Lecture 17: Mobile Computing Platforms: Android

Mythili Vutukuru

CS 653 Spring 2014

March 24, Monday

Mobile applications vs. traditional applications

- Traditional model of computing: an OS (Linux / Windows), libraries, user-space applications.
- Then came the move to web-based or cloud-based applications. Application hosted remotely, accessed by your desktop via browser / some other client.
- Initially, mobile applications followed the same principles as traditional desktop principles, except that the OS and applications were customized to run on different (resource-challenged) platforms. E.g., embedded linux.
- Now, with the advent of the smartphone, the model of mobile applications is changing.

Mobile applications today

- Mobile operating system: Android and iOS are the main players. We will focus on Android in this lecture.
- Android ecosystem is basically Linux + some extra libraries + Android runtime virtual machine (Dalvik VM) + application development framework + applications.
- Anyone can develop an “app” using the framework and any of the Linux libraries.
- Applications are written in Java and run on a Dalvik Virtual Machine (think of it as an optimized java virtual machine)

Android design principles (1)

- Traditional environment: users are the main entities by which permissions are assigned. Each user has a unique UID. In Android, each app has a unique UID, and runs in its own separate VM. This enforces isolation, and allows us to run untrusted third party applications.

Android design principles (2)

- Android enforces a fixed structure on applications. An application is composed of one of the following components:
  - Activity (user interface)
  - Service (background processing)
  - Content provider (data store)
  - Broadcast receiver (mailbox to receive notifications)
- The fixed structure results is easy development of apps, and helps monitor the apps for security.

Android design principles (3)

- Traditionally, many mechanisms for IPC (inter-process communication), like shared memory, pipes, sockets etc.
- Android restricts IPC to passing "intents" between components of applications.
- An intent is a message that specifies a target component and some arguments (action etc.). A component that receives an intent does a specific action in response to it.
  - Start activity or service
  - Bind to a service
  - Query content provider
- Intends can be explicit (identify target) or implicit (Android identifies suitable target)
- All intents pass through Android reference monitor that can monitor them for security.
- External communication via network sockets or Java libraries, much like in Linux.

Android design principles (4)

- Access control in Linux is by setting permissions based on users and groups.
- In Android, every component is protected by a permission label. Labels can be defined by system or app developers.
- Another component that wants to access it must have the required permissions.
  - Permission may be needed to send intent and start activity
  - Permission may be needed to receive intents from certain components.
- All these checks enforced by reference monitor.
- In addition to checking intents, permissions enforced by special Linux groups for access to network, camera etc.

Android design principles (5)

- Permissions are of many types:
  - Normal: harmless, given to any app that requests.
  - Dangerous: user must approve
  - System: only for system apps
- A manifest file declares all components, permissions etc. Permissions can be granted only at install time.
- User is required to grant permissions judiciously.

Android Security

- Security is a first principle in Android design, so fairly secure system.
- Some malicious activity still happens due to:
  - Linux bugs
  - User carelessness in granting permissions
  - Applications misuse permissions
- Tools to catch malicious activity
  - Static analysis of code
  - Dynamic analysis of application behavior

For more information..

- Please see the references on the class website for more details on Android architecture and security.