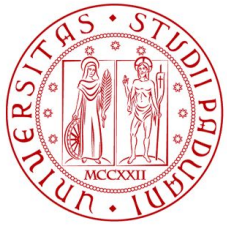
Mobile Security

Dr. Eleonora Losiouk
Department of Mathematics
University of Padua
elosiouk@math.unipd.it
https://www.math.unipd.it/~elosiouk/

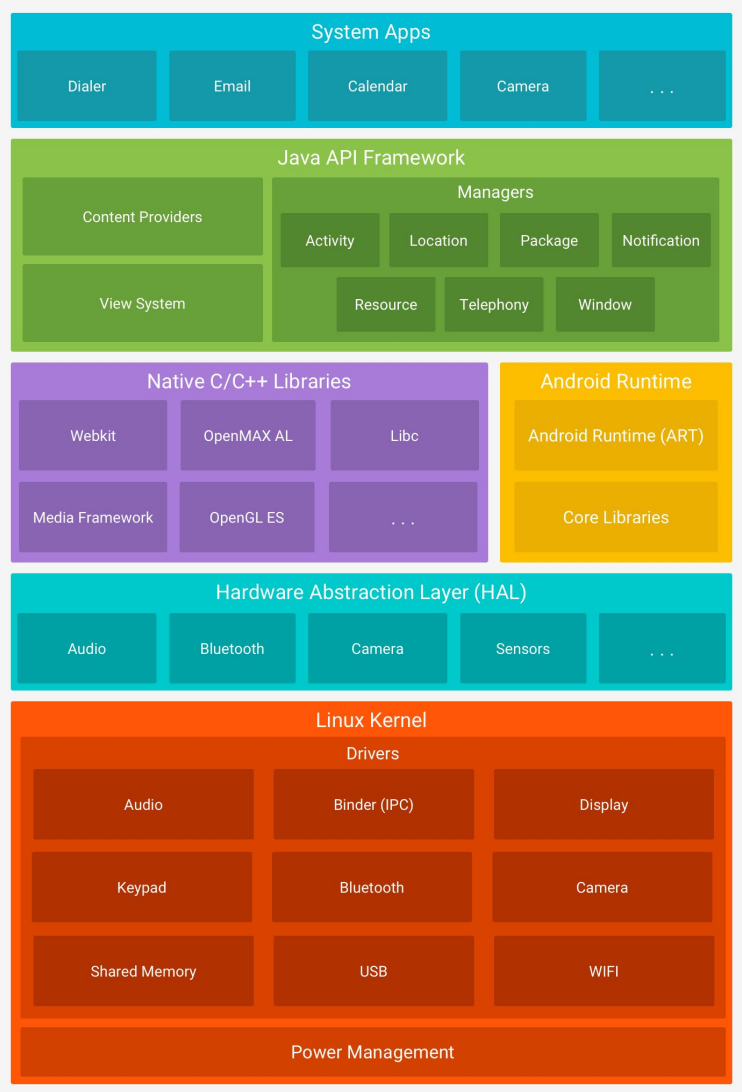


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP

DIPARTIMENTO
MATEMATICA

Android Architecture



Android Apps

- User installed apps
- Combination of loosely coupled components
  - Activities
  - Services
  - Broadcast receivers
  - Content providers
- Privilege separation (sandbox)
- Principle of least privilege (permissions)

AndroidManifest File

- Components, permissions and other metadata are specified in the AndroidManifest file
- Package name is an app unique identifier
  - Example: "com.facebook.katana"
- Package name constraints on an Android device and on the Play Store

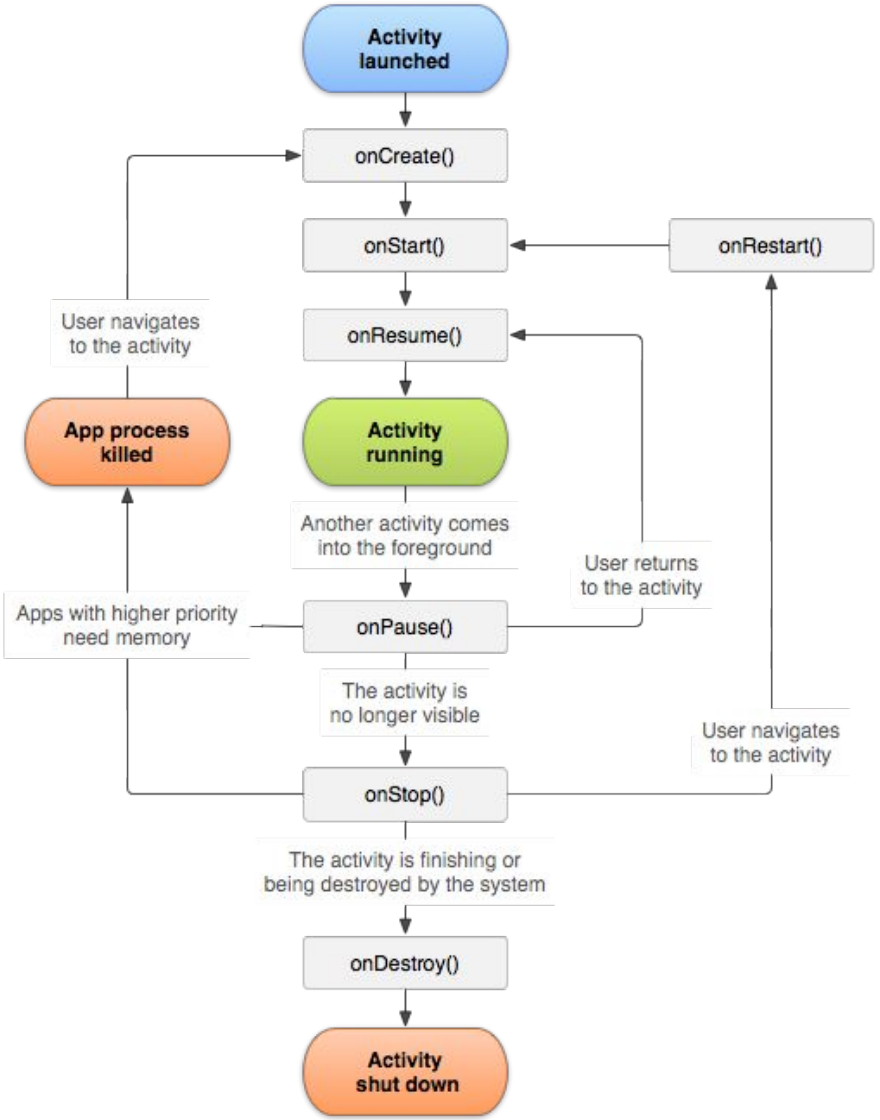
Basics on Android Apps

- There is no "main"
- The user interacts via the Graphical UI
  - Many types of UI Widgets: EditText, Button, ...
  - No command line interface
- Many APIs are "event-driven"
  - 1) You register a "listener" X
  - 2) X's callback is invoked later on

Activity (guide, ref)

- Entry point for interacting with the user. It represents a single screen with a user interface.
- You can have many: each of them defines a UI
- You can define which one is the "main" one
  - This is the chosen one when you start your app
- If the app allows it, an external app can start these activities at will

Activity Life Cycle



Service

- Performs an action in the background for some period of time, regardless of what the user is doing in foreground (the user could be switching between activities)
- Example: a music player service
- They do not provide a user interface

Broadcast Receiver

- They are meant to respond to system-wide events
- They have a well-defined entry point as well
- The system can deliver these events even to apps that are currently not running
- Example of events: battery charging, sms is received

Content Provider

- They manage a shared set of app data
- High-level API to access data so that other apps and services can query / interact with it
- They abstract away the storing mechanism
- Most often based on SQLite database (file-based)

Communication Between Apps

- IPC mechanisms built on top of the Binder component
  - Intents
    - commands and data delivered to components
  - Messengers
    - objects supporting message-based communication
  - Content providers
    - components exposing cross-process data management interface
  - AIDL
    - enables a client to call a remote object as if it was a local one

Communication Between Apps

- Use cases
  - Notation: "A.X" refers to app A's component X
  - A.X wants to start A.Y (Example: "Go to next activity")
  - A.X wants to send data to B.Z
  - Note: each component has its life cycle! A.Y could already be "started"

Explicit vs. Implicit Intents

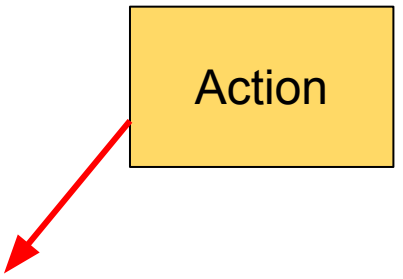
- Explicit
  - The intent "explicitly" specifies which component it wants to talk to
  - It specifies the target's full package name / component
- Implicit
  - The intent just describes the type of action to perform (and, optionally, some data)
- Good source of info / tutorial: link

Example of Explicit Intent


```
{
    ...
    Intent i = new Intent(this, SecondActivity.class);
    i.setData("Here is some data for act2");
    i.putExtra("arg1", "And here some more");
    startActivity(i);
    ...
}
```

Example of Implicit Intent


```
{
    ...
    String url = "http://www.google.com";
    Intent i = new Intent(Intent.ACTION_VIEW);
    i.setData(Uri.parse(url));
    startActivity(i);
    ...
}
```



Intent is sent around the system, with the hope that some other apps will do something about it

Intent Filters

- Intent filters are a mechanism for apps to declare something like:
  - "My component X can handle intents of type <TYPE>"

- When an app (a different one, or even itself!) sends an implicit intent, the "system" knows that it can count on X

Android Security Model

- Sandbox model
- Permission model
- App signature
- SELinux
- Verified boot

Android Sandbox Model

- Each app has its UID and dedicated data directory
- Isolation at the process level and at the file level
- The */data/system/packages.list* file contains all the information

```
com.google.android.email 10037 0
/data/data/com.google.android.email default 3003,1028,1015
```

Android Permission Model

- Due to the sandbox model, Android apps can access only to their own files and world-readable resources
- Permissions are fine-grained access rights
- Defined in the AndroidManifest file
- Granted in different moments according to their severity level
- Related permissions are mapped into the same GID

App signature

- Apps are signed by their developers
- There is no Certification Authority in Android signatures
- Signatures are used for updating apps
- System apps are signed by a number of platform keys
- Platform keys are generated by the entity responsible for the Android image running

SELinux

- Security Enhanced Linux (SELinux) is a MAC implementation for the Linux kernel
- Android integrates a modified version of SELinux
- SELinux isolates system daemons and apps in different security domains and it defines access policies for each domain
- Enforcing mode is applied to system daemons
- Permissive mode is applied to apps

Verified Boot

- The verification is performed by the kernel through an RSA public key saved into the boot partition
- Device blocks are checked at runtime
- Each device block is hashed and the hash value is compared to the one of the original block
- The kernel itself is verified through a key that is burned into the device