WILLIAM & MARY

CHARTERED 1693

CSCI 445:
Mobile Application Security

Lecture 2

Prof. Adwait Nadkarni

What is security?

- Garfinkel and Spafford (1991)
  - "A computer is secure if you can depend on it and its software to behave as expected."
- Harrison, Ruzzo, Ullman (1978)
  - "Prevent access by unauthorized users"
- Not really satisfactory – does not truly capture that security speaks to the behavior of others
  - Expected by whom?
  - Under what circumstances?

Security Goals

- *Confidentiality*: Prevention of unauthorized disclosure of information
- *Integrity*: Prevention of unauthorized modification of information
- *Availability*: Prevention of unauthorized withholding of information or resources

Brinkley and Schell, "Concepts and Terminology for Computer Security."

Security Goals (continued)

- *Authenticity*: Related to integrity, but also speaks to the sender, as well as freshness
- *Secrecy*: Similar to confidentiality, but often used when discussing specific mechanisms, e.g., access control
- *Non-repudiation*: Prevent a party from denying that some action took place
- *Privacy*: The ability/right to control access to one's information. There are many definitions. Often conflated with confidentiality/secrecy.

Risk

- *Assets* are valued resources that can be misused
  - Monetary, data (loss or integrity), time, confidence, trust
- *Risk* is the potential for an asset to be misused
  - Many different formulas, e.g., (Risk = likelihood * impact)
  - What does being misused mean?
    - Privacy (personal)
    - Confidentiality (communication)
    - Integrity (personal or communication)
    - Availability (existential or fidelity)
- Q: What is at stake in your life?

Threats

- A *threat* is a specific means by which an attacker can put a system at risk
  - The goal *and* abilities of an attacker (e.g., eavesdrop , fraud, access denial)
  - Independent of what can be compromised
- A *threat model* is a collection of threats that deemed important for a particular environment
  - A collection of attacker(s) abilities
  - E.g., A powerful attacker can read and modify all communications and generate messages on a communication channel

Vulnerabilities (attack vectors)

- A *vulnerability* is a systematic artifact that exposes the user, data, or system to a threat
- E.g., buffer-overflow, WEP key leakage
- What is the source of a vulnerability?
  - Bad software (or hardware)
  - Bad design, requirements
  - Bad policy/configuration
  - System Misuse
  - Unintended purpose or environment
    - E.g., student IDs for liquor store

Adversary

- An *adversary* is any entity trying to circumvent the security infrastructure (sometimes called *attacker*)
  - The curious and otherwise generally clueless (e.g., script-kiddies)
  - Casual attackers seeking to understand systems
  - Venal people with an ax to grind
  - Malicious groups of largely sophisticated users (e.g, chaos clubs)
  - Competitors (industrial espionage)
  - Governments (seeking to monitor activities)

Are users adversaries?

This is known as the insider adversary!

- Have you ever tried to circumvent the security of a system you were authorized to access?
- Have you ever violated a security policy (knowingly or through carelessness)?

Attacks

- An attack occurs when someone attempts to exploit a vulnerability
- Kinds of attacks
  - **Passive** (e.g., eavesdropping)
  - **Active** (e.g., password guessing)
  - **Denial of Service** (DOS)
    - Distributed DOS – using many endpoints

- A compromise occurs when an attack is successful
  - Typically associated with taking over/altering resources

Participants

- *Participants* are expected system entities
  - Computers, agents, people, enterprises, …
  - Depending on context referred to as: servers, clients, users, entities, hosts, routers, …
  - Security is defined with respect to these entites
    - Implication: every party may have unique view
- A *trusted third party*
  - Trusted by all parties for some set of actions
  - Often used as introducer or arbiter

Trust

- *Trust* refers to the degree to which an entity is expected to behave
- What the entity not expected to do?
  - E.g., not expose password
- What the entity is expected to do (obligations)?
  - E.g., obtain permission, refresh
- A *trust model* describes, for a particular environment, who is trusted to do what?
- Note: you make trust decisions every day
  - Q: What are they?
  - Q: Whom do you trust?

Trusted vs. Trustworthy

- *Trusted*: a trusted system or component is one whose failure can break the security policy
- *Trustworthy*: a trusted system or component is one that won't fail

Anderson, "Security Engineering."

Security Model

- A *security model* is the combination of a trust and threat models that address the set of perceived risks
  - The "security requirements" used to develop some cogent and comprehensive design
  - Every design must have security model
    - LAN network or global information system
    - Java applet or operating system
- The single biggest mistake seen in use of security is the lack of a coherent security model
  - *It is very hard to retrofit security* (design time)
- *Need for apps to be explicit about these things to be secure.*

Android Overview

- Impact: 73% global market share (https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/
- Low barrier for entry for developers
  - Have 25$ and a phone?
- Apps can be *installed* by the user: No need for an app store.
- The OS is most evolved in terms of:
  - Application inter-operability
  - OS extensibility: Even some crucial features can be substituted by third party apps (e.g., Launcher)

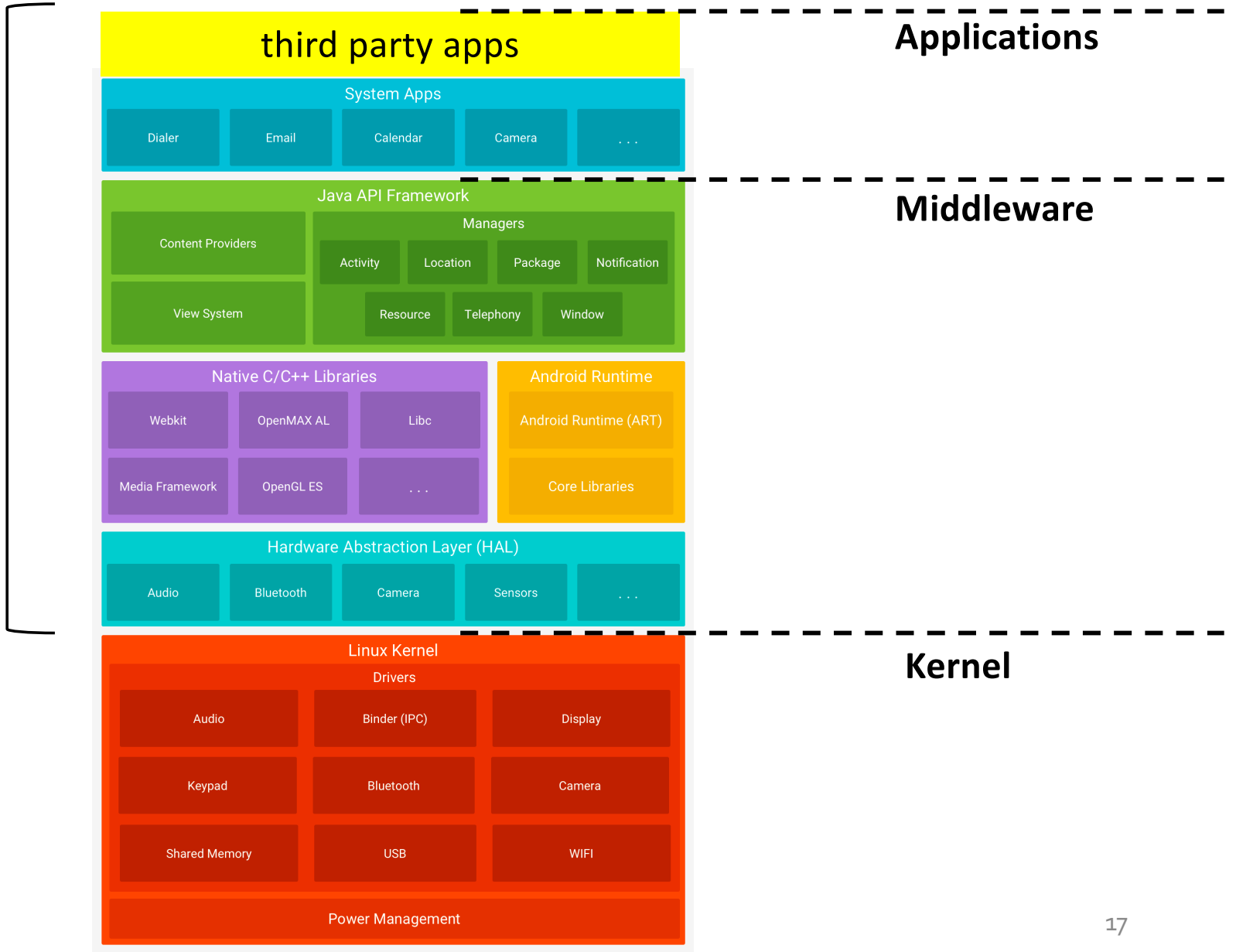
Architecture

- The Android OS is built upon Linux and includes many libraries and a core set of apps.
- The *middleware* makes it interesting
  - Not focused on UNIX processes
  - API for common resources, inter-app communication
  - Applications consist of *components* of distinct types
  - Apps interact via components
    - *Binder* framework for inter component communication.
  - We will focus on security with respect to components.

Architecture

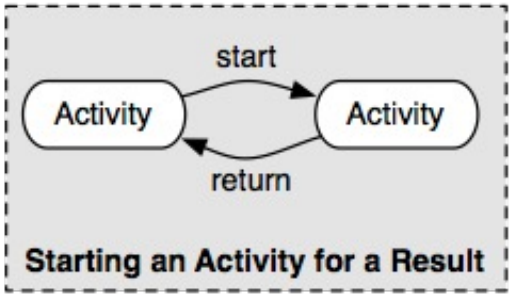
Userspace
(*reference monitor*
in middleware)

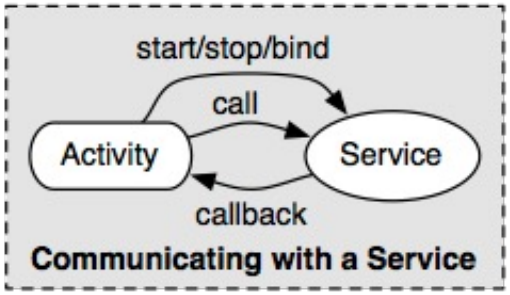
Kernel space
(*reference monitor*
in kernel)

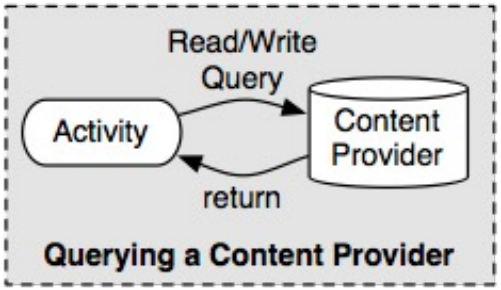


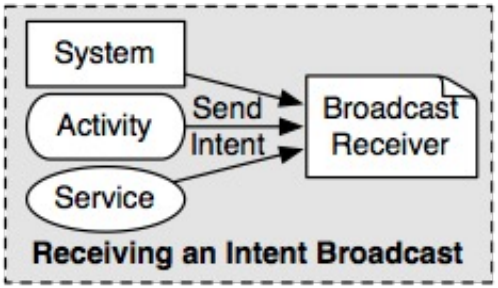
Component Model

- Four different types of components:









- Distinct *functions*, a *lifecycles*: Managed by the ActivityManager System service.
- Functional decomposition of the app: Great boundary for security! (in contrast w/ processes)

Component Model

- **Activities:** User Interaction (i.e., the UI)
- **Services:** Background computation
- **Content Providers:** Interface to data (storage, cloud, memory)
- **Broadcast Receiver:** Receive events
- *Different from processes*: An app's components may or may not exist in the same process (android:process)
- Interactions within/among apps are based on components
  - Target components in the same or different app
  - *But first ...*

Android Manifest

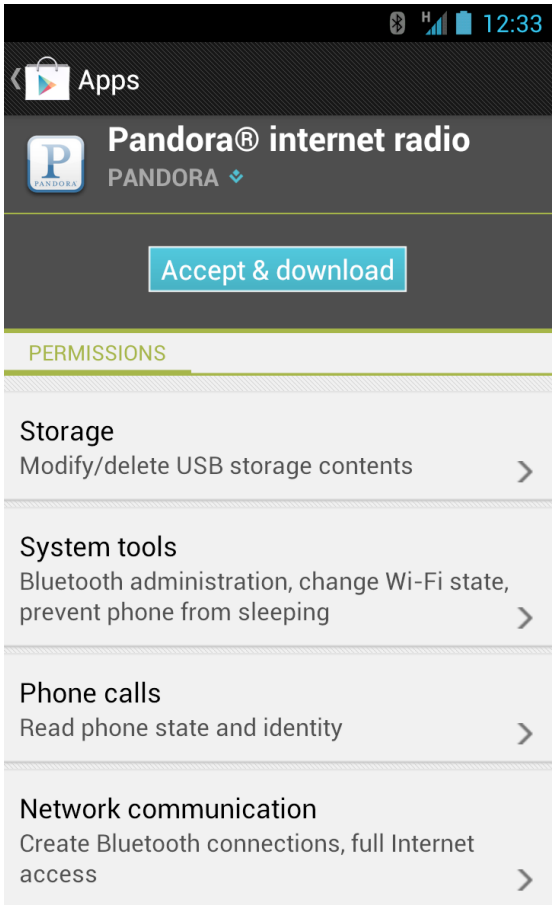
- Describes the application package:
  - Application components, and *access rules for components*
  - Intent filters (services provided by components)
  - Versioning information (minimum, target, maximum API)
  - Permissions:
    - That the application requires
    - Custom permissions declared by the app
  - You can tell a lot about an application by looking at its manifest
    - *Some of the initial security analysis in this class will involve Application manifests*

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.siislab.tutorial.friendtracker"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FriendTrackerControl"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <provider android:authorities="friends"
              android:name="FriendProvider"
              android:writePermission="org.siislab.tutorial.permission.WRITE_FRIENDS"
              android:readPermission="org.siislab.tutorial.permission.READ_FRIENDS">
        </provider>
        <service android:name="FriendTracker" android:process=":remote"
              android:permission="org.siislab.tutorial.permission.FRIEND_SERVICE">
        </service>
        <receiver android:name="BootReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"></action>
            </intent-filter>
        </receiver>
    </application>

    <!--  Define Permissions -->
    <permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></permission>
    <permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></permission>
    <permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></permission>

    <!--  Uses Permissions -->
    <uses-permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></uses-permission>

    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
</manifest>
```

Permissions

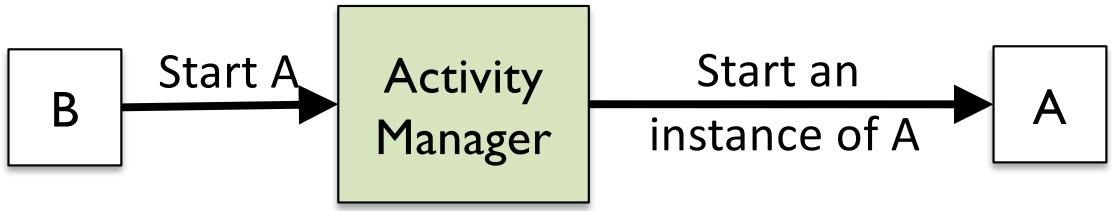
$$security == permissions$$

- For accessing device/user resources: SDcard, network, phone IMEI/IMSI, contacts, calendar data, ...
- For interacting with other apps
  - Custom permissions
- *Permissions define capabilities*
- Users make permission decisions
  - Install-time (< Android 6.0)
  - versus runtime (>=Android 6.0) ?
- Permissions provide:
  - User consent?
  - Review triaging
- Many believe permissions are not enough



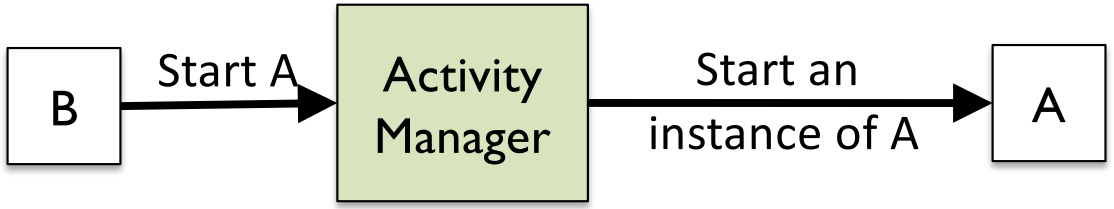
Intents

- Most common form of *Inter-component communication*
- Intents are objects used as inter-component messages
  - Starting the user interface for an app
  - Sending a message between two components
  - Starting/binding to a background service
- Two types: *explicit* or *implicit*
  - Explicit: start activity **A** from **app XYZ**"
  - Implicit: start an activity to **ACTION_VIEW** a **PDF**

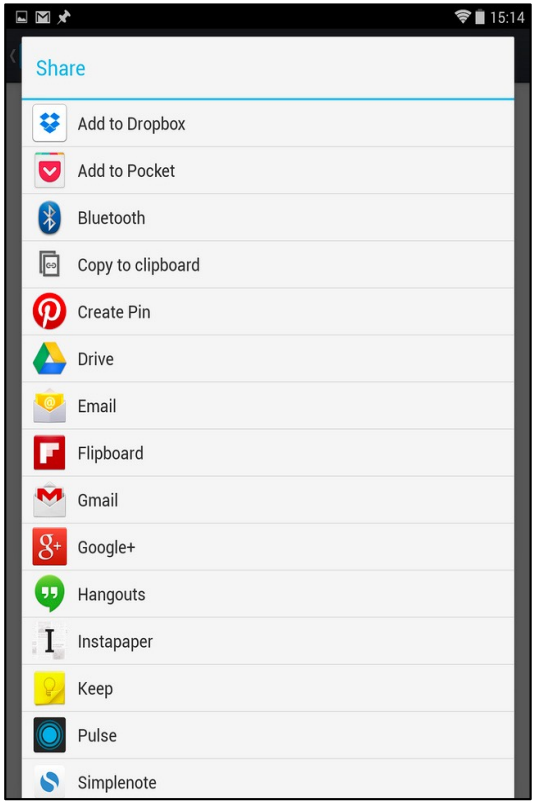


Intent Filters

- Intents are an *indirect* and *asynchronous* communication mechanism.
- Intent filters describe the service provided by a component: ACTION, DATA, CATEGORY, ...
- The system matches intents with filters

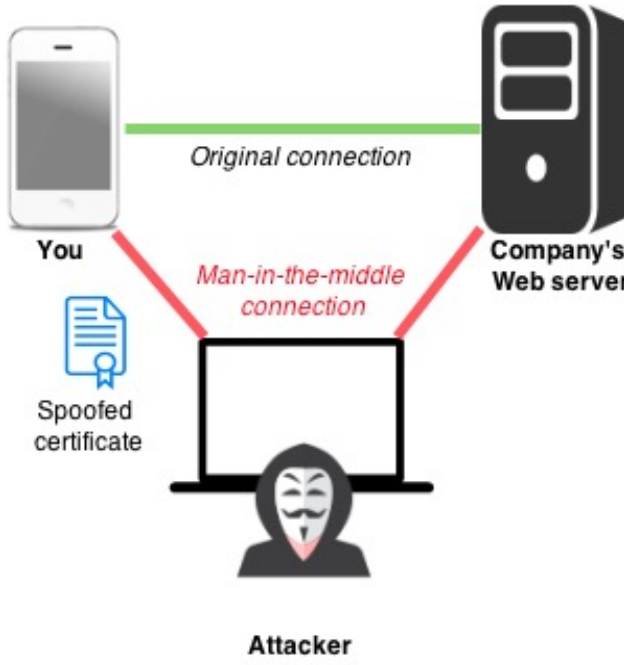


- But, what if there is more than one match?
  - Activity: Ask the user!
  - Service: ?
- Security implications? *Intent Hijacking*



Network, Storage access

- API provided by the OS
- Good enough for most application tasks
- *App developers do not use safe defaults. Why?*
  - *Debugging*
  - *Legacy code*
  - ...
- For example:
  - `openFileOutput(FileName)` vs `FileOutputStream(/path/to/file)`
  - Creating `MODE_WORLD_READABLE` files
  - Allow all certificates in *WebViews*
- *Lots of problems here:* This class will teach you what to look for; and how.



Recall: Project Proposal

- *Deadline: Feb 08, 11:59 PM (Thursday)*
  - At least five unique application ideas in order of interest
  - Each possessing at least *4 characteristics (i.e., network, storage, inter-app communication, permissions, Webviews, etc.).*
  - Part of one or more *real* features.
  - Names of **up to 4 group members**.
- **IMPORTANT:** Immediately after submitting the proposal to Blackboard, ***schedule a 30 minute meeting with me***
- I have the final say on your project and group

See https://www.adwaitnadkarni.com/teaching/csci445/project