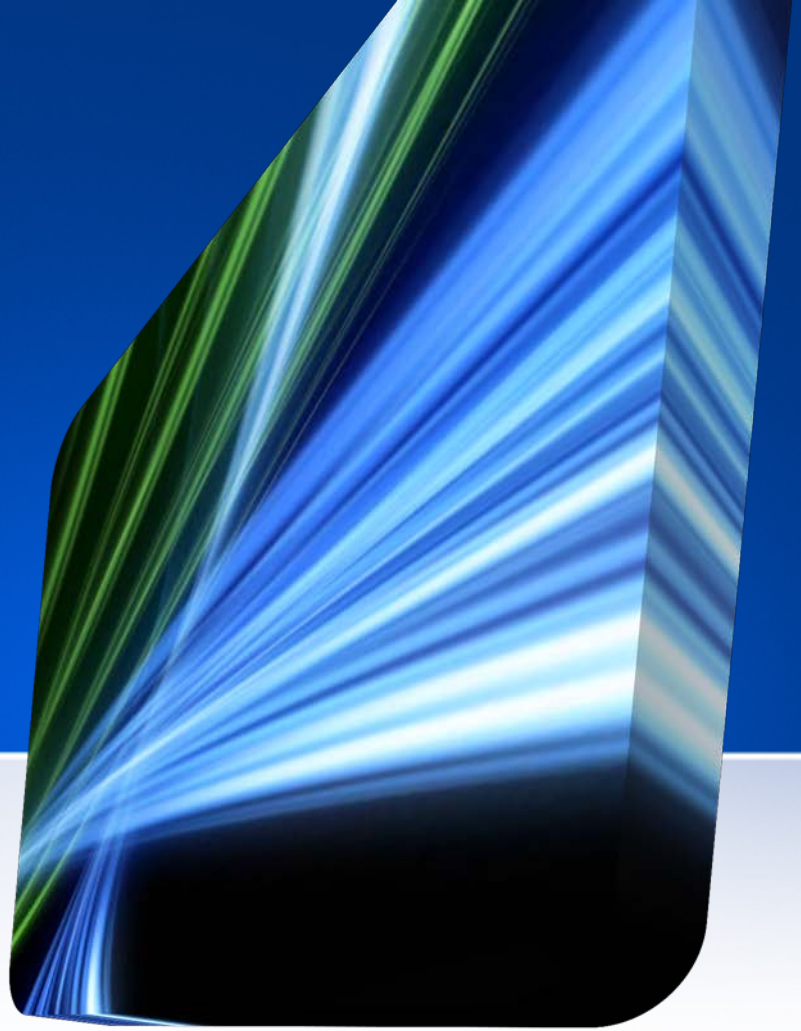
Android Security



Giovanni Russello

g.russello@auckland.ac.nz

N-Degree of Separation

- Applications can be thought as composed by
  - **Main Functionality**
  - **Several Non-functional Concerns**
- Security is a non-functional concern
- Moreover Security a **cross-cutting concern**

Specification Vs. Enforcement

- Security can affect several parts of application code
- However it is the enforcement that needs to be spread over the application code
- Specification of security policies can be done in more concise and precise way

Android Security Specification

- Android allows app developers to specify the security needs of their apps
- Each app comes with a Manifest file where the permissions listing the required permissions
- The user of the device has only two choices
  - Either install the app granting the whole set of permissions
  - Or not install the app
- **All-or-nothing model!**

Android Permission Levels

- Android provides a set of well-defined permissions
- *Normal Permissions* are assigned by default to apps
- *Dangerous Permissions* require user confirmation
- *Signature Permissions* are granted to apps signed by the same developer
- *System or Signature Permissions* are granted only to special apps installed in the data/system folder (i.e., apps signed by Google)

Permission example

- An app that wants to listen for incoming SMS has to declare in its manifest:

```
<uses-permission
android:name=android.permission.RECEIVE_SMS"/>
```

- The `RECEIVE_SMS` is consider a dangerous permission and the apps has to request it

Android Security Enforcement

- Android supports a security model that is enforced by two layers: Linux and Android middleware
- Linux enforces the DAC model
- Android middleware enforces a MAC model

Linux DAC in Android

- When an app is installed it gets a unique UID and GID
- Each app gets a home dir
  - `/data/data/<package_name>/`
- The UID and GID of the app get full access to its home dir and the files it contains
  - rwx,rwx,---

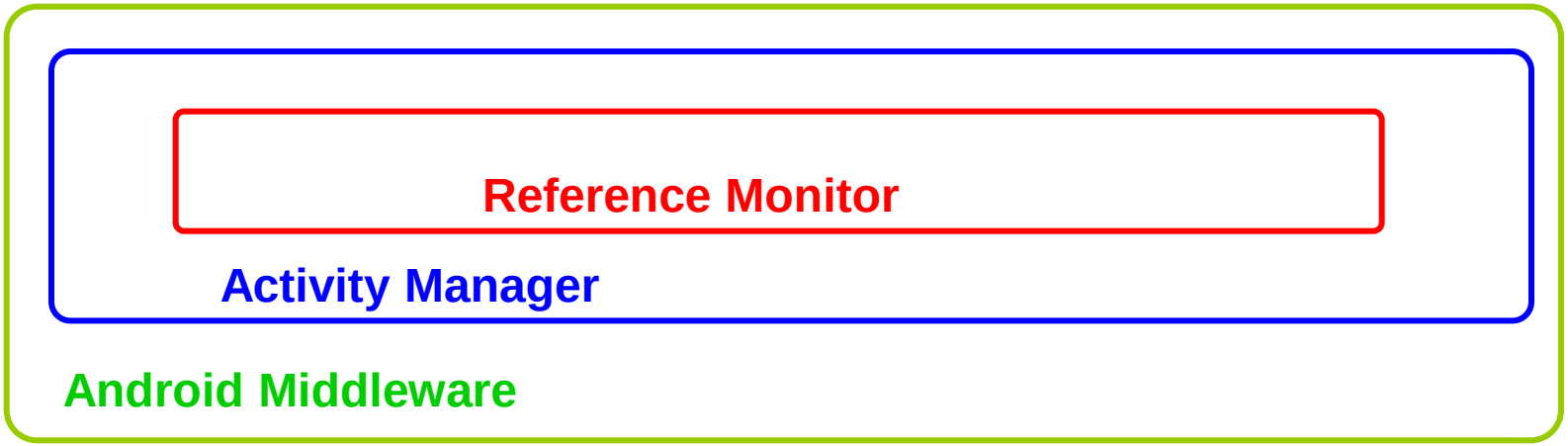
Linux Special Groups

- Linux also maintains special groups for the Internet, External Storage, and Bluetooth
- If an app asks for accessing Internet it is assigned to the Internet Group
- Similarly for the other two groups/permissions

Android Middleware MAC

- The Android Middleware controls the way in which apps use the ICC mechanism
- Each protected feature that is reachable through the ICC mechanism is assigned a label
- When the app asks for a permission in its manifest the corresponding label is assigned to the app

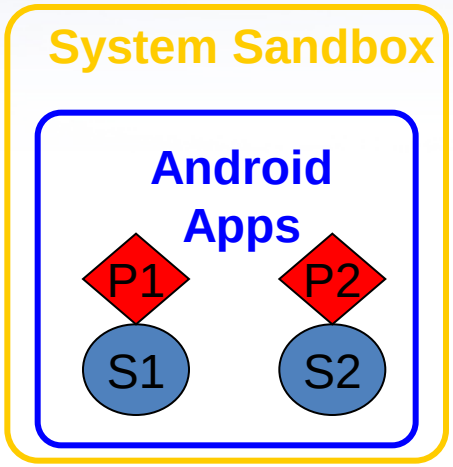
Android MAC Model

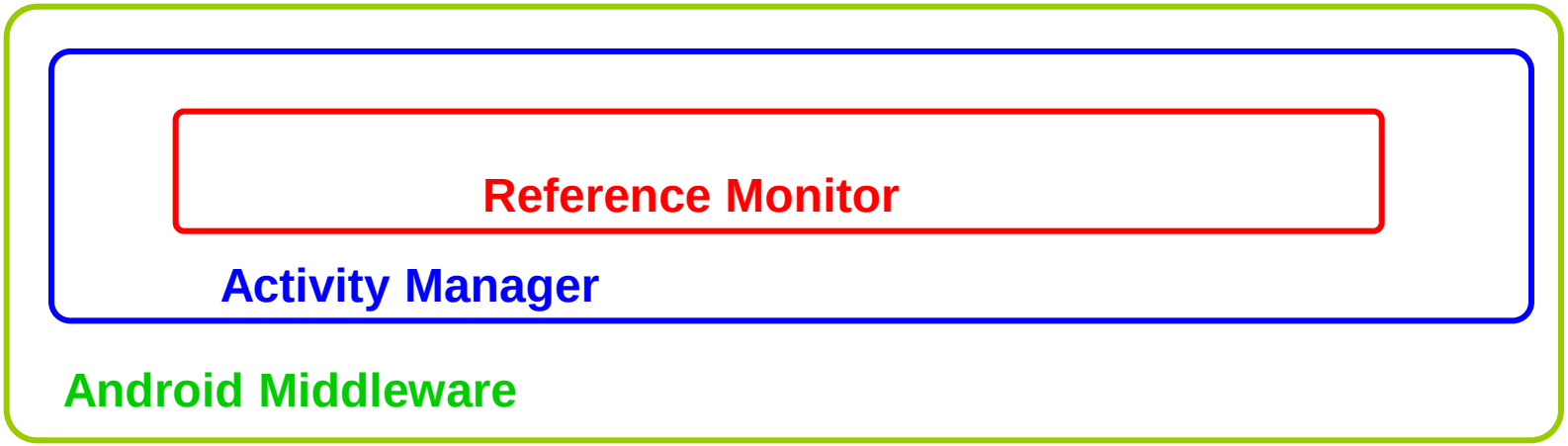


Protection Domain

S1 = Location Service

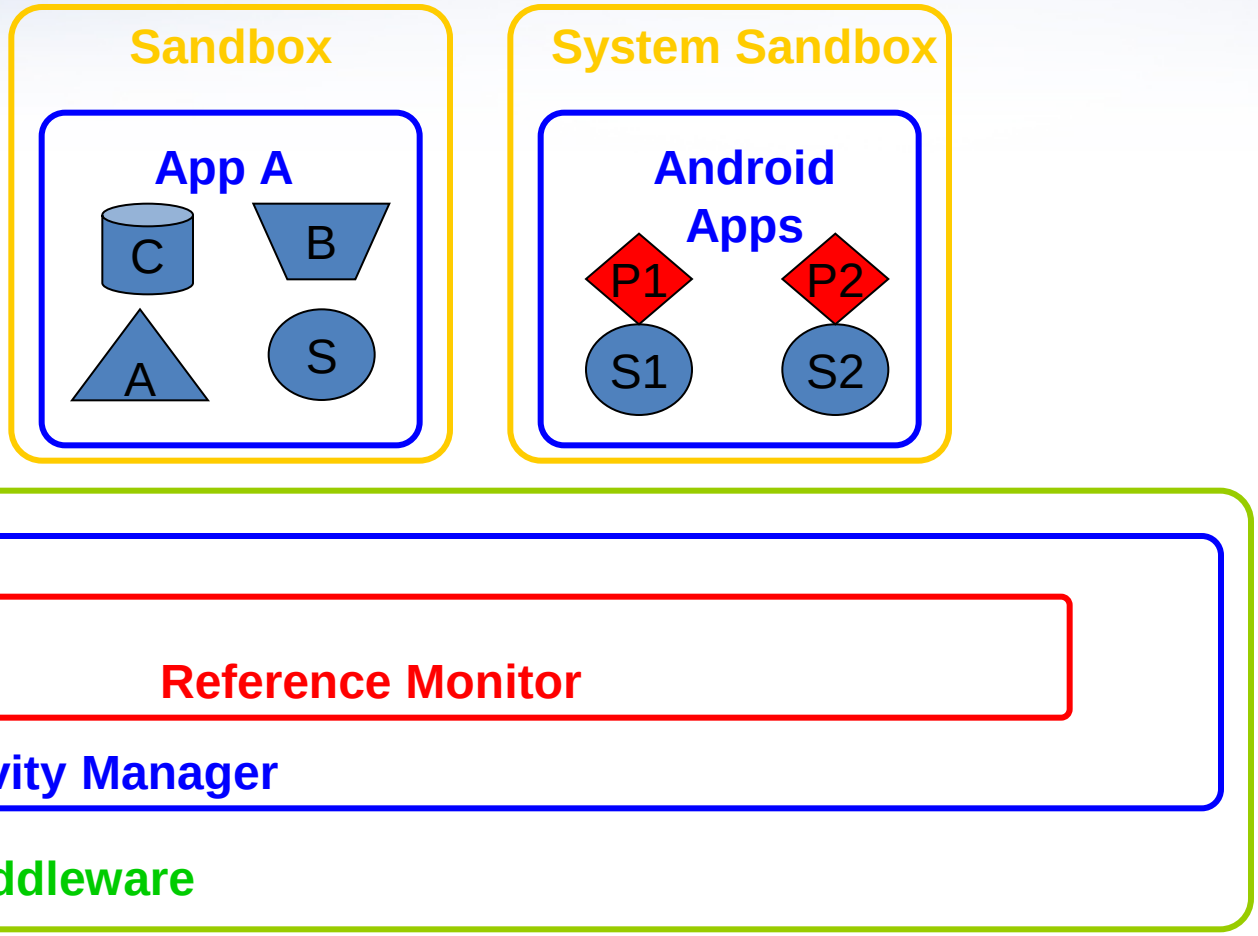
P1 = LOCATION_PERMISSION



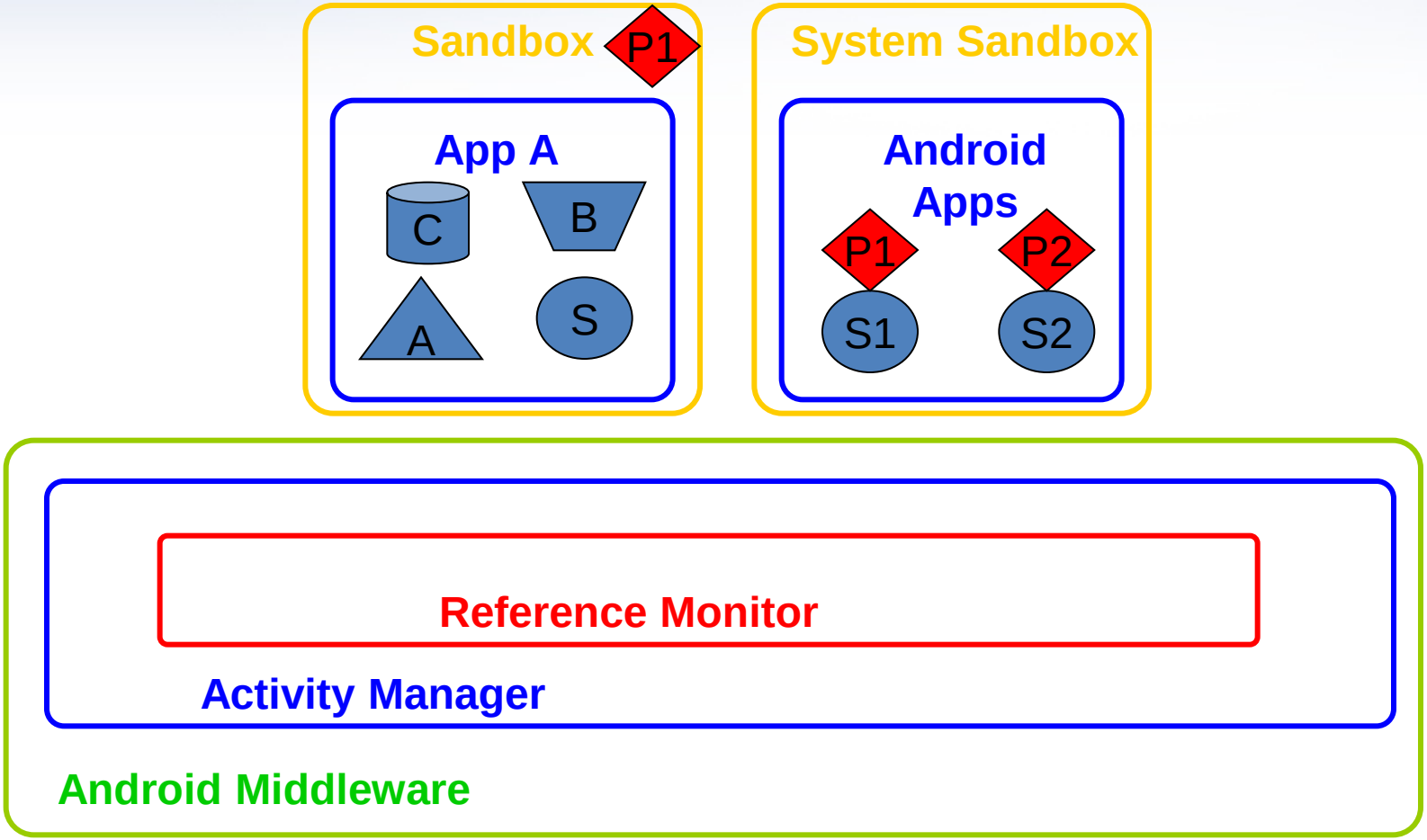


Assignment of Permissions

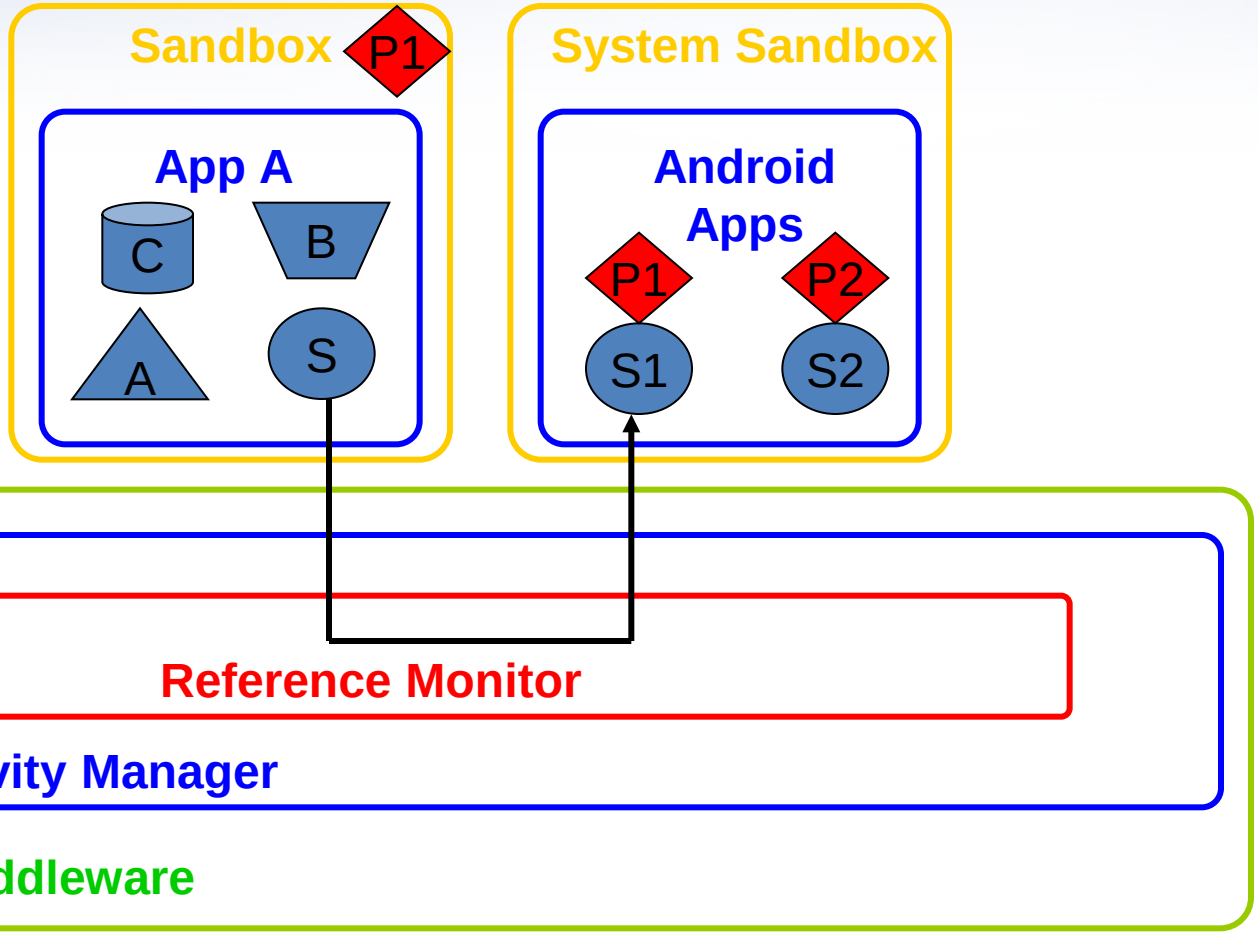
Install Time: Uses Permission = P1?



Using the Permission



Reference Monitor



Security Confinement

- Once the labels are assigned neither the app nor the user can change them
- Apps cannot delegate their permissions
- However, components can expose interfaces to other apps
- This makes difficult in standard Android to control information flow (can lead to severe attacks)

Android Security Refinements

Android Security Model allows developers to refine the security domain of their applications

- Through the standard mechanism using the Manifest
- Programmatically by using special parameters in the API

Bad move!!! Make everything murky and worst of all by default access is granted!!

Public vs Private Components

- By default any components that is not assigned a permission is public
- Developers can declare a component private by setting the `exported` flag to false in the manifest file
- Private components can only be accessed by other components in the same app
- Android can also infer if a component is private by other declarations in the manifest file (Do you trust it??)

Implicitly Open Components

- Public components have all their interface accessible to any other components
- Developers must explicitly assign permission labels to protect those interfaces

Broadcast Intent Protection

- When an intent is broadcasted, all installed apps are able to listen to those events
- This mechanism can be exploited by malicious apps that are listening for a certain event to happen
- It is possible to protect the intent programmatically:

```
sendBroadcast(intent, perm.MyPerm)
```

This means that the Manifest does not provide a complete view of app security

Service Hooks

- Android does not support a fine-grained mechanism to protect the interface of a Service
- Once a component has the permission label to access a service, the component can start, stop, bind the service
- Again programmatically it is possible to refine this mechanism by doing some extra checking at the code level, putting security policies in the app code
- Not a good security and software eng. practice!

Delegation

- Pending Intents that delegate to another app the parameters and time when an action is executed
  - Location service notifies registered apps when location changes
- URI delegation where an app delegates a component to perform an action on a resource
  - The app provides a capability to the component for performing the action
- Per se, there is nothing wrong with delegation. However, it deviates from the main MAC model

Concluding

- The Android security is based on two enforcement layers:
  - Linux DAC
  - Android Middleware MAC
- Specification is done mainly through the Manifest file

Main Drawbacks

- Specification can be done programmatically
  - Source code injection
- Open default policy
  - Developers have explicitly protect apps' interfaces
- Delegation
- No support for information flow control