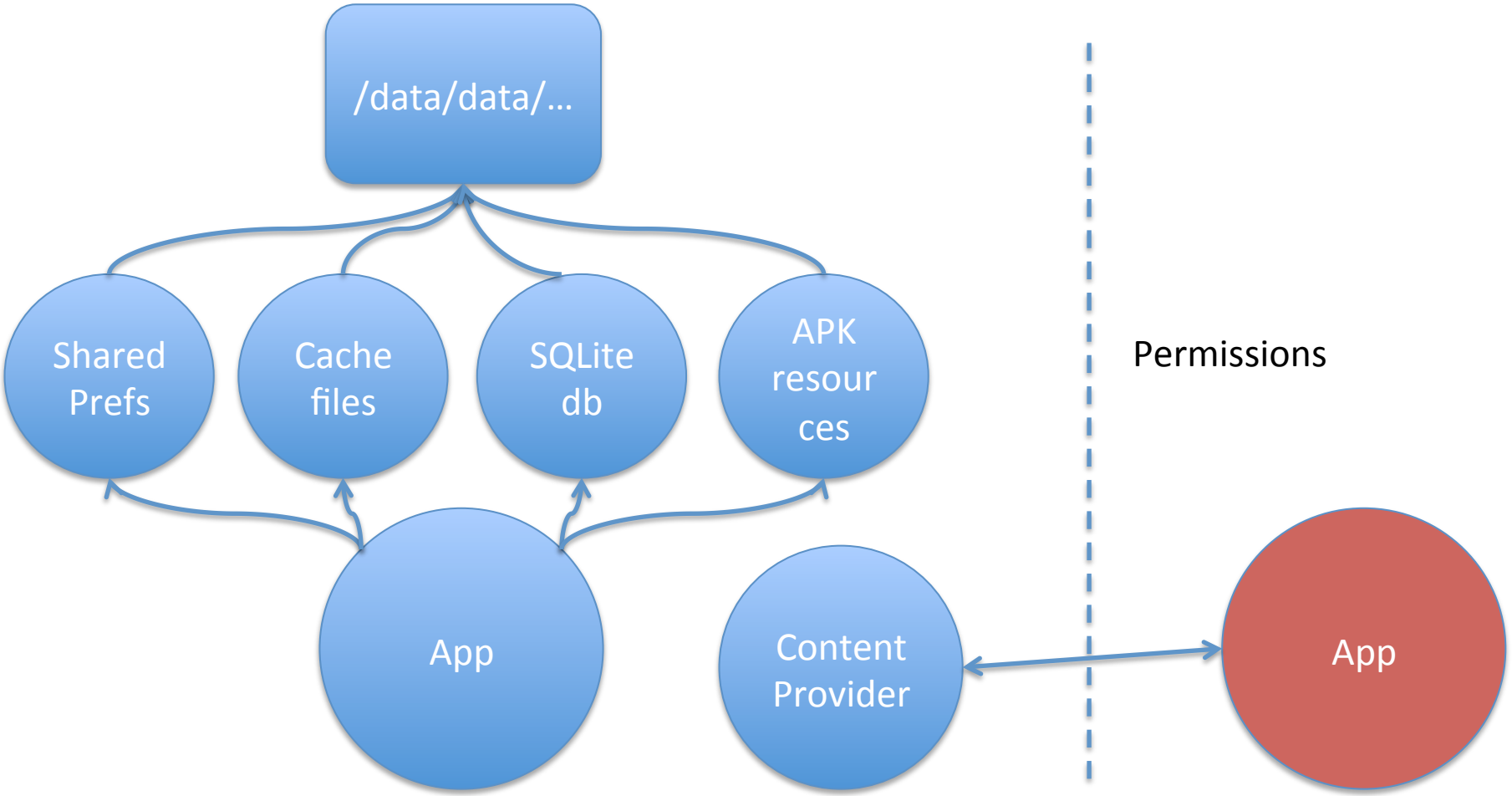
G54MDP
Mobile Device Programming

Lecture 12 – ContentProviders, Permissions and Security, BroadcastReceivers

git

- http://git-scm.com/
- https://github.com/mdf/g54mdp
- git clone https://github.com/mdf/g54mdp.git
- git pull

Sharing Data – if not



Creating a Content Provider

- Implement a storage system for the data
  - Structured data / SQLite
    - Values, binary blobs up to 64k
    - Database
  - Large binary blobs
    - Files
  - Photos / media manager
- Implement a ContentProvider
  - Implement required methods
  - query, add, update, insert etc
  - onCreate
  - getType
    - What type of data are we providing?
  - ParcelFileDescripter openFile()
- Tell Android we are a provider
  - Declare in the AndroidManifest

Contract

- Defines metadata pertaining to the provider
- Constant definitions that are exposed to developers via a compiled .jar file
  - Authority
    - Which app is responsible for this data
  - URI
  - Meta-data types
  - Column names
    - Abstraction of database architecture

URI Matching

- All of these methods (except onCreate()) take a URI as the first parameter
  - The object will need to parse it to some extent to know what to return, insert or update
  - Android provides android.content.UriMatcher to simplify this
    - Provides mapping between abstraction of contract class to concrete db implementation
  - Does the calling application want all data from a table, or just a row, or a specific table?
    - Or a “virtual” table

Let’s have a look…

Network

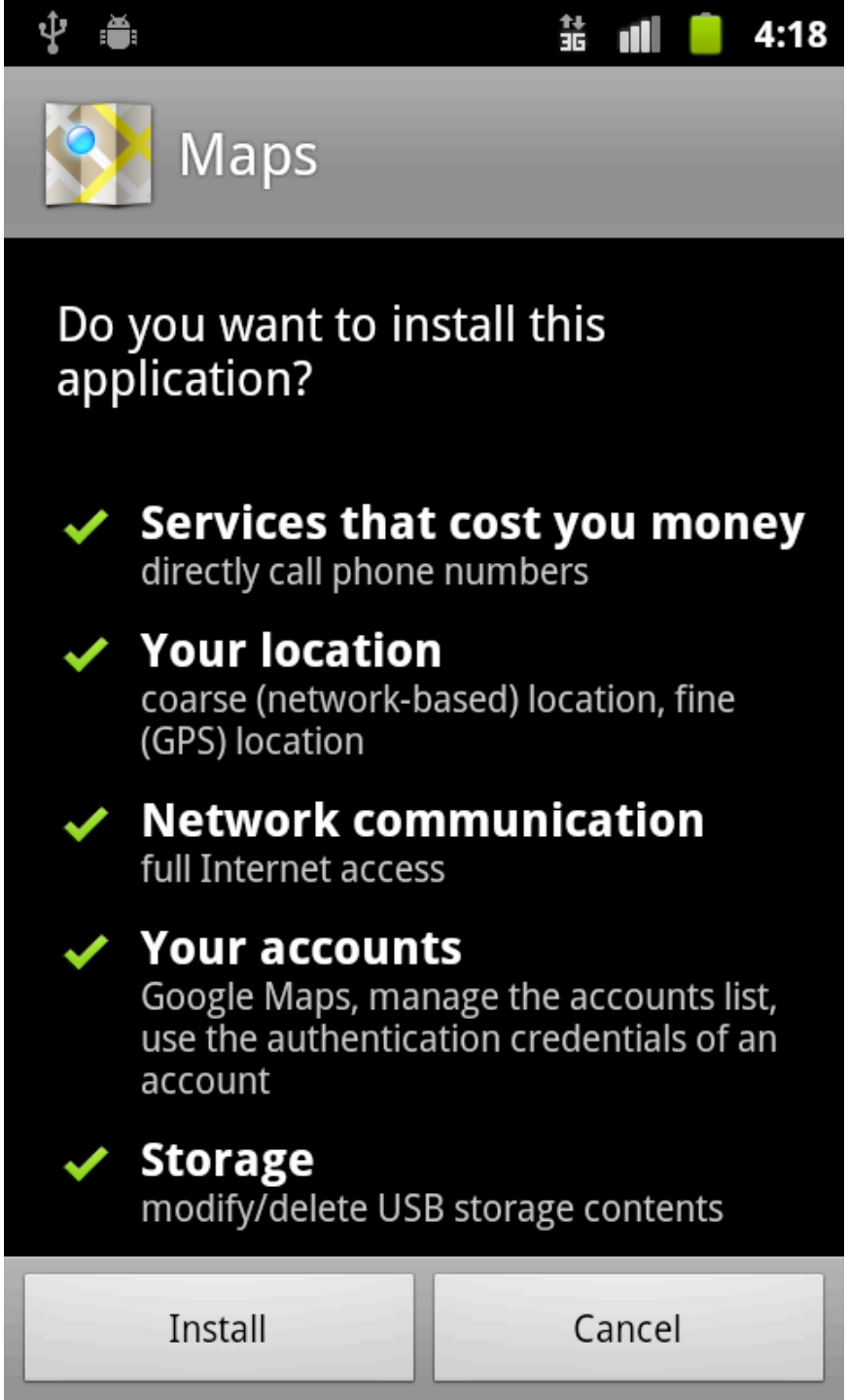
- One last type of data storage
  - Get it off the phone, and into the cloud
- Implement a SyncAdapter
  - Appears in the “Accounts and Sync” menu in the OS
  - Synchronizes a local database / content provider with a remote server
  - Make use of a Service to push data in the background
- http://developer.android.com/training/sync-adapters/creating-sync-adapter.html

Android Security

- Isolation by default
- Linux kernel
  - Filesystem / UID
    - Private, per-application file storage
  - Processes
    - Individual virtual machine instances
    - Native-code controlled by the application sandbox
- Restricted access to the root user
  - Most processes run as normal users
- IPC through specific interfaces
  - Services, Binders, Intents, Messages, ContentProviders
  - Intentional lack of APIs for sensitive functionality
    - Direct SIM card access

Permissions

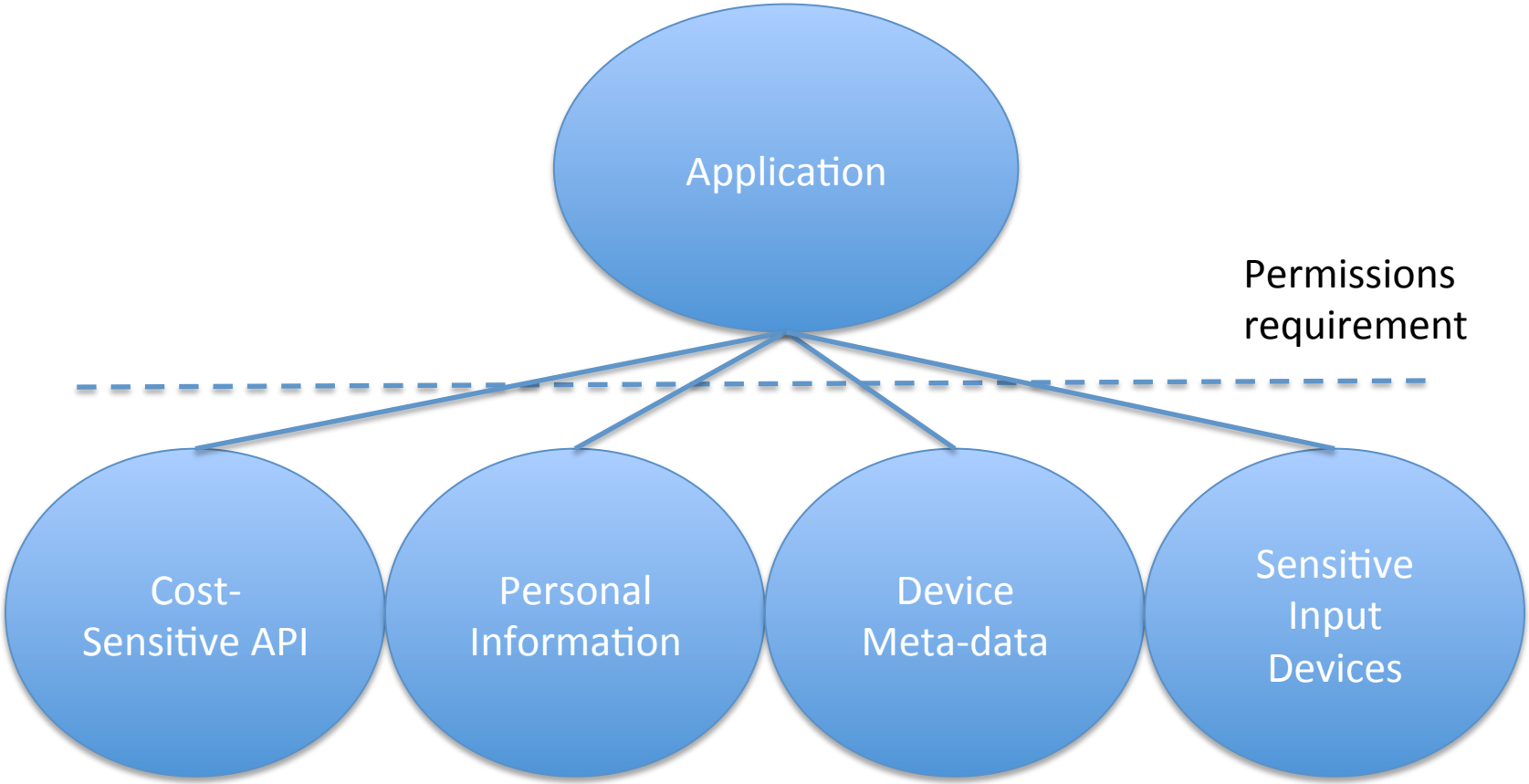
- No access by default
  - Control access to specific mechanisms
- Applications can offer protected access to resources and data with **permissions**
  - Permissions explicitly granted by users
- Permission architecture
  - Applications statically declare permissions
    - Required **of** components interacting with them
      - You must have this permission to interact with me
    - Required **by** components they interact wth
      - I will need these permissions
  - Android requires user's consent to specific permissions when an application is installed


4:18
Maps
Do you want to install this application?
Services that cost you money
directly call phone numbers
Your location
coarse (network-based) location, fine (GPS) location
Network communication
full Internet access
Your accounts
Google Maps, manage the accounts list, use the authentication credentials of an account
Storage
modify/delete USB storage contents
Install
Cancel

Permissions

- Show permissions required at install time
  - Not prompted again regarding permissions at run-time
- Why?
  - Not yet made a commitment (financial, mental) to the application
    - Can compare to other applications
  - Not per session / at run-time
    - "Seamless" switching between Activities / applications
    - Would slow down the user experience
    - Train users to click "ok" repeatedly without considering the implications

Permissions



Permissions

- Cost-Sensitive APIs
  - Telephony
  - SMS/MMS
  - Network/Data
  - In-App Billing
  - NFC Access
- Personal Information
  - Contacts, calendar, messages, emails
- Device Meta-data
  - System logs, browser history, network identifiers
- Sensitive Input Devices
  - Interaction with the surrounding environment
  - Camera, microphone, GPS

Permissions

- http://developer.android.com/reference/android/Manifest.permission.html

String	PROCESS_OUTGOING_CALLS	Allows an application to modify or abort outgoing
String	READ_CALENDAR	Allows an application to read the user's calendar
String	READ_CALL_LOG	Allows an application to read the user's call log.
String	READ_CONTACTS	Allows an application to read the user's contacts
String	READ_EXTERNAL_STORAGE	Allows an application to read from external storag
String	READ_FRAME_BUFFER	Allows an application to take screen shots and m the frame buffer data.
String	READ_HISTORY_BOOKMARKS	Allows an application to read (but not write) the u bookmarks.

Common Permissions

- android.permission.ACCESS_FINE_LOCATION
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.INTERNET
- android.permission.WAKE_LOCK

Using Permissions

- Applications can define new permissions in the manifest

  <permission android:name="android.permission.VIBRATE"

  …

  </>

  - Do we really need a new permission?
  - normal / dangerous / signed
  - "Readable" explanation of the new permission
- Applications can require components interacting with them to have a specified permission, set in the manifest
  - By default all permissions apply to all components hosted by the application
    - Activities, Services etc.
  - Or per component permission requirements

Using Permissions

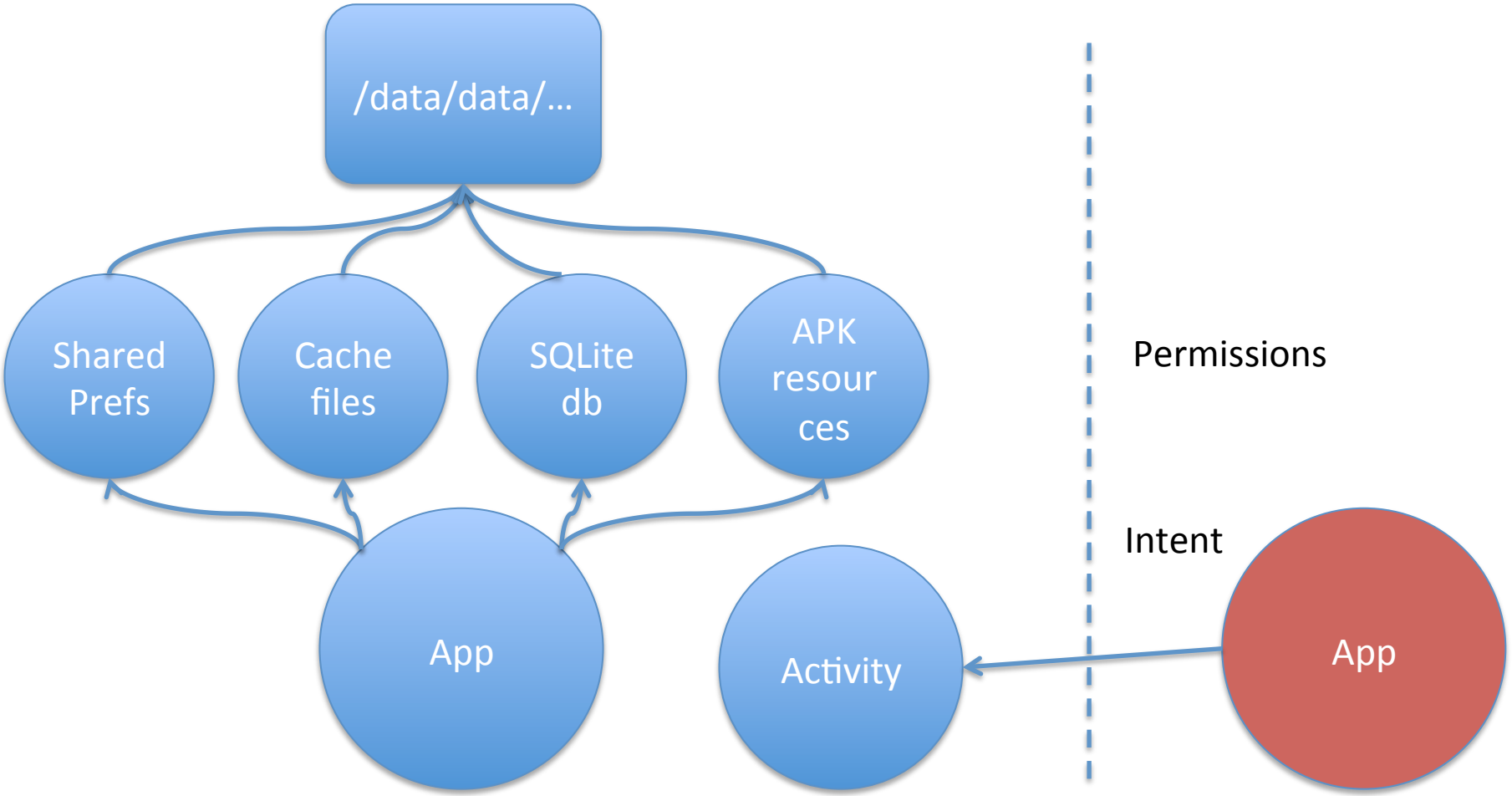
- Specify that an Application **uses** a permission

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

- Specify that an Application **requires** a permission
  - The app must **use** permissions it **requires**

```
<provider
    android:permission="android.permission.READ_CONTACTS”
    android:authorities="com.example.martincontentprovider.MyProvider”
    android:multiprocess="true”
    android:name="com.example.martincontentprovider.MyProvider">
</provider>
```

Sharing Data – is this good enough?



Component Permissions

- Activity
  - Restricts which components can start the activity
  - Checked within execution of:
    - startActivity()
    - startActivityForResult()
- Service
  - Restricts which components can start or bind to the associated service
  - Checked within execution of:
    - Context.startService()
    - Context.stopService()
    - Context.bindService()
- ContentProvider
  - Restricts which components can read or write to a ContentProvider
- Throw SecurityException on permissions failure
  - Usually as we've forgotten to ask for permission during installation

Querying Permissions

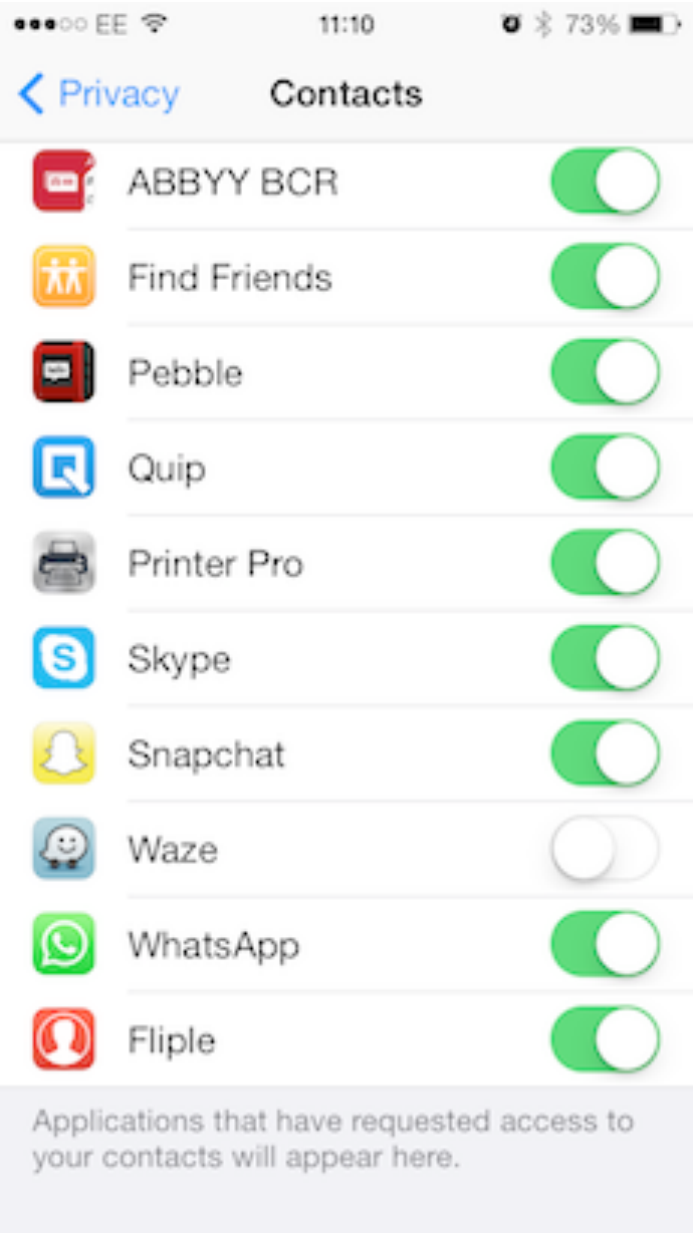
- Cannot really ask for more permissions at runtime
- Can query permissions before we try and do something / allow an app to do something
- Does the app using my Service have a permission?
  - Context.checkCallingPermission()
- Does a process have a permission?
  - Context.checkPermission()
- Does an installed app have a permission?
  - PackageManager.checkPermission()

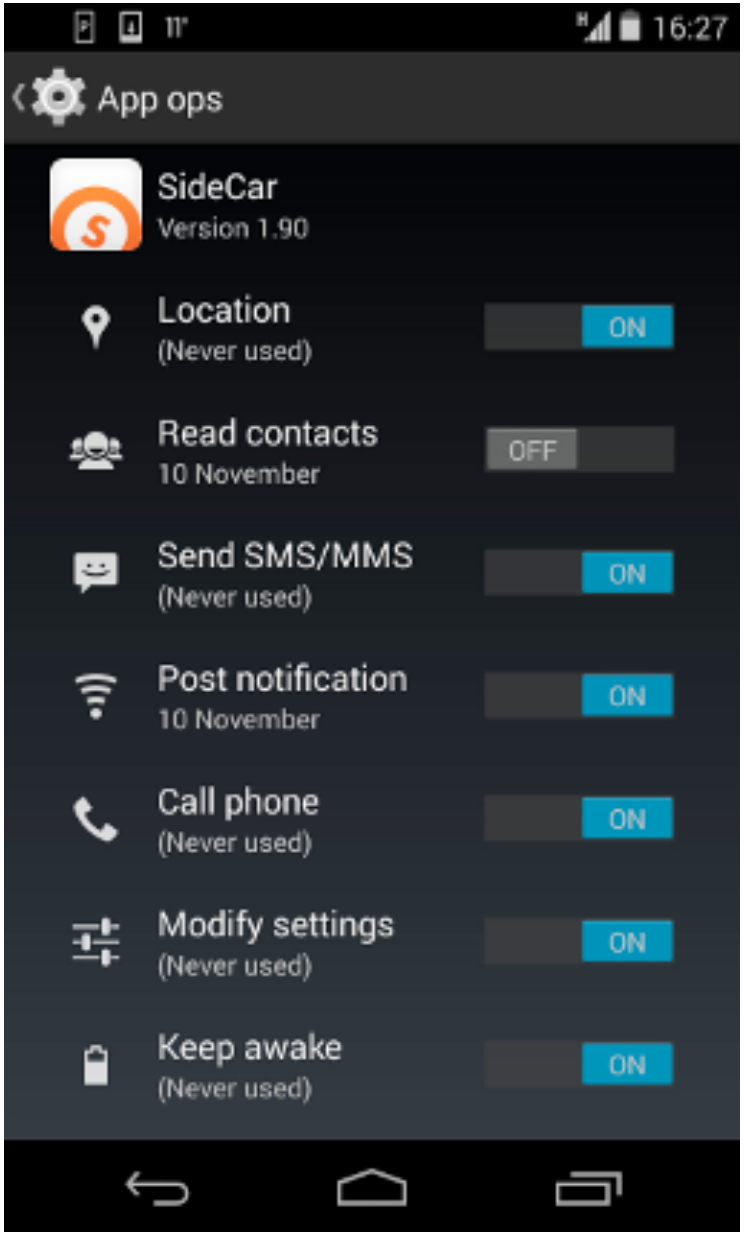
Temporary URI Permissions

- Applications making use of multiple Activities
  - "Access to the mail should be protected by permissions, since this is sensitive user data. However, if a URI to an image attachment is given to an image viewer, that image viewer will not have permission to open the attachment since it has no reason to hold a permission to access all e-mail."
  - Allow access to specific URIs, not the whole provider

```
android:grantUriPermissions="true"
myIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
```

- Temporary URI permissions last while the stack of the receiving Activity is active


EE
11:10
73%
Privacy
Contacts
ABBYY BCR
Find Friends
Pebble
Quip
Printer Pro
Skype
Snapchat
Waze
WhatsApp
Fliple
Applications that have requested access to your contacts will appear here.


16:27
App ops
SideCar
Version 1.90
Location
(Never used)
ON
Read contacts
10 November
OFF
Send SMS/MMS
(Never used)
ON
Post notification
10 November
ON
Call phone
(Never used)
ON
Modify settings
(Never used)
ON
Keep awake
(Never used)
ON

News › Technology › Android

Android torch app with over 50m downloads silently sent user location and device data to advertisers

US Federal Trade Commission charges 'deception' over app which turned on lights on Android smartphones - but also told advertisers about location and device information

4 Application Components

- Activity
  - A UI for part of a task
- Service
  - A long-running background task
- ContentProvider
  - Storage and provision of data
- …all driven by user activity within an application
- BroadcastReceiver

BroadcastReceiver

- Respond to system-wide broadcast announcements
  - The user has not necessarily done something, but the OS / phone / another application has
    - The screen has turned off, the battery is low, a new SMS has arrived, the phone has booted
  - Can be sent by an application, or received by an application from the OS
- Intents are sent to specific Activities / Services
- Broadcasts are sent to anything that cares to listen
  - System-wide Intent broadcasts

Broadcasts

- Broadcasts are Intents
  - Send an Intent to start an Activity
  - Send an Intent to start a Service
  - Send an Intent to trigger Broadcast Receivers that subscribe to that particular class of Intent
    - Can define our own Broadcast Intents
    - Cannot send system Intents (battery, screen etc), as the security model prevents it

BroadcastReceiver

- Declare in AndroidManifest.xml
  - <activity>
  - <service>
  - <provider>
  - **<receiver>**
- Specify broadcast intents we are interested in
  - Listening to system intents may require certain permissions
    - i.e. phone state – why?
- Receiver registered at boot-time / install-time
  - Again, another **entry point** to our application

```
<receiver android:name="MyReceiver" >
    <intent-filter>
      <action android:name=”com.example.martinbroadcast" />
    </intent-filter>
</receiver>
```

Implementing a BroadcastReceiver

- Subclass BroadcastReciever
  - Specify which Intents we are interested in receiving
    - Permissions permitting
- Implement the onReceive() method
- Then…
  - Send a Message to our Activity to change our behavior
    - Reduce the audio volume, play a sound
  - Start an Activity
    - Do we really want to interrupt the user?
  - Start a Service
  - Show a Notification
    - Alert the user that there is something that they need to interact with

BroadcastReceiver types

- Several broadcast methods available
- Normal or Ordered
  - Normal
    - Sent to all registered receivers
    - Undefined order
  - Ordered
    - Sequential processing in priority order
      - Who gets to know that a new message has arrived first?
    - Each step in the sequence can choose to squash the broadcast
- Sticky or Non-Sticky
  - Sticky
    - Store the Intent after the initial broadcast
    - Received by new BroadcastReceivers on registration
  - Non-Sticky
    - Discard the Intent after the broadcast

Caveats

- Either the sender or receiver of a broadcast can implement permissions
  - Control who can send broadcast intents to my application
  - Control who can receive the intents that my application broadcasts
  - Limit broadcasts to my application components only
- Lifecycle
  - A receiver handling broadcasts is considered to be running in the foreground (albeit briefly)
  - Process is aggressively killed once onReceive() has returned
    - No binding, no startActivtiyForResult
  - Long-running code should start a Service instead
    - As with any other Activity
- The application receiving the broadcasts must have been explicitly started / not explicitly stopped
  - Applications cannot intercept broadcasts without the user having some awareness that they have given it permission
- Can register / unregister for broadcast events programmatically
  - Most things that the manifest specifies can be done programmatically

Let’s have a look…

References

- http://developer.android.com/guide/topics/providers/content-providers.html
- http://developer.android.com/guide/components/fundamentals.html
- http://developer.android.com/reference/android/content/BroadcastReceiver.html