





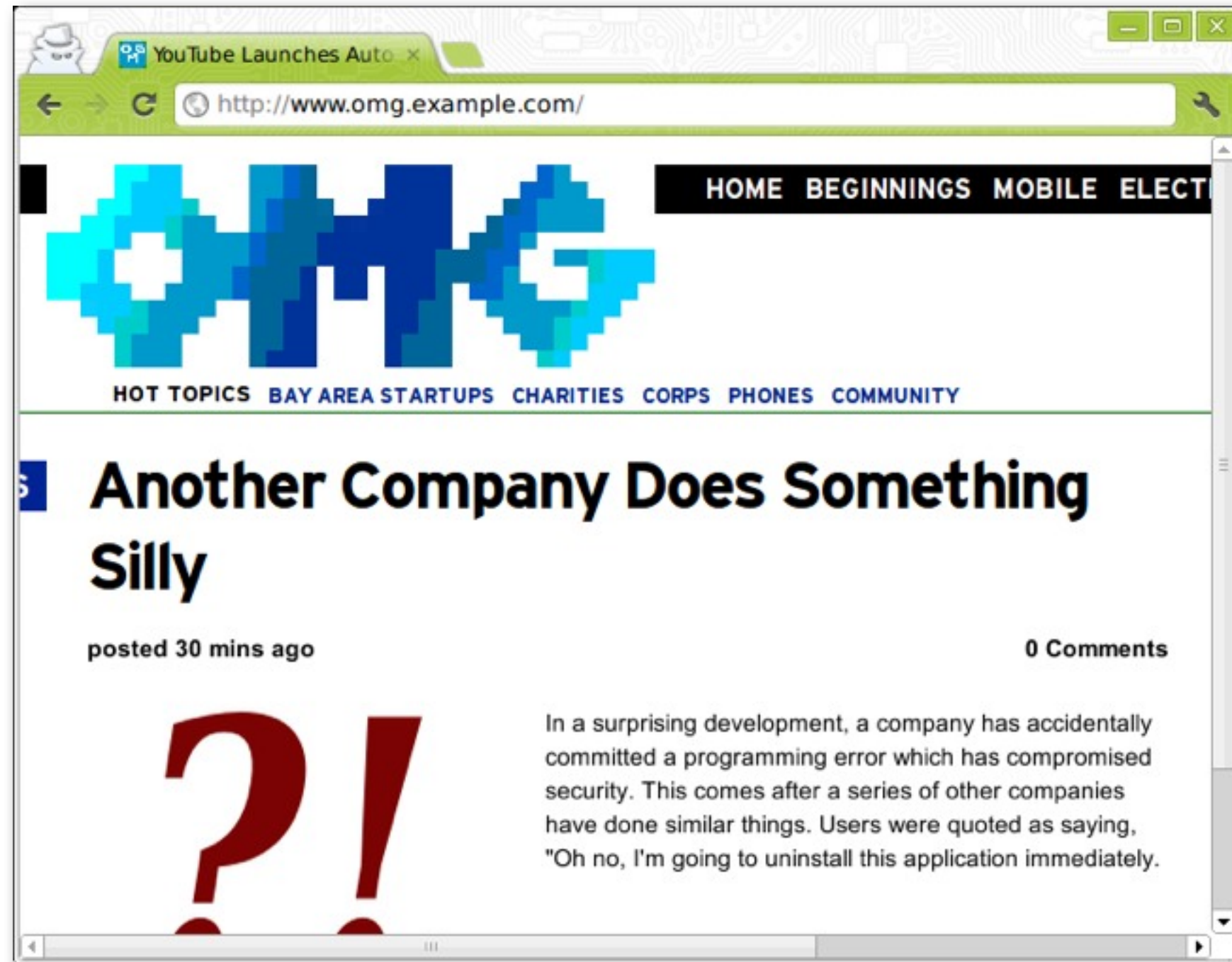
Security and Privacy in Android Apps

Jon Larimer - Security Engineer, Android Team
Kenny Root - Software Engineer, Android Team



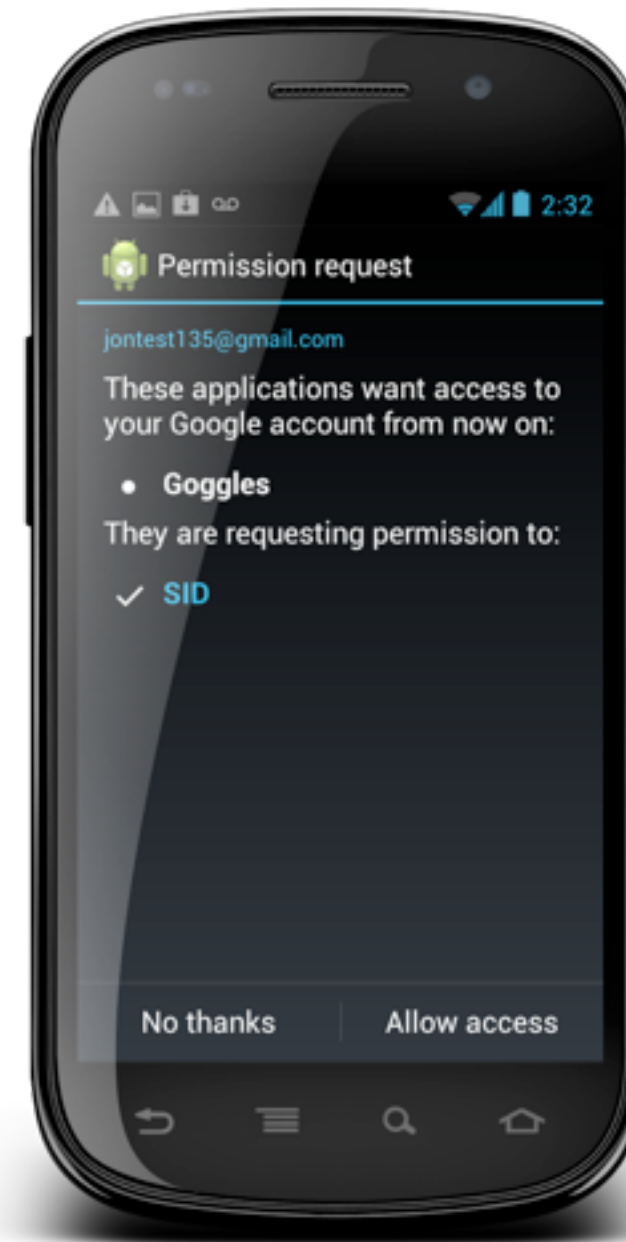
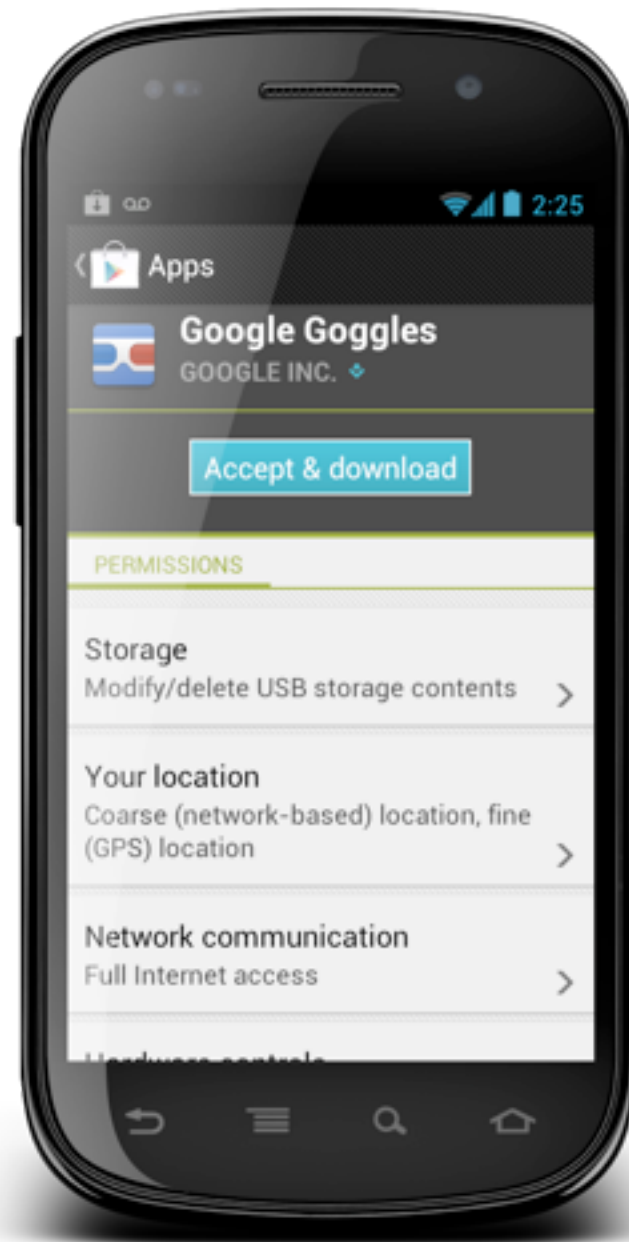
Another privacy breach in the news...

At least it wasn't your app this time!



Mobile devices are full of data...

Android protects access to sensitive data and device capabilities



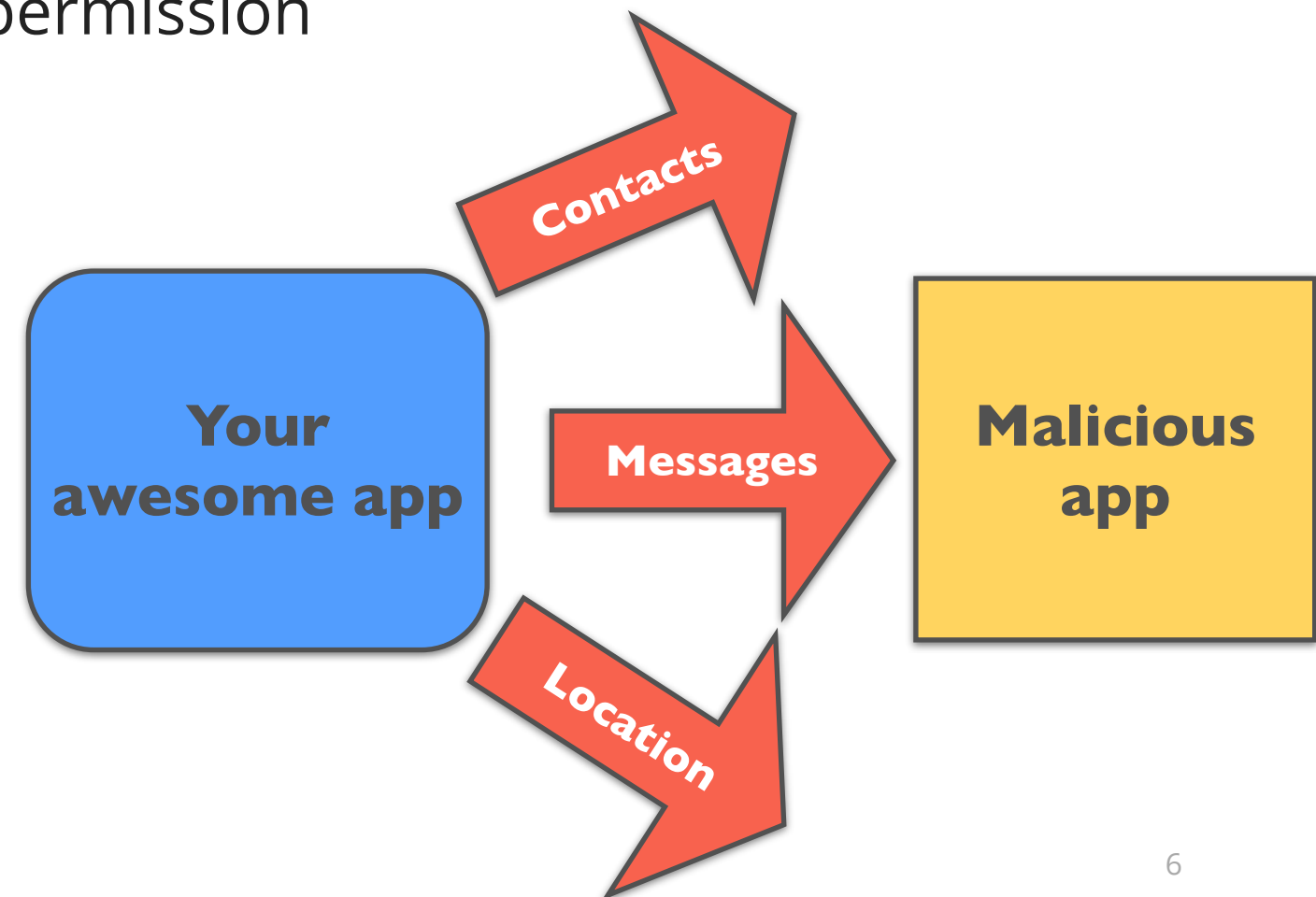
Apps need to respect the data on Android devices

- People generally don't like giving out their personal details to strangers
- Unscrupulous marketers want to mine mobile devices for data
 - User's phone number and email address could be harvested for SPAM
 - Same with the people on their contact lists
- Criminals want to steal your money
 - Sending premium-rate SMS messages from your phone
 - Intercept two-factor authentication messages



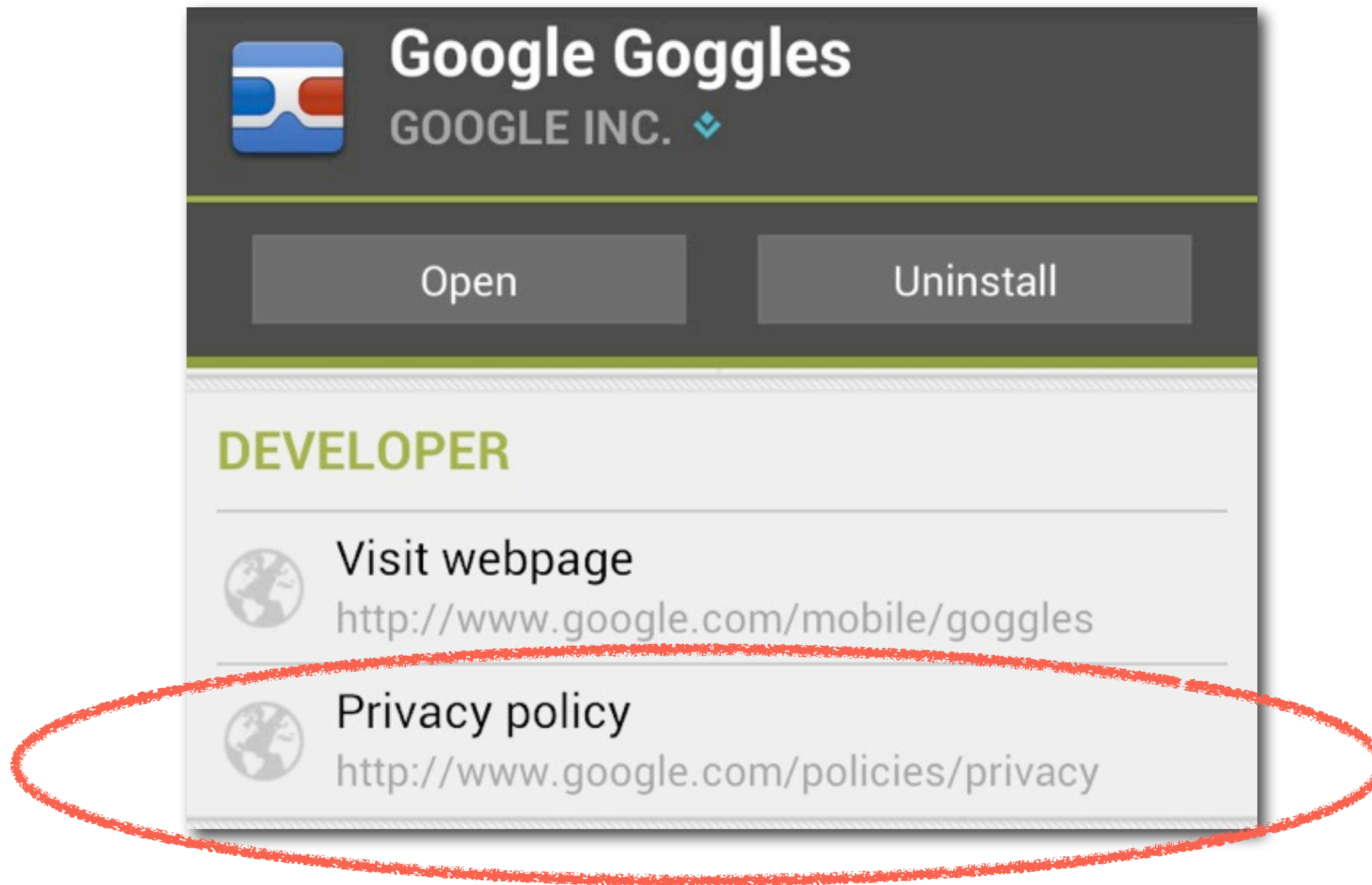
Insecure apps can grant unwanted access to data!

- **When a user allows your app to access some aspect of their phone, they're trusting you with it**
 - **Please don't let them down!**
- If your app requests permissions, a security vulnerability in your app can grant other apps access to the protected data or component without permission
 - Storing personal data in a world-readable file
 - Exporting an unprotected content provider
 - Logging personal data in logcat logs
- It's not just other apps that you need to think about
 - Insecure wireless networks
 - Lost and stolen devices



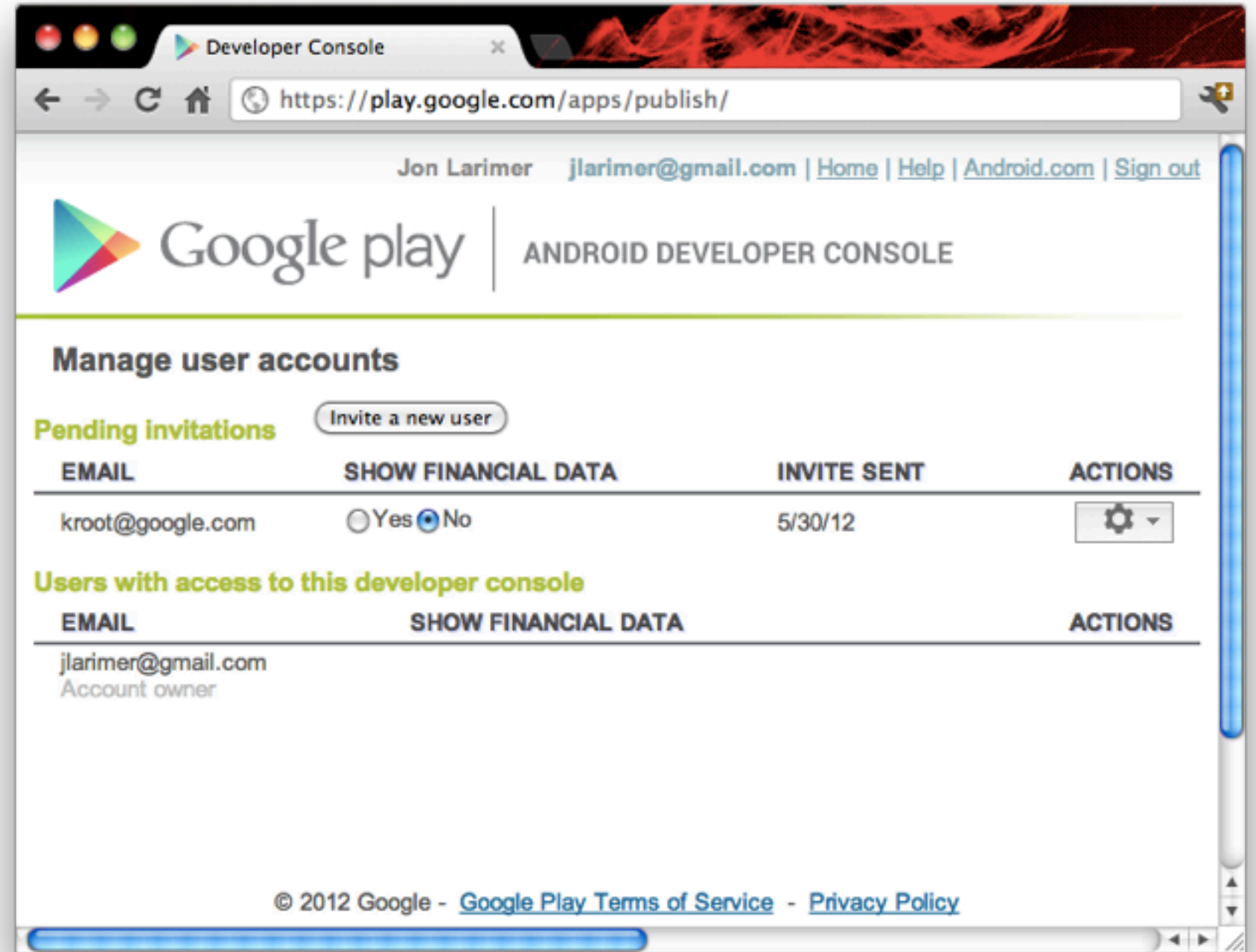
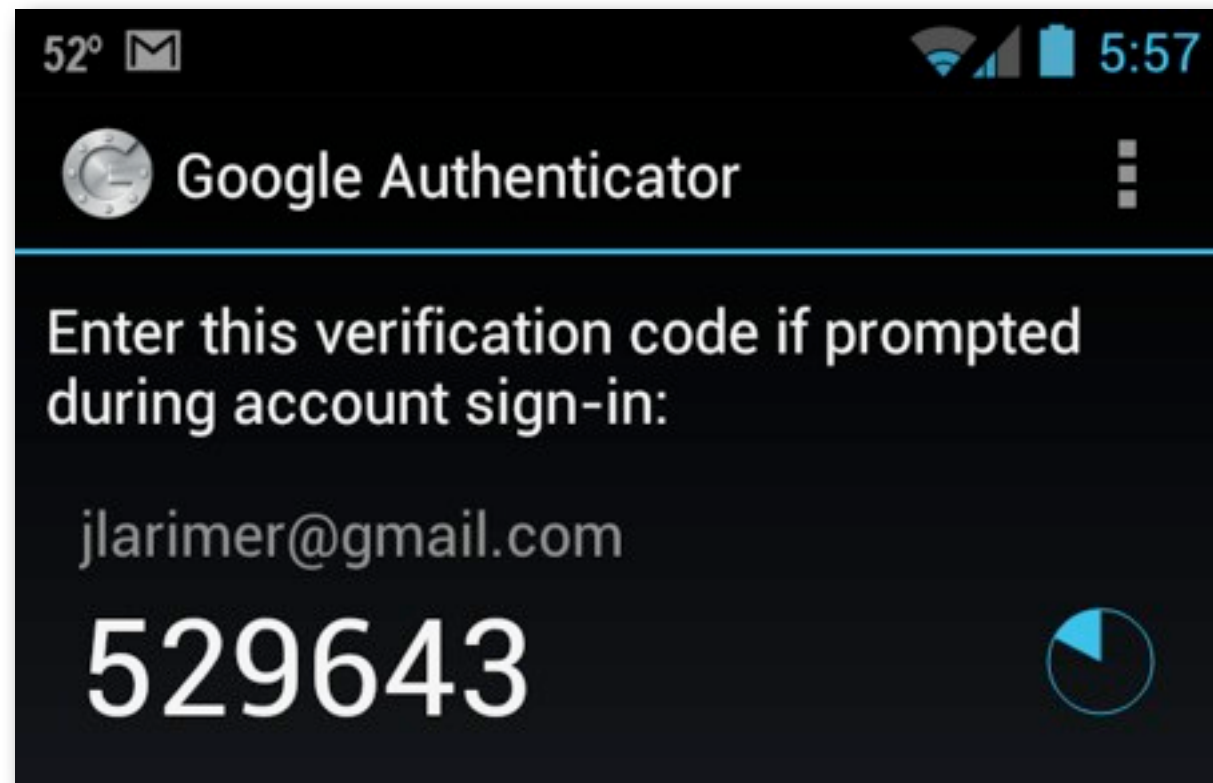
Upload a privacy policy for your app

Let users know what you're going to do with their data



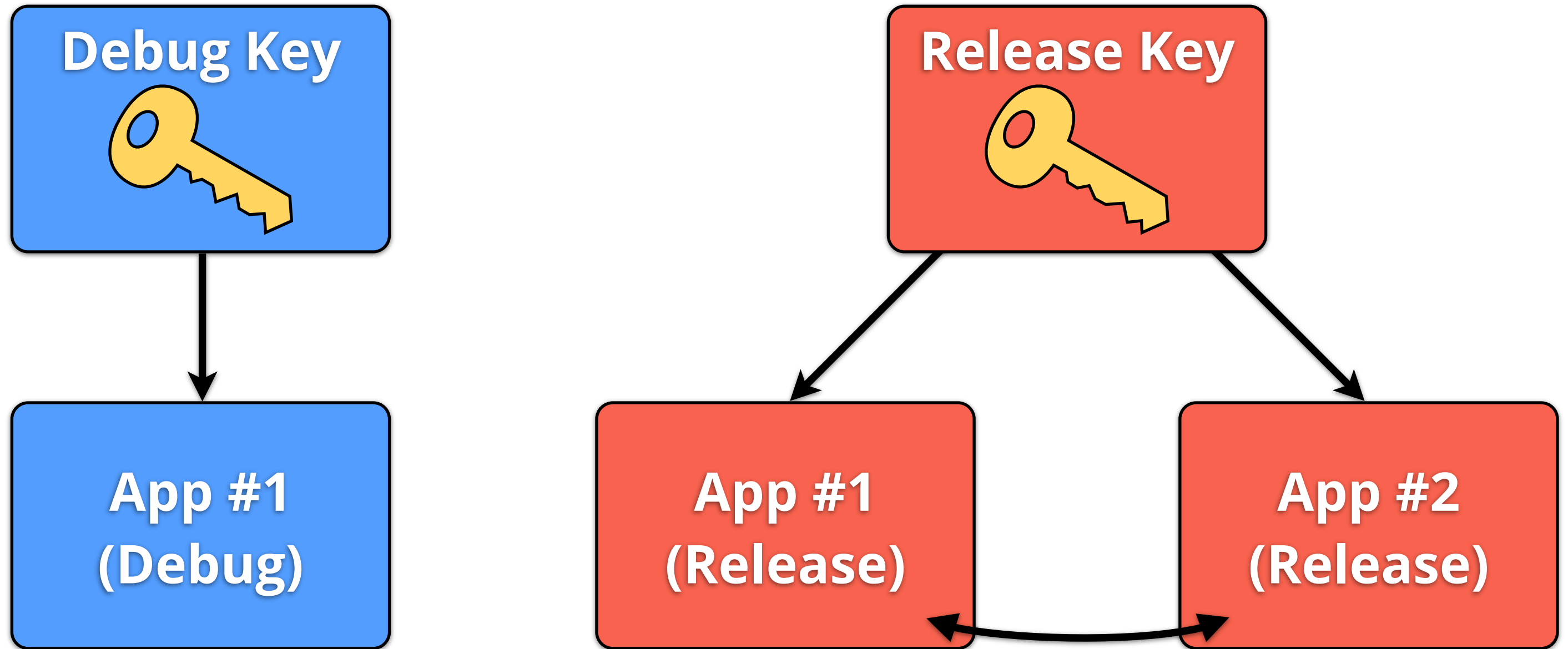
Developer account security

You don't want other people to publish apps as you



Application signing key

Your signing key is part of the identity of your app



Same signing key means `permissionLevel="signature"` works!



Signing key security

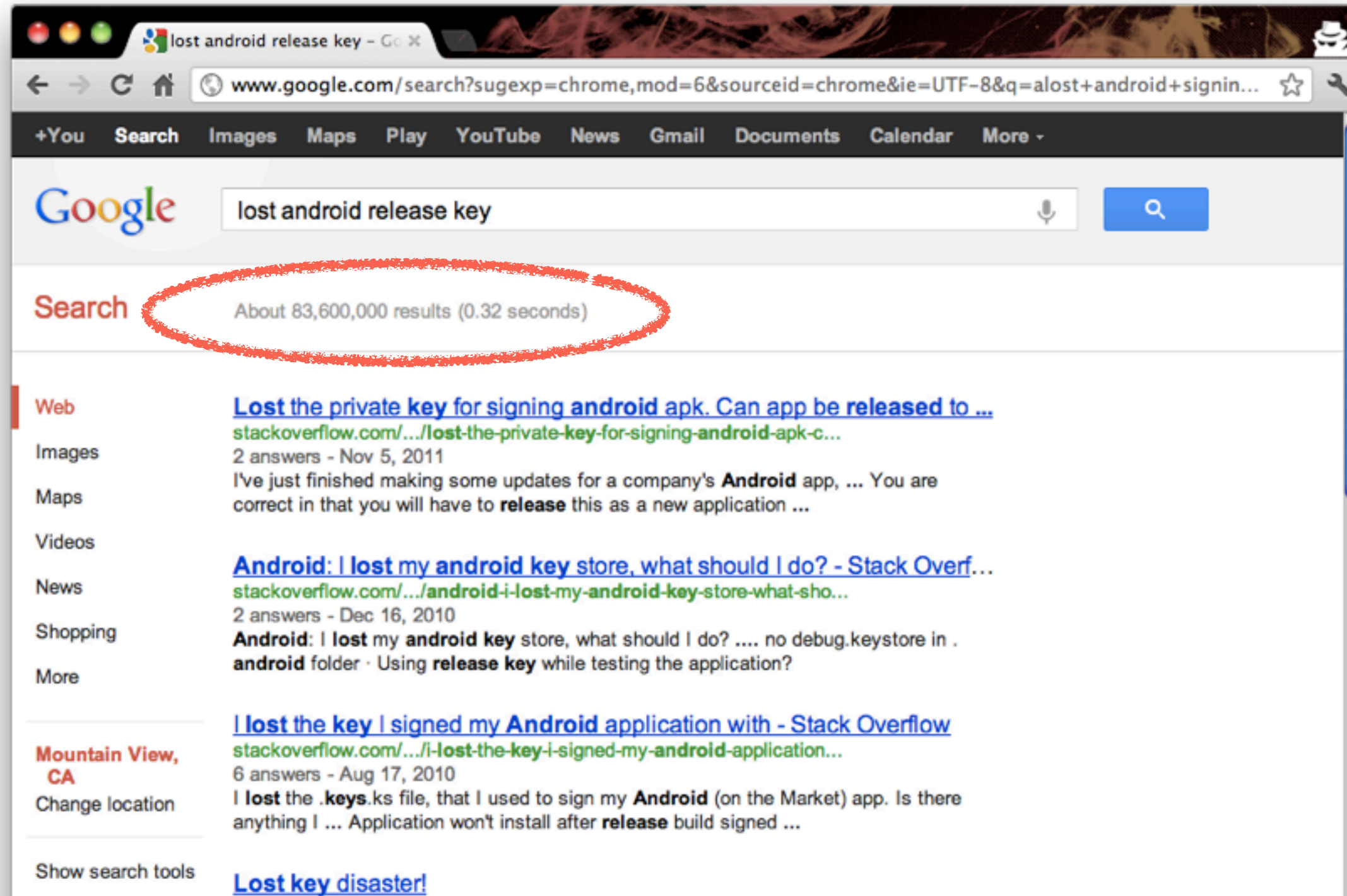
Don't accidentally give out your key!

```
kroot — example
$ unzip -v has-keystore.apk
Archive:  has-keystore.apk
Length  Method  Size  Cmpr  Date   Time   CRC-32  Name
-----  -----  -----  ---  -
  4792  Defl:N   1002   79%  2011-01-25  11:14  9e66f970  res/layout/main.xml
  2012  Defl:N    743   63%  2011-01-25  11:14  e8b11547  AndroidManifest.xml
  1944  Stored   1944    0%  2011-01-25  11:14  37d67f55  resources.arsc
  4691  Stored   4691    0%  2011-01-25  11:14  c7b06c20  res/drawable-hdpi/icon.png
  1537  Stored   1537    0%  2011-01-25  11:14  8ef78580  res/drawable-ldpi/icon.png
  2200  Stored   2200    0%  2011-01-25  11:14  99a4f90b  res/drawable-mdpi/icon.png
 17620  Defl:N   9196   48%  2011-01-25  11:14  f5f55b5e  classes.dex
   741  Defl:N    412   44%  2011-01-25  11:14  bc5c9864  META-INF/MANIFEST.MF
   794  Defl:N    441   45%  2011-01-25  11:14  136fe651  META-INF/CERT.SF
   771  Defl:N    598   22%  2011-01-25  11:14  b555638c  META-INF/CERT.RSA
  1262  Defl:N   1203    5%  2011-01-25  11:14  7ae790e7  prod.keystore
-----  -----  -----  ---  -
 38364                23967  38%
$
```



Signing key security

Don't lose your key!



A screenshot of a Google search page. The search bar contains the text "lost android release key". Below the search bar, the word "Search" is followed by "About 83,600,000 results (0.32 seconds)", which is circled in red. The search results are categorized by type: Web, Images, Maps, Videos, News, Shopping, and More. The "Web" category is selected, showing three search results from Stack Overflow. The first result is titled "Lost the private key for signing android apk. Can app be released to ..." and is dated Nov 5, 2011. The second result is titled "Android: I lost my android key store, what should I do? - Stack Overf..." and is dated Dec 16, 2010. The third result is titled "I lost the key I signed my Android application with - Stack Overflow" and is dated Aug 17, 2010. The location is set to Mountain View, CA.

lost android release key - Go X

www.google.com/search?sugexp=chrome,mod=6&sourceid=chrome&ie=UTF-8&q=alost+android+signin...

+You Search Images Maps Play YouTube News Gmail Documents Calendar More -

Google lost android release key

Search About 83,600,000 results (0.32 seconds)

Web [Lost the private key for signing android apk. Can app be released to ...](#)
stackoverflow.com/.../lost-the-private-key-for-signing-android-apk-c...
2 answers - Nov 5, 2011
I've just finished making some updates for a company's **Android** app, ... You are correct in that you will have to **release** this as a new application ...

Images

Maps

Videos

News [Android: I lost my android key store, what should I do? - Stack Overf...](#)
stackoverflow.com/.../android-i-lost-my-android-key-store-what-sho...
2 answers - Dec 16, 2010
Android: I lost my android key store, what should I do? no debug.keystore in .android folder · Using release key while testing the application?

Shopping

More

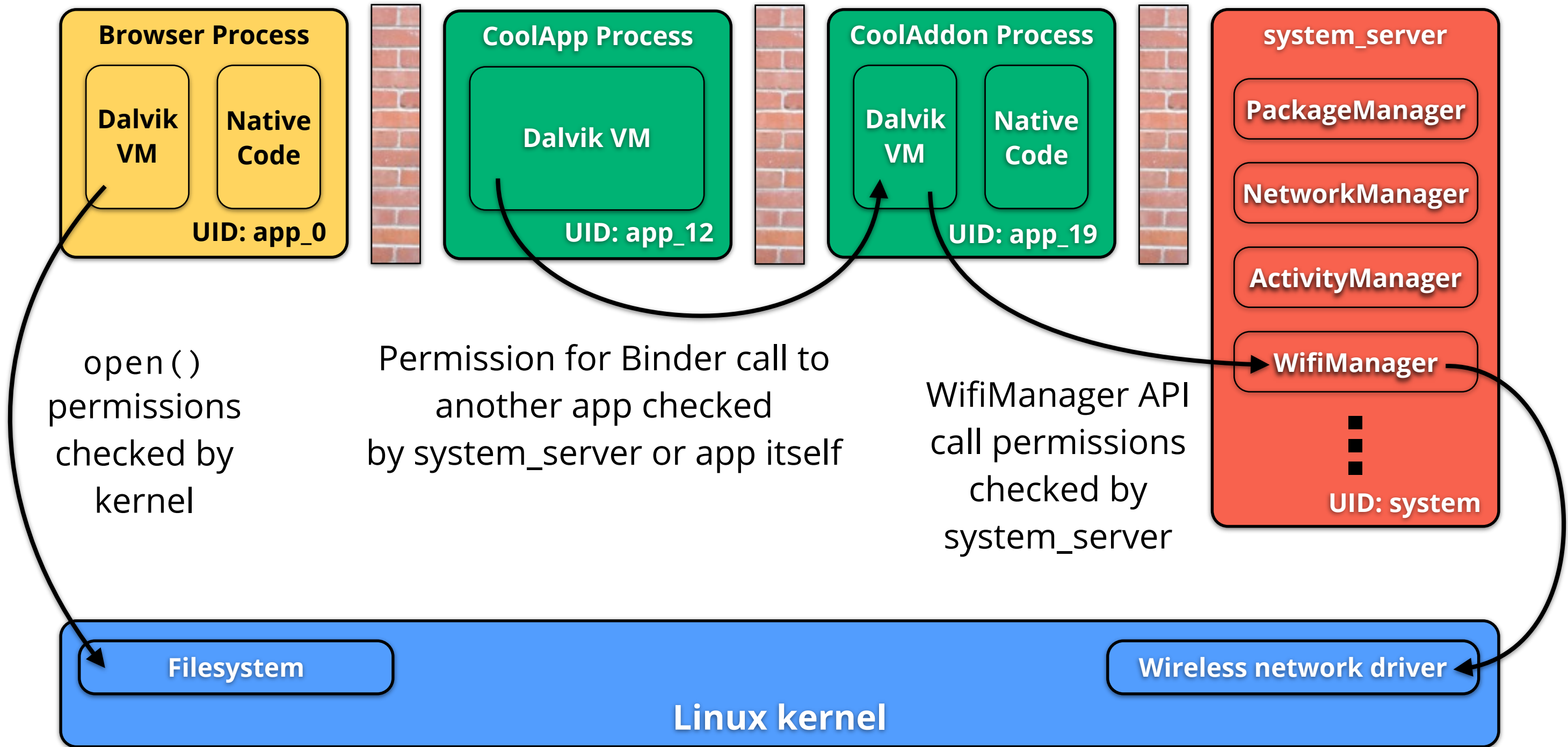
Mountain View, CA
Change location

Show search tools [I lost the key I signed my Android application with - Stack Overflow](#)
stackoverflow.com/.../i-lost-the-key-i-signed-my-android-application...
6 answers - Aug 17, 2010
I **lost** the .keys.ks file, that I used to sign my **Android** (on the Market) app. Is there anything I ... Application won't install after **release** build signed ...

[Lost key disaster!](#)



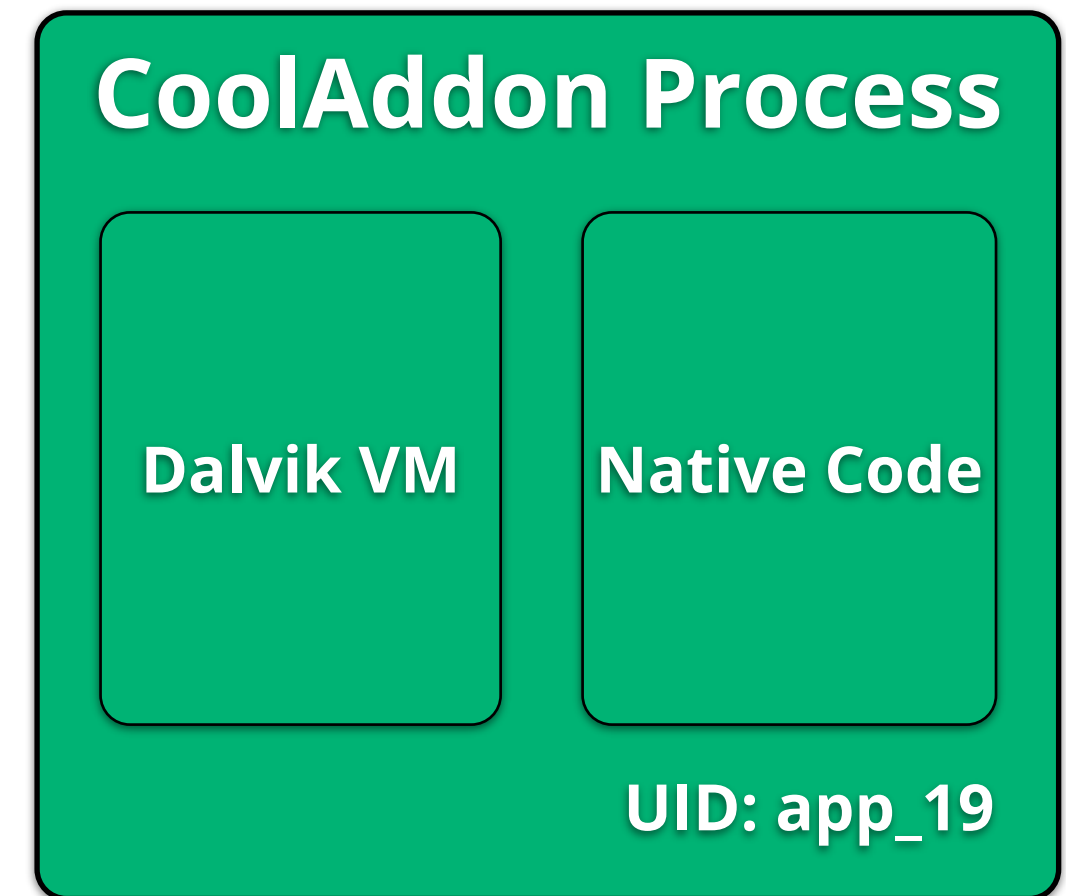
Security architecture of Android



Security for your app

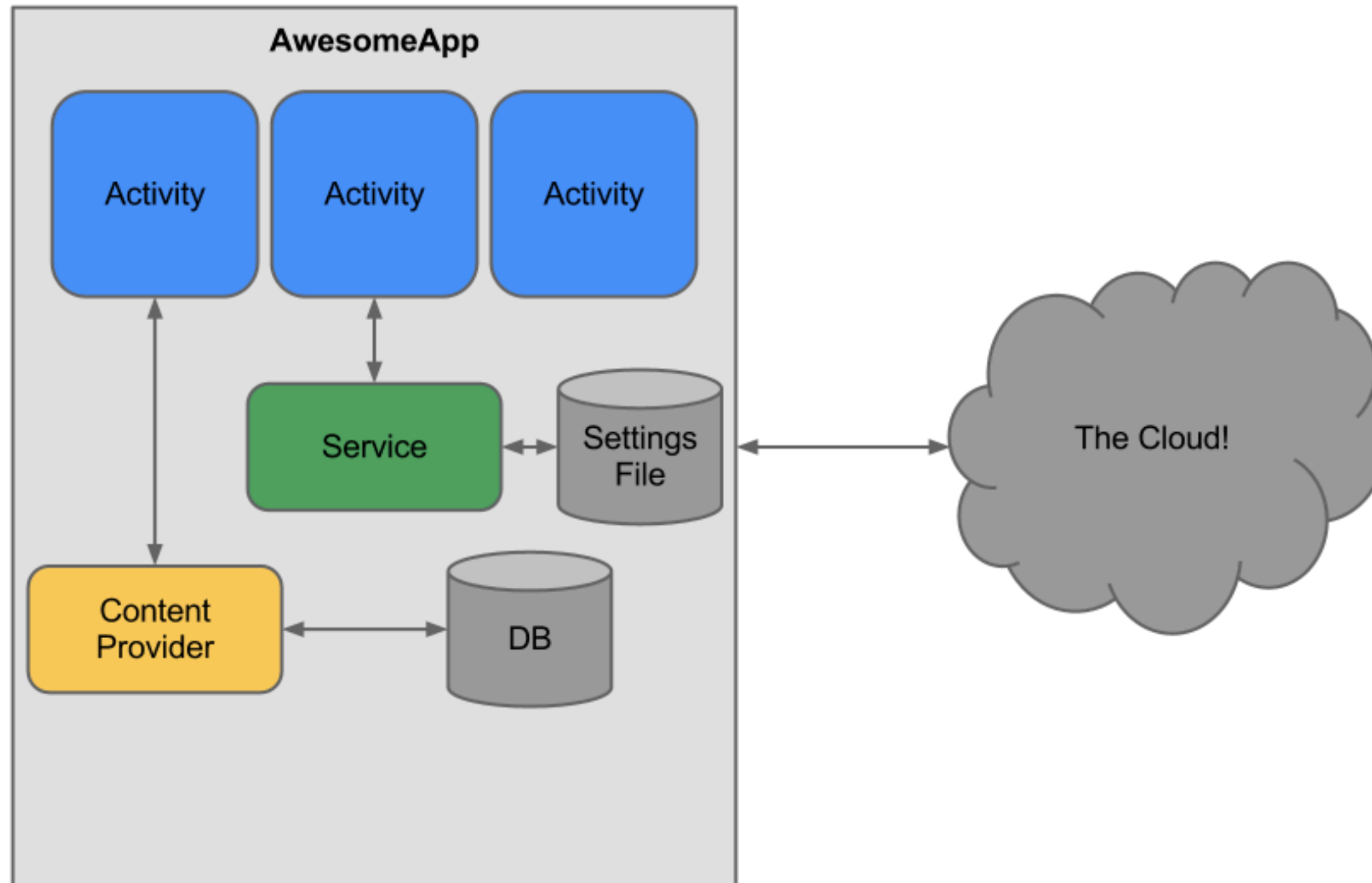
The application is in its own process sandbox.

- Dalvik gives you the freedom to add your own crypto implementations
 - Reflection can be used to bypass scoping
 - **private** and **protected** may be ignored
 - Native code can access and change data in the current process's Dalvik VM - don't rely on the VM to provide security!
- For inter-process communication, there are protections:
 - Intent filters
 - Permissions
 - Signatures



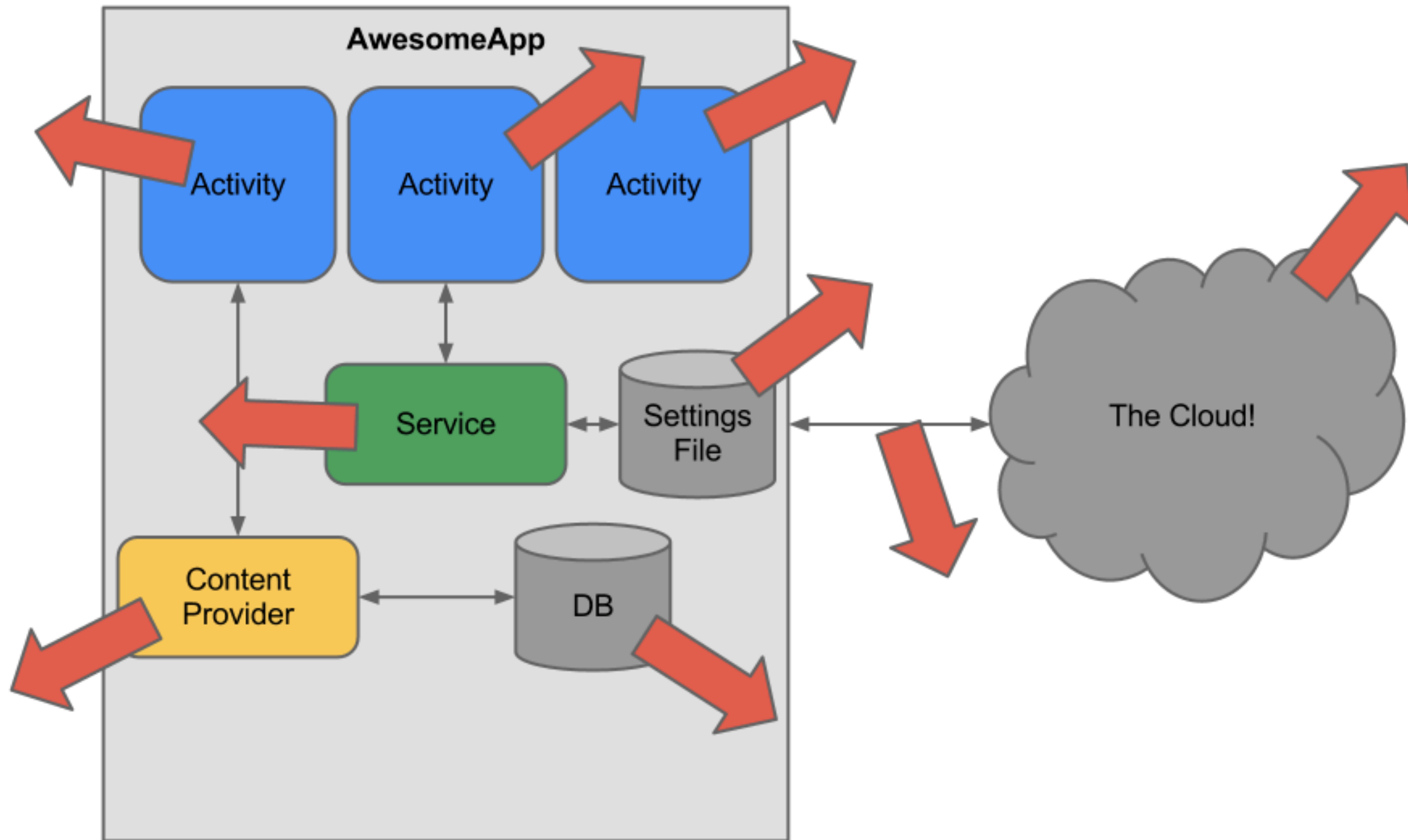
Typical application

Where's the attack surface?



Typical application

Where's the attack surface?



Protecting app components

App components and the AndroidManifest.xml file

- Accessible app components are declared in the **AndroidManifest.xml** file
 - Activities – **<activity>**
 - Services – **<service>**
 - Broadcast receivers – **<receiver>**
 - Content providers – **<provider>**
- Components specify what kind of **Intent** they accept with an **<intent-filter>** in the manifest
 - If a component has an **<intent-filter>** in the **AndroidManifest.xml** file, it's exported by default
 - Content providers are the exception: they export data by default
- Don't export app components unless you want other apps on the system to interact with your app



Limit access to components by external apps

This service has an intent filter so it must be explicitly marked as not exported

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.awesome">
    <application android:label="@string/app_name">
        ...
        <service android:name=".ServiceExample"
            android:exported="false">
            <intent-filter>...</intent-filter>
        </service>
        ...
    </application>
</manifest>
```



Permissions for application components

Using permissions on exported components

- There are different permission protection levels available for apps:
 - **protectionLevel="normal"** – A lower-risk permission that gives requesting applications access to isolated application-level features, with minimal risk to other applications, the system, or the user. This is the default protection level.
 - **protectionLevel="dangerous"** – A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user.
 - **protectionLevel="signature"** – Can be used to limit access to components to only apps signed with the same certificate.



Limit access to an exported component by permission

In this example an application signed with the same key can access the service

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.awesome">
    <permission android:name="com.example.awesome.EXAMPLE_PERM"
        android:label="@string/example_perm_desc"
        android:protectionLevel="signature" />
    <application android:label="@string/app_name">
        <service android:name=".ServiceExample"
            android:permission="com.example.awesome.EXAMPLE_PERM">
            <intent-filter>...</intent-filter>
            ...
        </service>
    </application>
</manifest>
```

Define a permission

Require the permission to access this service



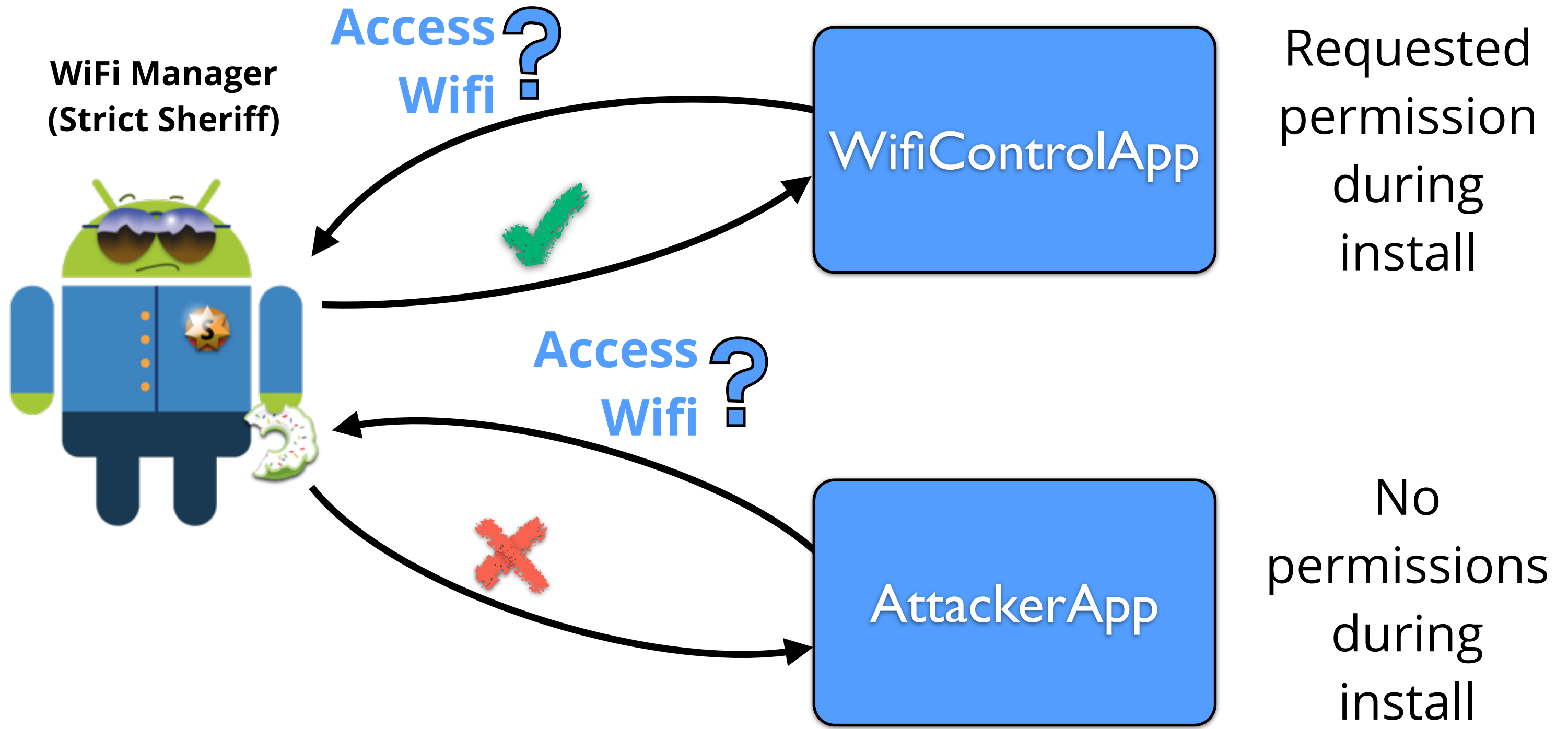
Checking permissions in code

Sometimes you want finer-grained control over how permissions are enforced

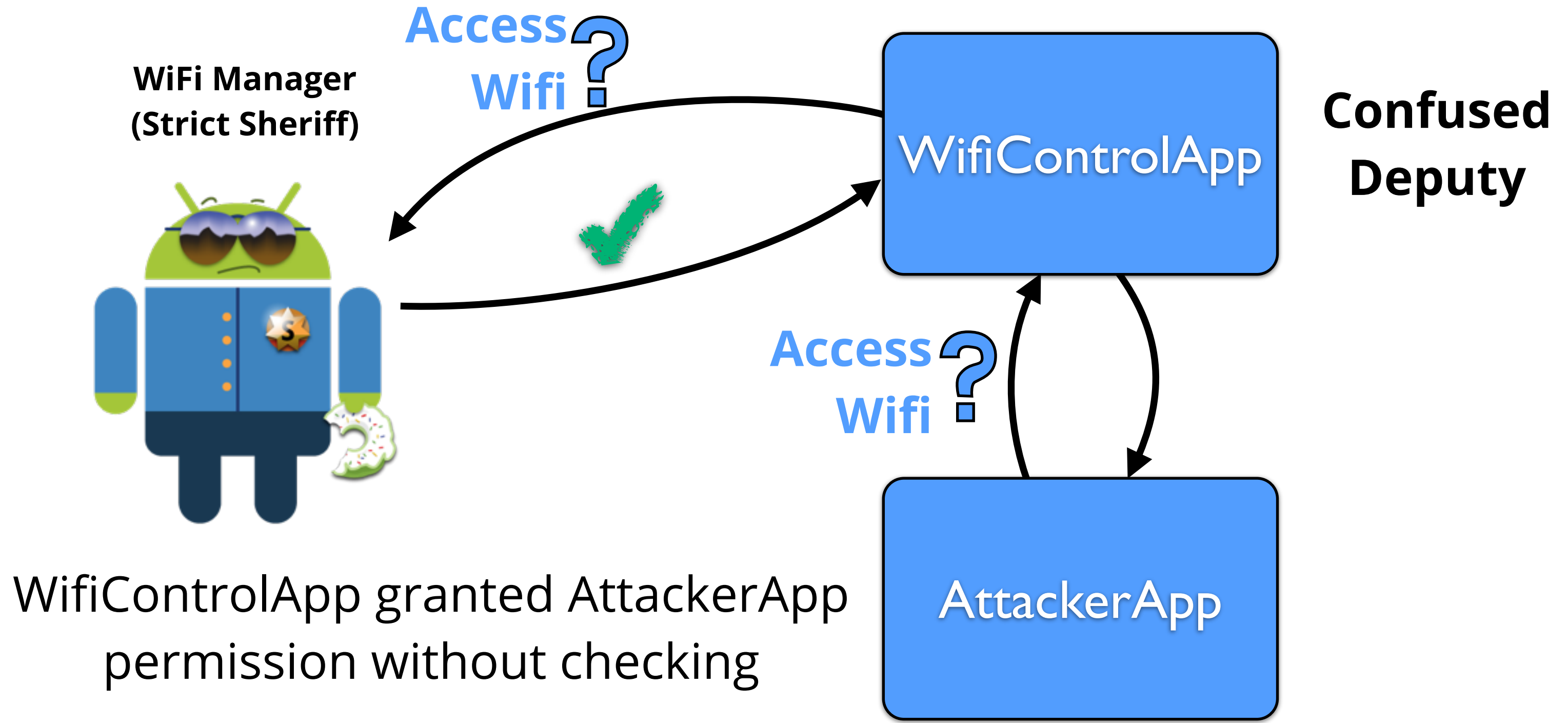
- The **AndroidManifest.xml** should be used whenever possible to declare required permission.
- However, if it's not possible, there are other ways:
 - **Context.registerReceiver(...)** can be used to register a BroadcastReceiver dynamically
 - There is a version of **registerReceiver(...)** which can be used to specify permission the broadcaster must hold for your dynamically-registered receiver to be invoked.
 - **Context.checkCallingPermission(...)** and **Context.enforceCallingPermission(...)** can be used in your source code to make sure the calling app holds the appropriate permission.
 - This can be used to implement fine-grained permissions if needed.
- Avoid the **confused deputy** problem:
 - If your app is using its granted permissions to respond to another app, check that the calling app has that permission as well.



Avoid being the confused deputy



Avoid being the confused deputy

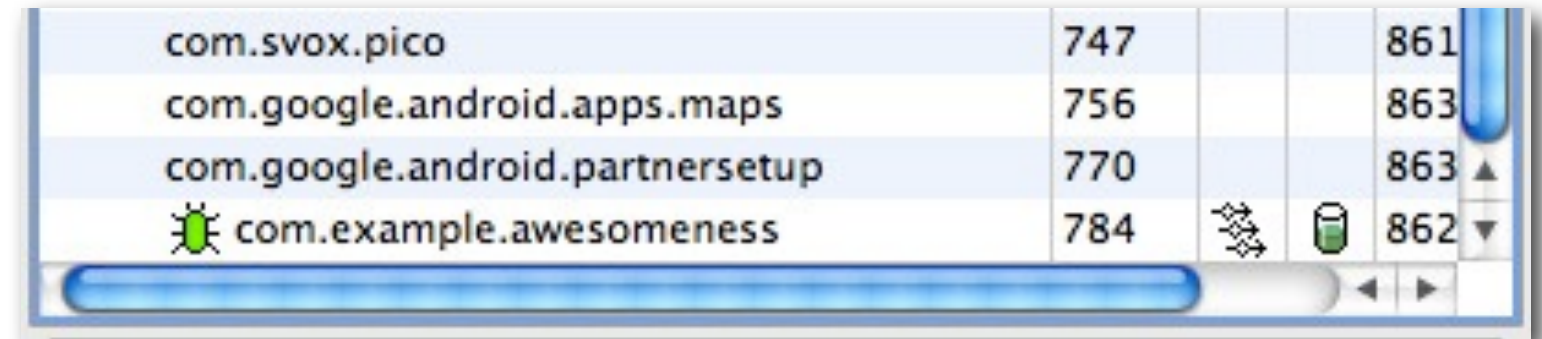




Protecting Android apps from users

Don't let users debug your apps

- **android:debuggable**

- Disabled by default
- Never leave this enabled in release code!
- Allows a user to debug your app - even without source code
- Users with physical access can run code as your app and access your app's data



com.svox.pico	747		861
com.google.android.apps.maps	756		863
com.google.android.partnersetup	770		863
 com.example.awesomeness	784		862

```
jlarimer-macbookair:~ jlarimer$ adb shell
shell@android:/ $ run-as com.example.awesomeness sh
shell@android:/data/data/com.example.awesomeness $ id
uid=10060(app_60) gid=10060(app_60)
shell@android:/data/data/com.example.awesomeness $ ls files/
secret_data.txt
shell@android:/data/data/com.example.awesomeness $ cat files/secret_data.txt
SECRETS!
```



Storing data

Avoid exposing personal or protected data to other apps

- Protect personal data and data that requires a permission to access
 - Use **MODE_PRIVATE** for data files, shared preferences, and databases
 - **openFileOutput()**, **openSharedPreferences()**, and **openOrCreateDatabase()** create files in your app's private data directory
 - External storage (sdcard) is shared storage
 - Don't store personal or protected data on external storage without user consent

```
-rw-rw-rw- app_53 app_53 8 2012-06-18 13:39 secret_data.txt
-rw-rw-rw- app_53 app_53 81544 2012-06-18 21:43 private_info.txt
```

- You can't trust files that other apps can write to
 - Don't store code libraries that are world writable or on external storage
 - Don't store paths to code libraries in files that are world writable or on external storage
 - Don't process data from writable files in native code - memory corruption vulnerabilities could allow apps to run arbitrary code with your app's ID



Protecting data files

There are no good reasons to make your app's private data files world readable

Good:

```
FileOutputStream fos = openFileOutput("private_data.txt", Context.MODE_PRIVATE);  
SharedPreferences prefs = getSharedPreferences("data", Context.MODE_PRIVATE);
```

Bad:

```
FileOutputStream fos = openFileOutput("private_data.txt", Context.MODE_WORLD_WRITEABLE);  
SharedPreferences prefs = getSharedPreferences("data", Context.MODE_WORLD_READABLE);
```



Data encryption doesn't solve all problems

Encryption is not authentication!

Chosen Ciphertext Attack

EncryptedMessage = Encrypt(K, "Login-OK=0")

AlteredMessage = EncryptedMessage ... XOR {..., 0x31}

Plaintext = Decrypt(K, AlteredMessage) = "Login-OK=1"



Use a peer-reviewed library like **keyCzar**

Encryption is not authentication!

```
java -jar KeyczarTool.jar create --location=/path/private.key \  
  --purpose=encrypt --name="My Server Key" --asymmetric=rsa \  
java -jar KeyczarTool.jar pubkey --location=/path/private.key \  
  --destination=app/res/raw/server_pub.key
```

On the host

```
Crypter crypter = new Crypter(new AssetReader(R.raw.server_pub));  
String ciphertext = crypter.encrypt("Secret message");
```

In your app



Leave inventing cryptography to the experts

Although, even experts make mistakes

Rivest, Shamir, and Adleman took 42 tries to discover the RSA algorithm.

The screenshot shows the NIST Computer Security Resource Center website. The main content area is titled "CRYPTOGRAPHIC HASH ALGORITHM COMPETITION". The text on the page includes: "NIST announced a public competition (Federal Register Notice) on Nov. 2, 2007 to develop a new cryptographic hash algorithm for use in digital signatures, message authentication codes, and other applications in the information security community. The winning algorithm will be named 'SHA-3', and will augment the hash algorithms currently specified in the Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard." Below this, it states: "NIST received sixty-four entries by October 31, 2008; and selected fifty-one candidate algorithms to advance to the first round on December 10, 2008, and fourteen to advance to the second round on July 24, 2009. A year was allocated for the public review of the fourteen second-round candidates." Further down, it says: "NIST received significant feedback from the cryptographic community. Based on the public feedback and internal reviews of the second-round candidates, NIST selected five SHA-3 finalists - *BLAKE*, *Groestl*, *JH*, *Keccak*, and *Skein* to advance to the third (and final) round of the competition on December 9, 2010, which ended the second round of the competition."

NIST received sixty-four entries by October 31, 2008

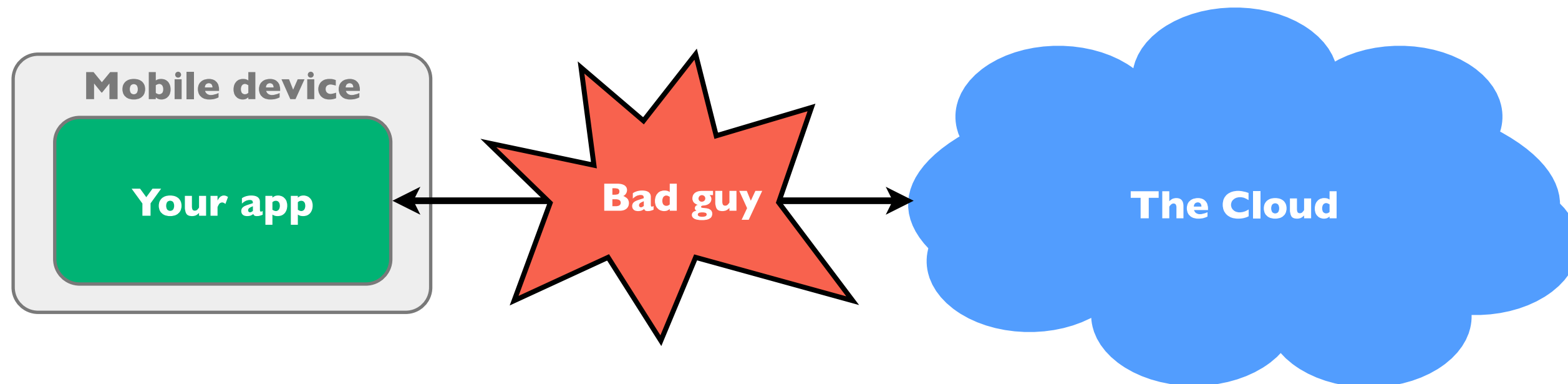
NIST selected five SHA-3 finalists



Protect network traffic

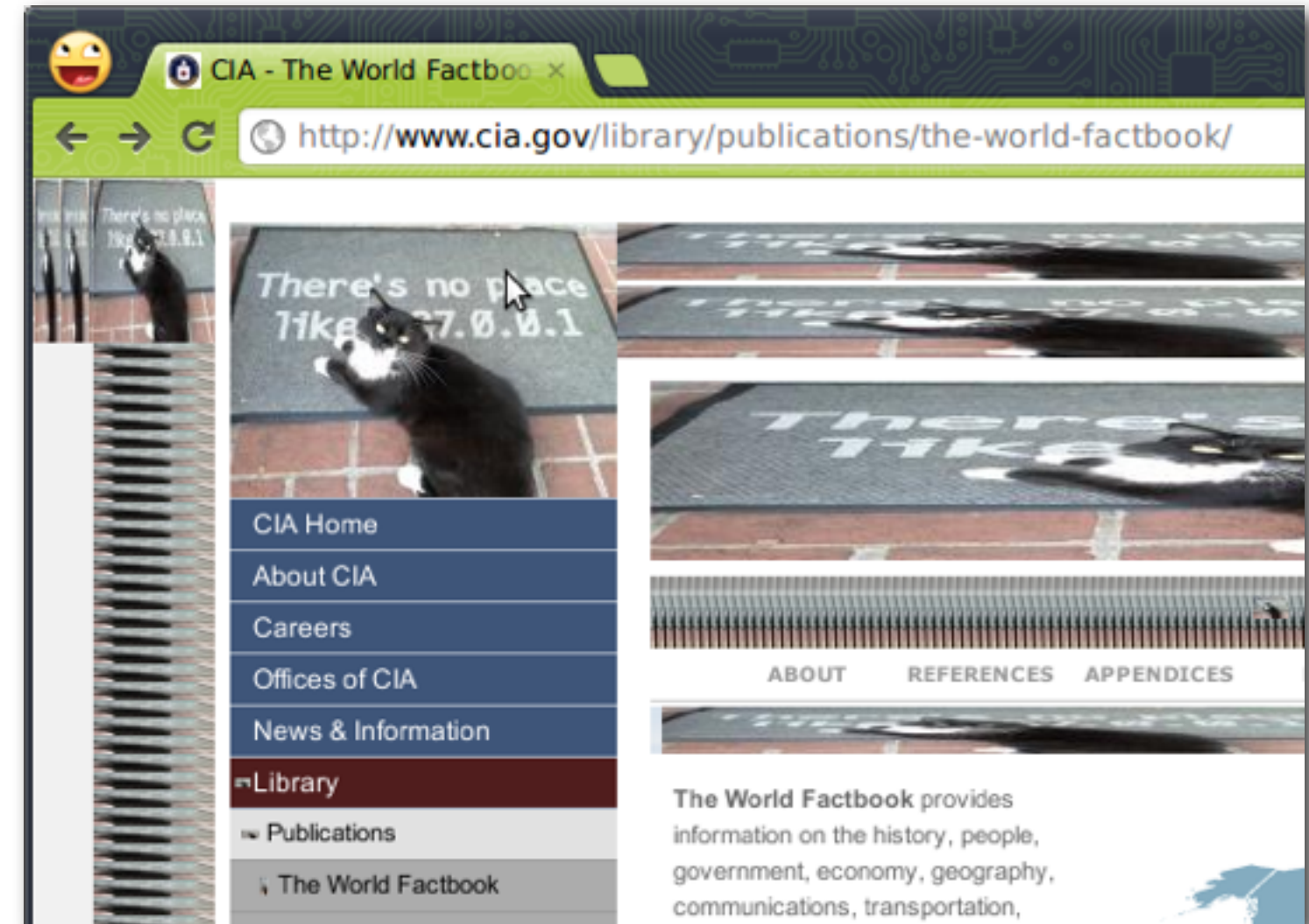
Attackers can eavesdrop on your app's communications

- Assume that there's a bad guy reading all of your app's network traffic
 - Public WiFi networks can't be trusted
 - Rogue cellular base stations can intercept mobile network data traffic
- You can't trust data coming from a server
 - Web servers can be compromised
 - Network traffic can be vulnerable to man-in-the-middle (MitM) attacks that insert malicious data into the network stream



Protecting network traffic

A man-in-the-middle attack can change your network traffic...



Practice safe networking

Encrypt your network requests

- Best practice is to always encrypt network communications
 - HTTPS and SSL can protect against MitM attacks and prevent casual snooping
 - Server certificate validity is checked by default

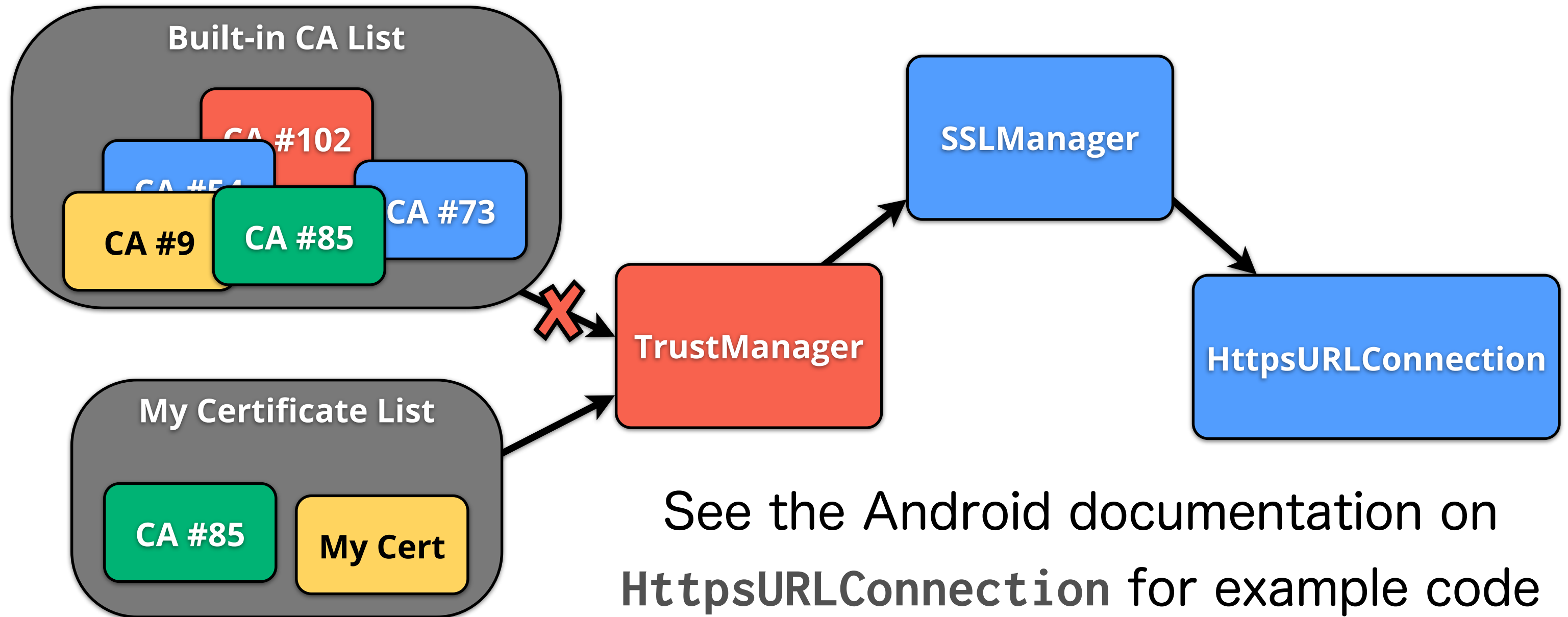
```
URL url = new URL("https://www.google.com/");  
URLConnection urlConnection = (URLConnection) url.openConnection();
```

- Be very careful running code retrieved over the network
 - Use cryptographic signing for any DEX or native code libraries that you load dynamically
 - Better yet, don't run code from the network



Certificate pinning

If you don't completely trust the entire CA ecosystem...



Using WebView

Don't turn web problems into Android problems

- Watch out for cross-site scripting (XSS) and cross-site request forgery (CSRF) vulnerabilities if JavaScript is enabled on your WebView
 - JavaScript is disabled by default
 - If you run a web app in your Android app, you now have all of the security concerns of writing an Android app plus all of the security concerns with running a website
- **addJavascriptInterface()** is dangerous
 - Avoid exposing protected or personal data to a JavaScript interface
 - Server or network could be compromised, you can't trust the code
 - If you do use it, ensure that you're using HTTPS for the WebView



Minimize requested permissions

Users don't like when your app requests too many permissions...

Insane permissions, poorly coded
★★★★★ Mike 6/15/12

I love the concept but I really don't know why this app needs SEND SMS MESSAGES, RECEIVE SMS, READ CONTACT DATA or WRITE CONTACT DATA. It doesn't appear to send SMS or edit my address book or record videos, so there's no need to ask for those permissions. The app looks like

Bad birdie
★★★★★ Ruben 6/14/12

Samsung Galaxy Nexus for an older version

Why are the birds stalking me? Now they want to know my location.

Uninstalled.
★★★★★ Zaib 5/2/12

Samsung Galaxy Note for an older version

Do not understand why it needs GPS permission.?? It's suspicious..

From 5 to 1 star.
★★★★★ Byron 6/14/12

EeePad Transformer TF101 for an older versi...

No way you're tracking my location.

Permissions
★★★★★ Linford 6/14/12

HTC Sensation 4G for an older version

Not having location permissions

Certainly does work...
★★★★★ Christine 5/15/12

Samsung Dart

However, the permissions are disturbing: "take pictures and videos Allows the app to take pictures and videos with the camera. This allows the app at any time to collect images the camera is seeing." And since the program also has network access, it could potentially take pictures of you or those around you and send them back to the programmer or who knows where else. "Your location coarse (network-based) location fine (GPS) location" For a flashlight? Why is this necessary? Last time I checked flashlights were not location-dependent. I'll continue to carry my 4 sevens quark. :P "Read phone state and identity Allows the app to access



Only request the permissions that your app requires

There are ways to access some Android capabilities without requesting permission

- Why minimize the amount of permissions your app requests?
 - One group of researchers found that 1/3 of apps request more permissions than they need
 - Security vulnerabilities can expose protected data
 - Users like apps that request few permissions
- Permissions aren't required if you launch an activity that has the permission
 - Getting a picture from the camera
 - Sending an SMS through the SMS app
- Permissions can be temporarily granted to apps by content providers
 - Letting the user pick a contact to share with your app



Get a camera pic without CAMERA permission

This prompts the user to take the picture, so they're in control of what your app gets

```
// create Intent to take a picture and return control to the calling application
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

// create a file to save the image
fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
// set the image file name
intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

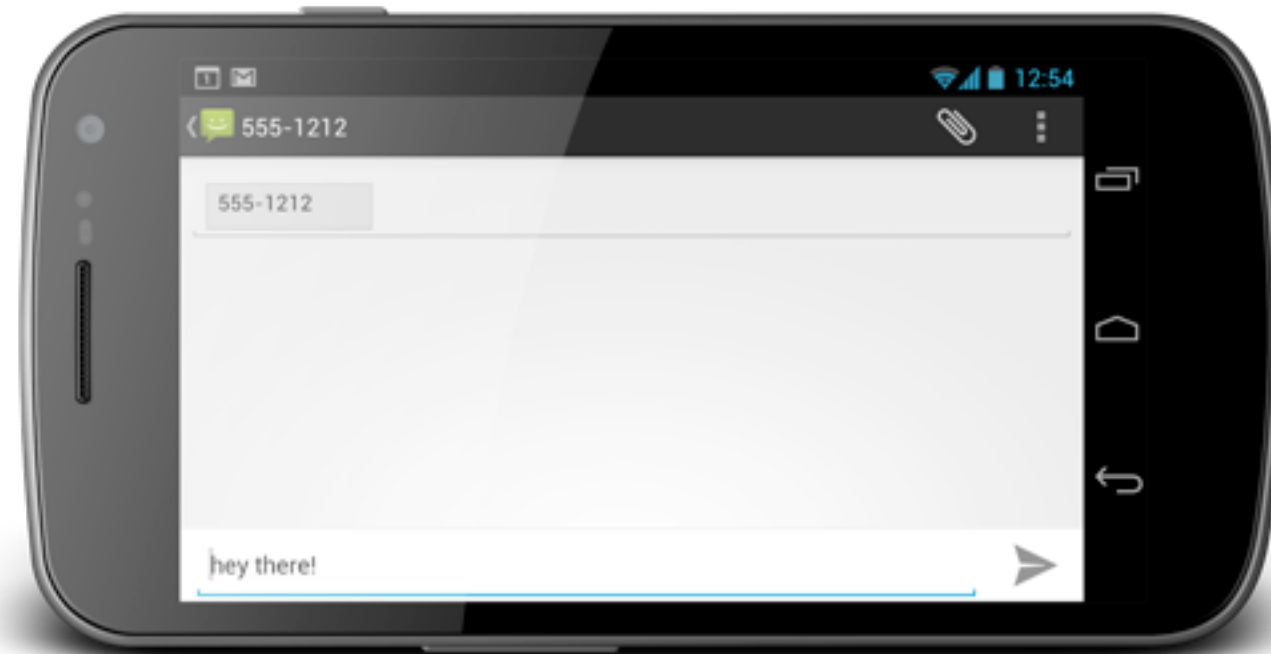
// start the image capture Intent
startActivityForResult(intent, MY_REQUEST_CODE);
```



Start the SMS app with a filled-in destination and message

Doesn't require the SEND_SMS permission

```
Uri smsNumber = Uri.parse("sms:5551212");  
Intent intent = new Intent(Intent.ACTION_VIEW);  
intent.setData(smsNumber);  
intent.putExtra(Intent.EXTRA_TEXT, "hey there!");  
startActivity(intent);
```



Let the user choose a contact with ACTION_GET_CONTENT

Retrieve the selected contact data without requesting READ_CONTACTS

```
Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
intent.setType(Phone.CONTENT_ITEM_TYPE);
startActivityForResult(intent, MY_REQUEST_CODE);
```

```
void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (data != null) {
        Uri uri = data.getData();
        if (uri != null) {
            try {
                Cursor c = getContentResolver().query(uri, new String[] {
                    Contacts.DISPLAY_NAME, Phone.NUMBER}, null, null, null);
            }
        }
    }
}
```



More minimizing requested permissions

More ways to reduce requested permissions

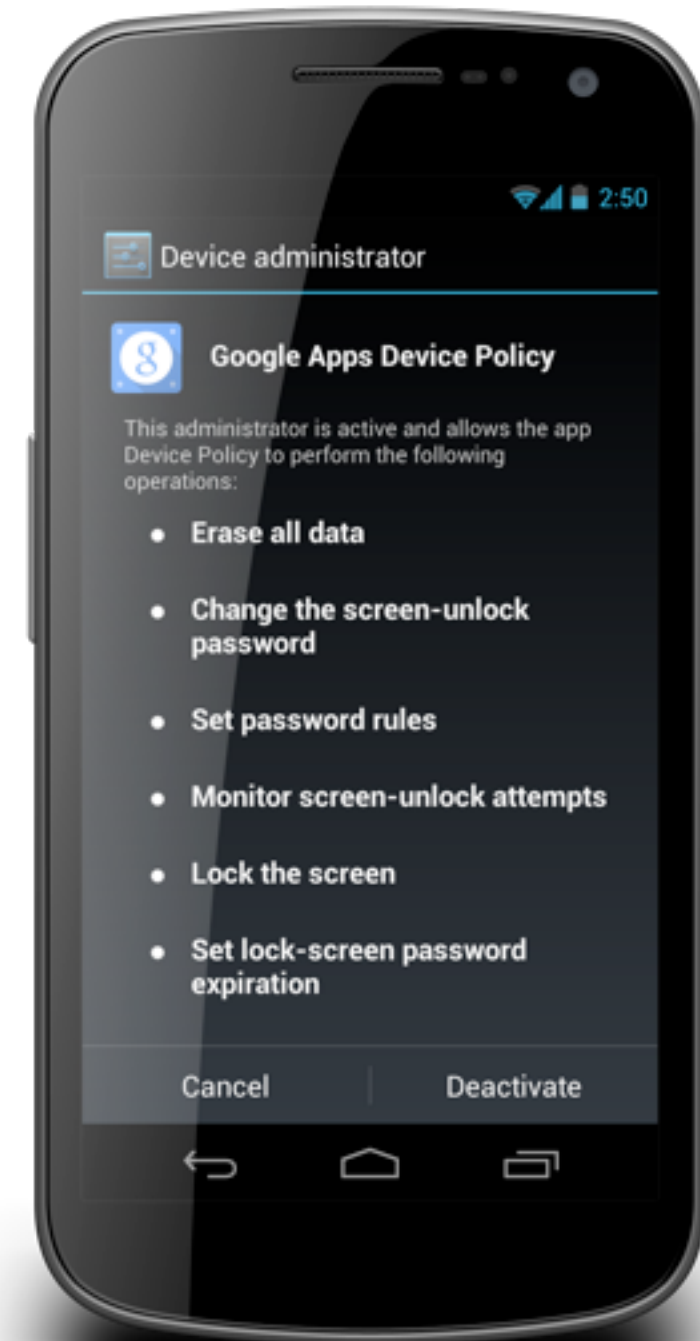
- Need a unique identifier?
 - **TelephonyManager.getDeviceId()** requires **READ_PHONE_STATE** permission
 - Hardware IDs are a poor choice for identity anyway - see <http://android-developers.blogspot.com/2011/03/identifying-app-installations.html>
 - **Settings.Secure.ANDROID_ID** doesn't require a permission, but still not perfect
- To identify an installation of your app
 - Generate a UUID when your app starts and store it in shared preferences:
 - **String id = UUID.randomUUID().toString();**
 - Use Android Backup Service to save the shared preferences to the cloud
 - See: <https://developers.google.com/android/backup/>



Device Administration access

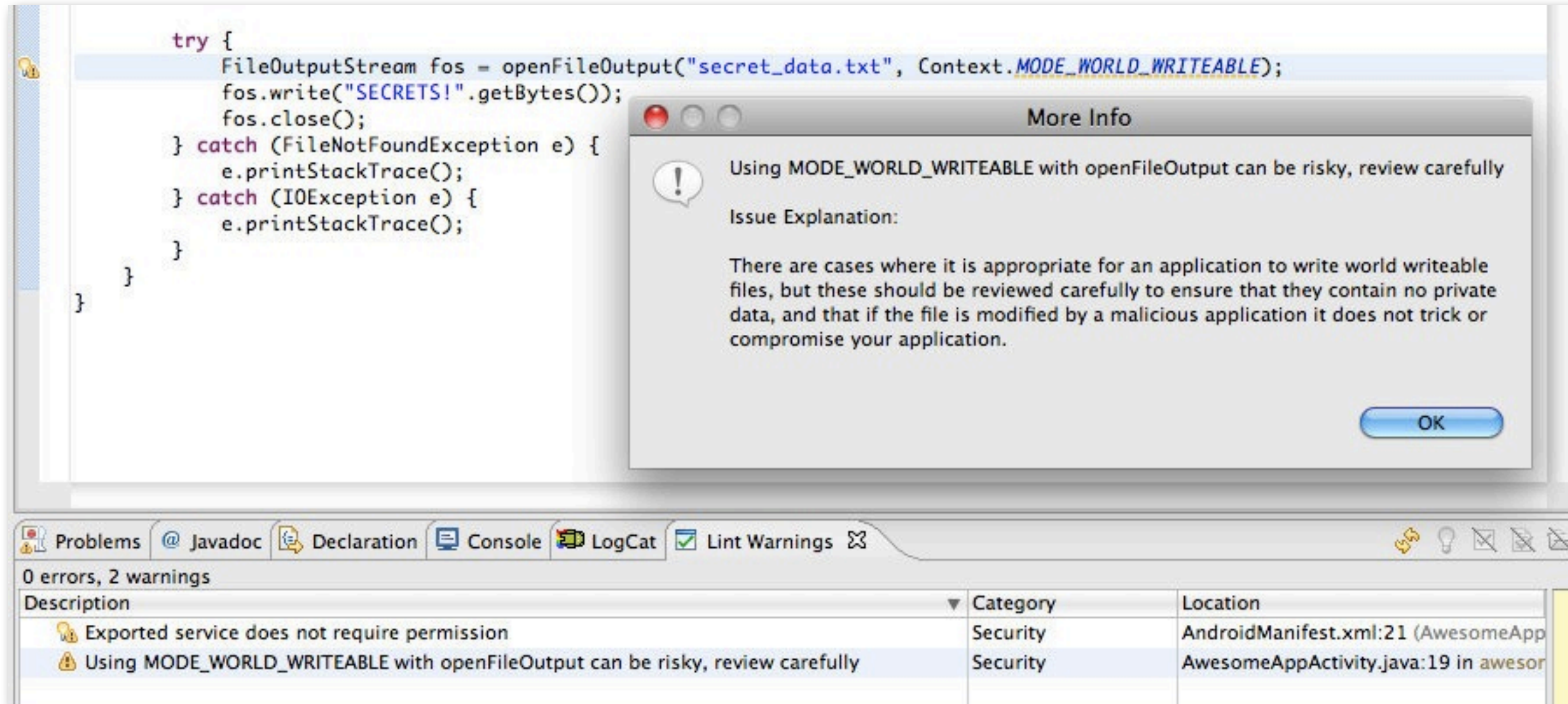
Designed for enterprise mobile device management (MDM) apps

- Device Administration API provides a lot of power, can be dangerous in the wrong hands
- Changing device security settings can have a serious impact on overall security
- Spend extra time auditing if your app can act as device administrator - you really don't want to leak these permissions!



Use Android Lint

Lint comes with the Android SDK and detects common programming errors



The screenshot shows an IDE window with a Java code snippet. A yellow warning icon is visible in the left margin. A 'More Info' dialog box is open, displaying the following text:

More Info

Using MODE_WORLD_WRITEABLE with openFileOutput can be risky, review carefully

Issue Explanation:

There are cases where it is appropriate for an application to write world writeable files, but these should be reviewed carefully to ensure that they contain no private data, and that if the file is modified by a malicious application it does not trick or compromise your application.

OK

The IDE's bottom toolbar includes 'Problems', 'Javadoc', 'Declaration', 'Console', 'LogCat', and 'Lint Warnings'. The 'Problems' tab is active, showing a table with the following content:

Description	Category	Location
Exported service does not require permission	Security	AndroidManifest.xml:21 (AwesomeApp
Using MODE_WORLD_WRITEABLE with openFileOutput can be risky, review carefully	Security	AwesomeAppActivity.java:19 in awesor



Developer documentation on security

See these sites for more information on what we talked about today



- **Android Security Overview:** <http://source.android.com/tech/security/index.html>
 - Describes how various security features are implemented in Android
- **Designing for Security:** <http://developer.android.com/guide/practices/security.html>
 - Teaches you how to write apps with security in mind
- **Security and Permissions:** <http://developer.android.com/guide/topics/security/permissions.html>
 - SDK documentation on the Android permission system
- **Application Security for the Android Platform: Processes, Permissions, and Other Safeguards,** Jeff Six, O'Reilly Media



<Thank You!>



Ask questions about writing secure apps: groups.google.com/group/android-security-discuss
Contact the Android security team: security@android.com

+Jon Larimer

jarimer@google.com

+Kenny Root

kroot@google.com



Google
Developers