



WILLIAM & MARY

CHARTERED 1693

CSCI 445: Mobile Application Security

Lecture 8

Prof. Adwait Nadkarni

Sandboxing, on Android

- Android is a *Linux-based system*
- Apps are security principles, *treated as users*
- Apps acquire *permissions* to access ...
- What separates apps from one another?
- What separates Apps from the kernel?
- What prevents apps from accessing arbitrary storage?



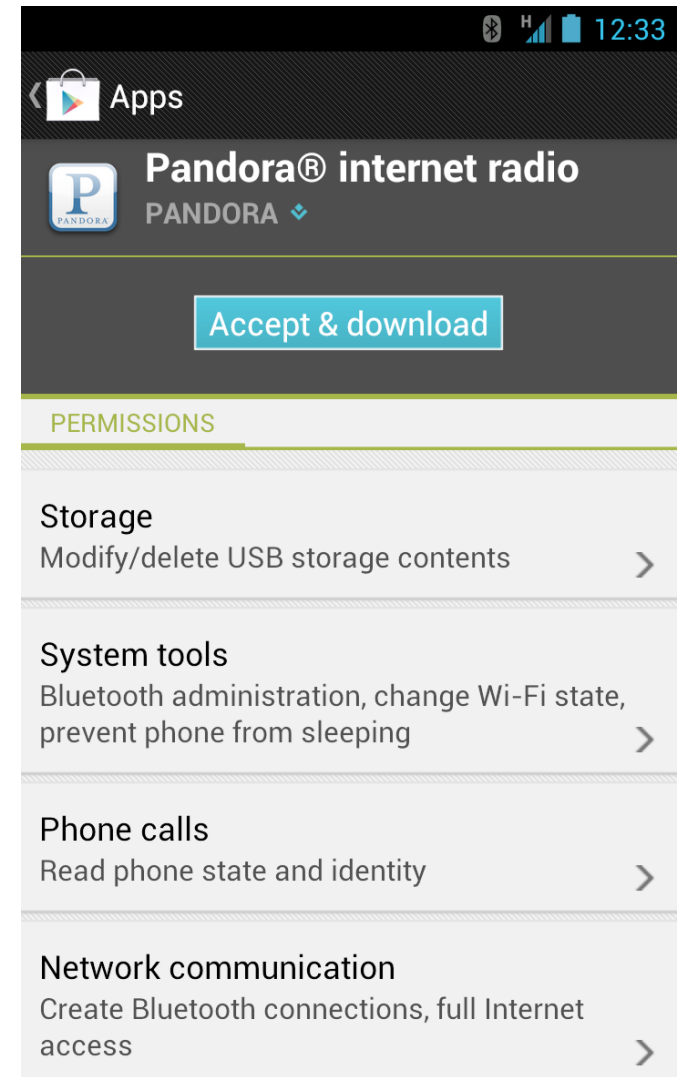
Sandboxing, on Android

- Applications run as different **Linux UIDs**
 - The low-level Android OS is protected by Linux file permissions and SELinux policy
 - All interactions outside the *sandbox* require authorization.
- The middleware ICC is protected by *permissions*
 - A permission is just a text string that gains semantics based on where it is used
(typically, `android.permission.<name>`)
 - Android defines many permissions for protecting resources and sensitive interfaces
(e.g., `android.permission.WRITE_CONTACTS`).
- *Can an app do damage without any permissions?*

Android's Permission Model

On Mobile OSes

- *Permissions define capabilities*
- For accessing objects belonging to the user/system:
 - E.g., SDcard, network, phone IMEI/IMSI, contacts, calendar data, ...
- For accessing objects belonging to other apps:
 - E.g., Interfaces to services exposed by other apps, files/data of other apps



Manifest File

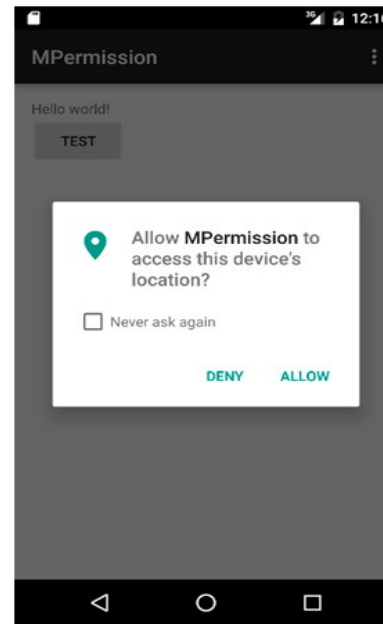
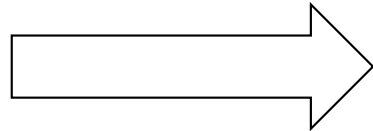
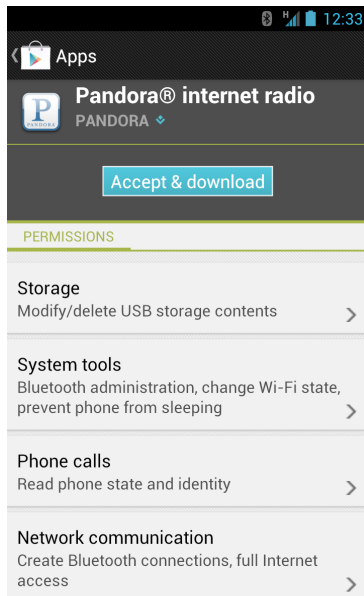
```
10     <action android:name="android.intent.action.MAIN" />
11     <category android:name="android.intent.category.LAUNCHER" />
12 </intent-filter>
13 </activity>
14 <provider android:authorities="friends"
15     android:name="FriendProvider"
16     android:writePermission="org.siislab.tutorial.permission.WRITE_FRIENDS"
17     android:readPermission="org.siislab.tutorial.permission.READ_FRIENDS">
18 </provider>
19 <service android:name="FriendTracker" android:process=":remote"
20     android:permission="org.siislab.tutorial.permission.FRIEND_SERVICE">
21 </service>
22 <receiver android:name="BootReceiver">
23     <intent-filter>
24         <action android:name="android.intent.action.BOOT_COMPLETED"></action>
25     </intent-filter>
26 </receiver>
27 </application>
28
29 <!-- Define Permissions -->
30 <permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></permission>
31 <permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></permission>
32 <permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></permission>
33
34 <!-- Uses Permissions -->
35 <uses-permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></uses-permission>
36 <uses-permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></uses-permission>
37 <uses-permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></uses-permission>
38
39 <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
40 <uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
41 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
42 </manifest>
```

Protection Levels

- **Normal:** Very little risk, *default value*
 - Automatically granted
- **Dangerous:** For high-risk protected operations
 - E.g., user's private data, functioning of the device/other apps.
 - **Must be explicitly granted by the user**
- **Signature:** Granted only when the requesting and declaring apps are signed with the same certificate.
 - **Privileged (flag) (signature/privileged):** May accompany platform-specific signature permissions, which must be *explicitly* allowed/denied to OEM apps, or the device won't boot.
- Q: *Which of these permissions are normal?* Internet, NFC, Bluetooth pairing, kill processes, receive boot complete, IR

Consent: Install vs run-time

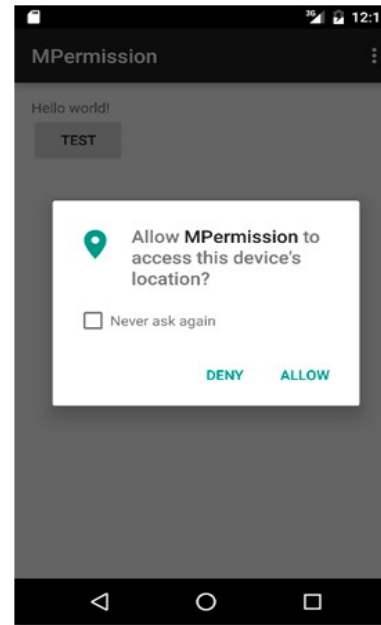
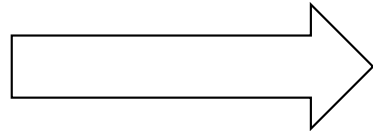
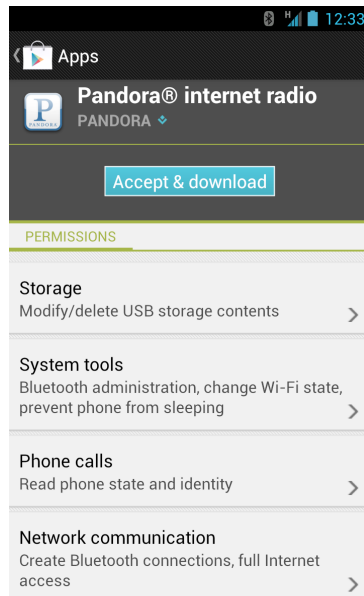
- >100 permissions



- Prior to Android 6.0
 - User accepts all permissions at *install-time*
 - *Class Exercise: Which is better? Why?*
- Android 6.0 and later
 - Apps ask for individual permissions at *run-time*

Install-time vs run-time

- *> 100 permissions*

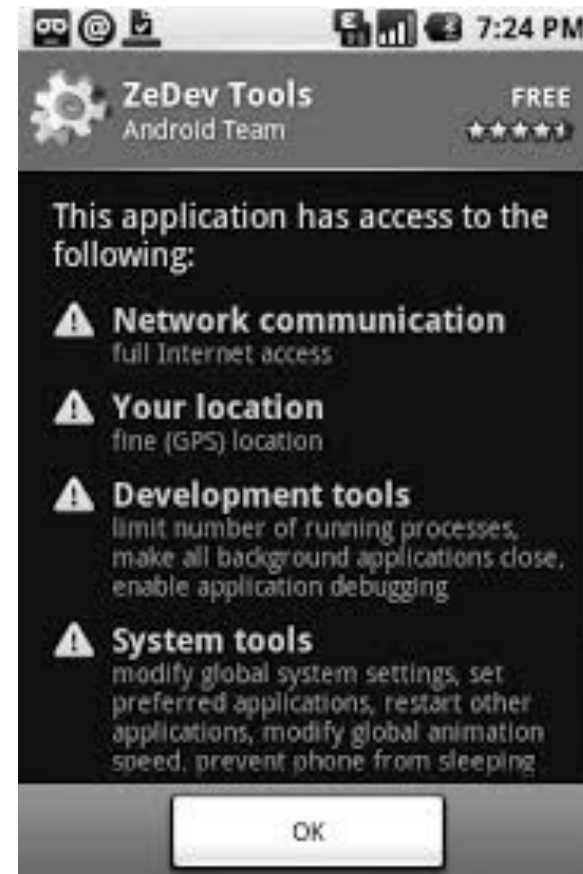


- Prior to Android 6.0
 - User accepts all permissions at *install-time*
 - *How can apps abuse the run-time model?*
- Android 6.0
 - Apps ask for individual permissions at *run-time*

Permissions and Users

Permission Effectiveness

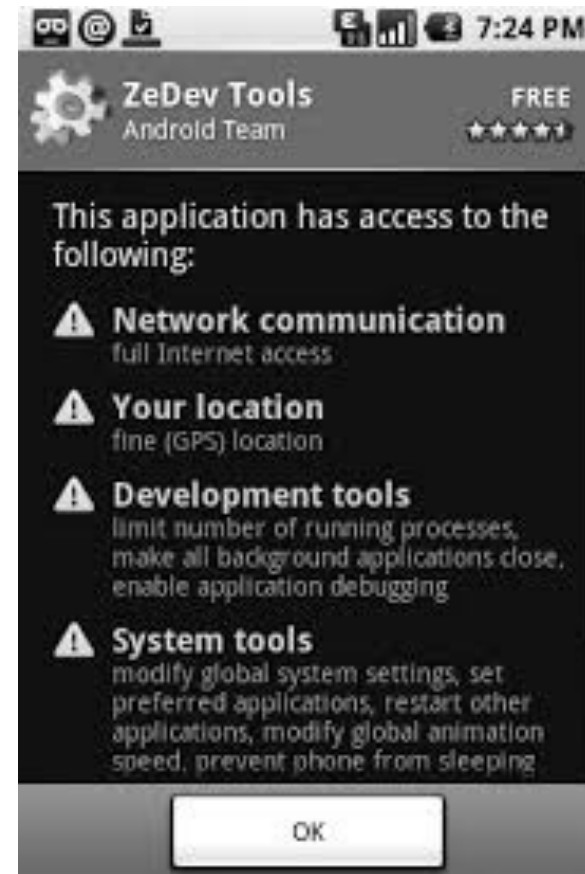
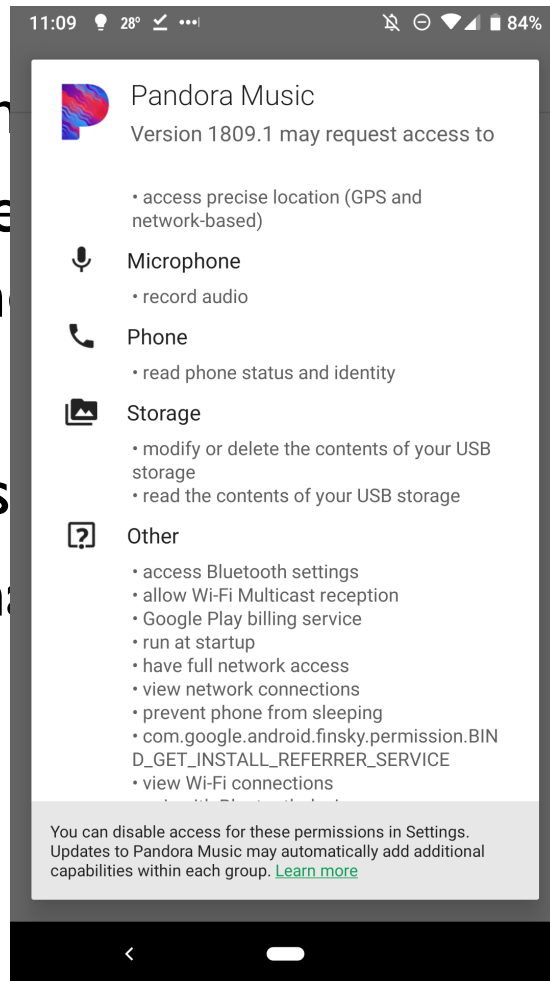
- >100 permissions
- Users often ignore, or do not understand, permissions [1]
- *Why?*
 - Risk vs. assets
 - Information Overload



[1] Felt, Adrienne Porter, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. "Android permissions: User attention, comprehension, and behavior." In *Proceedings of the eighth symposium on usable privacy and security*, ACM, 2012.

Permission Effectiveness

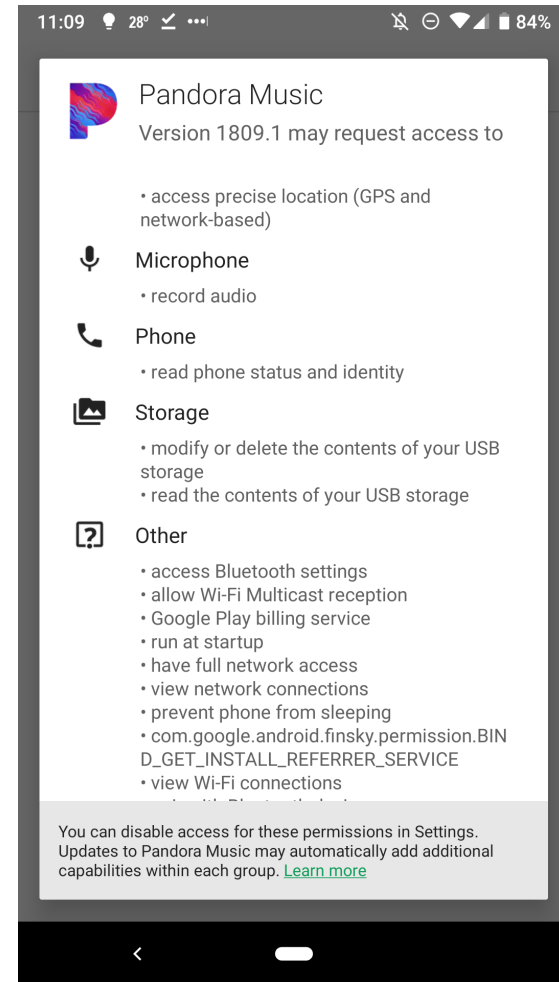
- >100 perm
- Users often understand
- Why?
 - Risk vs
 - Informa



[1] Felt, Adrienne Porter, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. "Android permissions: User attention, comprehension, and behavior." In *Proceedings of the eighth symposium on usable privacy and security*, ACM, 2012.

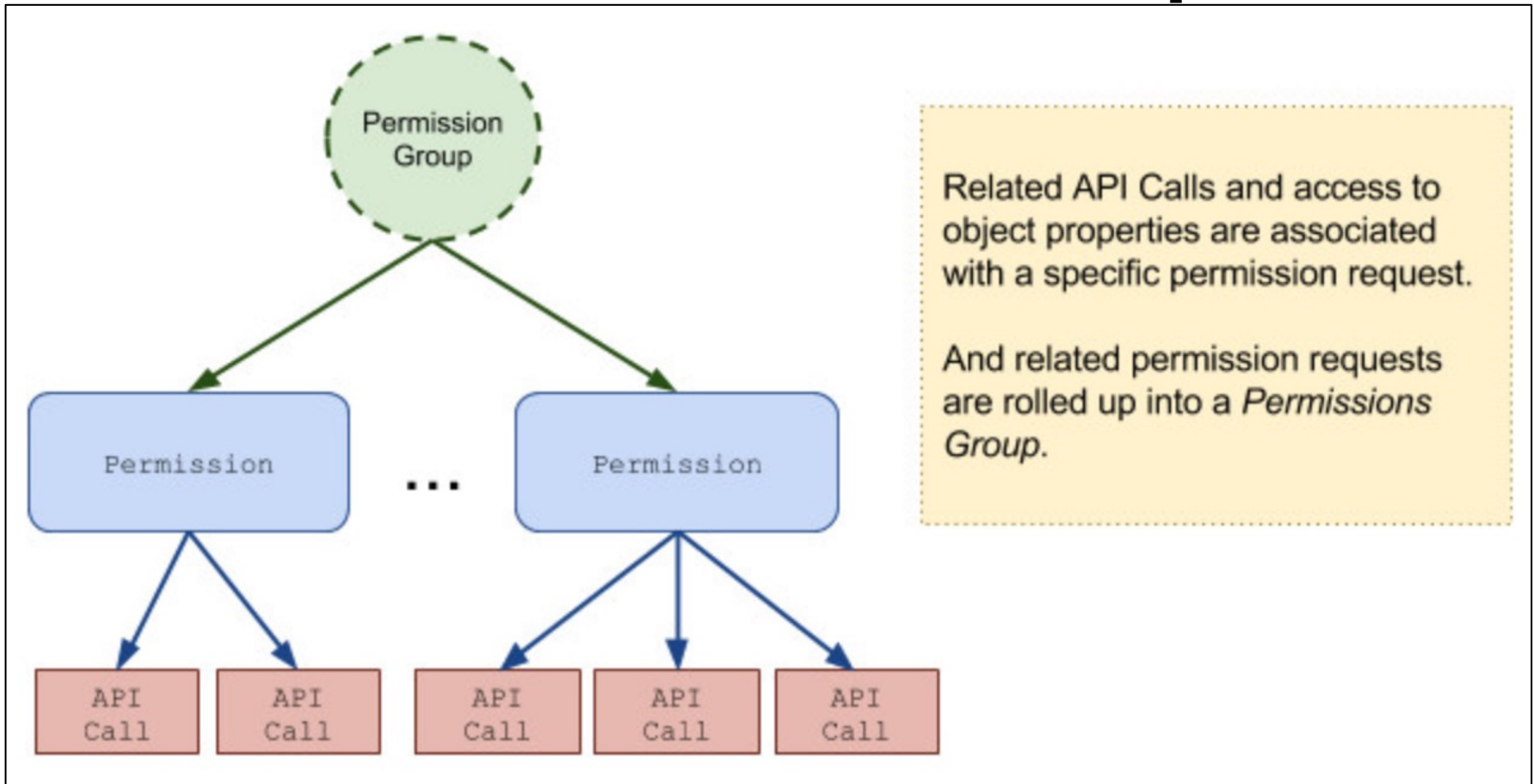
Permission Effectiveness

- >100 permissions
- Users often ignore, or do not understand, permissions [1]
- *Is it better to have fewer permissions?*
 - i.e., coarse-grained permissions for everything?



[1] Felt, Adrienne Porter, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. "Android permissions: User attention, comprehension, and behavior." In *Proceedings of the eighth symposium on usable privacy and security*, ACM, 2012.

Permission Groups



<https://developer.android.com/training/articles/user-data-overview.html>

- Access is granted to the *entire* group. *Implications?*

Permission Groups

- Access is granted to the *entire* group. *Implications?*
- In Group PHONE:
 - READ_PHONE_NUMBERS: Find out the user's phone number. *Relatively benign.*
- Also in Group PHONE:
 - CALL_PHONE: Call *any* phone
 - READ_CALL_LOGS
 - WRITE_CALL_LOGS
 - PROCESS_OUTGOING_CALL: Allows an application to see the number being dialed during an outgoing call with the option to *redirect the call to a different number or abort the call altogether.*

Protecting Users, best practices

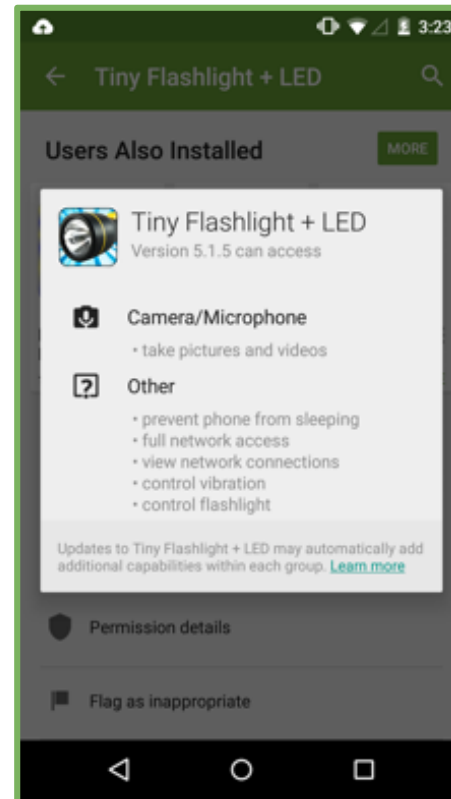
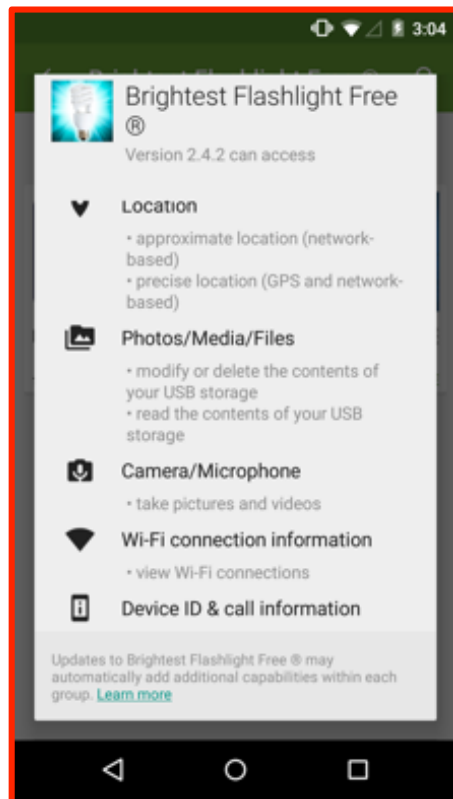
Recall: The Principle of Least Privilege

A system should only provide those rights needed to perform the processes function and no more.

- **Implication 1:** you want to reduce the protection domain to the smallest possible set of objects
- **Implication 2:** you want to assign the minimal set of rights to each subject
- **Caveat:** of course, you need to provide enough rights and a large enough protection domain to get the job done.

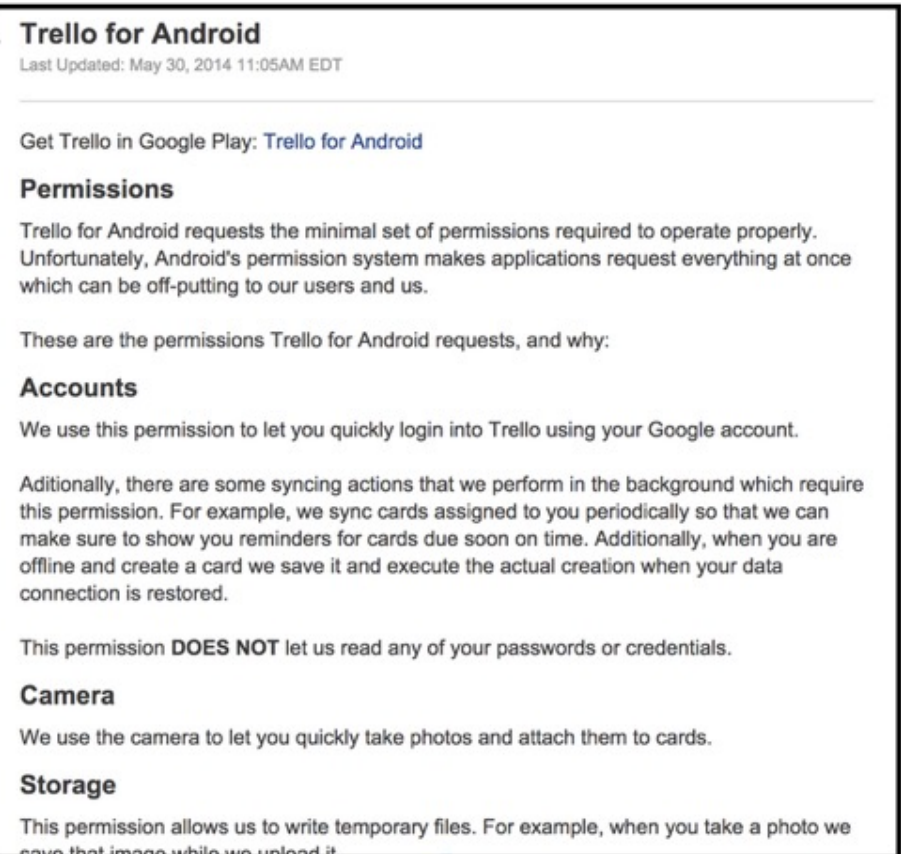
I. Principle of Least Privilege

- *Only request permissions your app requires.*
 - E.g., if you need *approximate* location, use **COARSE_LOCATION**.



2. Describe Permission Use

- Build User Confidence
- Helps you reconsider the permissions requested.



Trello for Android
Last Updated: May 30, 2014 11:05AM EDT

Get Trello in Google Play: [Trello for Android](#)

Permissions

Trello for Android requests the minimal set of permissions required to operate properly. Unfortunately, Android's permission system makes applications request everything at once which can be off-putting to our users and us.

These are the permissions Trello for Android requests, and why:

Accounts

We use this permission to let you quickly login into Trello using your Google account.

Additionally, there are some syncing actions that we perform in the background which require this permission. For example, we sync cards assigned to you periodically so that we can make sure to show you reminders for cards due soon on time. Additionally, when you are offline and create a card we save it and execute the actual creation when your data connection is restored.

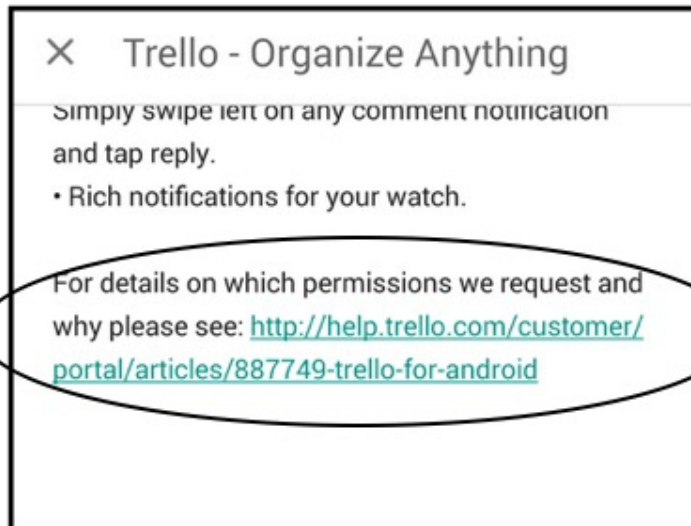
This permission **DOES NOT** let us read any of your passwords or credentials.

Camera

We use the camera to let you quickly take photos and attach them to cards.

Storage

This permission allows us to write temporary files. For example, when you take a photo we save that image while we upload it.



✕ Trello - Organize Anything

Simply swipe left on any comment notification and tap reply.

- Rich notifications for your watch.

For details on which permissions we request and why please see: <http://help.trello.com/customer/portal/articles/887749-trello-for-android>

3. Involve the User

- Often a non-permission option using a “trusted UI”
- Example: insert a contact using an *intent message* instead of requiring the *_CONTACTS permissions

```
// Creates a new Intent to insert a contact
Intent intent = new Intent(Intent.ACTION_INSERT);
// Sets the MIME type to match the Contacts Provider
intent.setType(ContactsContract.RawContacts.CONTENT_TYPE);
// Inserts a phone number
intent.putExtra(Intent.EXTRA_PHONE_NUMBER, mPhoneNumber.getText());
// Inserts an email address
intent.putExtra(Intent.EXTRA_EMAIL, mEmailAddress.getText());
// Sends the Intent
startActivity(intent);
```

3. Involve the User

- Using the camera:
 - Use `MediaStore.ACTION_IMAGE_CAPTURE` or `MediaStore.ACTION_VIDEO_CAPTURE`
 - The system asks for permission on your behalf, captures the photo/video via the camera app, and returns it.



4. Avoid fixed device identifiers

- Device identifiers such as the IMEI are *persistent tracking cookies* and lead to privacy failure
- Alternatives include:
 - Play Store statistics (why do it yourself?)
 - ANDROID_ID - requires OEM support, still some privacy concern

```
import android.provider.Settings.Secure;
...
private String android_id = Secure.getString(getContext().getContentResolver(), Secure.ANDROID_ID);
```

- UUID (track the installation, not the device)

```
import java.util.UUID;
...
String id = UUID.randomUUID().toString();
```

5. Pay attention to libraries

- Libraries (esp. Ad libraries, also SDKs) packaged with the app may request permissions.
 - *Why?*
- Libraries execute as the app's code
 - i.e., within the app's *security context*
- *What is the potential risk?*
 - Library code could misbehave!
 - From the user's perspective, the permission is from your app.



To avoid reading all that, read this ad.

For the problem of constipation, Dulcolax laxative. The only one with 6-8 hours programmed action.

If constipation drives you to hit the books while waiting, we've got news for you! Dulcolax laxative. Its programmed action guarantees results in just 6-8 hours after taking it, and this is something no other laxative can guarantee. From now on, Dulcolax laxative. Definite results and more importantly, always on time!

Always on time.

<https://www.coloribus.com/adsarchive/prints/dulcolax-laxatives-library-2130205/>

6. Some other things to avoid

- Avoid executing code that is not packaged in the app
 - *How is this possible?*
 - *Dynamic code loading!*
- Avoid asking for all runtime permissions at once
 - Ask at the time of API use

Takeaways

- Permissions allow user-input in the access control decisions
- However, there are trade-offs
 - Fine and course-grained permissions
 - Runtime vs install-time
- *Effectiveness depends on developers*