service())
- Foreground – visible
- Background – no UI
- Bound – as long as atleast one app com. is bound to it.

- Unbounded Services – startService() ; onCreate() ; onStart() ; onDestroy() ;
- Bounded Services – bindService() ; onCreate() ; onBind() ; onRebind() ; onUnbind() ; onDestroy() ;

(*) Content Provider – provide content/data to app from sys/app ; content resolver ; central repository ; hides database details ; text, img, vdo, doc etc.



(*) Permissions – protect the privacy of user ; AndroidManifest.xml ; <uses-permission>
- Normal
- Signature
- Dangerous

6.0 (28) → runtime 5.1.1 (20) → install time
checkSelfPermission() → requestPermissions() → onRequestPermissionResult() ;

(*) SIPC – sandboxing ; org data & signals blw processes ; Binder ; client → Proxy → BD → Stub → Service
\dev\binder ; context manager ; token.

(*) Android Boot Process –
1) Boot ROM Code Execution – device hardware ; boot media ; boot seq ; bootloader to internal RAM
2) Boot Loader – IPL (external RAM) ; SPL (Android OS ; boot modes) ; SPL → kernel
3) Linux Kernel – root file system (rootfs) ; system & user data ; userspace → init ; caches/MMU
4) Init Process – first/root process ; init.rc ; <android-source> /sys/core/init ; logo
5) Zygote & Dalvik – first init proc ; DVM ; shared code
6) System Server – telephony, nlw, other functions.

(*) Adb commands – devices, connect, kill-server, start-server, shell, push, pull, logcat, install

(*) Android Partitions & File System Hierarchy – Bootloader. Boot, recovery, userdata, system, cache, radio

cat proc/partition
cat /proc/mounts
df

ls -al /dev/block/platform/msm_sdcc.1/by-name
cat /proc/emmc
cat /proc/dumchar_info
cat /dev/block/platform/dw_mmc
ls -l

busybox fdisk -l /dev/block/sda

Hell

ANDROID SECURITY

UNIT 1: Introduction to Android Security –

Android Forensics :

- extracting, recovering & analyzing data on a mobile device; 2010 Times Square car bombings, Boston marathon bombings; forensic soundness.
- Investigation & Preperation → Seizure & Isolation (screen lock, USB debugging, screen wake-up, screen timeout, static bags, airplane mode, ADM, MDM, Faraday bags / tents, RF isolation box). → Acquisition (manual, logical, filesystem, physical) → Examination & Analysis → Reporting.
- Preventing data alteration; wide range of OS & models; inherent security features; legal issues

Introduction to Android :

- Open-source; Linux OS; Java/Kotlin; OHA - Google
- Oct 2003 Android Inc → 2005 Google → 2007 Android SDK → 2008 Android (1.0) Cupcake (↳ HTC manufacturer) → 2012 Android 4.1 (Jellybean) → 2014 Android 5.0 (Lollipop) → 2021 Android 12 → 2025 15.0
- Andy Rubin, Rich Miner, Chris White, Nick Sears; Google Bouncer - Google App Verifier
- Android also provides a hardware ~~extraction~~ abstraction layer for the developers to create software hooks b/w Android platform stack & the hardware they want to port.

Android Architecture :

(↳ why is android so popular?)

- OS - manages resource, enforces security, hardware abstraction; stack of layers; Apache 2.0 liscence.

1) Linux Kernel - abstraction; drivers → instruction; OS functions; USB, audio, display; ex-camera.
2) Libraries - diff types of data; C/C++; Linux-libc, Android-bionic; surface manager, media framework, SQLite, webkit, OpenGL.
3) DVM/ART - DVM-register, JVM-stack; DVM-JIT, ART-AOT (↳ 4.4/5.0); DVM-loco proc, performance based optimized byte code; .class → .dex → DVM
4) Application Framework - run/manage applications; Activity manager, View System; Pkg Manager; Telephony manager, Resource Manager, Location Manager, Notif. Manager
5) Applications - system apps (/system & /system/priv-app/); user installed (/data)

Android Application Framework :

- Folder structure - AndroidManifest.xml, MainActivity.java, res (activity_main.xml), build.gradle (↳ drawable, layout, mipmap, values)
- AndroidManifest.xml - root, build tools, components, pkg name, permissions, requirements (↳ activity, service, broadcast rec, content provider)
- Activity - single screen + UI; MAIN action & LAUNCHER category
- Intent - messaging object; startActivity, broadcastIntent, startService / bindService
  - ↳ Explict & Implicit.

#Application Sandboxing:

- Linux based protection model; UID; permissions & priviledges; native + OS apps.
- SE Linux - Android 4.3, DAC → MAC, permissive mode, entorcing mode (Android 5.0), vold/netd etc.
- Android uses a fine grained permission model which requires the application to predefine the permission before compiling the final application package.

#Secure Inter-Process Communication:

- to organize signals b/w processes; binder framework; world readable/writable; /dev/binder/
- client → proxy → binder driver → stub → service
- Register service with context manager → CM (name → handle) → token assigned to service (32-bit) → client rec. token → start comms

#Application Permissions:

- manifest file <uses-permission>; Normal, Dangerous, Signature.
- Android 6.0 (23) → runtime; else below → install time
- declare permission in manifest file → check if perm is granted (check-self-permission) NO → requestpermissions() → onrequestpermissionsresult()
- GID → UID; platform.xml @ /system/etc/permissions.

Application Signing - CA, self; Google Bouncer, keystore, jarsigner, eclipse

#Android Boot Process:

1) Boot ROM code execution - boot media, boot sequence, boot loader → internal RAM
2) Boot loader - IPL → external RAM, SPL → Android OS (fastboot/rec), SPL → kernel
3) Linux Kernel - OS functions, rootfs, MMU/cache → rootfs → init process.
4) Init - first process, init.rc <and-src>/system/core/init, sys service process
5) Zygote & Dalvik - first init proc, animation, shared code, ZygoteInitClass → registerZygoteSocket() → preload Classes() → preloadResources()
6) System Server - power manger, AM, telephony reg, pkg mgr, context mgr

#Android Partitions:

- Bootloader, Boot, Recovery, UserData, System, Cache, Radio

cat /proc/partitions cat /proc/mounts df

Windows → FAT32, NTFS Linux → EXT2, EXT4

#Android Filesystems:

cat /proc/filesystems

- Flash-memory FS, Media-based FS, Pseudo FS

ANDROID SECURITY

UNIT 5

Request Interception & Traffic Analysis

- Insufficient transport layer protection is the third biggest risk in mobile devices according to OWASP Top 10.

Example: HTTP → login credentials & HTTPS → authentication cookies

Types of N/w Traffic Analysis

Passive Analysis

no active interception is done
capture & open using n/w analyzer
ex: wireshark

Active Analysis

actively intercept all n/w comm.
analyze / assess / modify data
ex: burpsuite proxy.

#Passive Analysis :

Def^n: tcpdump - save all network information to a specific file.

-s : Snarf few bytes of data from each packet instead of 65535 default.
-v : verbose output
-w : filename to write raw packets to.

Commands / steps :

```
adb push tcpdump-arm /data/local/tmp/tcpdump
chmod 777 tcpdump
./tcpdump -v -s 0 -w output.pcap
adb pull /data/local/tmp/output.pcap output.pcap
chmod 666 output.pcap
```

adb shell /data/local/tmp/tcpdump -i any -p -s 0 -w /mnt/sdcard/output.pcap
emulator -avd Android_Pentesting --tcpdump trafficapture.pcap

#Active Analysis.

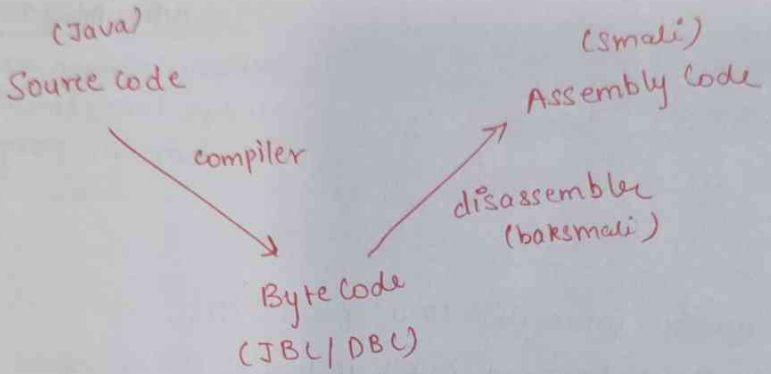
emulator -avd Android_Pentesting -http-proxy 127.0.0.1:8080
works for HTTP but HTTPs will give an error due to certificate mismatch and thus we wont be able to intercept traffic.

adb push portswiggerca.crt /mnt/sdcard/portswiggerca.crt

• Other ways to Intercept SSL Traffic - pulling cacerts.bks file from /system/etc/security; BouncyCastle ; Charles Proxy ; MITMProxy

Extracting sensitive files via packet capture:
(wireshark)

load .pcap file → search for "multipart" string → Follow TCP stream → Select RAW → Save with extension



.java → .class → .dex → .jar → .smali

Frida 10 Marks - Dynamic Instrument Toolkit
Scriptable, Portable, Free, Battle tested
Python, C, bidirectional, JS ; Commands,
Frida Tools - trace, ps, ls-devices, discover, kill
Flags - -U -Ua -Uai -D -j
API - Java.Enumerate loaded classes (), Java.android version, Java.available
Hooking - onCreate(), onResume, exit(), ret variable.

QARK - Quick Android Review kit
Static analysis, permissions, exp activities, code vuln, Intent vuln, webview, 3rd party
qark --apk <path> limitations - android, not perfect
↳ report generation.

MobSF -
static, dynamic, API testing, SCR, Integration, automation
Report - perm, activ, services, CP, BR, location etc
Frida +

• ADB Dumpsys - running services status ; adb shell service list.

Topic ..... Date .....

Book Information

→ Learning Android Forensics :

- 2010 Times Square Car Bombing, Boston Marathon bombings.
- "forensically sound" evidence in mobile forensics.
- Need for mobile forensics - personal info., online activity, crimes
- Mobile Forensics Process - Investigation Preparation, Seizure & Isolation, Acquisition, Examination & Analysis, Reporting.
- Chain of custody, anti-static bags, USB debugging, Android Device Manager (ADM), Mobile Device Management (MDM), faraday bags/tents, RF Isolation box.
- Acquisition Techniques - manual, logical, filesystem, physical
- Challenges in Mobile Forensics — data alteration, wide range of OS & models, inherent security features, legal issues.
- Why is Android so popular? - Apache 2.0 liscence.
- Each Android application runs its own instance of the DVM. [DVM - register based, JVM - stack based]
- DVM → JIT ; ART → 4.4/5.0, AOT
- System installed apps → /system ; /system/priv-app/ user installed - /data.
- Android security - app permissions, sandboxing, secure IPC.
- SELinux - Android 5.0, MAC → DAC, permissive/enforcing modes.
- Application Signing - CA or self, app developer → pvt. key, certificates
- init.rc → <android-source>/system/core/init ; PID 1
- Zygote - loadZygoteInitClass ; registerZygoteSocket() ; preloadClasses() ; preloadResources() → android.R file
- ADB → <sdk path>/platform-tools ; adbd ; port → 5555 - 5585 / 5037.
- adb devices - offline, device, no device.
- Types of logs - verbose, error, information, debug, warning, cricket, fatal, silent
- Rooting Android - superuser/root ; jailbreaking ; UID → /data/system/packages.xml
- Bootloader - recovery boot & fastboot → reflash partitions on your device.

/proc/partitions

- Partitions - bootloader, boot, recovery, userdata, system, cache, radio
- File Hierarchy - acct, cache (lost+found), d, data, dalvik-cache, dev, init, mnt proc, root, sbin, misc, sdcard, DCIM, system, build.prop, app, framework.

/proc/filesystems

- Filesystems - flash memory, media based, pseudo

Teacher's Sign. .....

- ADB Dumpsys - running services status ; adb shell service list.

→ Learning Pentesting for Android Devices :-

- Android provides a Hardware abstraction layer for the developers to create software hooks b/w the Android platform stack and the hardware they want it to port.
- Surface Manager, Media Framework, SQLite, WebKit, OpenGL.
- Linux (libc) → Android (bionic) ; radio & app-processes (adb)
- The core of Android security model is priviledge seperation.
- /system/bin , /system/xbin , /data/data/ , /data/app-priv.
- Android uses a fine grained permission model which requires the app to predefine the permissions before compiling the final application package.
- Group IDS : group gid = "net_bt" / "inet" / "camera"
- Google Bouncer , Google App Verifier. (jarsigner)
- A workspace is a location where all your Android application development projects & their files will be stored. ; AVD → ARM architecture.
- adb : -d (dumps of full log file) , -f (write to file) → logcat
- Shared Preferences are used by an application in order to save small sets of data for the application.
- reverse engineering : .dex → .smali (syntax similar to Jasmine)
- The main advantage of APKTool over JD-GUI is that it is bidirectional.
- All content providers have a uniform resource identifier (URI) in order to be identified and queried → content:// adb shell content query --uri
- Content provider leakage → android:exported = false
- OWASP Top 10 - weak server side controls, insecure data storage, insufficient transport layer protection, unintended data leakage, poor authorization & authentication, broken cryptography, client side injection, security decisions via untrusted inputs, improper session handling, lack of binary protections
- WebView Vulnerability - file://, data://

Topic ..... Date.....

Commands / code Syntax

Traffic Analysis :

→ Passive Analysis

adb push <tcpdump bin> /data/local

adb shell

cd /data/local

chmod 777 tcpdump

tcpdump -s 0 -v -w out.pcap

adb pull /data/local/tcpdump/out.pcap <path>

open out.pcap in wireshark .

→ Active Analysis — HTTP

emulator -avd Android_Test

--http-proxy 127.0.0.1:8080

WiFi settings in AVD

Burp > Options > invisible proxy > chk.

proxy in Browser > N/w tab — HTTPS

> manual proxy > Get cert > save

adb push ps.crt /mnt/sdcard/....

AVD > setting > Sec > Personal > SD

check ; (2) AVD /sys/etc/cacerts.bks

Bouncy Castle → mnt → mks.y offs 2

charles Proxy , MITM proxy.

Android Application Securing Auditing & Pentesting :

→ Drozer

adb connect — adb forward tcp:31415 tcp:31415 — drozer console connect

run app.package.list <module name> -f <search>

run app.package.info -a <pkg> — run app.package.attacksurface <pkg>

run app.package.debuggable <pkg> — run app.provider.finduri <pkg>

CP leakage / Injection ← run app.provider.query <uri>

run scanner.provider.injection -a <pkg> ← projection -- projection ; selection -- selection

Android Application Components.

BR: <receiver> , Context.registerReceiver() , android:priority , sendBroadcast()

CP: ContentResolver resolver = getContentResolver(); Cursor cursor = resolver.query();

S: startService() ; onCreate() ; onStart() ; onDestroy() ; stopService() ; stopSelf() ;

bindService() ; onCreate() ; onBind() ; onRebind() ; onUnbind ; onDestroy()

Teacher's Sign. .....

ADB Commands.

adb connect <ip>
adb devices
adb kill-server
adb start-server

adb shell
adb push <local> <remote>
adb pull <remote> <local>
adb logcat -b -c -d -f -v

adb install <file>
adb uninstall <apk>
df, ps, mount

Android Boot Process.

adb shell ps ps | grep "init"

Mobile Application Security Pen Testing:

1) Insecure logging:
logcat | grep "diva"
Log.e pidcat

2) Hardcoding Issues:
• equals ("vskey")

3) Insecure Data Storage.
privateSQLiteDatabase nDB;
createTempFile();
/data/data/shared_prefs;
getexternalstoragedirectory();

4) Input Validation
•'= '1' or '1 file://

5) Access control.
am start -a <intent filter>
chkpin = False --e3

Android Manifest.xml

<uses-permission android:name = "android.permission.SEND_SMS" />
checkSelfPermission(); requestPermissions(); onRequestPermissionsResult();
build.gradle - location of code entities when building project.
res → drawable, layout, menu, values

<activity android:name = "MainActivity" />
action = MAIN category = LAUNCHER

Reverse Engineering.

apktool d <file.apk>
apktool b <folder>
• java → .class → .dex → .apk

dexdump <apk file>
-f, -d, -o

d2j-dex2jar -d <apk>

jd-gui classes_dex2jar.jar
~~dexdump~~ hd -b -c -d m -o
jadx → .dex in gui.

Frida

frida-ps -U -Ua -Uai -D -j
frida-trace frida-discover -ls-devices
frida-kill frida -U -l <pkg name> -f

Intent

android.intent.action.BOOT_COMPLETED (*)
android.intent.action.REBOOT

Exp: Intent intent = new Intent(getAppContext(), Act.class);
startActivity(intent);
startService(intent);
context.sendBroadcast(intent);

Imp: Intent intent = new Intent();
intent.setAction = (Intent.Action_Dial);

<intent-filter>
<action android:name = "(*)" />
<category android:name = B,A,4,H,L />
<data android:scheme = "https" />
</intent-filter>.