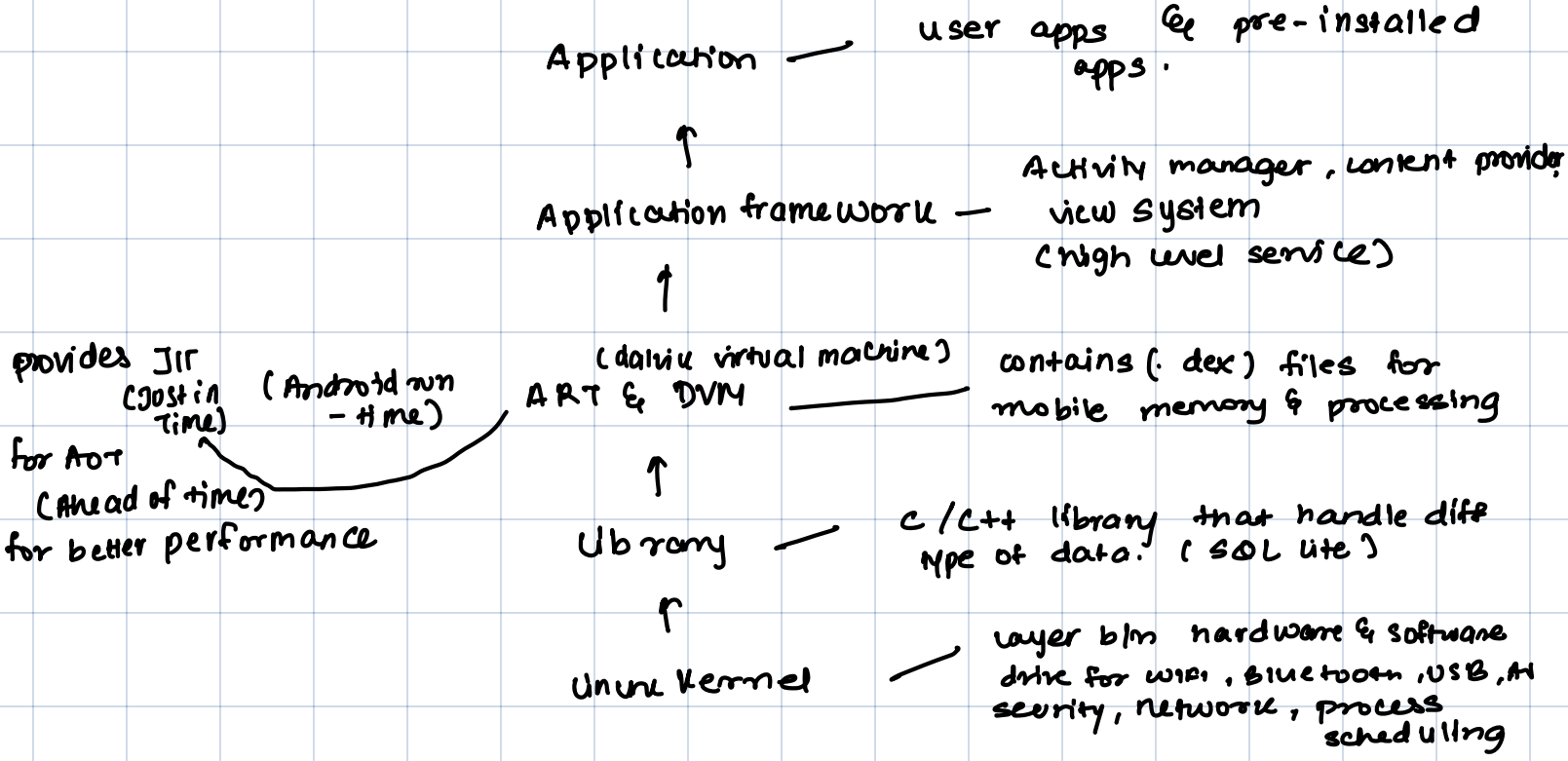
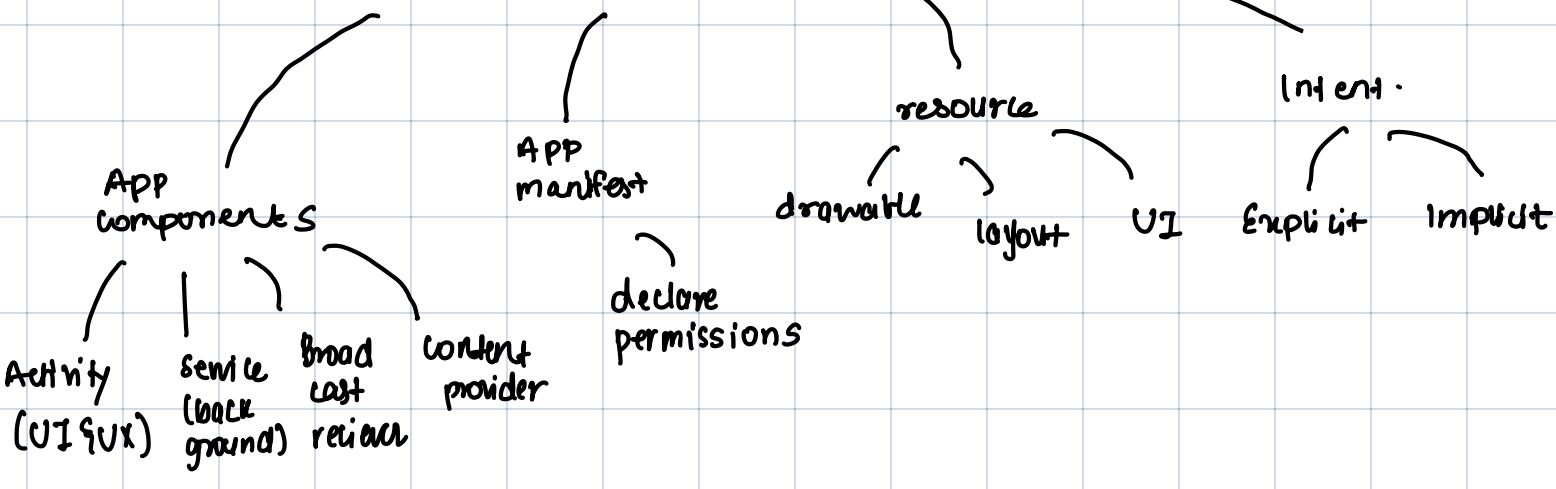
TA-1

Q.1) Architecture. (L L D AF A)



(C, M, R, I)

2) Android app framework



(A S B C)

Android application components

- **Activity** — UI & user integration
- **Service** — Background services of Android
- **Broadcast reciever** — Comm^n b/n OS & Application
- **Content Provider** — Handle data & db operation.

Eg - Alarm

App khola & set alarm - UI

alarm ka data save hoga - content provider

service continously time dekhega - service

Jaise time match hoga alarm bajega - Broadcast reciever.

Android manifest .xml

- saare activities define karo
- saare permission share karo.
- <activity> - <service> - <reciever> - Broadcast reciever <provider> - content provider
- what type of h/w /s/w our app needs

Intent

- Action (Action jo perform hoga)
- Category (Extra info abt action)

Explicit

To start a component in our app. Because you know class name

Implicit

They declare general action which allows component from other app to handle.

3) **sandboxing**

- uses linux user based
- unique UID
- Prevents unauth access
- Kernel-level isolation.

4) **Inter-process communication.** (BBCA)

- Binder framework
  - secure comm bln processes.
- Binder communication model
  - client server, use proxy
- context manager
  - resolving names to service for comm^n.
- Android IPC
  - Intent, content. AIDL, Messenger

5) **DVM**

- designed specifically for android to run apps
- Java to (dex)
- DVM runs apps in isolation
- replaced by ART
- register based unlike JVM

Java source code → Java compiler → Java Byte code → dex compiler → Dalvik Byte code

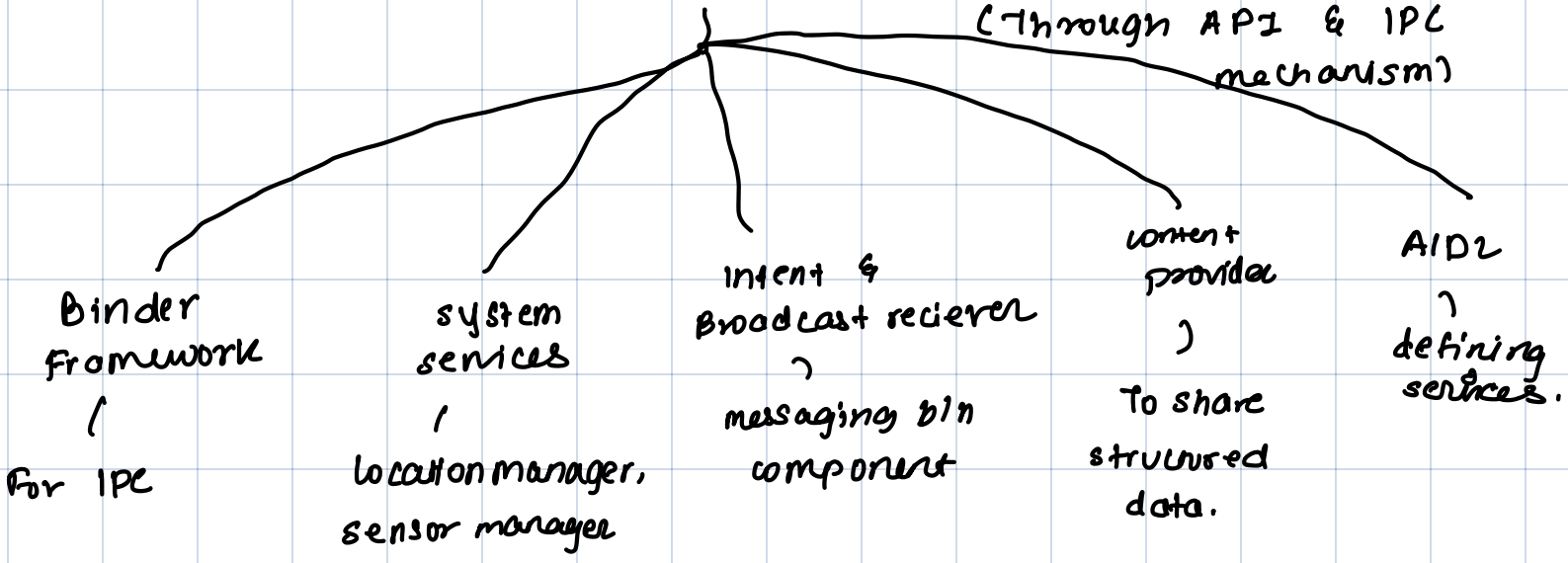
6) **Permission**

(apps must request permission before accesing data)

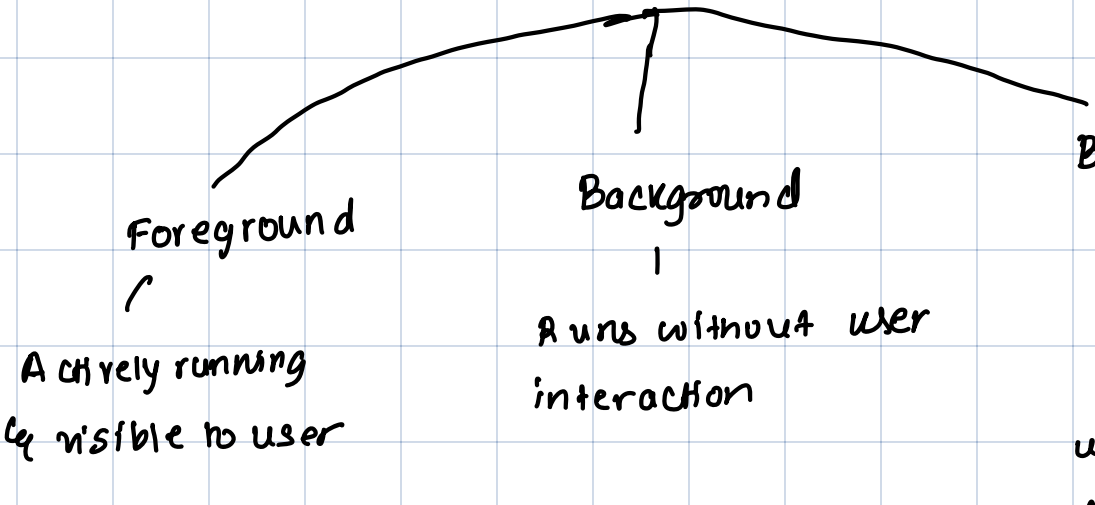
declared in android manifest.xml.

- normal
  - automatically granted eg- timezone
- signature
  - if requesting app asks same as defining ap
- dangerous
  - user approval.
  - eg- location, contacts

7) How do android communicate with system app



8) Diff type of services.



Lifecycle ⇒ onCreate() → onStart() → onBind

- onCreate(): when service created
- onStart(): called when service starts
- onBind: components Bind to service
- onDestroy(): service terminate

Q. Content Provider → Allows apps to share data w/ each other through URI (Uniform resource Identifier)

Q. Content resolver → Data ← content resolver → content provider
intermediatory bln data & content provider.

Q. cursor → retrieves & iterates over query results from db.

MAD - SEM ....

* ADB commands

1) adb devices - lists all devices (offline, device, no-device)
2) adb connect device_ip-address
3) adb kill-server - server band karta hai
4) adb start-server
5) adb pull <remote> <local> - remote se local file lena
6) adb push <local> <remote> - Local se remote ko file bhej na
7) adb logcat - sare device ke logs for debugging.

Android Boot Process

Boot ROM code execution

- Dono Physical memory (RAM) & NAND (processor) ko jagata hai. (load)

Boot Loader

Before OS starts

1. IPL - Initial prog. Load
2. SPL - secondary prog. Load

↓

Linux kernel

- Responsible for process managment memory managment, enforcing security. cache start hoga kernel se init process chalu krega

↓

Init process

- first process. root process of all other process. It will parse init.rc
- You will se android logo

↓

zygote & dalvik

- zygote - First init process
- It initialises Dalvik VM & multiple instances for each process.
- Uses shared code across VM to save memory.

↓

System server

- all core features like telephony, network & Activity, context manager

Pen-testing Strategy / Approach

Topics
- drozer
- Apk pentesting
- JD GUI
- Genymotion

Black-Box

- No access to source code
- payloads in URL, text field, variables. Provide valid & invalid i/p to see response
- understanding of payload injection
- Good when external attacker & no access to external code
- Time consuming

White-Box

- Full access to source code
- Understand the front & Back end.
- static & dynamic analysis
- understanding of reverse engineering
- used on mobile apps when u have > 50 % of code
- Expertise in binary code analysis reqd.

Dalvik

- original runtime environ. It executes .dex
- converts Java into .dex
- Java → .dex → DVM
- Hard to modify .dex
- like Java bytecode
- running eff. apps on And

smali

- Human readable rep. of dalvik byte code. It is used for reverse eng.
- Used to disassemble & modify .dex files.
- .dex → smali → modify → Back to .dex.
- easy to modify
- like assembly code
- Reverse eng / Pentesting apps

(convert .dex to Java)

JD-GUI (used to decompile .Java class file)

- It compiles .class to readable code & .Jar file.
- Used for reverse engineering, debugging JAVA apps
- Java .class file
  ↓
  Java source code
  ↓
  method, variable, class structure
- Easy with a GUI, no need for CLI.
- Helps recover lost code, analyse JAVA malware.

Hexdump

- converts binary to hexadec/ASCII
- Helps in inspect raw file content to find pattern
- used in forensic, pen testing, debugging executable
- Binary file
  ↓
  Human readable
  ↓
  display in Hexa.

✱ OWASP Top 10 security risk. (P D C A I C A C C R L)

(Plat, data, com, Auth, insuff cryp, Authoriz, client code, C, rev, lack of BP)

- M1 - Improper Platform usage
- M2 - Insecure data storage
- M3 - Insecure comms
- M4 - Insecure Authentication
- M5 - Insuff. Crypto
- M6 - Insecure Authorization
- M7 - Client code quality
- M8 - Code Tampering
- M9 - Reverse Engineering.
- M10 - Lack of Binary Protection

Drozer

(leading security testing framework).

- open-source audit & attack framework
- Helps security identify vul.
- Allow testers to
  - simulate attack
  - execute Java code

Features

- Automates testing
- Pre-build Android exploits
- Malicious to observe IPC endpoints
- CLI-based commands

Drozer client - send commands to server

Server - Runs on target Android device

Santoku Linux - Security testing Linux based.

Installation-

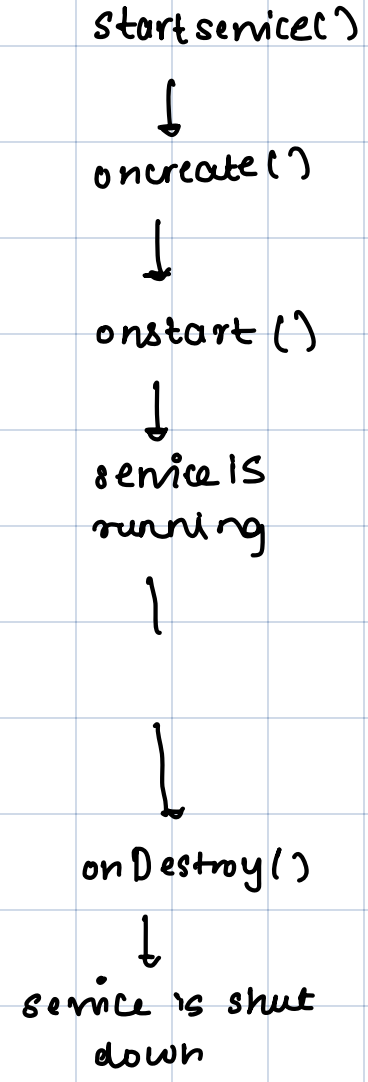
1) install drozer client
2) install drozer apk
3) Start drozer server
4) connect using ADB.
5) open drozer console.

commands

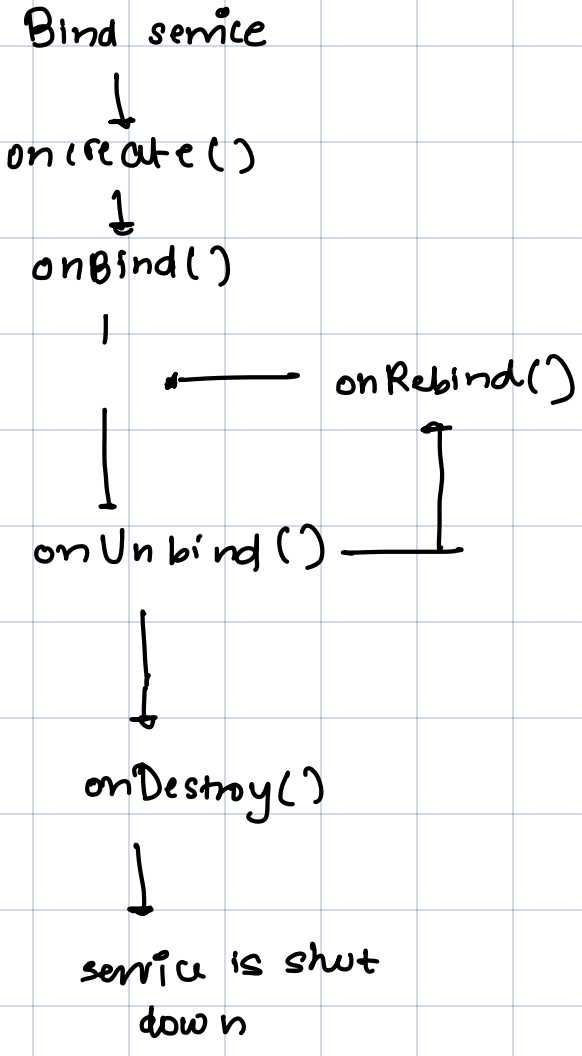
```
dz> run app.package.list
dz> run app.package.attacksurface [package_name]
```

genymotion.

Life cycle of Android Service

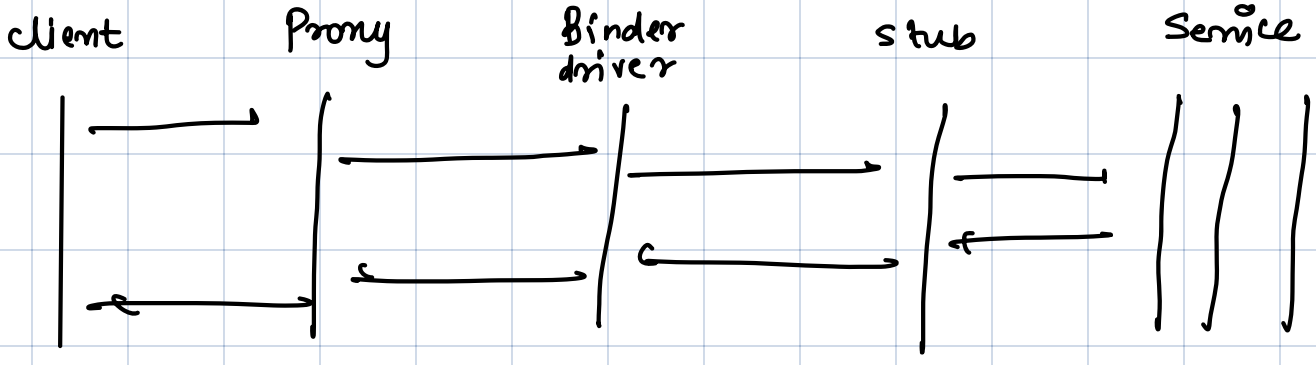


unbound service



Bounded service.

Secure Interprocess Communication.



Android File

Based on linux

Har app ka sandbox

Format
- ext4 (secure)

priv- /data
pub- /sdcard

root user can access/ modify

Android Partition (BRSOC)

Boot → Recovery → System → User → Cache

Boot (Initialise)

Recovery (used for maintenance)

System (Android data)

User (personal)

Cache (Temp. File)

Gapps (Google Apps)

→ pre. installed

cloud backup

doesn't run in custom (ROM) due to it being google app

supports location

can't run play store

used for authentication

APK Pentesting

Reconnaissance (collecting info about apk)

↓

static Analysis (reviewing source code & config files)

↓

Dynamic Analysis (Running the app & monitoring behaviour)

↓

Exploitation (Attempting attacks)

↓

Post-Exploitation (Gaining access & data extraction)

↓

Report & Mitigation (documenting vulnerability & recommending fixes).

Genymotion

High perf. emulator
simulates diff android device
faster & more features

Features
- multiple android d
- Fast & lightweight
- Root access
- GPS simulation
- network sim.
- H/W sim.

preferred over AVD
for pentesting & forensic research.

End- sem

Frida

(dynamic instrumentation toolkit that allows to inject scripts in running android apps).

Used

- runtime hooking (also function hooking)
- Bypassing security mechanism
- Inspecting & modifying memory
- Reverse engineering without modifying apk.

commands

frida-ps - list process

frida-trace - Hooks native methods

frida-ls-devices - lists connected devices

send() - msg from hook to console

Workflow

1) Install Frida server

↓

Run Frida on PC & connect via ADB.

↓

Attach Frida to an app. (scripts)

↓

Inject scripts to hook or replace function

↓

Observe / alter behaviour in real time

Features.

1) Hook fn. Hook into Java function to watch them change their behaviour

2) Bypass sec - We can bypass root/ Jailbreak

3) See how app behaves when certain actions are performed.

4) No repackaging needed (no need to modify APK.

Mobile-SF (All in one automated pen-testing framework)

Use-case

- Reverse Eng, vulnerability det., malware analysis

Features

1) Static Analysis - Analyses APK without running the app

   decompiles app & checks for hardcoded secrets, insecure permissions

   decompiles .apk to smali/Java using JADX

2) Dynamic analysis - Runs the app in emulator to observe run-time behaviour (network req etc.)

   calls for API to check security (HTTP instead of HTTPS).

4) gives an overall security score

Working

1) upload apk

2) static

- extract permission, manifest, activities
- Hardcoded credentials -

Dynamic

- observe n/w traffic, run-time permission & fil sys changes

3) Generates a detailed PDF.

eg :- android:allowBackup : "true"
http instead of https

* Fully automated. generates reports. detects insecure components, permissions, code & run-time behaviour

DIVA

(Damn insecure vulnerability app)

- It covers most of the OWASP Top 10

1) SQL injection

- App uses unsanitised input in SQL, allowing attacker to modify the logic

comm - ' OR '1' = '1

sol - use parameterised queries (not string concat enation).

2) Insecure logging

sensitive data (password) is written to system logs

comm - 'adb logcat

sol - grep "login activity"

3) Access control issue

exported activity accesible externally w/o permission check.

comm - adb shell am start -n <pkg> /. AdminActivity

soln - set android:exported = "false".

4) Hardcoded secret

secret like API key embedded in the code - attacker can extract them.

comm^n - decompile apk with JDAX

soln - store in keystore API or fetch from backend.

5) Webview Injection

User input is injected into a webview → leads to javascript injection (XSS).

comm - <script> alert(1) </script>

soln - solve it using Html.escapeHtml()

6) Shared preference

storing user creds or tokens in unencrypted xml file in the internal storage.

comm

soln

7) SD card storage

saving sensitive data on external storage (/sdcard/) where any app can read it.

comm - adb shell cat /sdcard/data.txt

soln - use priv internal storage

8) Temp file storage

sensitive data written to temporary file (eg. cache) - left behind & exposed

comm - adb shell ls /data/data/<package>/cache/

soln - Delete temp file immed.

9) SQLite storage

plaintext sensitive data inside app database can be dumped using ADB or sqlite shell

comm - adb shell sqlite3 /.../dbname.db

soln - Encrypt the DB.

Unit 5

Traffic Analysis

- App. can sensitive info in their n/w data, so finding it is very imp.

Passive

Active

Passive

- save all the n/w info to a specific file & later analyse it.
- Use tcpdump. Later analyse it using wireshark.

work flow

Install tcpDump
(command line utility to capture traffic)
(cd / tcpdump)

↓

save traffic dump to file
(tcpdump -s 0 -v -w out.pcap)

↓

open the pcap in wireshark.

Active

(Actively monitor & control traffic b/n android app & the internet using proxy tool like Burpsuite).

Workflow

Install & open Burpsuite on laptop

[1) open Burp suite
2) setting > proxy > proxy listener.
3) set port → 8080]

↓

set proxy on android

[1) set lap & phone on same wifi
2) Go to proxy on WIFI by pressing modify network
3) proxy : manual
Hostname : Laptop IP
Port : 8080]

↓

Install Burp cert on phone

[1) chrome pe http://Burp
2) download CA cert.
3) setting > security > Install an choose. crt file]

↓

open any app

[If traffic should be shown in Burp Intercept tab]

Features

- view request / response
- modify data
- catch session cookies
- Bypass weak security