MOBILE SECURITY

Mobile devices contain most sensitive info. It is always connected to outside world all time. There are chances info. can go out.

Android → most vulnerable OS ∵ it is open source. Any app can be installed.

Firmware → Android/IOS.
→ large users
→ more choices
→ vulnerable

Applications → store, access & send data.

Even naive user can develop app. It can have bugs or be malicious, so our private info. can be read/leaked.

Android App. is modular in nature i.e., consists of modules which we can plug & play.

(I) Building block of apps → component

① Activity → responsible for GUI interface through which user can interact. (I/O).
(most Imp. component).

whenever user is doing some activity, he/she can't Input/output some other activity.
At a pt. of time, only one activity interacts with user in terms of I/O.

(2) Service → runs as a background process, user can't interact.

(3) Broadcast Receiver → receiving notification. always stays connected to a port.

(4) Content-Provider → provides interface for app to interact with sql database if it exists.

All these components are java classes:-
define abstract class, extend it and make a component out of it. (helps in debugging) since modular. (Reuse the component in other app as well). Reduce error space.

Android based on linux kernel
↳ Java. ↳ C/C++

Seperate JVM created for each app in android.
(DVM)
↳ Dalvik virtual machine.
to keep our apps isolated from each other.
This property is called Sandboxing.
If app1 has some bugs, other apps will run smoothly. we don't want entire system to shut down. If malicious app gets installed,

Shared resources → camera (H/W), GPS, system apps (contacts), SMS, SD card, sensors

GIDs → group ids (How apps will use these shared resources).

Each shared res. is given unique no. When app installs, we must as a developer know what shared resources app will use.

it won't affect other apps which are running.

Local variables of one app can't be accessed by other apps.

How this isolation is implemented?

Each app is assigned separate user id's (UIDs). Each app is separate user. System apps (root) also given id.

(II) Manifest File (.xml) (most imp. file).

All metadata about component.

→ info about all components that app have.

→ privileges which are associated with this particular app. (what resources it can access).

→ For each GID, UIDs are listed and android framework does this.

Developer will mention it in manifest file. in the form of labels/permission.

<user-permission> grant SD card access.

string → GID. (install)

Front & back camera → same GID.

SD card & external storage → same GID.

→ Restrictions → who can access my app.

Define some labels for components.

for user we define some permission.

For eg:- accessing locat" is a privelege the app has but with whom we can share it is restriction.
Attach permission → viewer → only. friends.
(user-defined permissions)
permiss" associated with component & not app.

→ Capabilities of app. are defined by action strings. (intent filter is associated with component). With each component, we can apply various Intent filters.

As an app, I am the one who is capable to perform a special action.
To open a pdf, whenever a pdf file comes, ask me to perform the action.
A component can perform multiple actions.
~~Ea~~
Entire app is signed.

Sometimes, ~~the~~ app1 may want to use or access some data from app2. We want mech. for interapp communication.

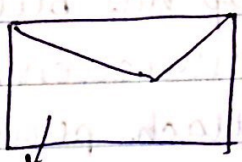
Inter-App communication

Intents are message passing objects used for Inter-App communication.

App1 talking to App2 is similar to their components talking. We call is inter-component communicatn (ICC). generic.
Components can be on same app or diff.
Same-app → intra-app, diff app → inter-app

Intent has 3 imp. components —

① Component name.



Package Name (URL) + Component Name = Add
ACTION -

address → comp name (Package Name & comp Name).

Domain Name.
com.android.example.ApplicationName.
Package Name

We can't install 2 apps with same pkg name (not domain) in one device.
We give UID to package name.

Reference monitor channelizes all the intents generated.
App1 sends intent to RM, it will see for whom it is meant for & check all restrictions on the component.
Developer just mentions all restrictions in simple string.
Checking & coding → modular.

Intent → system call → goes via kernel.

I can send an intent saying what actn need to be performed. App1 wants to open some content in browser. Need not specify recipient always. Check all apps registered for the actn.

* If registratn is via activity, all the apps registered are shown to users and they can choose.
* service → what if there are multiple apps that can provide actn, but GUI absent, so no choice. RM will randomly choose service app from available optn.
* Broadcast Receiver → RM will send the notificatn to all the apps. (all apps that can perform the actn.)
* CP has no such thing called Intent filters.

whenever an app wants to send a message to other app which has to be sent via broadcast receiver, then the message is sent to all the (multiple) ~~its~~ broadcast apps registered.

There are 3 security mechanism adopted by Android:-

1. Sandboxing.
2. Signature → all apps are signed by develope

then only app can be installed or uploaded on store. This provides accountability.

③ Permissions.
Protect our sensitive info.

Manifest

→ component info → Tags
→ Privileges → Permissions (App-level).
→ Restrictions → user-defined permissions (component level)
→ capabalities → intent filters (standard for IPC in android).

Explicit intent → where the exact name of component to which info. has to be sent. (which comp. of which app)

Implicit intent → when action needs to be performed, doesn't matter which app performs it, automatically chosen from available by user/system.

Intent contains component name, action and data.

Types of permissions —

1) Normal Permissions
2) Dangerous Permissions
3) Signature Permissions

(1) Not very much related to sensitive (or private) info.
eg:- setting wallpaper, alarm.
Doesn't cause harm/ risk. (~~No~~ Ad. don't ask).

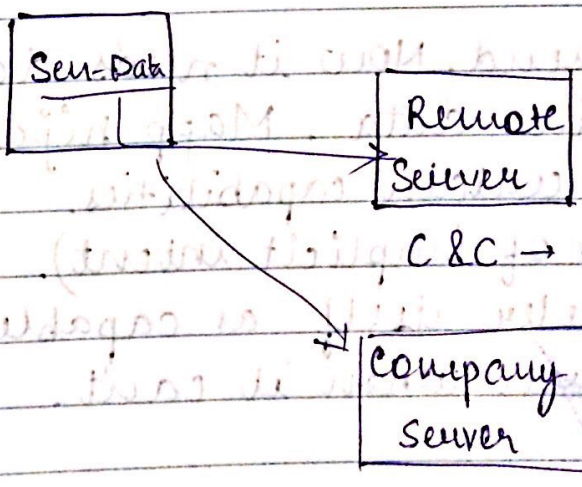
(2) which directly affect personal info.
eg:- gallery, contacts.

(3) defined by particular developer. Google can put some permissn so that all apps developed by Google can share info.

* system and signature permissn → when apps are system apps and signature permissn is given to them.

Attacks

→ Ex- filtration (leakage) of data when data goes out of device to server.



IPC in case of android, no permissn / restrictn is required/given.
So, we can't mitigate intent Hijacking attack.

Some apps come and sit on our device and send messages to the company at high cost without user concern.

→ Rooting → super user permission.
(isolatn breaks down)

→ Repackaging → malicious soft is developed, antivirus companies can identify malicious code.
Benign app $\xrightarrow[\text{(malware writers)}]{\text{modify}}$ Malicious.

- more & more users will install.
- antivirus mechanism will get signature which is in white list/trusted. App can bypass & antivirus.

→ Ransomewares.

→ Intent Hijacking attack.
(can't do anything, depends on Android Framework)

Mapp needs Access & send permissn.

Mapp → only send. Now it needs data.
Intent contains data. Mapp hijacks it.

Mapp Registers for certain capabilities.
(only in case of implicit intent).
So, Mapp register itself as capable of everything even when it can't.

To remove this, put restriction ~~on~~ on permiss" on component which does some critical task so that our app doesn't get exploited.

Mapp has only access.

→ Component Injection attack!

↳ not system app. (bcoz Mapp will need permiss" to send sms).

App1 has service component which can send SMS to a no.

Mapp gives the info & no. of receiver, provoke service comp. and message is sent.

→ Priviledge Escalat" Attack → Not having priviledge but using some other app to make my work done.