Hacking Androids for ~~fun and~~ Profit

Riley Hassell, Founder
Privateer Labs

HITB KUALA LUMPUR 2011

- The App market is like the wild, wild, west. Open, accessible, unrestricted.
- No need to coerce a user to download your app and install it from a remote website.
- Permission based security model is new and puts the average consumer in charge of the critical security decision making process.

Why are we so interested in Android?

- Apps are not adequately reviewed before being placed on the market for public consumption.

Android Marketplace

(The biggest W@r3Z site in the world)

(Besides third party markets…)

Permission Model

- Users are prompted with a permission list that is at best vaguely described, even in SDK documentation.

"READ_PHONE_STATE - Allows read only access to phone state."

Might be better to say: "…is a permission that grants the application to read your unique cell phone serial, phone number, SIM card serial number, and much more!"

Impersonation

- Apps vendors are not validated.
- Malicious developers can publish apps that masquerade as legitimate products.
- Jon Oberheide provided an entertaining example. (RootStrap - Twilight)

Risks to Android Users

- Malware
- Autorun
- WiFi
- Phishing
- Rootkits
- Botnet Node
- Network Traversal
- Jailbreaking

- Don’t believe everything you read. In the press. It’s not that bad. At least for the Android Market…
- Android malware is advancing in sophistication much faster that on previous computing platforms.
- Introducing “Trend Trojans”.

Malware

- Things to look for when selecting apps for your mobile device:
  - Has the app been on the market for more than 90 days?
  - Does the app have decent ratings?
  - Developers a well known and respected?
  - What permissions is the app asking for?

Malware - Protecting yourself

Autorun

- Apps run without being “Clicked”.
- Apps can be invoked from automated system events.
- Since security apps typically scan post install due to framework limitations this leaves a window open for attackers to exploit.

WiFi Hazards

- Many apps do not encrypt your data beforing rifling them to backend servers.
- Most public access WiFi AP(s) are not encrypted.
- Even the phone is not in use many apps auto-sync in the background.
- Hackers can hijack your app accounts!
  - See FaceSniff

- Uncheck connect when within range features.

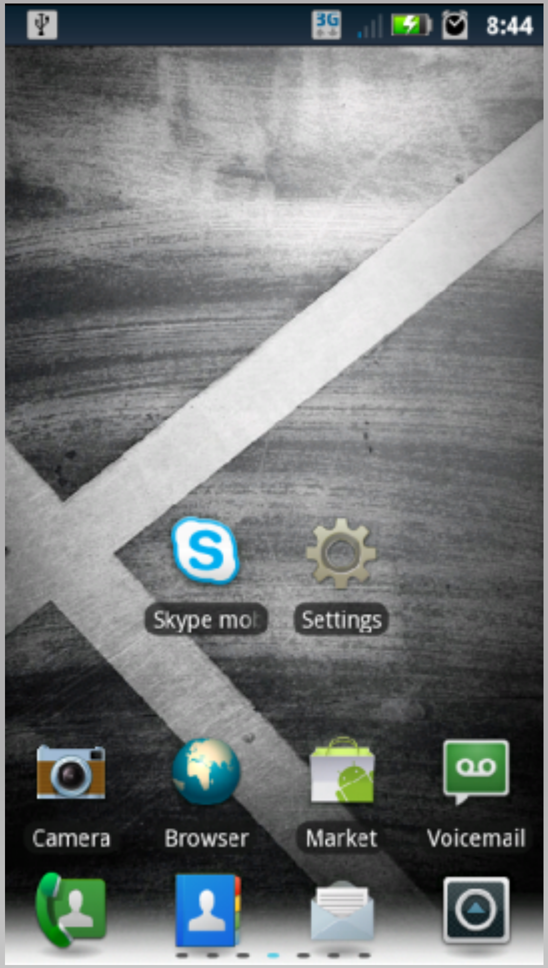
- Disable any other auto connect functionality.

WiFi Hazards – Protecting Yourself

- Rogue apps can masquerade as legitimate apps you trust.
- App waits for activity (UI Element) of interest to spawn.
- Phishing app will then overlay it's own interface, tricking the user into entering sensitive information into the phishing app.

Think "clickjacking" for Droid

App Phishing



App Phishing - Demo

- Exercise caution when granting apps these permissions:
  - *READ_LOGS*
  - *GET_TASKS*

- Phish apps will usually be unable to populate the fake login screen with *saved* credentials.

App Phising – Protecting Yourself

Rootkits

- Proof-of-Concepts have been around for a while, see:
  - DEFCON 18 Spiderlabs Android Rootkit
- None currently reported in the markets. At the moment we've only seen them coupled with 0day for targeted attacks.
- Apps can utilize jail break exploits to gain root privileges and install them.

- No recommendations at this time.
- Rootkit on your mobile == you SOL

Rookits –Protecting Yourself

- A few proof of concepts frameworks have circulated in the last several months.
- Imagine an army of mobile phones configured to listen to background noise, translate to text and target keywords, perform voiceprints, all why tracking an individuals every move with a live video feed.

Enter Mobile Echelon.

Botnet

- Usually deployed from malware.
- Exercise caution when installing apps (as discussed in prior malware section).

Botnet – Protecting Yourself

- Interesting attack variation supplied by mobile computing platforms.
- Compromised mobiles can be used to attack each network that the mobile gains access to.

Traversal Physical Boundaries

- Disable “connect when within range” features
- Exercise caution when installing apps.
- Consider installing a firewall app. Hackers fail to plan for security products.
- Checkout Anti app from Itz. Metasploit for Android ;)

Traversal Physical Boundaries – Protecting Yourself

- Su apps default to implicitly allow current and future process attempts for root escalation.
- Majority of jailbreak users trust shell (bin/sh).
- Malicious apps can simply invoke the shell from their app and “su” to root without prompting user.

Jailbreaking

- Don’t jailbreak your phone until a better escalation solution is available.
- When asked to approval an app for escalation uncheck the “remember” checkbox.

Jailbreaking – Protecting yourself

Hacking Android

- **DEX2JAR** – Convert compiled DEX object code to a JAR that can be decompiled with JAD.

- **APKTOOL** – Disassembler and binary xml translator built in. Produces Jasmin like syntax that can be reviewed by your favorite editor. Also supports apk rebuilding.

- **DED** (http://siis.cse.psu.edu/ded/) – Decompiler for Android DEX that while requires a little more setup but provides much more reliable results than other decompilers.

Your Toolkit

Your Toolkit - cont

- **Source Insight** – Industry favorite code analyzer. You can create custom SMALI/ JASMIN parsers to visually render your code as your desire.
- **010 Editor** – Fantastic hex editor. Also supports templates.
- **IDA** – The only tool for examining machine code. Cough up the cash, you need it ;)

- **Ubuntu 64bit Install**
  - You'll need this to build your own source so you can hack with symbols.
- **Android Prebuilt binaries**
  - gdbserver
  - tcpdump
  - strace
  - Busybox
  - bash
  - valgrind

Your Toolkit - cont

- Android Permissions
- Activity Reuse
- SQL Injection
- XML Injection
- Package Name Trust
- Traversing Webviews
- Info Leaks

Things to look for…

- Requested permissions offer us a valuable first stab at an attack surface area assessment, e.g.:
  - READ_LOGS – What happens when malicious log entries are injected into the system logs?
  - INTERNET – MITM/Leak Potential
  - RECEIVE_SMS ← Can they app be exploited with a text message?

Android Permissions

Norton Security 2.2.0.305

- Feature allows “buddy” to remotely lock, locate, and wipe your phone in case of theft. Requires origin phone number and password.

- SMS Message Syntax: cmd password, e.g.
  - “lock SecretPassword”
  - “locate SecretPassword”
  - …

- SMS origin is easily spoofed (if buddy system worked as intended).

- Buddy verification is broken, anyone can issue remote commands.
- No password strength guideline and phone.
- Limit for failed SMS authorization failures is not in place.
- User is not warned of failed attempts.

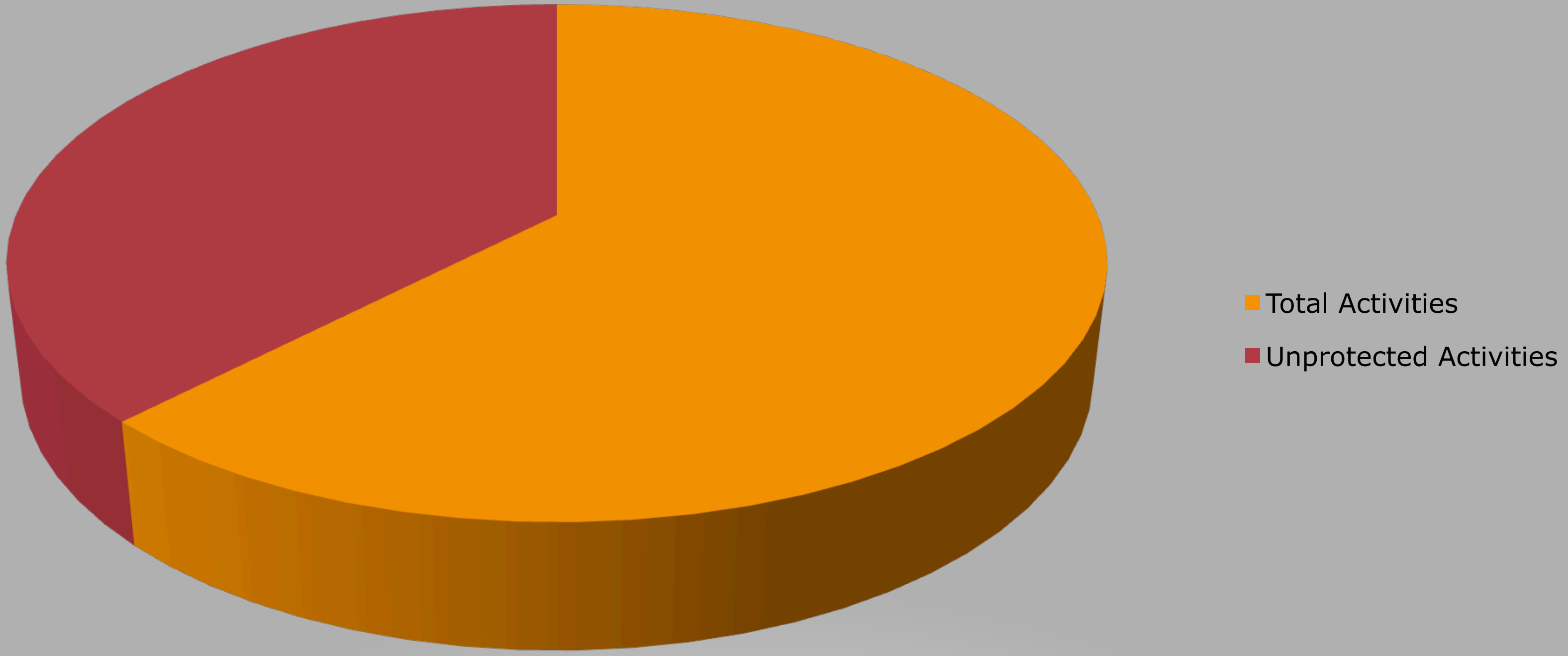
Norton Security - Cont

Activity Reuse

- Exported app Activities can be invoked by external app:
  - Activity exported by declaring the “android:export” attribute on the Activity
  - Activity applied an Intent filter (“intent-filter”).
  - Activities that do not utilize either of these are traditionally considered private and are not accessible.

Privateer Labs performed a review of 618 apps that contained a total of 3592 Activities.

Exported Activities Requiring Permissions

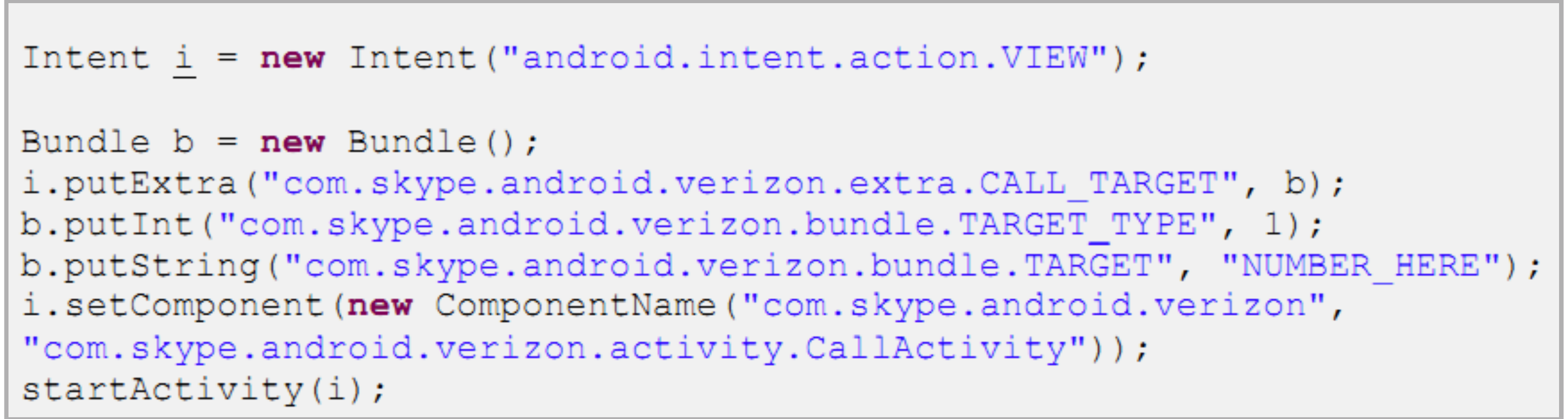


This yielded 2176 Activities that do not enforce permissions and are publicly accessible.

Activities Survey

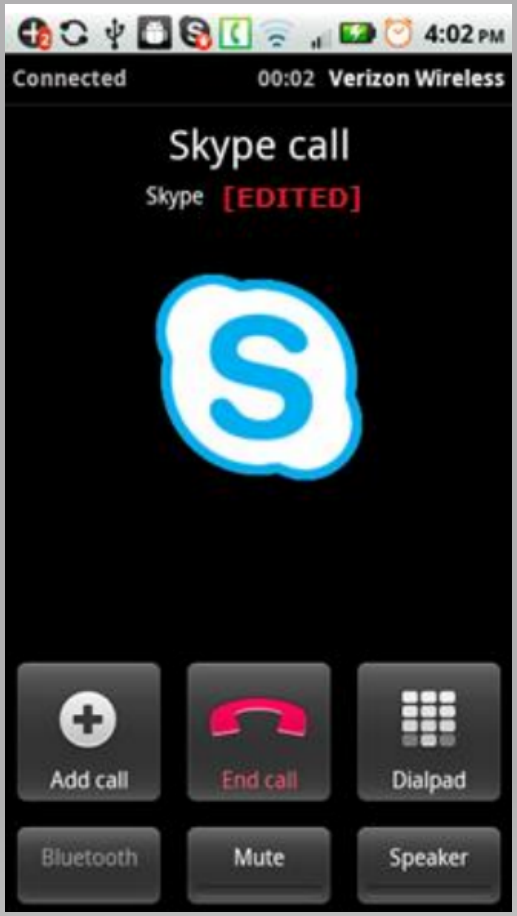
Activity Reuse in Skype

- Reported earlier this year.



```
Intent i = new Intent("android.intent.action.VIEW");

Bundle b = new Bundle();
i.putExtra("com.skype.android.verizon.extra.CALL_TARGET", b);
b.putInt("com.skype.android.verizon.bundle.TARGET_TYPE", 1);
b.putString("com.skype.android.verizon.bundle.TARGET", "NUMBER_HERE");
i.setComponent(new ComponentName("com.skype.android.verizon",
"com.skype.android.verizon.activity.CallActivity"));
startActivity(i);
```



We can make phone calls without the needed permission!

SQL Injection

- Android developers are recommended to use the parameterized query options to mitigate the risk of SQL Injection.
- …Although many developers build string queries via the execSQL() method.

XML Injection

- Preferred by new developers (vs SQLite)
- App developers rarely sanitize XML input.
- Began researching potential for XML injection when I found an example in one of my apps… We all make mistakes ;)

- Input sources typically user supplied and therefor should not be trusted.

- Android SharedPrefs properly encode problem characters ☺

XML Injection - Cont

- Test values were pushed to app.
- App was installed onto the Android phone.
- Then pulled and examined to verify the lack of secondary encoding on special values

Validation

XML Injection

Multiple key fields in the app manifest do not filter special characters:

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0 OR \'); | >) ;\'s"
package="com.privateer.vs"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8" />
    <application android:label="1.0 OR \'); |> ) ;\'s" android:icon="@drawable/icon"
android:debuggable="true">
        <activity android:label="@string/app_name" android:name=".VerizonSyncActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- Reported to Android Security team.
- Since reporting a new SDK has been published that does not allow characters used in XML injection, e.g. '>' to be supplied in ApplicationManifest fields.
- Attackers can still add these fields using other means… ☹

XML Validation – Android SDK

- Two package names cannot exist on the market at the same time.
- Don’t assume that package names can be trusted.
- Packages are sometimes deployed by vendor and not placed on the market.
- Packages name may be available on another (third party) market.

Package Name Trust

- HeroLED 😐
  - https://market.android.com/details?id=com.mclaughlin.HeroLED&rdid=com.mclaughlin.HeroLED&rdot=1&pli=1

- Advanced Task Killer has a feature to ignore “trusted” packages when displaying the task list to the user.

Package Name Trust

- If we were evil we would have published names of every possible app we could think off so we could "squat" them.

- Android package squatting… to be continued.

Package Name Trust

- Rich content apps relying on web views.
- Separate store than the browser.
- Prevents browser based XSS, CSRF, etc…
- These remote app web views can be accessed.

Traversing Webviews

Traversing Webviews

- BROWSABLE

```
<activity android:name="com.target.app.schemehandler">
     <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="httpx" />
    </intent-filter>
</activity>
```

- In this scenario “schemehandler” is an activity that receives the browse intent and acts on it. Often this is simply a Web View request containing data supplied by the user.

E.g.:

httpx://user?add=<script here>

Traversing Webviews - Cont

- Apps frequently fail to encrypt sensitive network communications.

- Setup MITM so you can review the network data delivery of your target apps.

- Worked for me:
  - Android -> Ubuntu 10.x PPTPD -> iptables port redirects -> Burp Proxy

Info Leaks

- Android Browser
  - Codecs/Plugins are compiled with NDK ;)
  - Lots of bugs here so far... instant code execution on the phone if exploited (root with jailbreak payload)

Browser Attack Surface

- Examine your target app’s code for calls to isLoggeable(). Grab the tag name supplied and set the loglevel to enable verbose logging.

E.g. Enabling web debugging:

```
setprop log.tag.HttpOperation VERBOSE
setprop log.tag.httpclient.wire.header VERBOSE
setprop log.tag.httpclient.wire.header VERBOSE
setprop log.tag.httpclient.wire.content VERBOSE
setprop log.tag.httpclient.wire.content VERBOSE
…
```

Tricks of the Trade

- Decompile an app.
- Insert your own classes to exposed extra debugging information, auto-validate all certs, etc…
- Very important when auditing apps.

Instrumentation

Android OS Vulnerabilities

Liblog

- Very Buggy…
- Log devices are world writeable (/dev/log/*).
- Arbitrary log writing possible.
- Logcat uses liblog.

Liblog - Cont

- Logcat instances can be exploited to disable log monitoring functionality in many apps.
- Code execution may be possible due to nature of vulnerabilities (heap corruption).
- Proof of Concept to be released following HITB.
- Possibility exists of exploiting the logging vulnerabilities remotely due to nature of vulnerabilities.
- Similar bugs found in library previously.

Logcat - Tombstones

- Android developer friendly version of a core dump.
- Located in /data/tombstones

```
Build fingerprint: 'verizon/
shadow_vzw/cdma_shadow:
2.3.3/4.5.1_57_DX5-3/110323:user/
release-keys'pid: 6367, tid: 6367
>>> ./logcat <<<signal 11 (SIGSEGV),
code 1 (SEGV_MAPERR), fault addr
deadbaad r0 00000027  r1 deadbaad  r2
00000000  r3 00000000 r4 00000000  r5
…
```

Logcat - Tombstone

- SQLQueryBuilder uses string concatenation internally to build queries. ☹

- Sanitize input before passing into WHERE and ORDERBY clauses of query() or managedQuery() as they are built by query builder.

SQL Injection in Framework

Mobile Vendor Vulnerabilities

- Justin Case and Travis Eckhart recently disclosed that demonstrate HTC propagates sensitive data into it’s own store that is accessible by hackers.

HTC

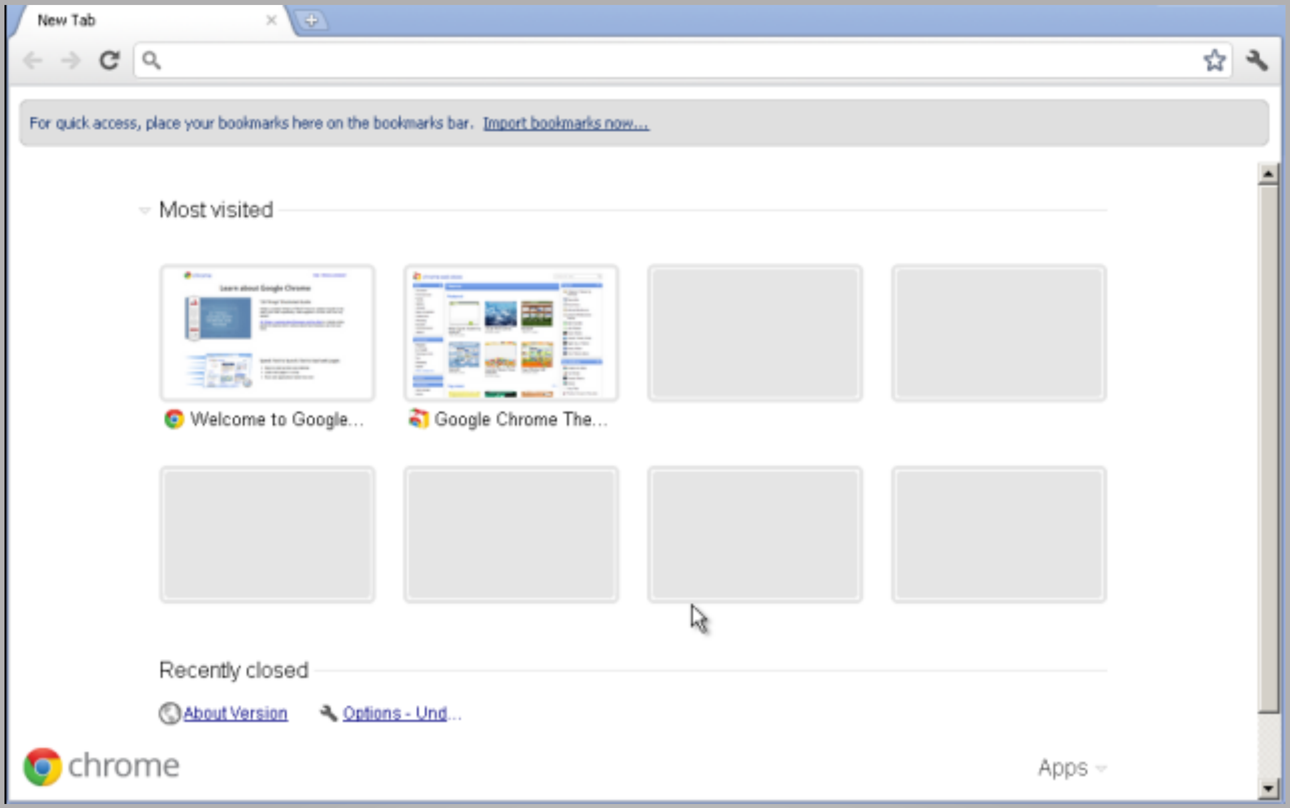
- Motorola Blur exposes OAUTH tokens during update checks!

Motorolla!

- Run a network capture on your Android and see what else vendors sending ;)

Many apps leak sensitive information…

Remote Application Install (RAI)



Google Account Linking

- Luckily there has never been a cross-scripting in any google services… ;)

Google Account Linking

- Google is pretty good about hardening their services.

- Cookies are usually set HTTP ONLY and SECURE (not accessible through script or exposed over HTTP)

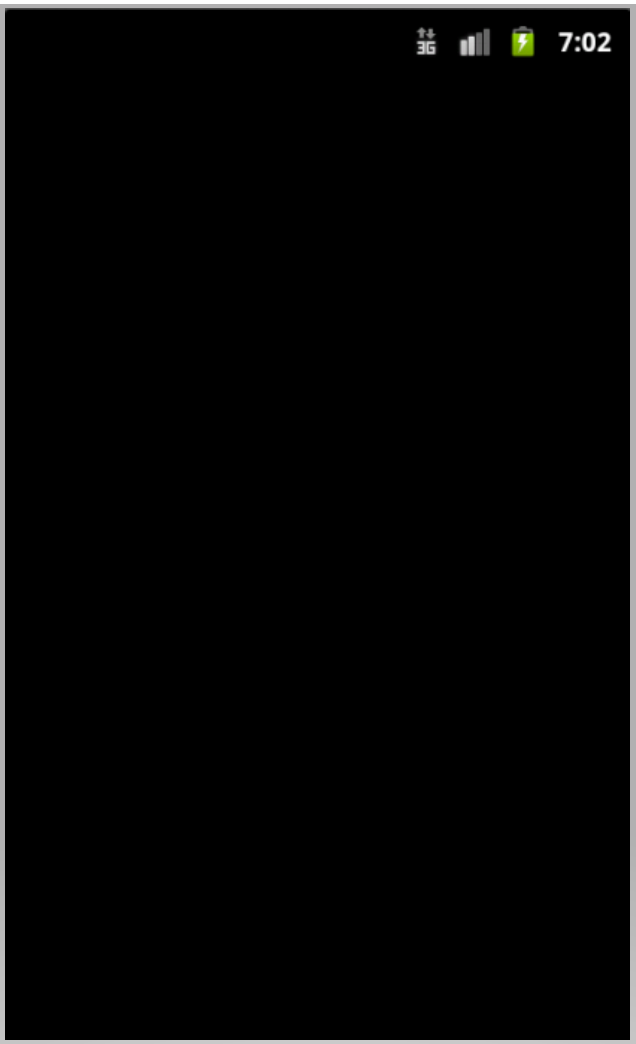
I know what your're thinking…

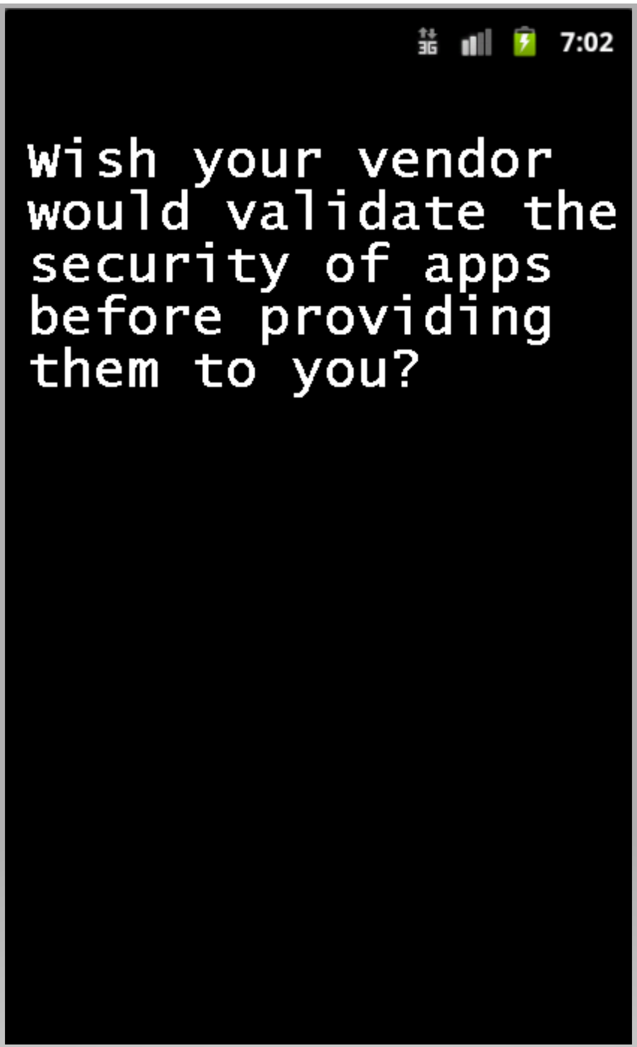
- [Insert new certicom bug here]
- You could be one null byte away from having your Android phone rootkited ;)

Certificate Validation

- Thanks to certificate validation we don’t have to worry about MITM.
- Along the way to KUL…

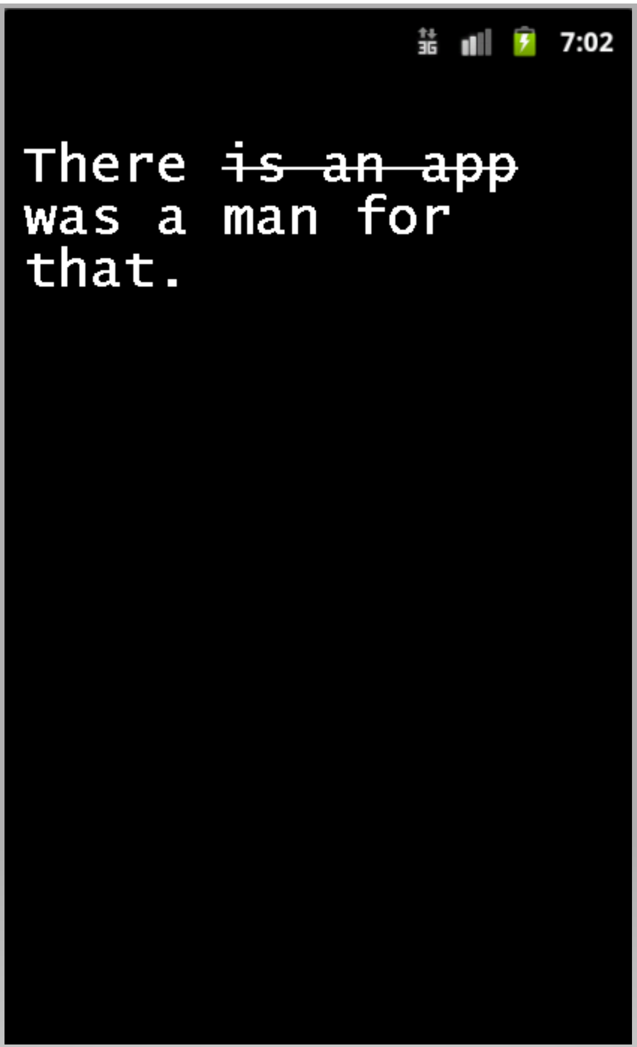
MITM


3G
7:02

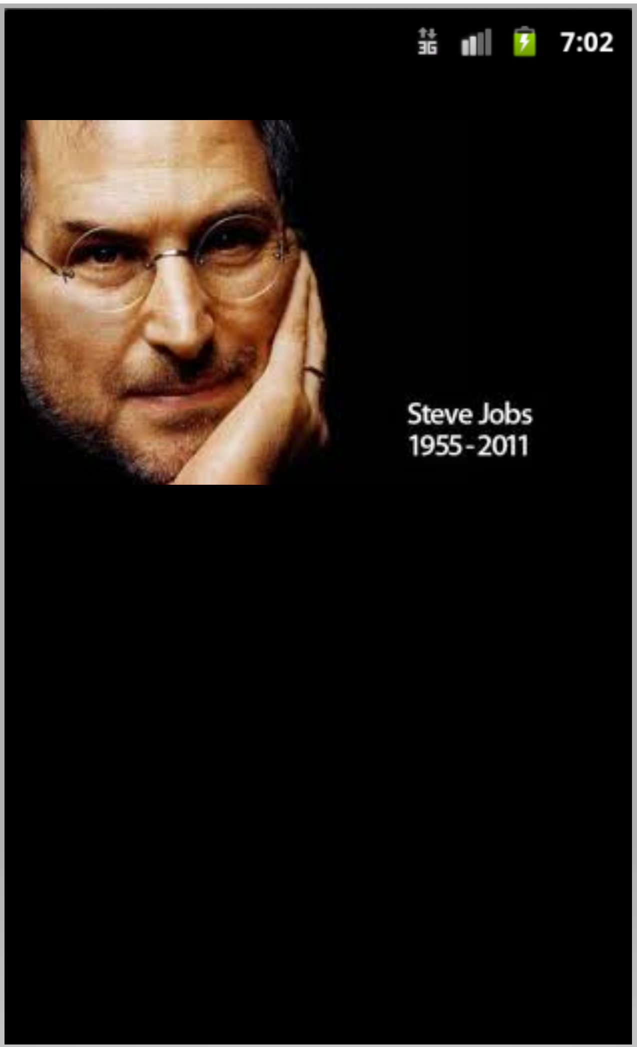


Wish your vendor would validate the security of apps before providing them to you?

Meanwhile in gate (??)...


7:02

There ~~is an app~~ was a man for that.


7:02
Steve Jobs
1955-2011

- Exercise caution when installing apps.
- Avoid free WiFi use on your mobile until the privacy leaks are plugged.
- Consider installing a mobile security app. There are many great security apps that offer decent protection for free.

Conclusion

Conclusion

- App developers typically do not have the budget to hire professionals to perform security audits of their apps.
- Marketplace operators do not currently perform vulnerability scans of apps.

Questions & Comments:

research@privateerlabs.net