COMP30510 Mobile Application Development

Android Security

Dr. Abraham Campbell

University College Dublin

Abey.campbell@ucd.ie

Outline

- Software security today
- Android's infrastructure overview
  - Device
  - Users
  - Platform Owner
  - Developers
- Pro's and Con's
- Creating more secure apps

State of Software Security

- Complex firewalls
- Sophisticated IDS
- System administrators
- Code audit
- Enhanced security mechanisms

=>

- Still, more systems are hacked and there is no sign of stopping.

Desktop World

- Any programme can do anything?!?
- Things started to change:
    - Windows Vista/7 – protecting crucial parts of the system, following Mac OS X/Linux/Unix
    - Digital Signatures for drivers, packages, etc
- However, even today, once installed, nothing stops any programme from reading/writing/deleting all user files, connecting to the internet (firewalls to the rescue), etc...

Solutions?

- Use software only from trusted sources
- Use software that is free & open source
- Sandbox every application, giving it precisely as many privileges as it needs to function correctly (and you being comfortable with it)

Implications?

- Trusted sources – you need to actually trust these sources, *i.e.* trust company/organization
- Open source – problem reviewing all the code you use, even for programmers. There is Just. Too. Much. Code. Goes back to Trust. Impossible to make everything open source...
- Sandboxing – performance penalty for isolating and controlling each app, constant user annoyance with confirmation dialogues

Desktop Vendors

- Desktop vendors usually use a combination of approaches (or not):
- GNU/Linux approach: trusted source (repositories), open source (mostly)
- Mac/Windows – trusted source and a bit of sandboxing/DRM

- But it is still not enough and it’s not working Remember how personnel a phone is to its user.

Android's Solution

- Static sandboxing of all installed applications, application signing

  =>

- Reasonable balance of protection vs. User annoyance
- Minimal performance penalty due to efficient use of the Linux kernel security mechanisms where possible

Android's Solution Cont'd

- Android's solution is still not perfect – a lot of people do not bother reading app permissions, install it, then wonder where their phone credit went
- Apple's 'Walled Garden' vs Open Internet approach, freedom vs. responsibility trade offs , also depends on trusting Apple exclusively

Device Security

- Security managed by Linux kernel built-in capabilities
- Each app runs in its own process
- Each app gets unique UID/GID
- Each app runs its own Dalvik VM

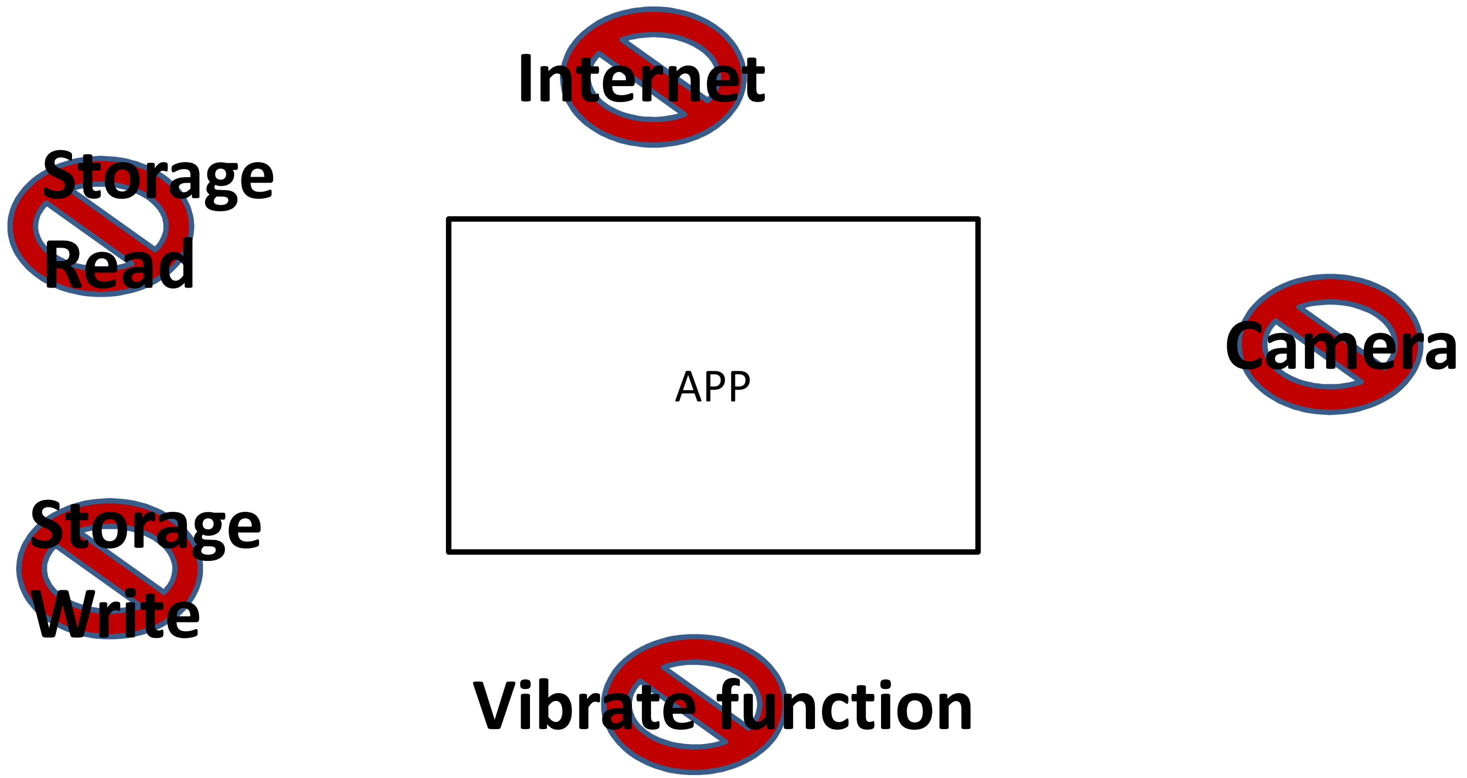
- This way the processes are isolated and crash of one app does not bring down the whole system

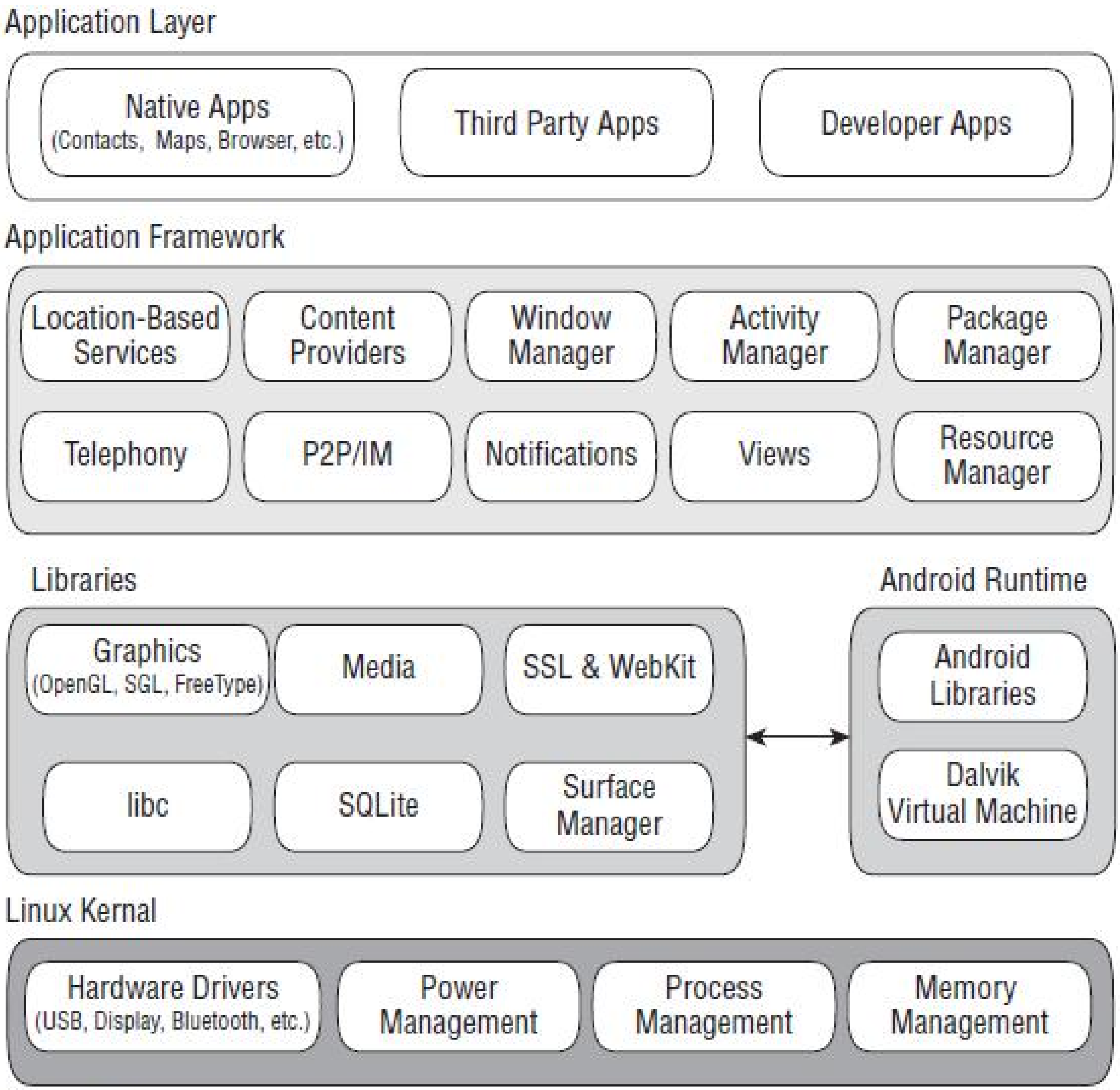
Device Security Cont'd

- Low-level permissions are managed by the Linux kernel instead of Dalvik VM, *i.e.* no sandboxing is done at the VM level
- This allows for execution of Dalvik code, native code, or hybrid (Dalvik + native) code
- Access to other parts of the system are also tightly controlled

Permissions Cont'd

- Without explicit permissions your app see its world like this





Application Layer

Native Apps (Contacts, Maps, Browser, etc.)

Third Party Apps

Developer Apps

Application Framework

Location-Based Services

Content Providers

Window Manager

Activity Manager

Package Manager

Telephony

P2P/IM

Notifications

Views

Resource Manager

Libraries

Graphics (OpenGL, SGL, FreeType)

Media

SSL & WebKit

libc

SQLite

Surface Manager

Android Runtime

Android Libraries

Dalvik Virtual Machine

Linux Kernal

Hardware Drivers (USB, Display, Bluetooth, etc.)

Power Management

Process Management

Memory Management

Figure 1-1

© Reto Meier

Native Android Permissions

- Too many to mention all, just open up AndroidManifest.xml, switch to 'Permissions' and try to add one... ;-) ~100 overall
- Most popular:
  - Full access to Internet (check network state)
  - Read/write to SD Card
  - Read phone state/identity
  - Access location (coarse)
- Permissions are static, they can't be modified at runtime once programme is installed (requires reinstall & manual confirmation)

Native Android Permissions Cont'd

```
<manifest ... >
<usespermission
android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

Creating and Enforcing Permissions

- Your application can create its own permissions, specifying who or what has ability to call it.
- Permissions can be specified for all building blocks of the Android, *i.e.*
  - Activities
  - Services
  - BroadcastReceivers
  - ContentProviders

Creating and Enforcing Permissions

```
<manifest ...>
<permission
android:name="com.me.app.myapp.permission.DEADLY_ACTIVITY"
android:label="@string/permlab_deadlyActivity"
android:description="@string/permdesc_deadlyActivity“
android:permissionGroup="android.permissiongroup.COST_MONEY"
android:protectionLevel="dangerous" />
</manifest>
```

Permission Properties

- **ProtectionLevel**: normal, dangerous, signature, signatureOrSystem
- PermissionGroup
- Label
- Description

Android Users can

- Encrypt device
- Set password/pin/pattern/face unlock
- Turn off selected networking
- Turn off untrusted/unknown install sources
- Install various apps from Android Market
  - During install an app shows required permissions, which user chooses to grant (by installing) or not to grant (by refusing to install an app)
- ● Report on installed software/rate/comment

Google as a platform owner can

- Remove any piece of software that violates Google Play's tems of service (ToS)
- If software is reported to be malicious, remove it from individual handsets automatically
- Pull an info (name, address, CC details, *bank details*) on any registered developer

Developers can

- Ask for specific permissions during application install and know they would be granted if application runs
- Utilize the power of API provided by Android software stack to create robust, secure applications by following straightforward list of guidelines

Creating Securer Apps

- Do not use dynamic class loading from insecure sources
  - Common storage, insecure (HTTP) download
- Do not use internal files that are world readable/ writable via **inter-process communication**
- Do use input validation when reading from user input, external storage or the internet
- Use parametrized query methods to avoid SQL injections

Creating Securer Apps Cont'd

- Make unavailable broadcast receivers, activities and services that are not meant to be called by other apps
- Drop sensitive permissions if you are not using them
- Prefer HTTPS to HTTP
- Do not use localhost or INADDR_ANY for communication, use Android IPC instead

Cause for Concern?

- Cloud-centric devices are dangerous
  - Single point of failure
  - Location of data and access to it is unknown
  - Central control over individuals, central authority ultimately deciding what one can and cannot have on their handsets