Android Security Model

intents = messages
intent filters = *NOT security
activities = receiver enforced permissions
services = receiver enforced permissions
broadcast receiver = sender enforced permissions
binder files

^ these are all in your manifest file (definition of the interface between your application and the android system)

1. Each application is initialized and run as a separate user
2. All sharing of data is explicit.

Android Permissions - applications can define their own permissions and are granted/disallowed on install

Some example default apps/processes: Location_fine, Location_coarse, contacts, sms_send
So what if a new app wants to interact w/ your shopping cart app? It might try to access:
Add_to_shoppingList;
Read_shoppingList;
Delete_from_shoppingList;
^^these are App defined permissions
**Android Interprocess Communication**

For more info: [http://androidapps.org.ua/androidintro_ipc.html](http://androidapps.org.ua/androidintro_ipc.html)

**IPC subsystem** - must provide UID of callers to callees. This is done by the Kernel as you can see in the diagram above.

**Intents** - These enable an application to select an Activity based on the action you want to invoke and the data on which they operate. In other words, you don't need a hardcoded path to an application to use its functions and exchange data with it. Data can be passed in both directions using Intent objects, and this enables a convenient, high-level system of interprocess communication.

**Activities** - small actions that are performed by one application for another.
1. sender constructs intent
2. invokes activities w/ that intent
3. when using applications in general the intent should not contain sensitive info since it may be routed to a surprising receipt

**Activities implementations** - can check permissions of invoker against user granted permissions decisions
   *This punts the security decision to user.......

**Broadcast Receivers**
- Senders can require permissions on receivers
- can be safe to send sensitive information

**Binder**
- Low level IPC system
- Binder objects: `Transact()` ∈ Public, `onTransact()` ∈ protected
  `^` this object is parcelable

---

Client and Service connection flow

![Binder connection flow diagram](image-url)
1. Don’t give binder references to applications you don’t want talking to you. This is because one app can give it to another. **Trust is transitive.**

![Diagram of binder references between apps A, B, and C](image)

*Now App C can get App B’s binder reference.*

**Capabilities**

*Note that these are different than Linux Capabilities*

- Unforgeable
- Explicit

**Ambient Authority:**

Confused Deputy Problem:

Programs generally take actions on the behalf of other programs or people. Therefore programs are deputies, and need appropriate permissions for their duties. The ConfusedDeputyProblem happens when a program which has permissions given to it for one purpose applies those permissions for some other purpose that is contrary to the original intent of the permission, and therefore allows something that it shouldn't.

**OS level capabilities:**

```c
int main (int initCap)
{ fd = open (initCap, “foo.txt”, RW);
```

**Root/Super User Capabilities:**

By default android operating systems don’t allow users to run programs as root user (UID 0). The process of rooting varies widely by device. It usually includes exploiting a security weakness in the firmware shipped from the factory. Once a user “roots” their phone they usually install an App known as SuperUser which manages whether or not all other applications on the phone can be run as UID 0. Many users root their phone in order to run a custom ROM/OS(most common and stable is Cyanogen Mod), uninstall mandatory apps from the phone, and more.