

CS475/575

Permissions

Manifest

# Outline

- Introduction
- Where defined (Manifest)
- Details
- Difference between normal and dangerous
- Behavior based on API (23 and above are harder)
- Demos

# Introduction

- Apps isolated (sandboxed), cannot access hardware or software by default
- Must explicitly request permissions in manifest

# Requesting permissions for an App

- Declared in manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.solution_color"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

# Where can I find a comprehensive list?

- You can find a list of Android permissions here:

<https://developer.android.com/reference/android/Manifest.permission.html>

`ACCESS_COARSE_LOCATION`

Allows an app to access approximate location.

`ACCESS_FINE_LOCATION`

Allows an app to access precise location.

`ACCESS_LOCATION_EXTRA_COMMANDS`

Allows an application to access extra location provider commands.

`ACCESS_NETWORK_STATE`

Allows applications to access information about networks.

`ACCESS_NOTIFICATION_POLICY`

Marker permission for applications that wish to access notification policy.

`ACCESS_WIFI_STATE`

Allows applications to access information about Wi-Fi networks.

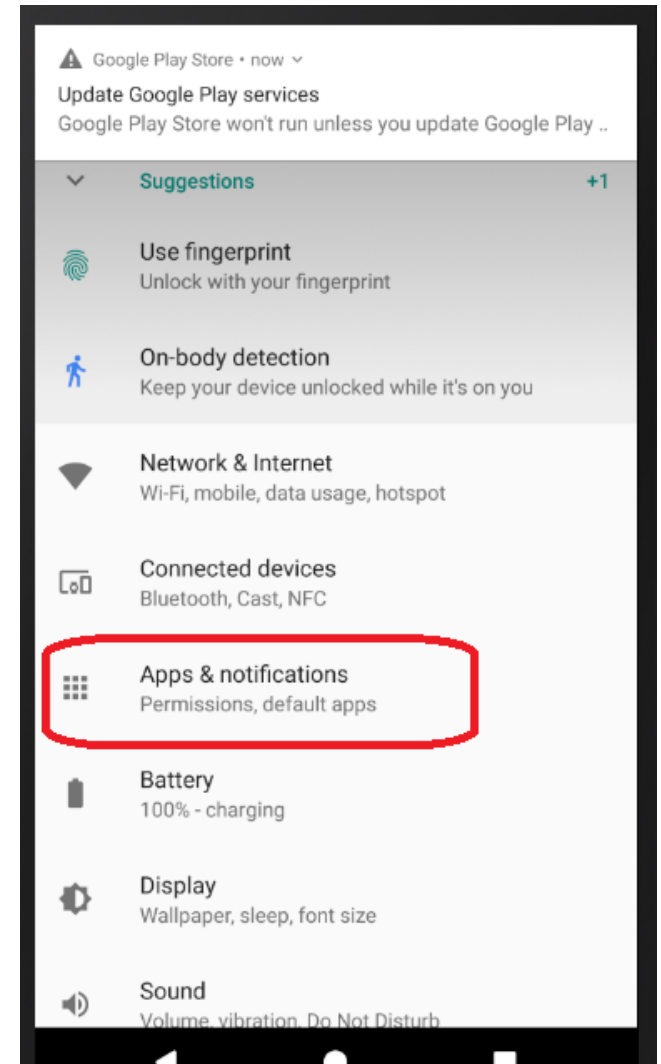
(You can also define your own)

# Is this done in secret?

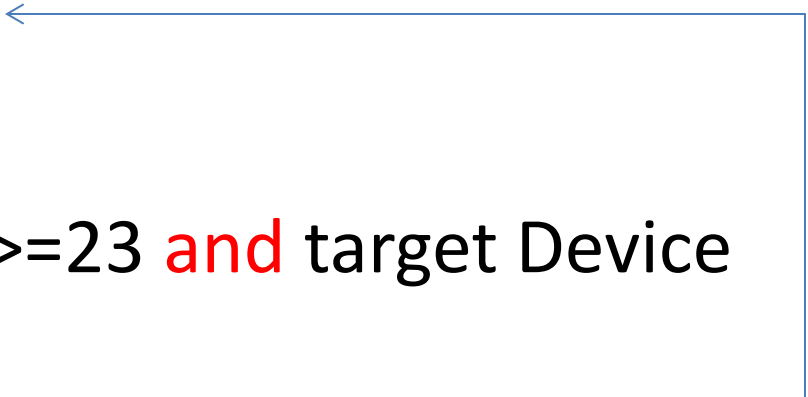
- Are your customers told that they have granted these permissions?
  - Adb install – no
  - Apk email install – no
  - Google Playstore – yes, prior to install
- However...

# How to view permissions

- You can see permissions an app has via Settings - Apps & notifications
- You can also en/disable them



# Where are permissions checked?

- Permissions granted dependent on API
  - If targetSDKversion <23 **or** target Device running <23 then
    - At install (manifest) 
  - If targetSDKversion >=23 **and** target Device running >=23 then
    - Normal permissions – granted at install (like above)
    - **MUCH HARDER** - Dangerous permissions – must ask user at runtime, user can revoke permissions



# Where is targetSdkVersion?

- In build.gradle (app)

```
apply plugin: 'com.android.application'
```

```
android {  
    compileSdkVersion 23  
    buildToolsVersion "23.0.2"  
  
    defaultConfig {  
        applicationId "com.example.solution_color"  
        minSdkVersion 15  
        targetSdkVersion 23  
    }  
}
```

# List of Normal Permissions

<https://developer.android.com/guide/topics/permissions/overview.html#normal-dangerous>

## Normal permissions

*Normal* permissions cover areas where your app needs to access data or resources outside the app's sandbox, but where there's very little risk to the user's privacy or the operation of other apps. For example, permission to set the time zone is a normal permission.

If an app declares in its manifest that it needs a normal permission, the system automatically grants the app that permission at install time. The system does not prompt the user to grant normal permissions, and users cannot revoke these permissions.

As of API level 26, the following permissions are classified as `PROTECTION_NORMAL`:

- `ACCESS_LOCATION_EXTRA_COMMANDS`
- `ACCESS_NETWORK_STATE`
- `ACCESS_NOTIFICATION_POLICY`
- `ACCESS_WIFI_STATE`
- `BLUETOOTH`
- `BLUETOOTH_ADMIN`



The list goes on, as of API 26 there are about 40 of these

# List of Dangerous Permissions

<https://developer.android.com/guide/topics/permissions/overview.html#normal-dangerous>

## Dangerous permissions

*Dangerous* permissions cover areas where the app wants data or resources that involve the user's private information, or could potentially affect the user's stored data or the operation of other apps. For example, the ability to read the user's contacts is a dangerous permission. If an app declares that it needs a dangerous permission, the user has to explicitly grant the permission to the app.

To use a dangerous permission, your app must prompt the user to grant permission at runtime. For more details about how the user is prompted, see [Request prompt for dangerous permission](#).

Permission Group	Permissions
CALENDAR	<ul style="list-style-type: none"><li>• <a href="#">READ_CALENDAR</a></li><li>• <a href="#">WRITE_CALENDAR</a></li></ul>
CAMERA	<ul style="list-style-type: none"><li>• <a href="#">CAMERA</a></li></ul>
CONTACTS	<ul style="list-style-type: none"><li>• <a href="#">READ_CONTACTS</a></li><li>• <a href="#">WRITE_CONTACTS</a></li><li>• <a href="#">GET_ACCOUNTS</a></li></ul>
LOCATION	<ul style="list-style-type: none"><li>• <a href="#">ACCESS_FINE_LOCATION</a></li><li>• <a href="#">ACCESS_COARSE_LOCATION</a></li></ul>



The list goes on, as of API 26 there are about 30 of these

# What do I have to do for normal permissions?

- Define permission in manifest
- It works!

# What do I have to do for dangerous permissions?

- Define permission in manifest
- If API <23 **or** targetSDKVersion <23
  - It Works!

# What do I have to do for dangerous permissions?

- Define permission in manifest
- If API >23 **and** targetSDKVersion >23
- Check if permission already granted at runtime
- If so call your API
- If not ask the user for permission
- Define callback for users choice
- In callback, if yes use permission  
if no you don't have it, notify user  
(maybe give another chance?)

# Cant I just do it one way for all APIs?

- Yep, the dangerous permissions way works for both
- Asks user for API 23 and above
- Silently grants access for below API 23

# Permissions Demo

- Install `Android_RuntimePermissionsBasic`
- On API 19 device
  - Never asks for permissions
- On API 26 device
  - First install asks for permissions and then periodically thereafter

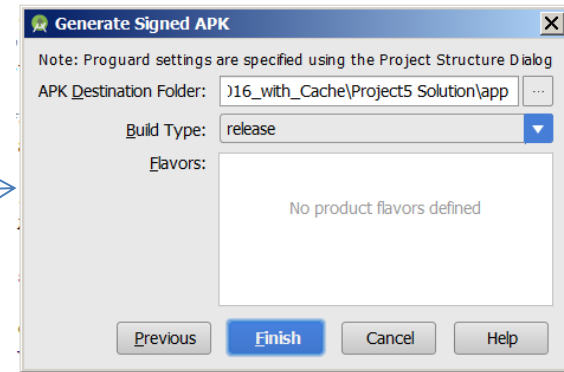
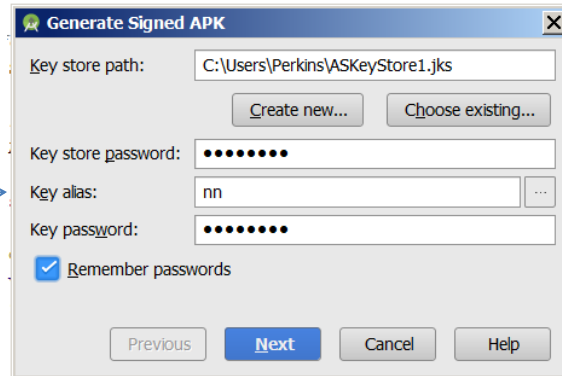
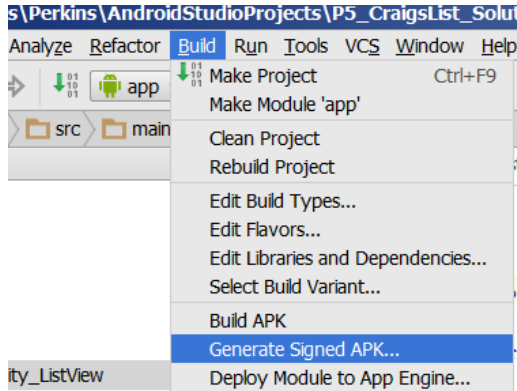


# Do I have to do this?

- Yep... with a very few exceptions
- For instance, can you use an implicit intent to ask Android to find another app to do what you want?
- If so the app it finds will run with its own permissions.
- ie. Ask Android to find a camera app for you verses using the camera itself.

# Done

# Signing an apk



- Uses a self signed digital certificate (ie no 3rd party vouching)
- Developer generates public/private key pair
- Developer makes a hash of the apk and sign(encrypt) the hash using the private key, then attaches to apk
- User of apk verifies that app unmodified by:
  - Making a hash of apk
  - Uses Developer public key to unwrap(decrypt) attached hash
  - Compares 2 hashes if == then owner of priv key generated the hash, if not its been modified
- Signing used by Google play, authors need to have same key to update app on play