lookout™
MOBILE SECURITY

Dex Education: Practicing Safe Dex

Tim Strazzere - Blackhat USA 2012

whoami

Senior Security Engineer
@ Lookout Mobile Security

- Reversed the Android Market/Google Play
- Always enjoyed reversing "exotic" platforms, writing tools to automate the mundane tasks
- Junkie for reversing mobile malware, creating write ups, teaching others all this fun stuff!
- the "different" tim at work, so I go by "diff"

Agenda

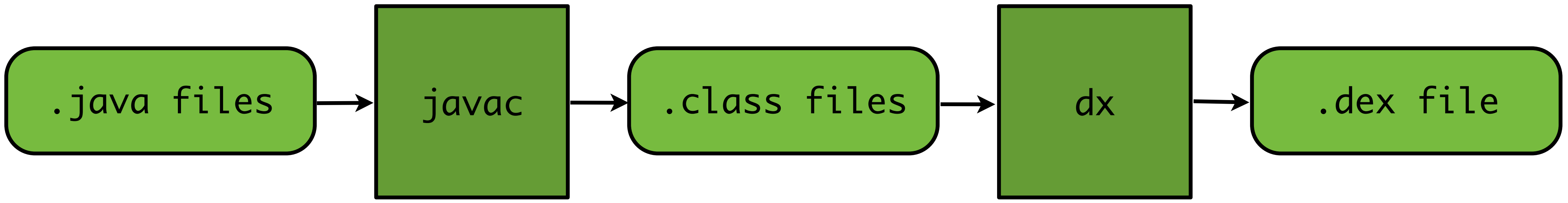
- Dex Education: Let's talk about Dex and what it's all about
- How are attackers hiding/breaking things currently?
- Hiding from, and breaking the tools - Anti-Analysis
- Breaking the toolbox - Anti-emulator/Anti-VM
- PoC Tool and Lessons learned!

lookout™ MOBILE SECURITY

Dex Education

What is this?

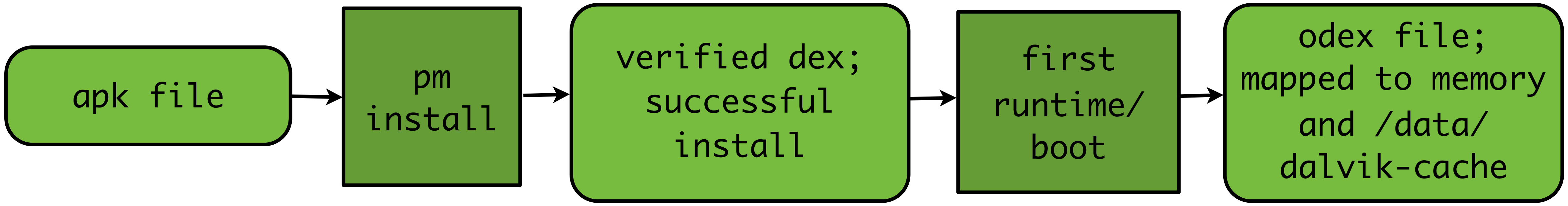
- DEX stands for **D**alivk **EX**ecutable
- Bundled class files which run inside the Dalvik VM
- Packaged inside of the APK file (essentially a jar/zip)



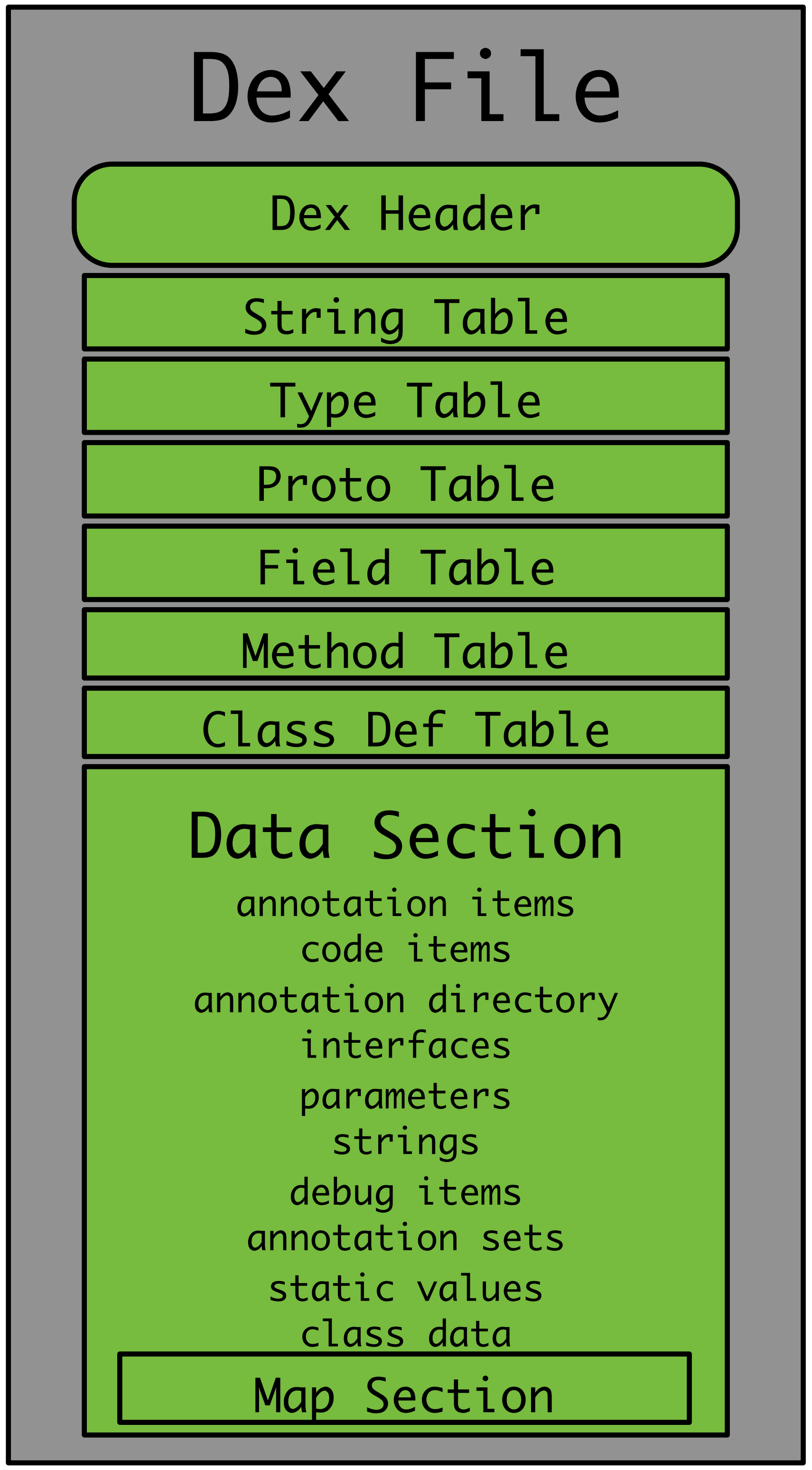
Dex Education

What is this? continued

- Upon install, checked dex file extracted and verified for integrity
- Upon first runtime/boot dex optimized, converted to "odex" (optimized dex)
- odex dropped to /data/dalvik-cache and loaded into memory on execution


apk file
pm install
verified dex; successful install
first runtime/ boot
odex file; mapped to memory and /data/ dalvik-cache

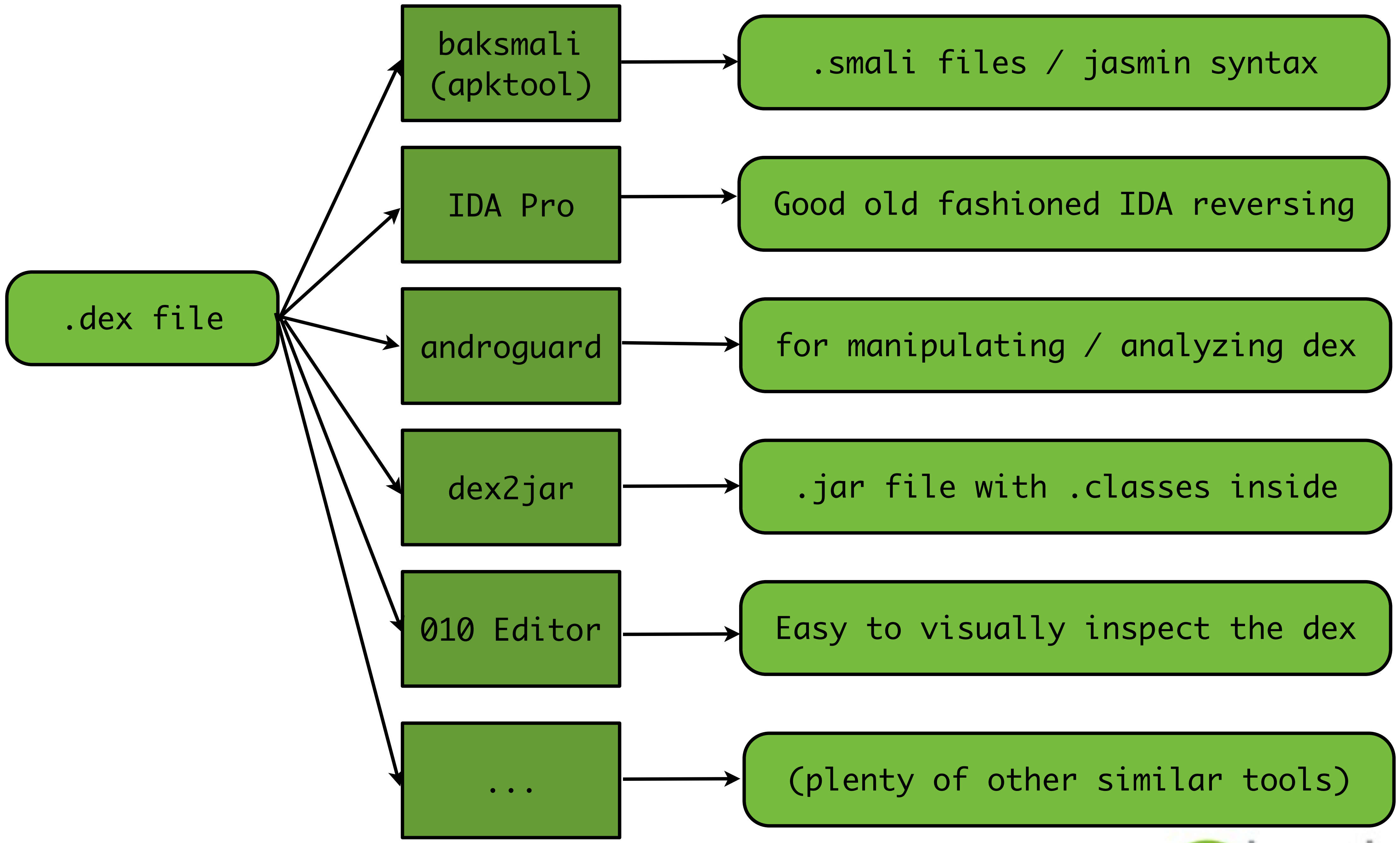
Dex Education: Dex File Format



- Header contains offsets/sizes to all sections
- Tables contain struct data, references to each other and offsets into data
- Data contains the meat of the file

Dex Education

How can we examine dex files?



.dex file
baksmali (apktool)
.smali files / jasmin syntax
IDA Pro
Good old fashioned IDA reversing
androguard
for manipulating / analyzing dex
dex2jar
.jar file with .classes inside
010 Editor
Easy to visually inspect the dex
...
(plenty of other similar tools)

lookout MOBILE SECURITY

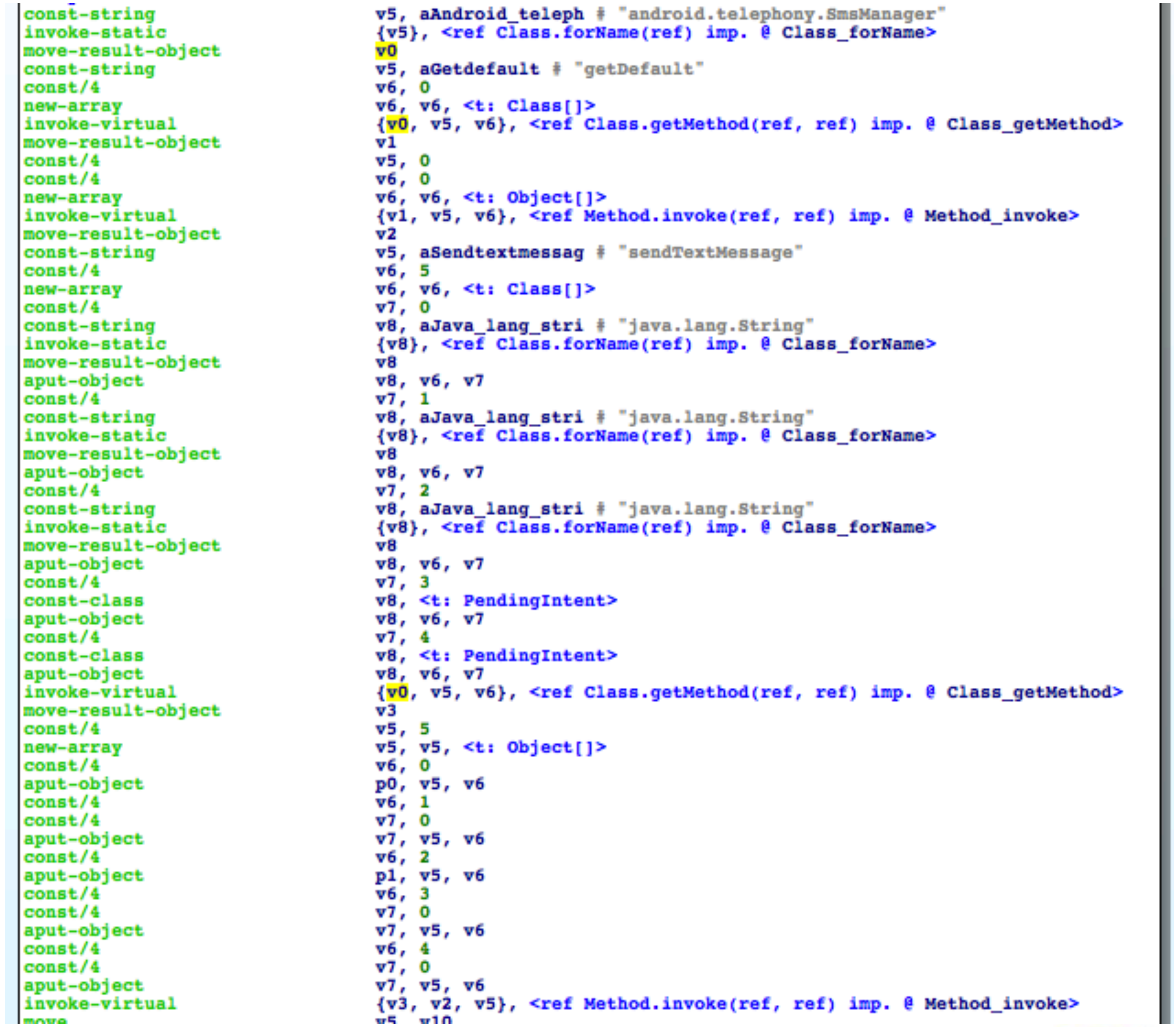
How are attackers hiding currently?

Simple methods employed

- Some use simple reflection to call “sensitive” function
- Keep dex file “clean” and load data from assets
- Find valid dex nuances which break some tools

lookout™ MOBILE SECURITY

Reflection to hide calls



```
const-string              v5, aAndroid_teleph # "android.telephony.SmsManager"
invoke-static             {v5}, <ref Class.forName(ref) imp. @ Class_forName>
move-result-object        v0
const-string              v5, aGetdefault # "getDefault"
const/4                   v6, 0
new-array                 v6, v6, <t: Class[]>
invoke-virtual            {v0, v5, v6}, <ref Class.getMethod(ref, ref) imp. @ Class_getMethod>
move-result-object        v1
const/4                   v5, 0
const/4                   v6, 0
new-array                 v6, v6, <t: Object[]>
invoke-virtual            {v1, v5, v6}, <ref Method.invoke(ref, ref) imp. @ Method_invoke>
move-result-object        v2
const-string              v5, aSendtextmessag # "sendTextMessage"
const/4                   v6, 5
new-array                 v6, v6, <t: Class[]>
const/4                   v7, 0
const-string              v8, aJava_lang_stri # "java.lang.String"
invoke-static             {v8}, <ref Class.forName(ref) imp. @ Class_forName>
move-result-object        v8
aput-object               v8, v6, v7
const/4                   v7, 1
const-string              v8, aJava_lang_stri # "java.lang.String"
invoke-static             {v8}, <ref Class.forName(ref) imp. @ Class_forName>
move-result-object        v8
aput-object               v8, v6, v7
const/4                   v7, 2
const-string              v8, aJava_lang_stri # "java.lang.String"
invoke-static             {v8}, <ref Class.forName(ref) imp. @ Class_forName>
move-result-object        v8
aput-object               v8, v6, v7
const/4                   v7, 3
const-class               v8, <t: PendingIntent>
aput-object               v8, v6, v7
const/4                   v7, 4
const-class               v8, <t: PendingIntent>
aput-object               v8, v6, v7
invoke-virtual            {v0, v5, v6}, <ref Class.getMethod(ref, ref) imp. @ Class_getMethod>
move-result-object        v3
const/4                   v5, 5
new-array                 v5, v5, <t: Object[]>
const/4                   v6, 0
aput-object               p0, v5, v6
const/4                   v6, 1
const/4                   v7, 0
aput-object               v7, v5, v6
const/4                   v6, 2
aput-object               p1, v5, v6
const/4                   v6, 3
const/4                   v7, 0
aput-object               v7, v5, v6
const/4                   v6, 4
const/4                   v7, 0
aput-object               v7, v5, v6
invoke-virtual            {v3, v2, v5}, <ref Method.invoke(ref, ref) imp. @ Method_invoke>
move                      v5, v10
```

Reflection to hide calls

- This allows attackers to "hide" sensitive calls
- Easy to detect, just look for reflection
- If obfuscation is added; automation becomes harder
- Easier to detect via dynamic analysis

Hiding in Resources

```
champagne:gamex/assets tstrazzere$ ls -l logos.png
-rw-r--r--@ 1 tstrazzere  staff    42K Mar 29 16:33 logos.png
champagne:gamex/assets tstrazzere$ file logos.png
logos.png: data
champagne:gamex/assets tstrazzere$ hexdump -C logos.png | head
00000000  42 59 11 16 18 12 12 1a  12 12 5e 91 6f 52 a4 12  |BY........^.oR..|
00000010  55 4c 58 49 12 12 58 49  12 12 1d 12 15 12 73 61  |ULXI..XI......sa|
00000020  61 77 66 61 3d 7b 71 7d  7c 3c 62 7c 75 ec d8 12  |awfa={q}|<b|u...|
00000030  12 12 12 12 50 4b 03 04  0a 00 00 08 00 00 af 84  |....PK..........|
00000040  7c 40 93 03 c4 66 bc 00  00 00 bc 00 00 00 0f 00  ||@...f..........|
00000050  07 00 61 73 73 65 74 73  2f 69 63 6f 6e 2e 70 6e  |..assets/icon.pn|
00000060  67 fe ca 00 00 00 00 00  39 35 25 24 31 38 32 38  |g.......95%$1828|
00000070  24 33 39 34 25 3e 32 38  34 3f 39 34 39 36 38 39  |$394%>284?949689|
00000080  3c 3b 29 31 39 29 25 3d  3d 39 3d 31 29 3d 38 29  |<;)19)%==9=1)=8)|
00000090  25 3d 31 3c 30 25 34 3c  3b 34 25 3c 38 31 29 25  |%=1<0%4<;4%<81)%|
```

Hiding in Resources

```
champagne:gamex/assets tstrazzere$ unzip -l logos.png
Archive:  logos.png
warning [logos.png]:  52 extra bytes at beginning
  or within zipfile
  (attempting to process anyway)
  Length     Date   Time    Name
 --------    ----   ----    ----
      188  03-28-12 16:37   assets/icon.png
      311  03-29-12 16:25   assets/logo.png
     5666  03-27-12 22:15   res/drawable/ic_launcher.png
     2704  03-29-12 16:26   AndroidManifest.xml
      792  03-29-12 16:26   resources.arsc
    27408  03-29-12 16:26   classes.dex
      472  03-29-12 16:26   META-INF/MANIFEST.MF
      525  03-29-12 16:26   META-INF/CERT.SF
     1077  03-29-12 16:26   META-INF/CERT.RSA
 --------                   -------
    39143                   9 files
```

```
champagne:games/assets tstrazzere$ unzip -l logos.png.xored
Archive:  logos.png.xored
  Length     Date   Time    Name
 --------    ----   ----    ----
    23370  03-29-12 16:26   assets/icon.png
      188  03-28-12 16:37   assets/logo.png
     7359  03-27-12 22:24   res/drawable/ic_launcher.png
     3344  03-29-12 16:26   AndroidManifest.xml
      792  03-29-12 16:26   resources.arsc
    14932  03-29-12 16:26   classes.dex
      472  03-29-12 16:26   META-INF/MANIFEST.MF
      525  03-29-12 16:26   META-INF/CERT.SF
     1077  03-29-12 16:26   META-INF/CERT.RSA
 --------                   -------
    52059                   9 files
```

- Decompressing w/o decryption results in different APK file

Hiding in Resources

- Was this meant to happen? Possibly
- Embed into valid image file?
- Could have done a "better" job making it stealth -- i.e. unzip w/o decrypt is a harmless game
- Still an interesting and maybe most advanced attempt at "hiding" bad code!

Hiding in Resources Part Deux

```
champagne:assets tstrazzere$ file mylogo.jpg
mylogo.jpg: JPEG image data, JFIF standard 1.01

champagne:assets tstrazzere$ hexdump -C mylogo.jpg | grep "ELF\|JFIF" -B 2 -A 2
00000000  ff d8 ff e0 00 10 4a 46  49 46 00 01 01 01 00 60  |......JFIF.....`|
00000010  00 60 00 00 ff e1 00 5a  45 78 69 66 00 00 4d 4d  |.`.....ZExif..MM|
00000020  00 2a 00 00 00 08 00 05  03 01 00 05 00 00 00 01  |.*..............|
--
000051b0  7a 63 c9 52 3b f3 4c 79  38 ab 46 6c 89 9f 34 c6  |zc.R;.Ly8.Fl..4.|
000051c0  6a 91 9f 34 c6 63 54 88  63 68 a2 8a a2 42 8a 28  |j..4.cT.ch...B.(|
000051d0  a0 0f ff d9 7f 45 4c 46  01 01 01 00 00 00 00 00  |.....ELF........|
000051e0  00 00 00 00 02 00 28 00  01 00 00 00 f0 8e 00 00  |......(.........|
000051f0  34 00 00 00 50 9d 00 00  02 00 00 04 34 00 20 00  |4...P.......4. .|
--
00008440  8c 9a 8b 8f 8d 90 8f 00  8d cf d1 9d 90 8b d1 96  |................|
00008450  9b 00 00 00 8d cf d1 9d  90 8b d1 9c 97 00 00 00  |................|
00008460  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00008470  02 00 28 00 01 00 00 00  70 9f 00 00 34 00 00 00  |..(.....p...4...|
00008480  dc 65 00 00 02 00 00 04  34 00 20 00 07 00 28 00  |.e......4. ...(.|
```

Simple Studio

- Valid, viewable jpg with malicious ELF payloads

Hiding in Resources Part Deux

- Requires a looking at all resource files more in-depth
- Looks like a JPG, Smells like a JPG
- Could have done a "better" job making it stealth -- i.e. unzip w/o decrypt is a harmless game
- Still an interesting and maybe most advanced attempt at "hiding" bad code!

lookout™ MOBILE SECURITY

Simple Tool Breakage

```
champagne:linklocked tstrazzere$ baksmali protected.apk -o wontwork
UNEXPECTED TOP-LEVEL EXCEPTION:
org.jf.dexlib.Util.ExceptionWithContext: This dex file has a link section, which is not supported
        at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithContext.java:54)
        at org.jf.dexlib.Item.addExceptionContext(Item.java:176)
        at org.jf.dexlib.Item.readFrom(Item.java:78)
        at org.jf.dexlib.DexFile.<init>(DexFile.java:390)
        at org.jf.baksmali.main.main(main.java:254)
Caused by: java.lang.RuntimeException: This dex file has a link section, which is not supported
        at org.jf.dexlib.HeaderItem.readItem(HeaderItem.java:84)
        at org.jf.dexlib.Item.readFrom(Item.java:76)
        ... 2 more
header_item
```

- Seen being used by developers to thwart pirates, and pirates to thwart developers
- Seen used by Lohan+ (AntiLVL) / jcase / other devs

Simple Tool Breakage

- simple breakage == simple fix
- link size/offset inside the dex header was being set
- "data used in statically linked files" according to docs, no further details, appears to not be used currently
- Most tools just ignored this section

Hiding from, and breaking the tools

Adventures in Anti-Analysis

Step 1: Put on an evil hat and pretend to be evil
Step 2: Target most popular tools*
Step 3: Break the tools
Step 4: ???
Step 5: Report breakages/patches if possible
Step 6: ???
Step 7: Sell breakages... Wait - crap :\
Step 8: ~~$$$~~

*No data to back this up - just personal pref :)

Adventures in Anti-Analysis

- Targeting:

  Baksmali - used almost universally (apktool/antilvl as well)
  dex2jar - seems almost every uses (and relies heavily on this... barf)
  IDA Pro - most companies have this around
  androguard - seems to be popular, I don't personally use it :)
  others ???

Adventures in Anti-Analysis: Kickin` It Old School

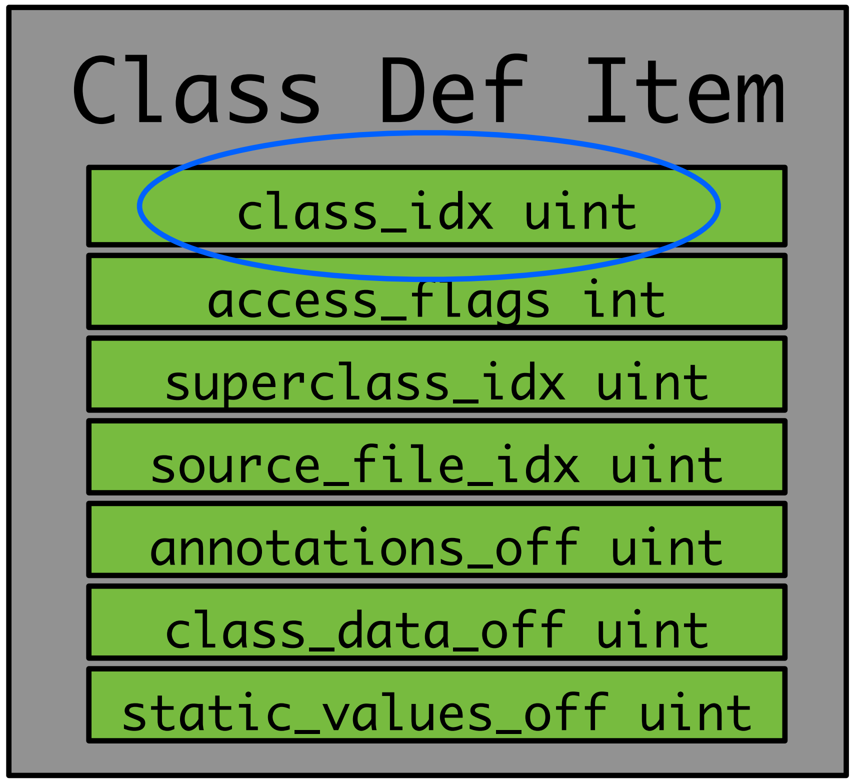
- APK is just an zip/jar, why not just use some old school jar hacks?
- Remember looking at old jars where files might be larger than 255+ characters?
- Required editing the jar file itself since contained files had no character limitation
- Damn those filesystem limitations!

Adventures in Anti-Analysis: Kickin` It Old School

- How can we port this to the dex file for breakage?
- Need the class to be 255+ chars

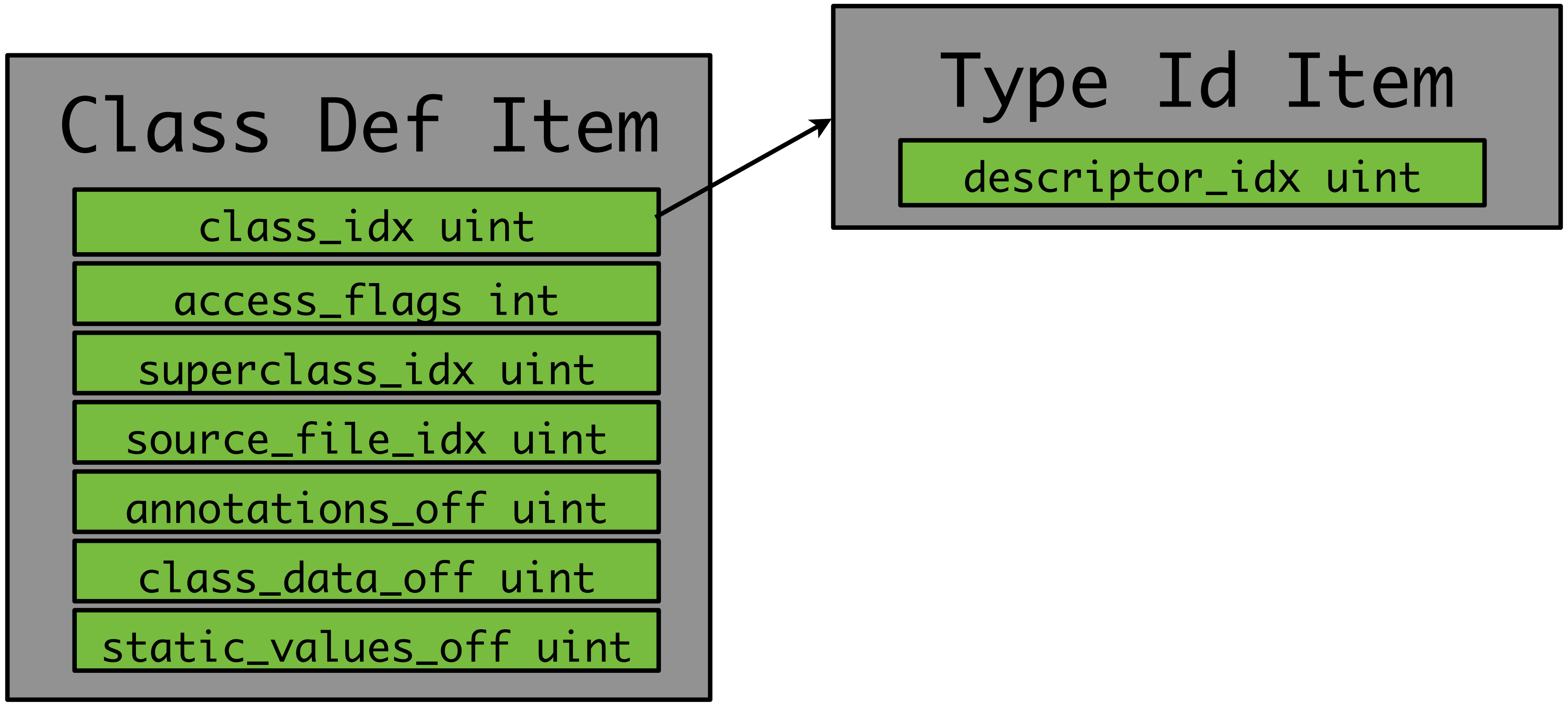
Adventures in Anti-Analysis: Kickin` It Old School

- How can we port this to the dex file for breakage?
- Need the class to be 255+ chars



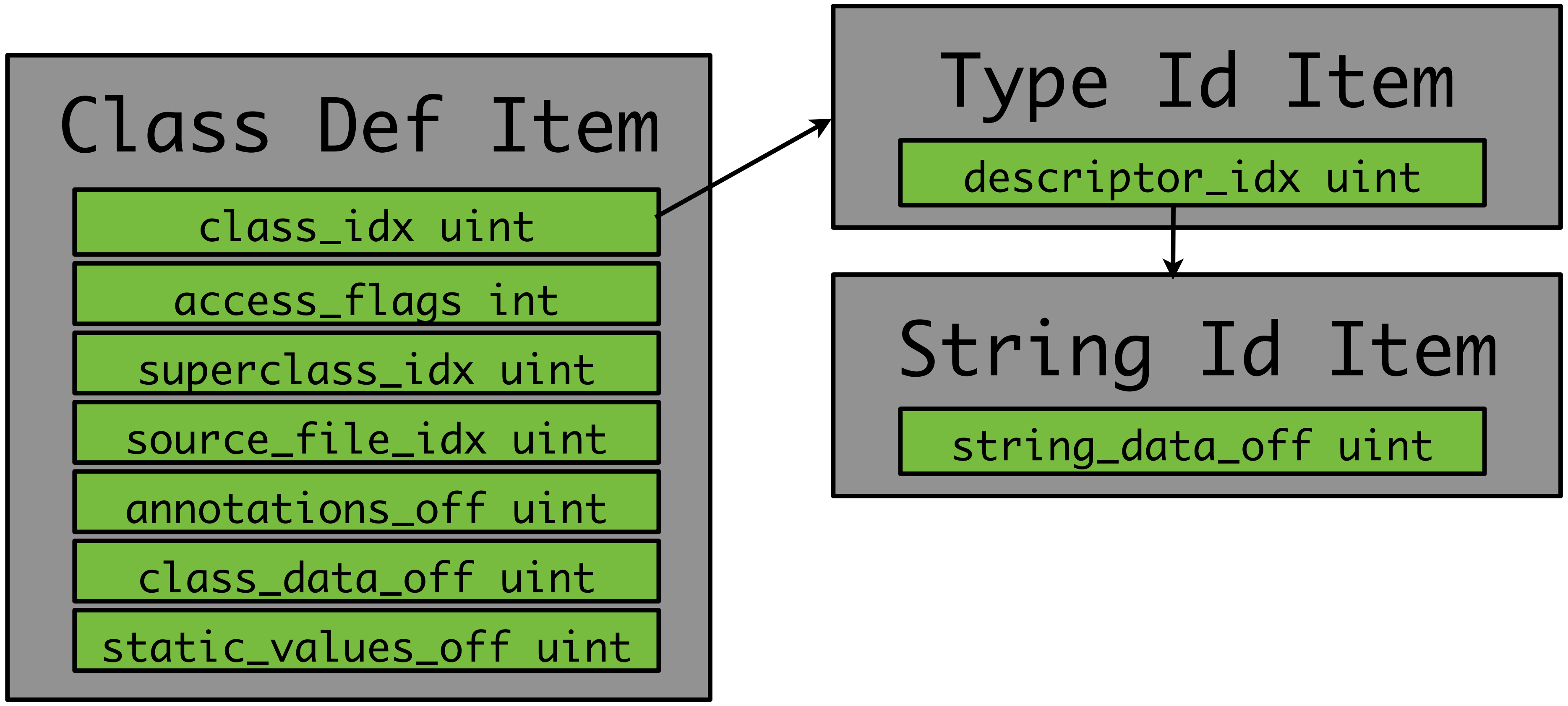
Adventures in Anti-Analysis: Kickin` It Old School

- How can we port this to the dex file for breakage?
- Need the class to be 255+ chars



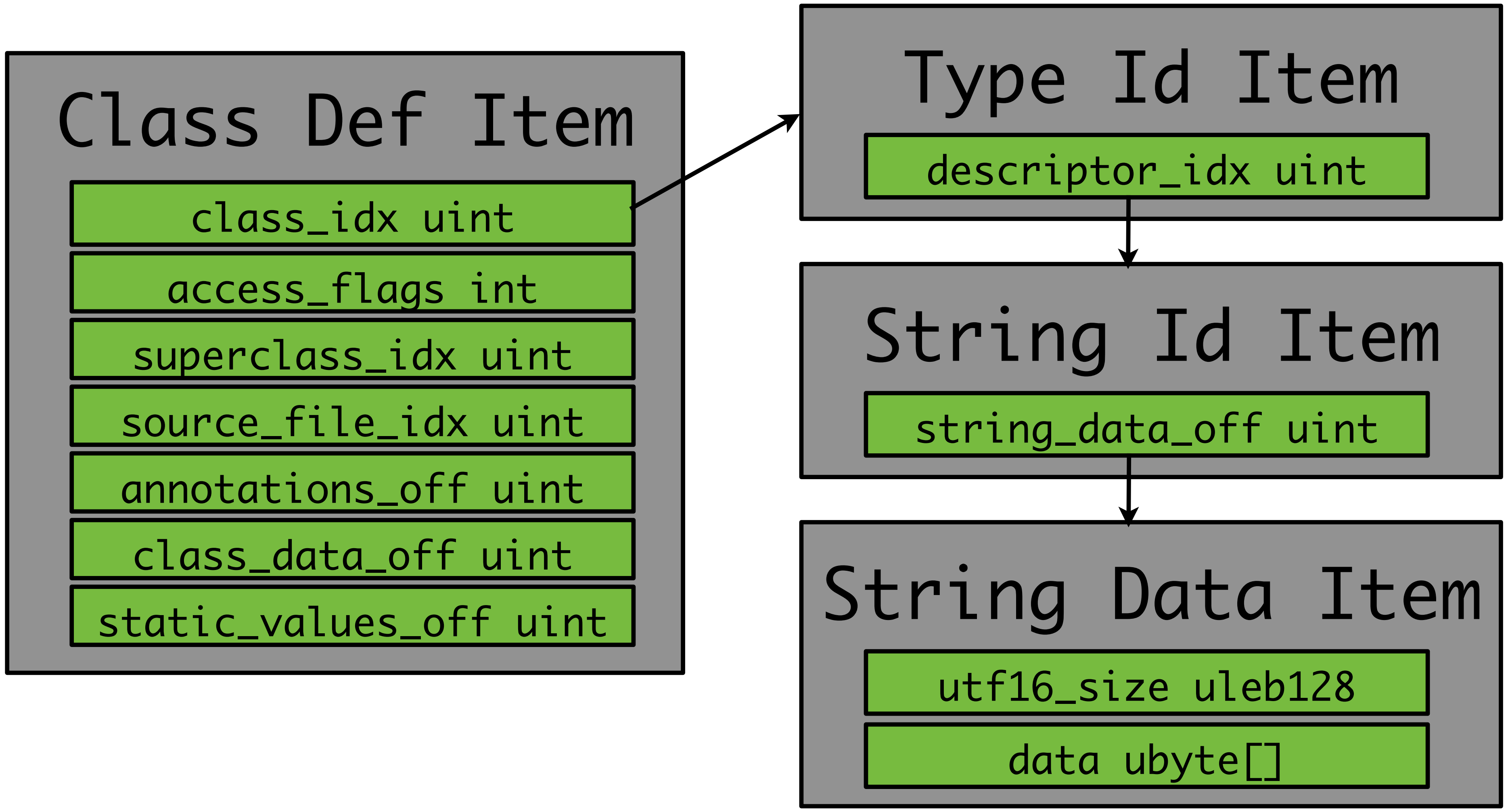
Adventures in Anti-Analysis: Kickin` It Old School

- How can we port this to the dex file for breakage?
- Need the class to be 255+ chars



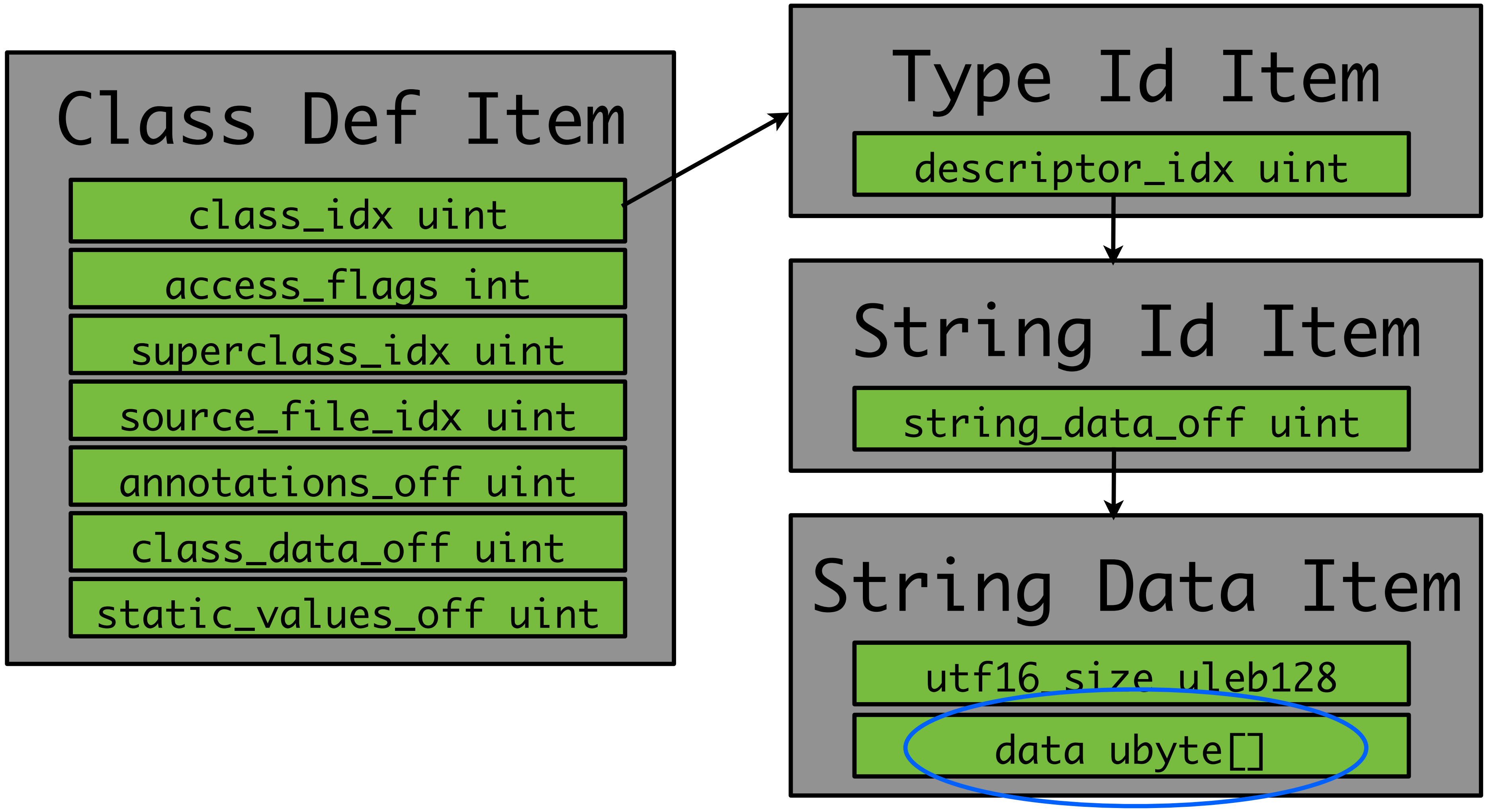
Adventures in Anti-Analysis: Kickin` It Old School

- How can we port this to the dex file for breakage?
- Need the class to be 255+ chars



Adventures in Anti-Analysis: Kickin` It Old School

- How can we port this to the dex file for breakage?
- Need the class to be 255+ chars



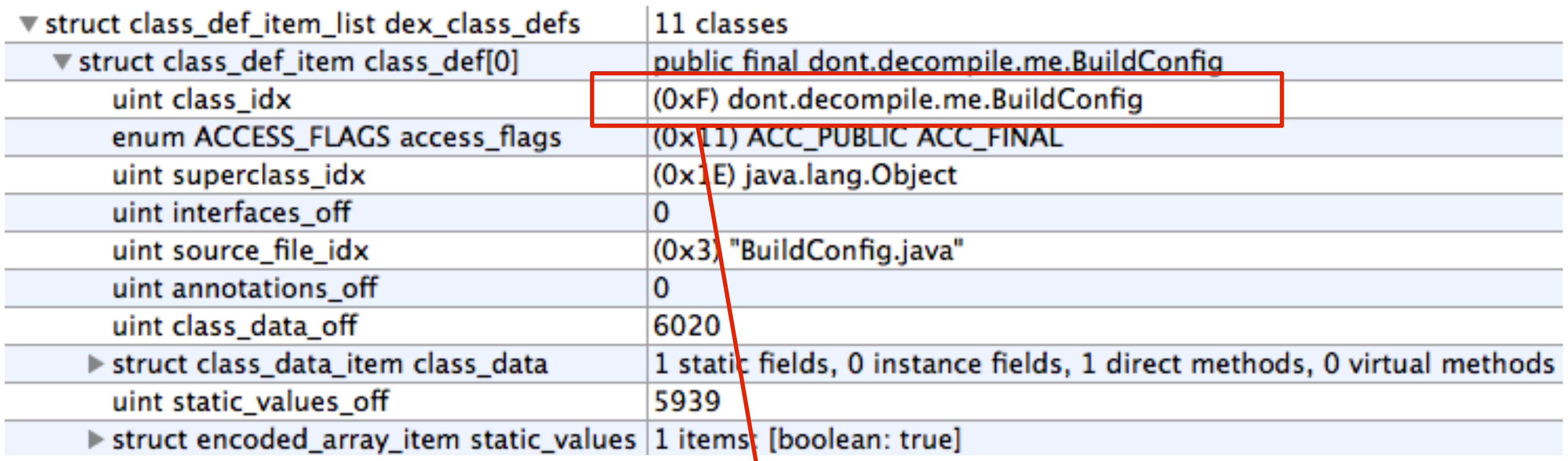
lookout MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

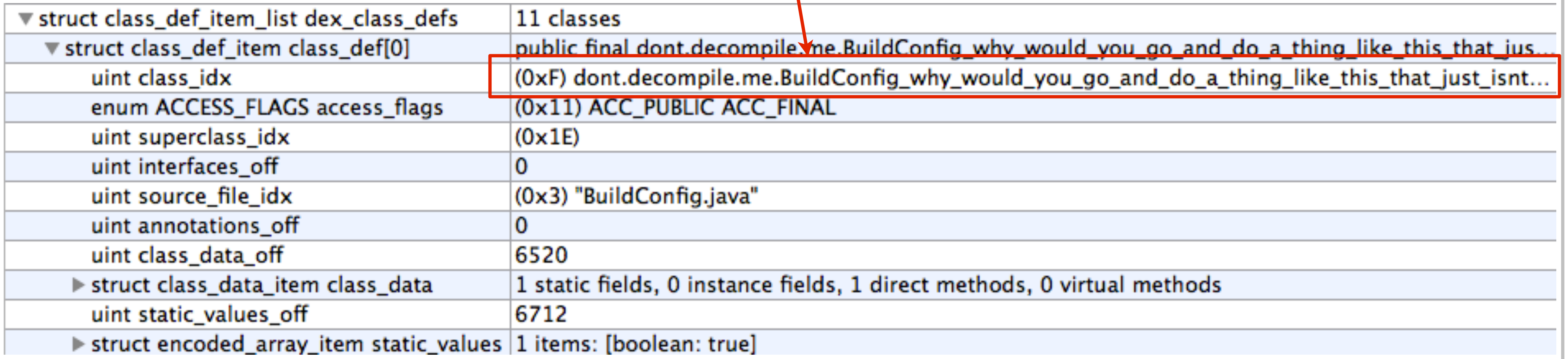
- Change the class name in the string data to a valid, but large class name
- Minor gotcha -- string table must *always* be in alphanumeric order!
- Just append data to the end of an early processed class (avoids having to reassemble tables)
- Wave our magic wand and see if this works...

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School



▼ struct class_def_item_list dex_class_defs	11 classes
▼ struct class_def_item class_def[0]	public final dont.decompile.me.BuildConfig
uint class_idx	(0xF) dont.decompile.me.BuildConfig
enum ACCESS_FLAGS access_flags	(0x11) ACC_PUBLIC ACC_FINAL
uint superclass_idx	(0x1E) java.lang.Object
uint interfaces_off	0
uint source_file_idx	(0x3) "BuildConfig.java"
uint annotations_off	0
uint class_data_off	6020
▶ struct class_data_item class_data	1 static fields, 0 instance fields, 1 direct methods, 0 virtual methods
uint static_values_off	5939
▶ struct encoded_array_item static_values	1 items: [boolean: true]



▼ struct class_def_item_list dex_class_defs	11 classes
▼ struct class_def_item class_def[0]	public final dont.decompile.me.BuildConfig_why_would_you_go_and_do_a_thing_like_this_that_jus...
uint class_idx	(0xF) dont.decompile.me.BuildConfig_why_would_you_go_and_do_a_thing_like_this_that_just_isnt...
enum ACCESS_FLAGS access_flags	(0x11) ACC_PUBLIC ACC_FINAL
uint superclass_idx	(0x1E)
uint interfaces_off	0
uint source_file_idx	(0x3) "BuildConfig.java"
uint annotations_off	0
uint class_data_off	6520
▶ struct class_data_item class_data	1 static fields, 0 instance fields, 1 direct methods, 0 virtual methods
uint static_values_off	6712
▶ struct encoded_array_item static_values	1 items: [boolean: true]

Adventures in Anti-Analysis: Kickin` It Old School

- Does it install? Yes!(after some work of course...)

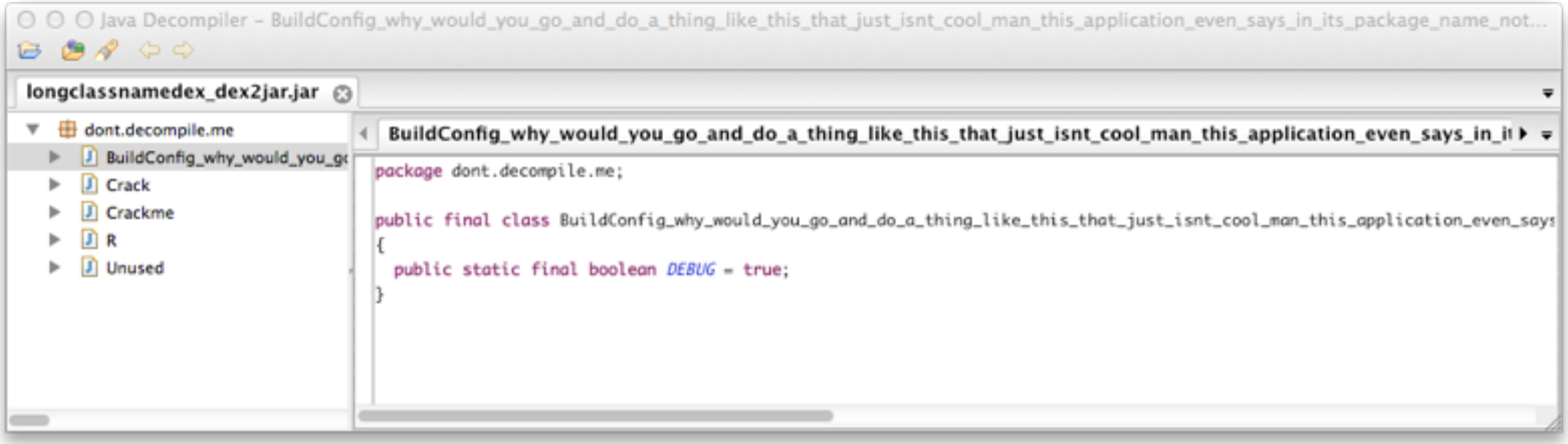
```
champagne:long-class-name tstrazzere$ adb install longclassnamedex.apk
1627 KB/s (9917 bytes in 0.005s)
        pkg: /data/local/tmp/longclassnamedex.apk
Success
```

- Does it break any tools? Sort of...

```
champagne:long-class-name tstrazzere$ baksmali longclassnamedex.apk -o wontwork


Error occured while disassembling class
Ldont.decompile.me.BuildConfig_why_would_you_go_and_do_a_thing_like_this_that_just_isnt_cool_man_this
_application_even_says_in_its_package_name_not_to_decompile_it_have_you_no_manners_______someday_
someone_might_decompile_you_then_youll_understand_the_feelings_this_poor_little_dex_file_is_feeling_
right_at_this_moment; - skipping class
java.io.IOException: File name too long
        at java.io.UnixFileSystem.createFileExclusively(Native Method)
        at java.io.File.createNewFile(File.java:883)
        at org.jf.baksmali.baksmali.disassembleDexFile(baksmali.java:195)
        at org.jf.baksmali.main.main(main.java:293)
```

Adventures in Anti-Analysis: Kickin` It Old School

- Results;
  IDA works
  Dex2jar works (depending on tool/use afterwards)
  Androguard works
  Baksmali works (sort of -- except for that class)



Java Decompiler – BuildConfig_why_would_you_go_and_do_a_thing_like_this_that_just_isnt_cool_man_this_application_even_says_in_its_package_name_not...
longclassnamedex_dex2jar.jar
dont.decompile.me
BuildConfig_why_would_you_go
Crack
Crackme
R
Unused
BuildConfig_why_would_you_go_and_do_a_thing_like_this_that_just_isnt_cool_man_this_application_even_says_in_it

```
package dont.decompile.me;

public final class BuildConfig_why_would_you_go_and_do_a_thing_like_this_that_just_isnt_cool_man_this_application_even_says
{
  public static final boolean DEBUG = true;
}
```

- We want something to break more!

Adventures in Anti-Analysis: Kickin` It Old School

- Easy to detect and work around
- Class name > 255 chars? alert! (maybe)
- Someone might be trying to mess with analysis systems
- Someone *might* just not understand that they have a horrible class name...

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

- Lots of interesting things you can do with this style of hack
- Targeting a mac user/case-sensitive filesystem? Ii == iI (Fixed in baksmali a little bit back)
- Throwing some nasty ASCII characters in there: ÀÀÁÀÀ - they're a pain to work with on the command line :(

lookout MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

- What about throwing some bad opcodes at these guys? Sort of like other malware on the PC that would cause tools to die

- Plan of attack:
  Inject "dead code" into the dex file which will never be executed, therefore the devices won't care!

lookout™
MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

- Goal to inject:

```
1201        // Load 0 into v1
3801 0300   // A conditional jump which should always succeed, jumps over
            // next bytes
FFFF        // Bad opcodes
```

Into a class we want to protect

```
champagne:long-class-name tstrazzere$ adb install bad-opcodes-1.apk
1627 KB/s (9917 bytes in 0.005s)
        pkg: /data/local/tmp/bad-opcodes-1.apk
Success
```

- Now lets rub our hands together and cackle like an evil genius!

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

```
W/dalvikvm( 2567): VFY: invalid instruction (0xffff)
W/dalvikvm( 2567): VFY:  rejected Ldont/decompile/me/Crackme;.access$0 (Ldont/decompile/me/Crackme;)Landroid/widget/
EditText;
W/dalvikvm( 2567): Verifier rejected class Ldont/decompile/me/Crackme;
W/dalvikvm( 2567): Class init failed in newInstance call (Ldont/decompile/me/Crackme;)
D/AndroidRuntime( 2567): Shutting down VM
W/dalvikvm( 2567): threadid=1: thread exiting with uncaught exception (group=0x40d08300)
E/AndroidRuntime( 2567): FATAL EXCEPTION: main
E/AndroidRuntime( 2567): java.lang.VerifyError: dont/decompile/me/Crackme
E/AndroidRuntime( 2567):  at java.lang.Class.newInstanceImpl(Native Method)
E/AndroidRuntime( 2567):  at java.lang.Class.newInstance(Class.java:1319)
E/AndroidRuntime( 2567):  at android.app.Instrumentation.newActivity(Instrumentation.java:1053)
E/AndroidRuntime( 2567):  at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1974)
E/AndroidRuntime( 2567):  at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2084)
E/AndroidRuntime( 2567):  at android.app.ActivityThread.access$600(ActivityThread.java:130)
E/AndroidRuntime( 2567):  at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1195)
E/AndroidRuntime( 2567):  at android.os.Handler.dispatchMessage(Handler.java:99)
E/AndroidRuntime( 2567):  at android.os.Looper.loop(Looper.java:137)
E/AndroidRuntime( 2567):  at android.app.ActivityThread.main(ActivityThread.java:4745)
E/AndroidRuntime( 2567):  at java.lang.reflect.Method.invokeNative(Native Method)
E/AndroidRuntime( 2567):  at java.lang.reflect.Method.invoke(Method.java:511)
E/AndroidRuntime( 2567):  at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:786)
E/AndroidRuntime( 2567):  at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:553)
E/AndroidRuntime( 2567):  at dalvik.system.NativeStart.main(Native Method)
W/ActivityManager(  306):   Force finishing activity dont.decompile.me/.Crackme
```

fail :(

Adventures in Anti-Analysis: Kickin` It Old School

- What happened? Why the failure?
- Dalvik verifier facepalmed us while executing all relevant code paths, it didn't actually skip over the dead code as originally expected :(
- So... if we can avoid the verifier, then we should be able to avoid this!
- Back to the drawing board

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

- New goal, to inject:

```
1201        // Load 0 into v1
3801 0300   // A conditional jump which should always succeed, jumps over
            // next bytes
FFFF        // Bad opcodes
```

Into a class we ~~want to protect~~ don't care about

```
champagne:long-class-name tstrazzere$ adb install bad-opcodes-2.apk
1627 KB/s (9917 bytes in 0.005s)
	pkg: /data/local/tmp/bad-opcodes-2.apk
Success
```

- This time... this time we will get it right!

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

```
I/ActivityManager(  306): START {act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] 
flg=0x10200000 cmp=dont.decompile.me/.Crackme u=0} from pid 538
I/ActivityManager(  306): Start proc dont.decompile.me for activity dont.decompile.me/.Crackme: pid=3464 
uid=10073 gids={1015, 1028}
I/dalvikvm( 3464): Turning on JNI app bug workarounds for target SDK version 3...
I/don't decompile me( 3464): please?
V/PhoneStatusBar(  390): setLightsOn(true)
I/ActivityManager(  306): Displayed dont.decompile.me/.Crackme: +242ms (total +7m18s529ms)
```

- Awesome, what tools can we break now?

lookout MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

```
champagne:bad-opcodes tstrazzere$ baksmali bad-opcodes-2.apk -o wontwork


UNEXPECTED TOP-LEVEL EXCEPTION:
org.jf.dexlib.Util.ExceptionWithContext: Unknown opcode: ff
        at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithContext.java:54)
        at org.jf.dexlib.Code.InstructionIterator.IterateInstructions(InstructionIterator.java:92)
        at org.jf.dexlib.CodeItem.readItem(CodeItem.java:154)
        at org.jf.dexlib.Item.readFrom(Item.java:76)
        at org.jf.dexlib.OffsettedSection.readItems(OffsettedSection.java:48)
        at org.jf.dexlib.Section.readFrom(Section.java:143)
        at org.jf.dexlib.DexFile.<init>(DexFile.java:431)
        at org.jf.baksmali.main.main(main.java:265)
Caused by: java.lang.RuntimeException: Unknown opcode: ff
        at org.jf.dexlib.Code.InstructionIterator.IterateInstructions(InstructionIterator.java:56)
        ... 6 more
Error occured at code address 12
code_item @0x1804
```

- Baksmali - check!
  (fixed in revision 2f81aec886d2 7/28)

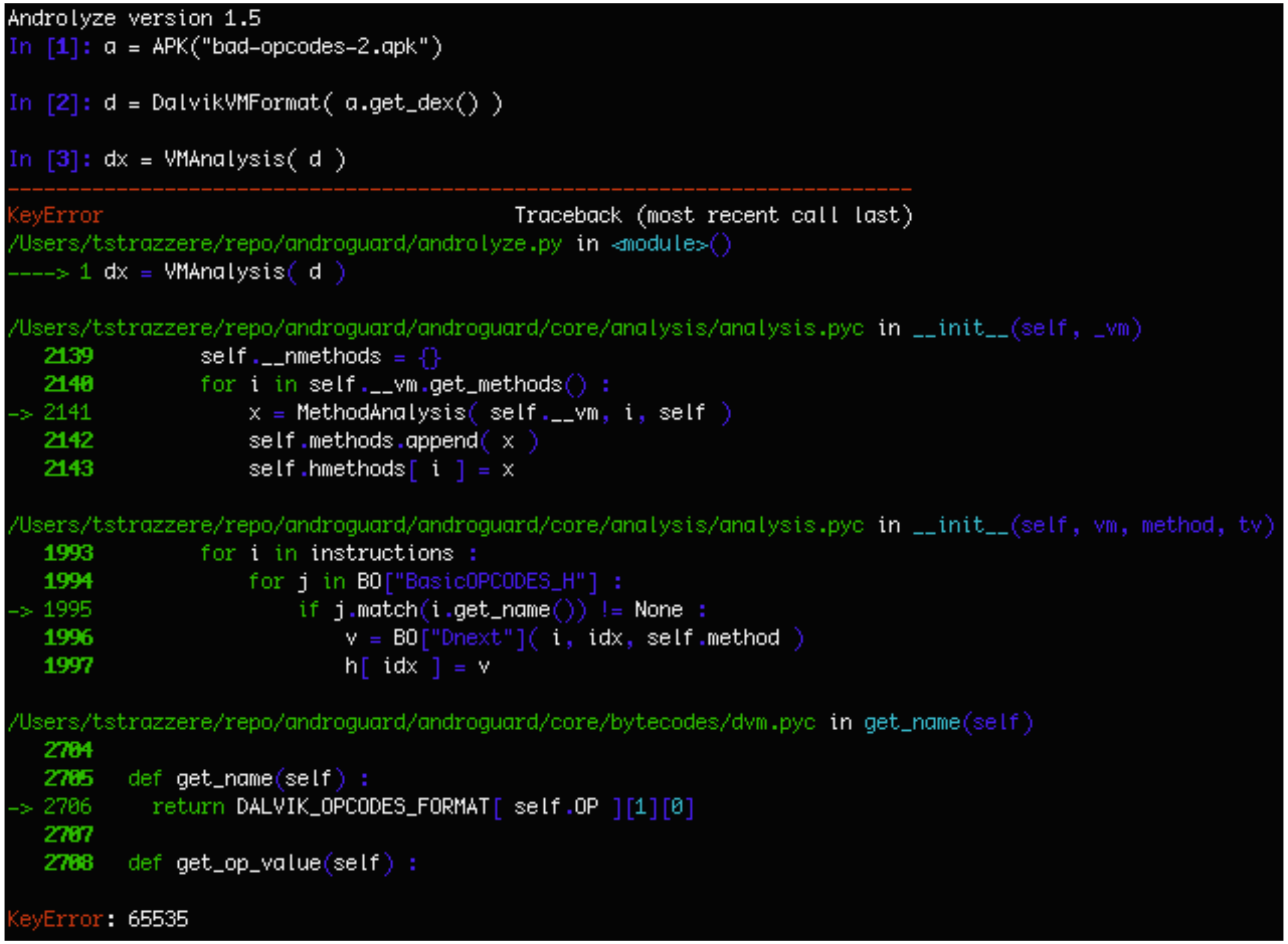
Adventures in Anti-Analysis: Kickin` It Old School

```
champagne:bad-opcodes tstrazzere$ dex2jar bad-opcodes-2.apk
dex2jar version: translator-0.0.9.8
dex2jar bad-opcodes-2.apk -> bad-opcodes-2_dex2jar.jar
com.googlecode.dex2jar.DexException: while accept method:[Ldont/decompile/me/Unused;.<init>()V]
	at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:705)
	at com.googlecode.dex2jar.reader.DexFileReader.acceptClass(DexFileReader.java:446)
	at com.googlecode.dex2jar.reader.DexFileReader.accept(DexFileReader.java:333)
	at com.googlecode.dex2jar.v3.Dex2jar.doTranslate(Dex2jar.java:82)
	at com.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:191)
	at com.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:182)
	at com.googlecode.dex2jar.v3.Main.doData(Main.java:43)
	at com.googlecode.dex2jar.v3.Main.doData(Main.java:35)
	at com.googlecode.dex2jar.v3.Main.doFile(Main.java:63)
	at com.googlecode.dex2jar.v3.Main.main(Main.java:85)
Caused by: com.googlecode.dex2jar.DexException: while accept code in method:[Ldont/decompile/me/Unused;.<init>()V]
	at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:695)
	... 9 more
Caused by: java.lang.RuntimeException: opcode format for 65535 not found!
	at com.googlecode.dex2jar.reader.OpcodeFormat.get(OpcodeFormat.java:362)
	at com.googlecode.dex2jar.reader.DexCodeReader.findLabels(DexCodeReader.java:85)
	at com.googlecode.dex2jar.reader.DexCodeReader.accept(DexCodeReader.java:287)
	at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:692)
	... 9 more
Done.
```

- Dex2jar - check!

lookout™ MOBILE SECURITY

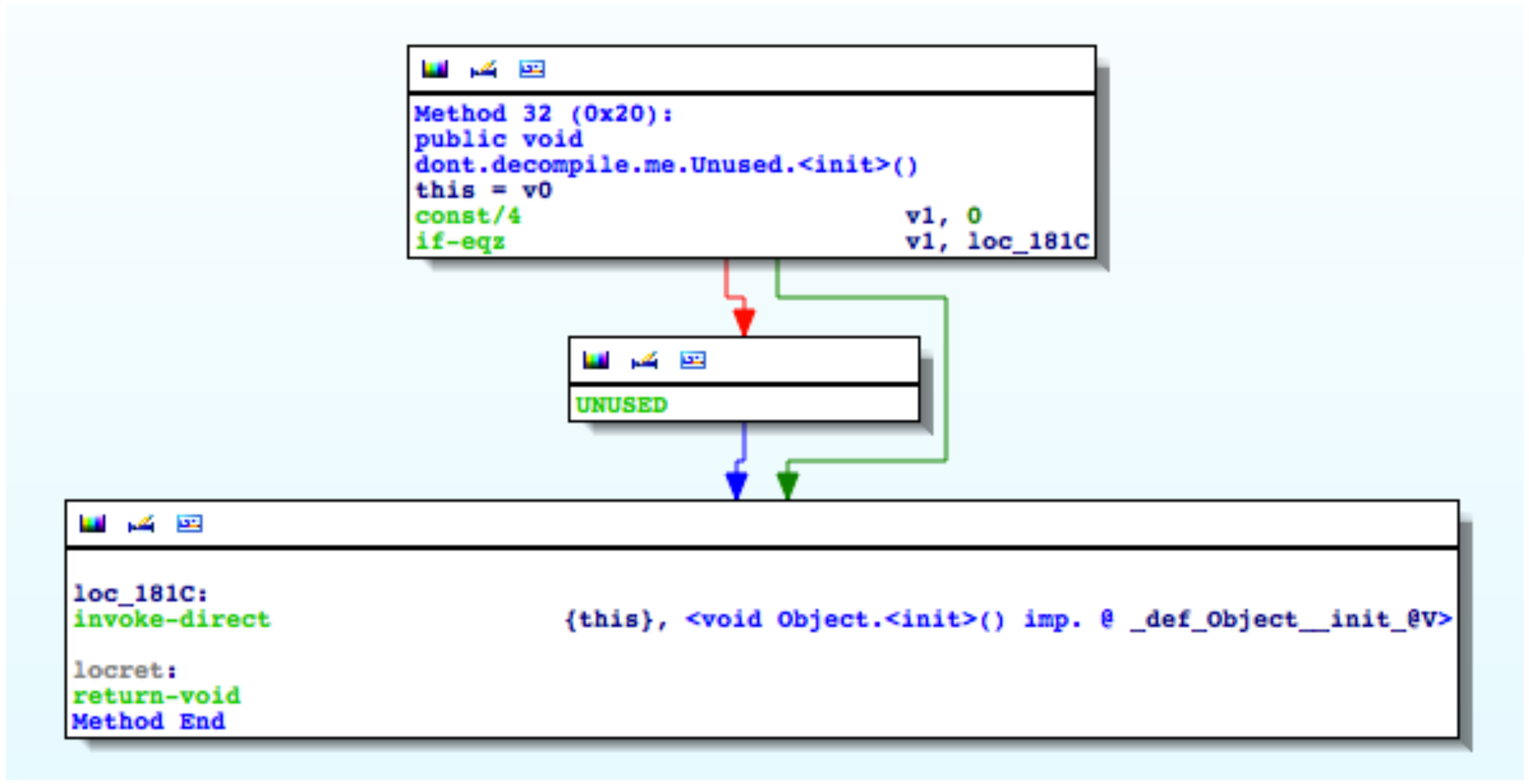
Adventures in Anti-Analysis: Kickin` It Old School



```
Androlyze version 1.5
In [1]: a = APK("bad-opcodes-2.apk")

In [2]: d = DalvikVMFormat( a.get_dex() )

In [3]: dx = VMAnalysis( d )
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/Users/tstrazzere/repo/androguard/androlyze.py in <module>()
----> 1 dx = VMAnalysis( d )

/Users/tstrazzere/repo/androguard/androguard/core/analysis/analysis.pyc in __init__(self, _vm)
   2139         self.__nmethods = {}
   2140         for i in self.__vm.get_methods() :
-> 2141             x = MethodAnalysis( self.__vm, i, self )
   2142             self.methods.append( x )
   2143             self.hmethods[ i ] = x

/Users/tstrazzere/repo/androguard/androguard/core/analysis/analysis.pyc in __init__(self, vm, method, tv)
   1993         for i in instructions :
   1994             for j in BO["BasicOPCODES_H"] :
-> 1995                 if j.match(i.get_name()) != None :
   1996                     v = BO["Dnext"]( i, idx, self.method )
   1997                     h[ idx ] = v

/Users/tstrazzere/repo/androguard/androguard/core/bytecodes/dvm.pyc in get_name(self)
   2704
   2705     def get_name(self) :
-> 2706       return DALVIK_OPCODES_FORMAT[ self.OP ][1][0]
   2707
   2708     def get_op_value(self) :

KeyError: 65535
```

- Androguard - check! (fix commited already/soon)

Adventures in Anti-Analysis: Kickin` It Old School



```
Method 32 (0x20):
public void
dont.decompile.me.Unused.<init>()
this = v0
const/4                                  v1, 0
if-eqz                                   v1, loc_181C
```

```
UNUSED
```

```
loc_181C:
invoke-direct                    {this}, <void Object.<init>() imp. @ _def_Object__init_@V>

locret:
return-void
Method End
```

- IDA Pro - doh!

Adventures in Anti-Analysis: Kickin` It Old School

```
champagne:bad-opcodes tstrazzere$ ded-0.7.1 -d $PWD -j ~/Downloads/jasminclasses-2.4.0.jar bad-opcodes-2.apk
Processing class #0: Ldont/decompile/me/BuildConfig;
Processing class #1: Ldont/decompile/me/Crack;
Processing class #2: Ldont/decompile/me/Crackme$1;
Processing class #3: Ldont/decompile/me/Crackme;
Processing class #4: Ldont/decompile/me/R$attr;
Processing class #5: Ldont/decompile/me/R$drawable;
Processing class #6: Ldont/decompile/me/R$id;
Processing class #7: Ldont/decompile/me/R$layout;
Processing class #8: Ldont/decompile/me/R$string;
Processing class #9: Ldont/decompile/me/R;
GLITCH: zero-width instruction at idx=0x0003
Processing class #10: Ldont/decompile/me/Unused;
```

- Ded - check! With a bonus of hogging a core until you kill the process!

lookout MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

- Round up? Only IDA Pro worked out of what I tested
- Why? Resilience to unexpected op codes
- Easy fix, baksmali has it done, androguard already had it done - but 0xFF was missed! ;)
- A good "edge case" to think of when developing these types of tools

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Kickin` It Old School

- Good thing to look for with automated analysis
- Used illegal opcodes? alert!
- Could be your tool chain is out of date, need to update to support more opcodes
- Could be someone trying to break your tool chain -- probably something worth looking at for that reason alone!

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Pushing the Bounds

- So illegal opcodes work(ed)
- What about legal opcodes, to bad objects?

```
1201       // Load 0 into v1
3801 0300  // A conditional jump which should always succeed, jumps over
           // next bytes
1a00 FF00  // Load const-string at index 255 (doesn't exist)
```

- We still want to avoid the verifier, but this is more "valid" than bad opcodes

Adventures in Anti-Analysis: Pushing the Bounds

```
champagne:bad-offsets tstrazzere$ baksmali bad-string-offsets.apk -o wontwork
UNEXPECTED TOP-LEVEL EXCEPTION:
org.jf.dexlib.Util.ExceptionWithContext: Index: 255, Size: 141
        at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithContext.java:54)
        at org.jf.dexlib.IndexedSection.getItemByIndex(IndexedSection.java:77)
        at org.jf.dexlib.Code.InstructionWithReference.lookupReferencedItem(InstructionWithReference.java:88)
        at org.jf.dexlib.Code.InstructionWithReference.<init>(InstructionWithReference.java:57)
        at org.jf.dexlib.Code.Format.Instruction21c.<init>(Instruction21c.java:63)
        at org.jf.dexlib.Code.Format.Instruction21c.<init>(Instruction21c.java:40)
        at org.jf.dexlib.Code.Format.Instruction21c$Factory.makeInstruction(Instruction21c.java:112)
        at org.jf.dexlib.Code.InstructionIterator.IterateInstructions(InstructionIterator.java:84)
        at org.jf.dexlib.CodeItem.readItem(CodeItem.java:154)
        at org.jf.dexlib.Item.readFrom(Item.java:76)
        at org.jf.dexlib.OffsettedSection.readItems(OffsettedSection.java:48)
        at org.jf.dexlib.Section.readFrom(Section.java:143)
        at org.jf.dexlib.DexFile.<init>(DexFile.java:431)
        at org.jf.baksmali.main.main(main.java:265)
Caused by: java.lang.IndexOutOfBoundsException: Index: 255, Size: 141
        at java.util.ArrayList.RangeCheck(ArrayList.java:547)
        at java.util.ArrayList.get(ArrayList.java:322)
        at org.jf.dexlib.IndexedSection.getItemByIndex(IndexedSection.java:75)
        ... 12 more
Error occured while retrieving the string_id_item item at index 255
Error occured at code address 12
code_item @0x1804
```

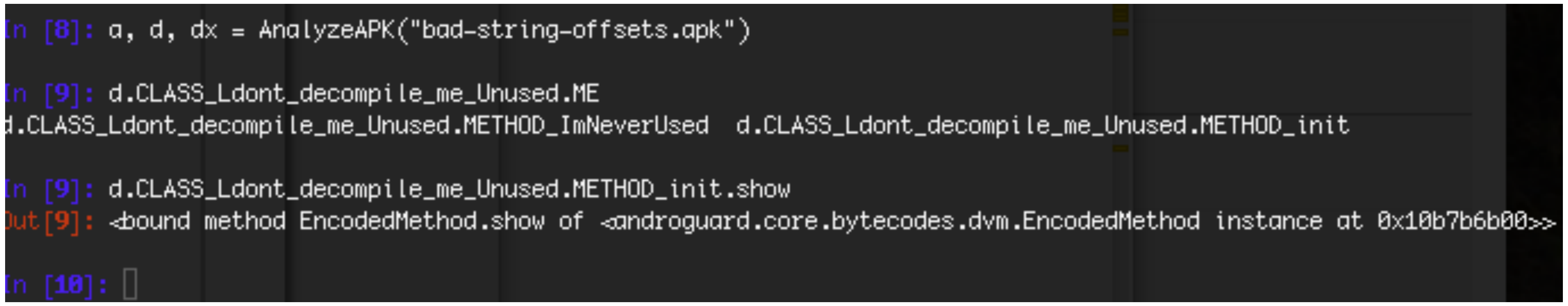
- Baksmali - check!

Adventures in Anti-Analysis: Pushing the Bounds

```
champagne:bad-offsets tstrazzere$ ./dex2jar.sh bad-string-offsets.apk
dex2jar version: translator-0.0.9.8
dex2jar bad-string-offsets.apk -> bad-string-offsets_dex2jar.jar
com.googlecode.dex2jar.DexException: while accept method:[Ldont/decompile/me/Unused;.<init>()V]
	at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:705)
	at com.googlecode.dex2jar.reader.DexFileReader.acceptClass(DexFileReader.java:446)
	at com.googlecode.dex2jar.reader.DexFileReader.accept(DexFileReader.java:333)
	at com.googlecode.dex2jar.v3.Dex2jar.doTranslate(Dex2jar.java:82)
	at com.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:191)
	at com.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:182)
	at com.googlecode.dex2jar.v3.Main.doData(Main.java:43)
	at com.googlecode.dex2jar.v3.Main.doData(Main.java:35)
	at com.googlecode.dex2jar.v3.Main.doFile(Main.java:63)
	at com.googlecode.dex2jar.v3.Main.main(Main.java:85)
Caused by: com.googlecode.dex2jar.DexException: while accept code in method:[Ldont/decompile/me/Unused;.<init>()V]
	at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:695)
	... 9 more
Caused by: java.lang.IllegalArgumentException: Id out of bound
	at com.googlecode.dex2jar.reader.DexFileReader.getString(DexFileReader.java:537)
	at com.googlecode.dex2jar.reader.DexOpcodeAdapter.x1c(DexOpcodeAdapter.java:129)
	at com.googlecode.dex2jar.reader.DexCodeReader.acceptInsn(DexCodeReader.java:386)
	at com.googlecode.dex2jar.reader.DexCodeReader.accept(DexCodeReader.java:292)
	at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:692)
	... 9 more
Done.
```

- Dex2jar - check! (fails only on that file)

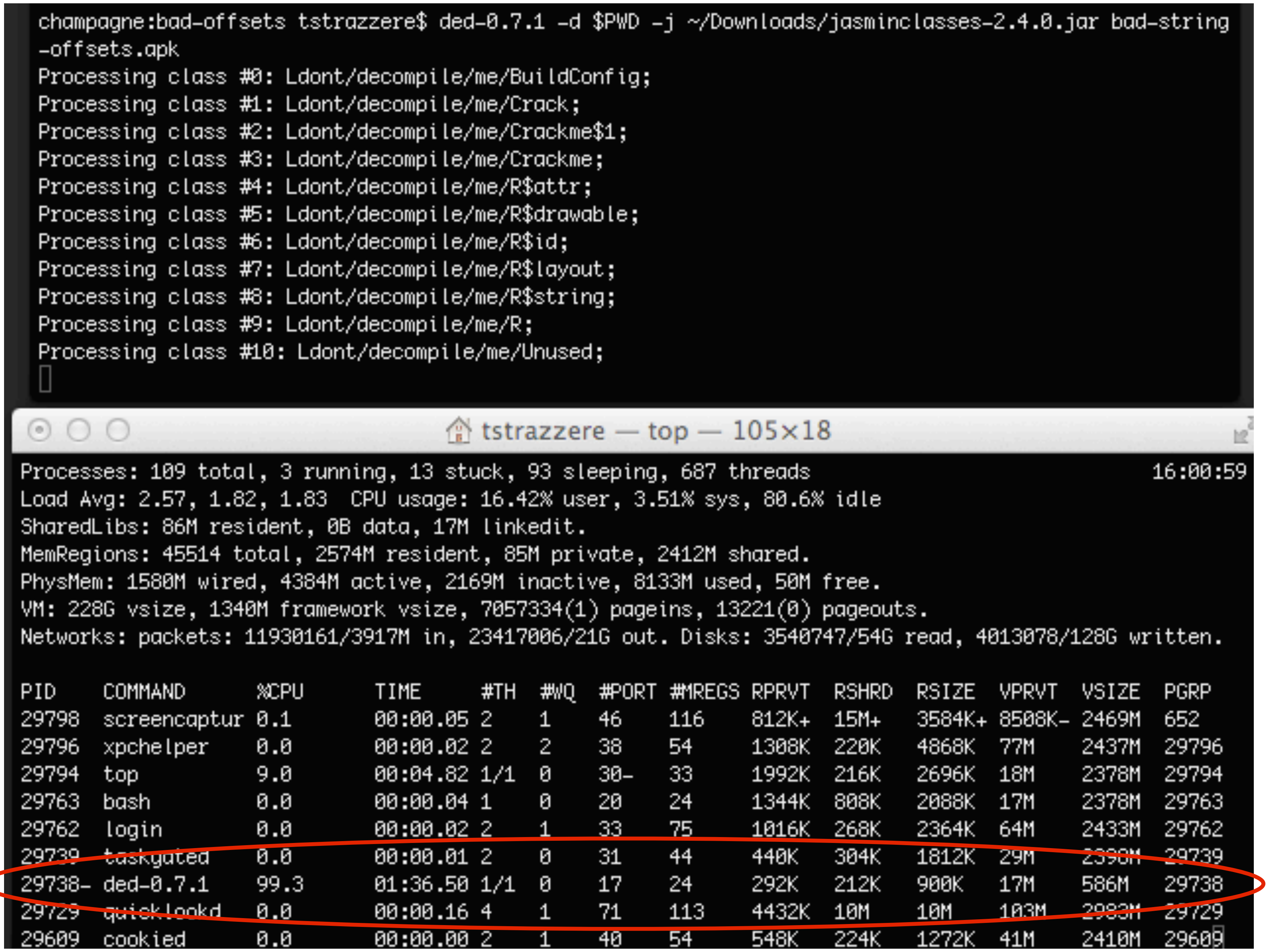
Adventures in Anti-Analysis: Pushing the Bounds



```
In [8]: a, d, dx = AnalyzeAPK("bad-string-offsets.apk")

In [9]: d.CLASS_Ldont_decompile_me_Unused.ME
d.CLASS_Ldont_decompile_me_Unused.METHOD_ImNeverUsed  d.CLASS_Ldont_decompile_me_Unused.METHOD_init

In [9]: d.CLASS_Ldont_decompile_me_Unused.METHOD_init.show
Out[9]: <bound method EncodedMethod.show of <androguard.core.bytecodes.dvm.EncodedMethod instance at 0x10b7b6b00>>

In [10]: 
```

- Androguard - check, sort of - only on the function which is using it

Adventures in Anti-Analysis: Pushing the Bounds



```
champagne:bad-offsets tstrazzere$ ded-0.7.1 -d $PWD -j ~/Downloads/jasminclasses-2.4.0.jar bad-string
-offsets.apk
Processing class #0: Ldont/decompile/me/BuildConfig;
Processing class #1: Ldont/decompile/me/Crack;
Processing class #2: Ldont/decompile/me/Crackme$1;
Processing class #3: Ldont/decompile/me/Crackme;
Processing class #4: Ldont/decompile/me/R$attr;
Processing class #5: Ldont/decompile/me/R$drawable;
Processing class #6: Ldont/decompile/me/R$id;
Processing class #7: Ldont/decompile/me/R$layout;
Processing class #8: Ldont/decompile/me/R$string;
Processing class #9: Ldont/decompile/me/R;
Processing class #10: Ldont/decompile/me/Unused;
```

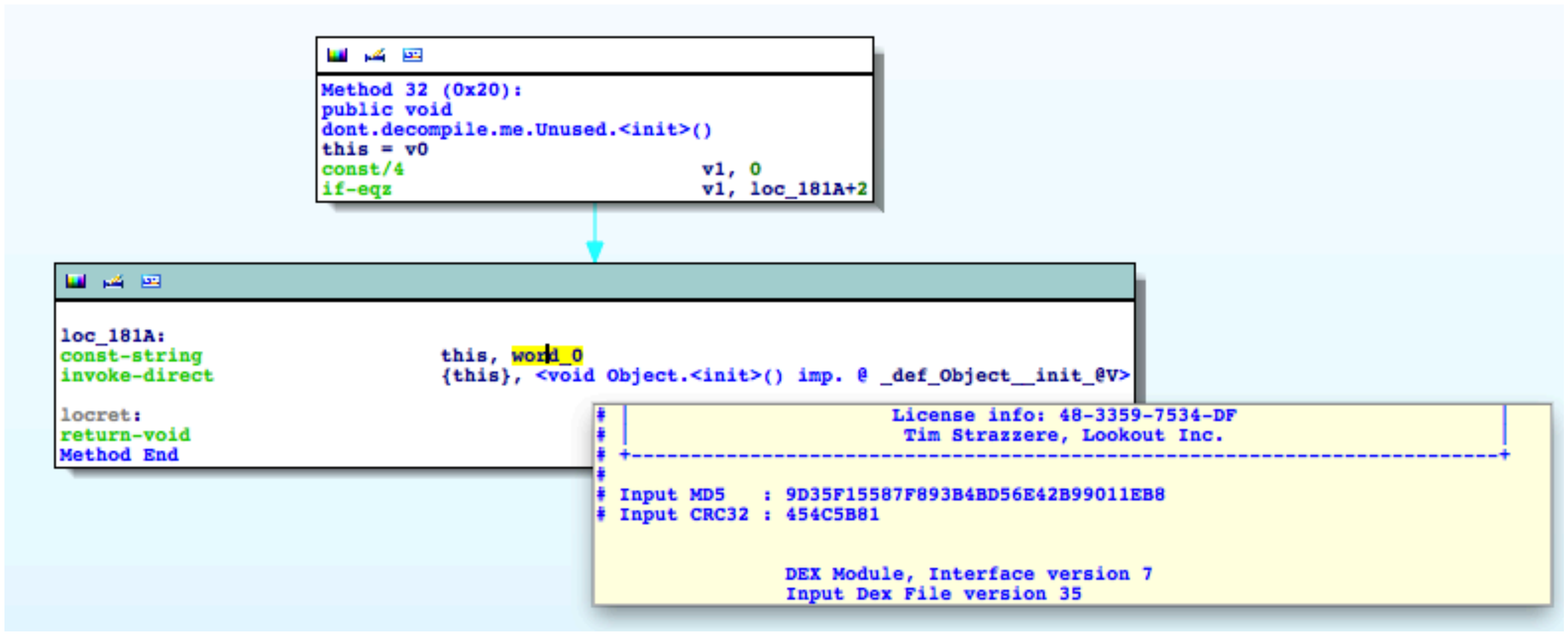
tstrazzere — top — 105×18

```
Processes: 109 total, 3 running, 13 stuck, 93 sleeping, 687 threads                               16:00:59
Load Avg: 2.57, 1.82, 1.83  CPU usage: 16.42% user, 3.51% sys, 80.6% idle
SharedLibs: 86M resident, 0B data, 17M linkedit.
MemRegions: 45514 total, 2574M resident, 85M private, 2412M shared.
PhysMem: 1580M wired, 4384M active, 2169M inactive, 8133M used, 50M free.
VM: 228G vsize, 1340M framework vsize, 7057334(1) pageins, 13221(0) pageouts.
Networks: packets: 11930161/3917M in, 23417006/21G out. Disks: 3540747/54G read, 4013078/128G written.

PID    COMMAND      %CPU      TIME     #TH   #WQ  #PORT #MREGS RPRVT  RSHRD  RSIZE  VPRVT  VSIZE  PGRP
29798  screencaptur 0.1       00:00.05 2     1    46    116    812K+  15M+   3584K+ 8508K- 2469M  652
29796  xpchelper    0.0       00:00.02 2     2    38    54     1308K  220K   4868K  77M    2437M  29796
29794  top          9.0       00:04.82 1/1   0    30-   33     1992K  216K   2696K  18M    2378M  29794
29763  bash         0.0       00:00.04 1     0    20    24     1344K  808K   2088K  17M    2378M  29763
29762  login        0.0       00:00.02 2     1    33    75     1016K  268K   2364K  64M    2433M  29762
29739  taskgated    0.0       00:00.01 2     0    31    44     440K   304K   1812K  29M    2390M  29739
29738- ded-0.7.1    99.3      01:36.50 1/1   0    17    24     292K   212K   900K   17M    586M   29738
29729  quicklookd   0.0       00:00.16 4     1    71    113    4432K  10M    10M    103M   2983M  29729
29609  cookied      0.0       00:00.00 2     1    40    54     548K   224K   1272K  41M    2410M  29609
```

- Ded - check!

Thursday, July 26, 12

Adventures in Anti-Analysis: Pushing the Bounds



```
Method 32 (0x20):
public void
dont.decompile.me.Unused.<init>()
this = v0
const/4                                v1, 0
if-eqz                                 v1, loc_181A+2
```

```
loc_181A:
const-string                    this, word_0
invoke-direct                   {this}, <void Object.<init>() imp. @ _def_Object__init_@V>

locret:
return-void
Method End
```

```
# |                   License info: 48-3359-7534-DF                        |
# |                    Tim Strazzere, Lookout Inc.                         |
# +------------------------------------------------------------------------+
#
# Input MD5    : 9D35F15587F893B4BD56E42B99011EB8
# Input CRC32 : 454C5B81


               DEX Module, Interface version 7
               Input Dex File version 35
```

- IDA Pro - sort of worked?
  Pointing at the header for what string to load

Adventures in Anti-Analysis: Pushing the Bounds

- Not hard to fix - don’t blindly follow an index that doesn’t exist
- loading index > table size ? alert!
- No real (good) reason someone to do this legitimately - if seen, clearly should look at that file!
- This attack is applicable for essentially all index references to the tables (use against type, etc as well)

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Fighting the Decompilers

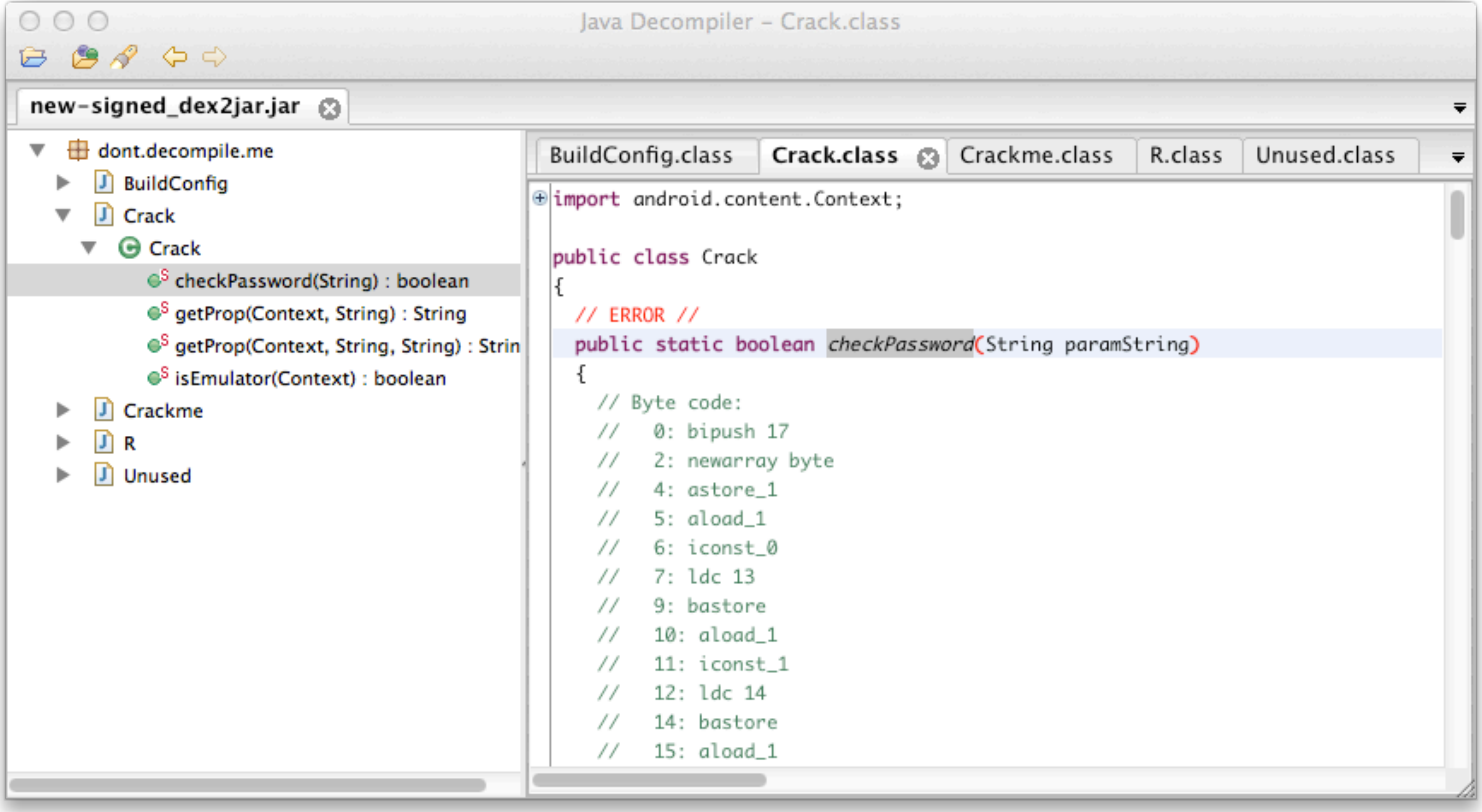
- What about dex2jar?
- Seems popular, so many people talking about using it to leverage other java tools / JD-Gui / JAD
- As a clarification - dex2jar does *not* give you the exact source back, it's a java representation of Dalvik optimized code
- This can result in "odd" java code when attempting to view with JAD/JD-Gui

Adventures in Anti-Analysis: Fighting the Decompilers

- JAD / JD-Gui works (roughly) by looking for known Java patterns
- dx (Dalivk compiler) does not use the same patterns that Java might have been using
- What does this result in?

lookout MOBILE SECURITY

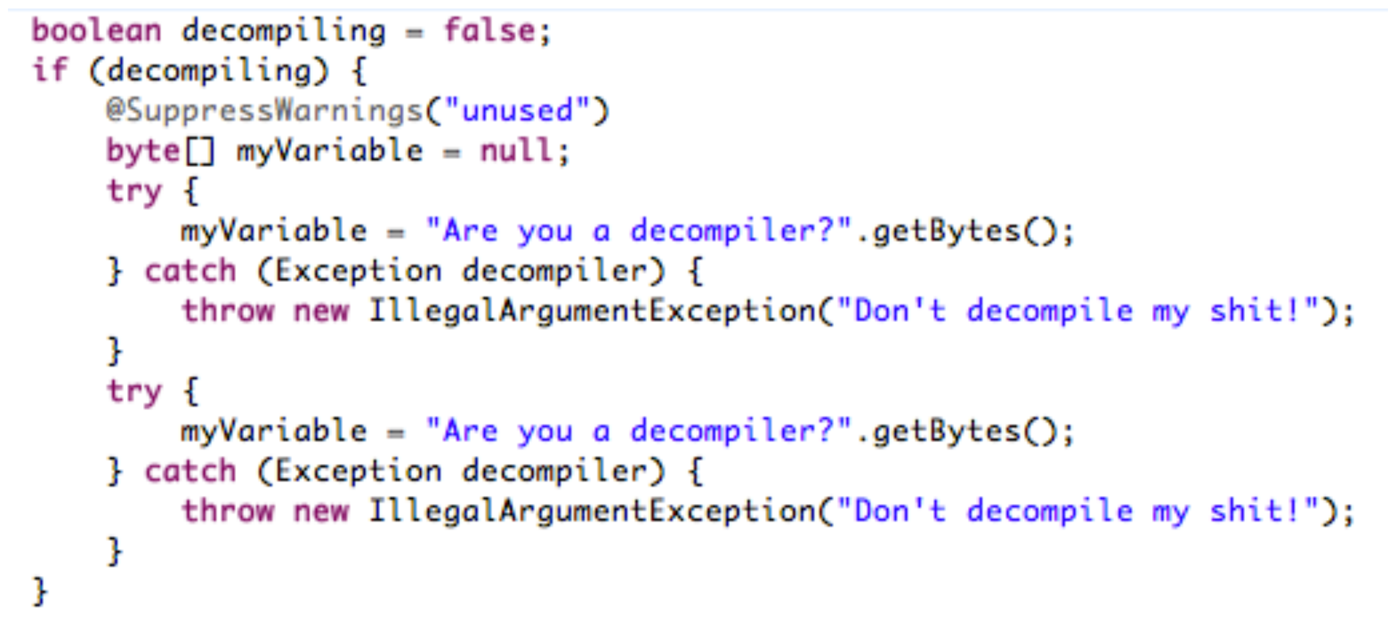
Adventures in Anti-Analysis: Fighting the Decompilers


Java Decompiler – Crack.class
new-signed_dex2jar.jar
dont.decompile.me
BuildConfig
Crack
Crack
checkPassword(String) : boolean
getProp(Context, String) : String
getProp(Context, String, String) : Strin
isEmulator(Context) : boolean
Crackme
R
Unused
BuildConfig.class
Crack.class
Crackme.class
R.class
Unused.class

```
import android.content.Context;

public class Crack
{
  // ERROR //
  public static boolean checkPassword(String paramString)
  {
    // Byte code:
    //   0: bipush 17
    //   2: newarray byte
    //   4: astore_1
    //   5: aload_1
    //   6: iconst_0
    //   7: ldc 13
    //   9: bastore
    //   10: aload_1
    //   11: iconst_1
    //   12: ldc 14
    //   14: bastore
    //   15: aload_1
```

Adventures in Anti-Analysis: Fighting the Decompilers

- Confusion via exceptions!



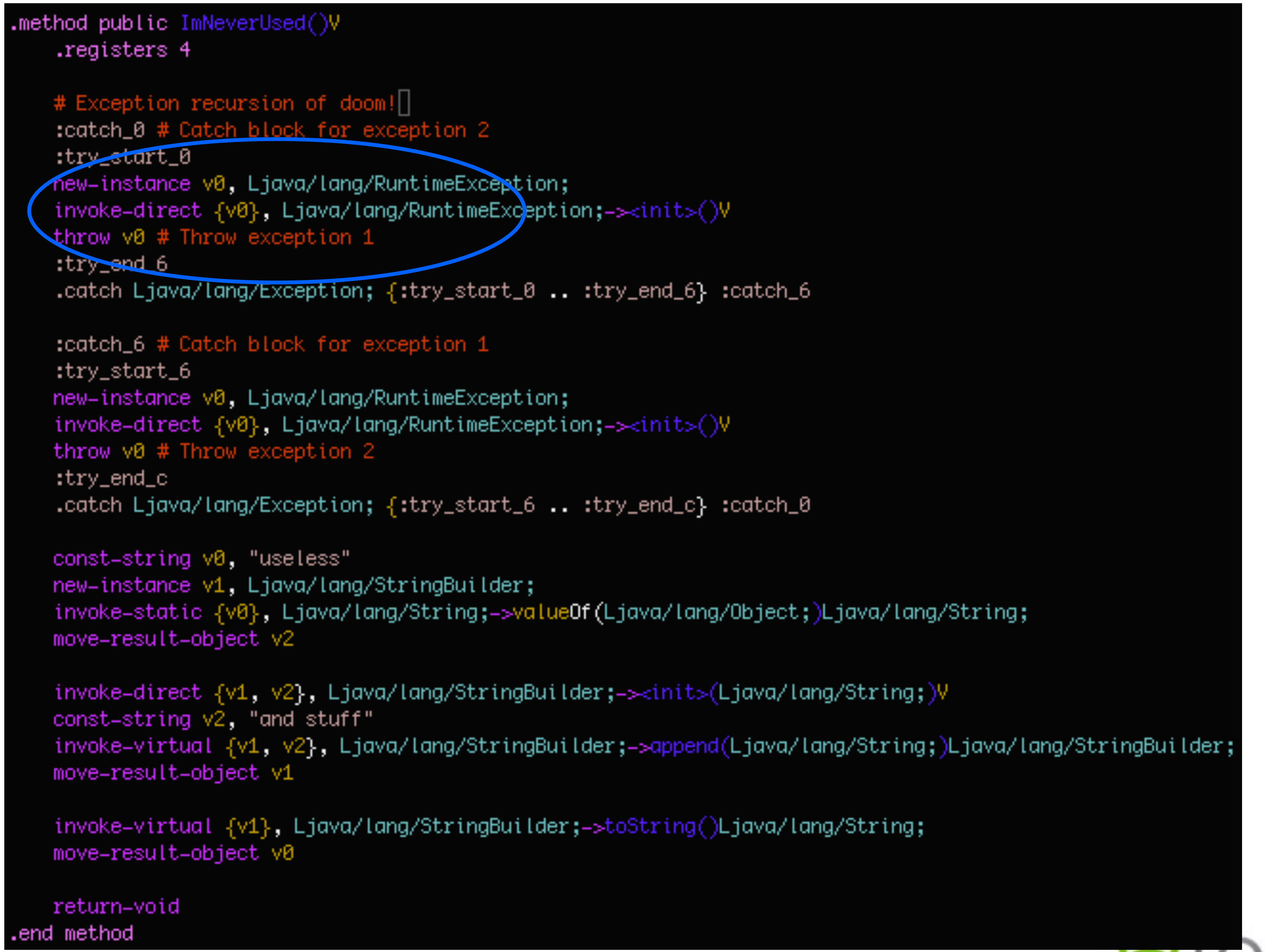
```
boolean decompiling = false;
if (decompiling) {
    @SuppressWarnings("unused")
    byte[] myVariable = null;
    try {
        myVariable = "Are you a decompiler?".getBytes();
    } catch (Exception decompiler) {
        throw new IllegalArgumentException("Don't decompile my shit!");
    }
    try {
        myVariable = "Are you a decompiler?".getBytes();
    } catch (Exception decompiler) {
        throw new IllegalArgumentException("Don't decompile my shit!");
    }
}
```

- Causes JD-Gui to explode on any method - JAD decompiles it fine though

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Fighting the Decompilers

- More exceptional fun! (example lifted from JF)

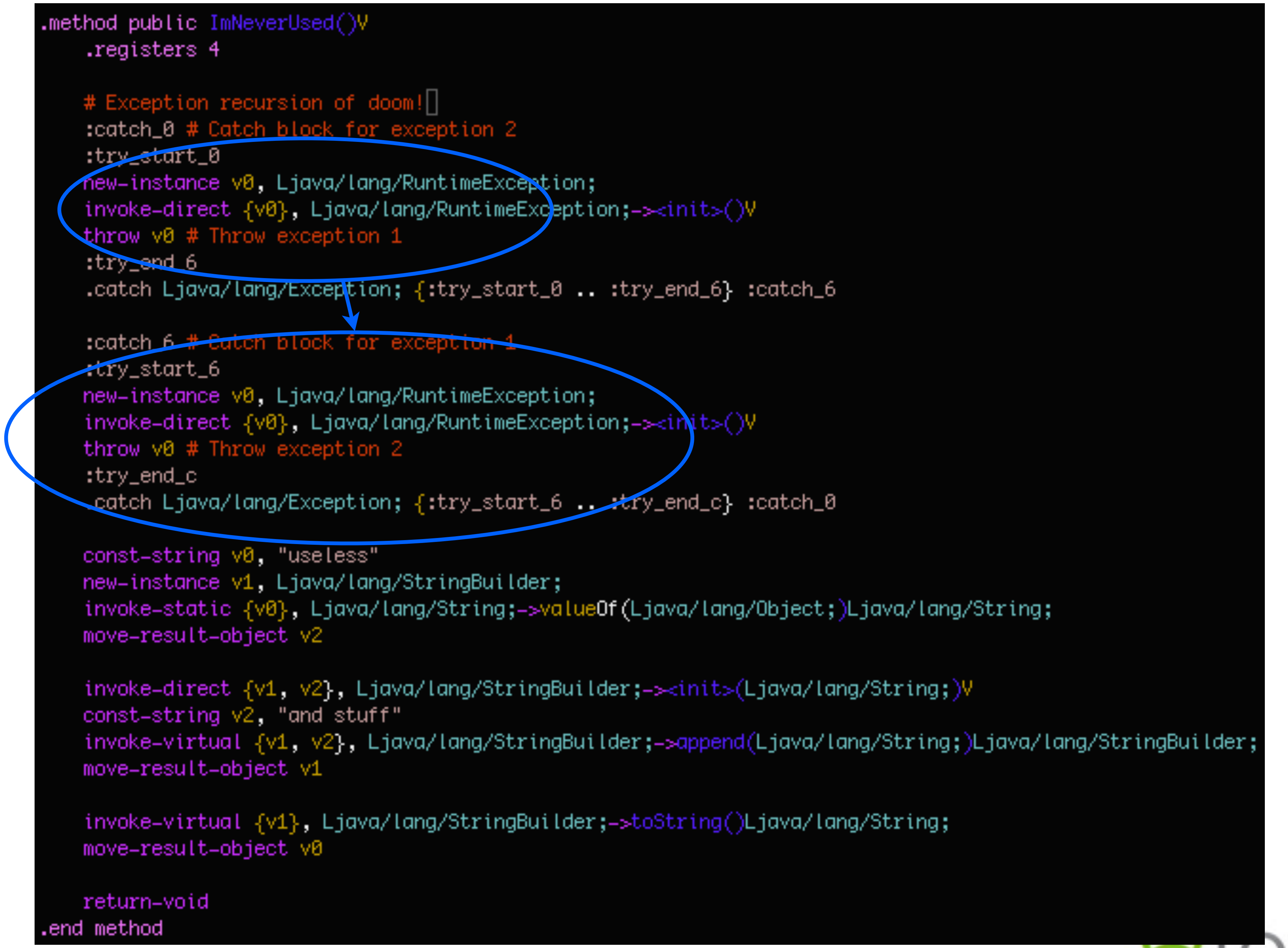


```
.method public ImNeverUsed()V
    .registers 4

    # Exception recursion of doom!
    :catch_0 # Catch block for exception 2
    :try_start_0
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 1
    :try_end_6
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_6} :catch_6

    :catch_6 # Catch block for exception 1
    :try_start_6
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 2
    :try_end_c
    .catch Ljava/lang/Exception; {:try_start_6 .. :try_end_c} :catch_0

    const-string v0, "useless"
    new-instance v1, Ljava/lang/StringBuilder;
    invoke-static {v0}, Ljava/lang/String;->valueOf(Ljava/lang/Object;)Ljava/lang/String;
    move-result-object v2

    invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V
    const-string v2, "and stuff"
    invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
    move-result-object v1

    invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
    move-result-object v0

    return-void
.end method
```

lookout MOBILE SECURITY

Adventures in Anti-Analysis: Fighting the Decompilers

- More exceptional fun! (example lifted from JF)

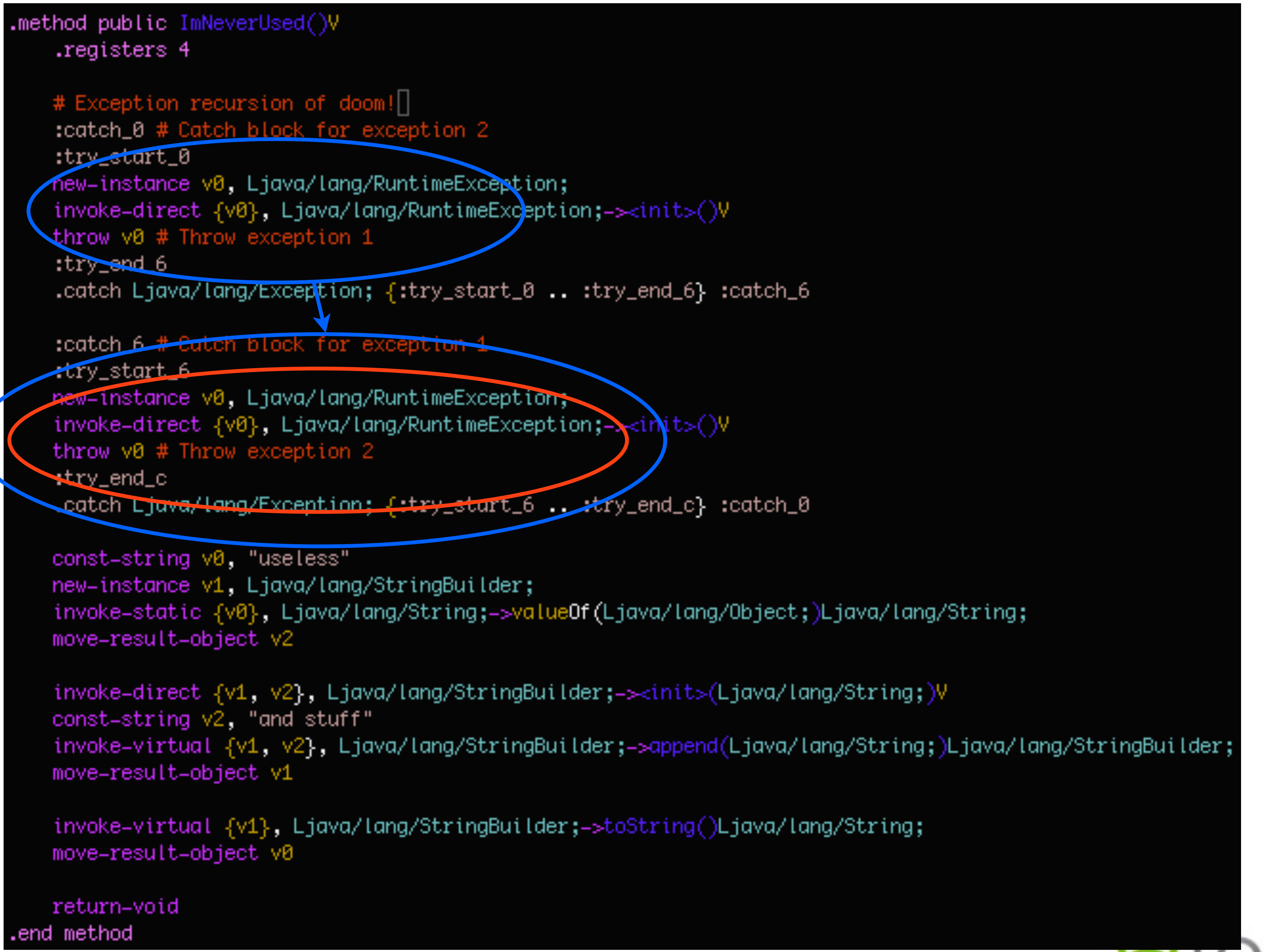


```
.method public ImNeverUsed()V
    .registers 4

    # Exception recursion of doom!
    :catch_0 # Catch block for exception 2
    :try_start_0
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 1
    :try_end_6
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_6} :catch_6

    :catch_6 # Catch block for exception 1
    :try_start_6
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 2
    :try_end_c
    .catch Ljava/lang/Exception; {:try_start_6 .. :try_end_c} :catch_0

    const-string v0, "useless"
    new-instance v1, Ljava/lang/StringBuilder;
    invoke-static {v0}, Ljava/lang/String;->valueOf(Ljava/lang/Object;)Ljava/lang/String;
    move-result-object v2

    invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V
    const-string v2, "and stuff"
    invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
    move-result-object v1

    invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
    move-result-object v0

    return-void
.end method
```

lookout™
MOBILE SECURITY

Adventures in Anti-Analysis: Fighting the Decompilers

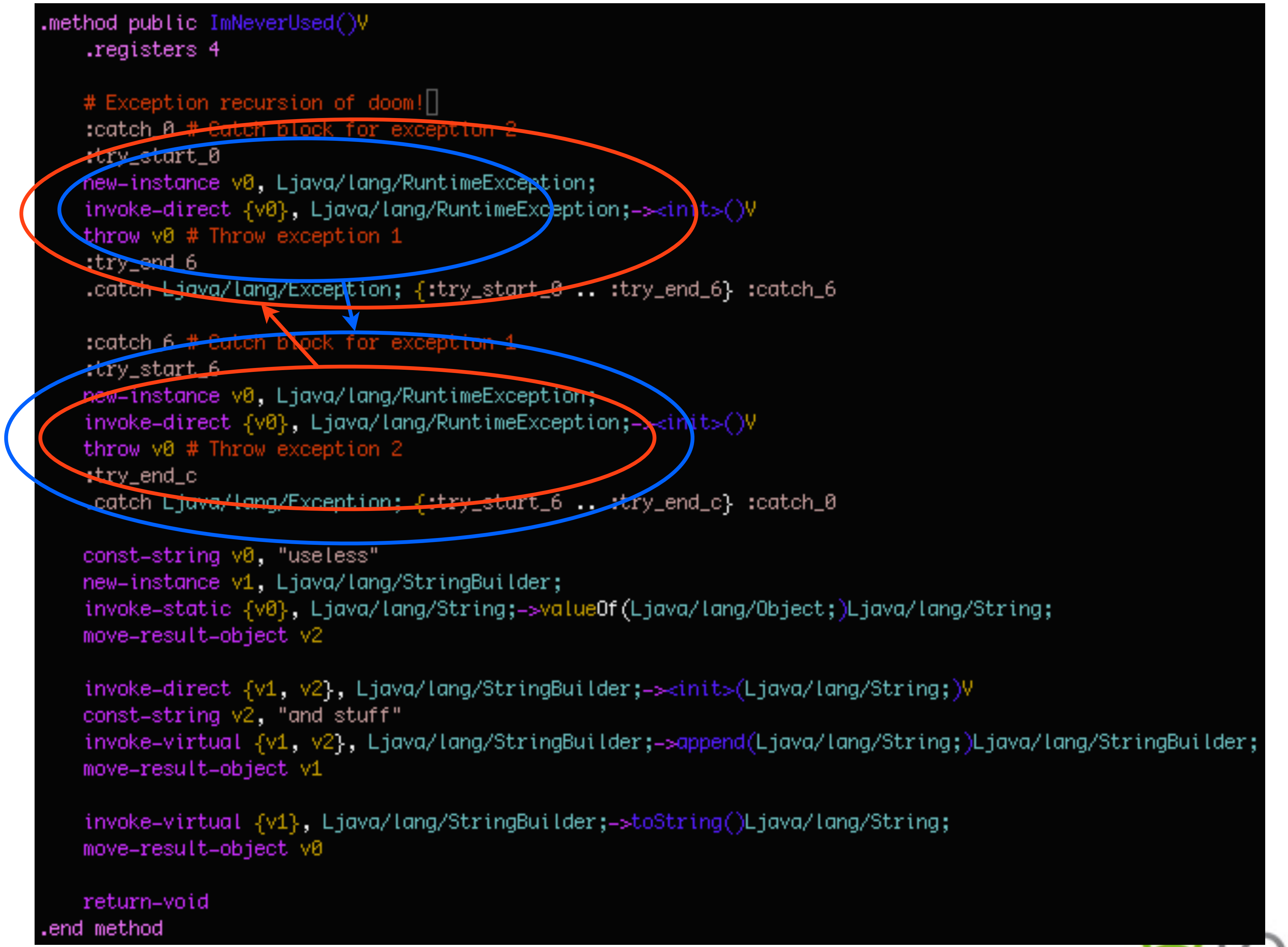
- More exceptional fun! (example lifted from JF)



```
.method public ImNeverUsed()V
    .registers 4

    # Exception recursion of doom!
    :catch_0 # Catch block for exception 2
    :try_start_0
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 1
    :try_end_6
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_6} :catch_6

    :catch_6 # Catch block for exception 1
    :try_start_6
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 2
    :try_end_c
    .catch Ljava/lang/Exception; {:try_start_6 .. :try_end_c} :catch_0

    const-string v0, "useless"
    new-instance v1, Ljava/lang/StringBuilder;
    invoke-static {v0}, Ljava/lang/String;->valueOf(Ljava/lang/Object;)Ljava/lang/String;
    move-result-object v2

    invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V
    const-string v2, "and stuff"
    invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
    move-result-object v1

    invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
    move-result-object v0

    return-void
.end method
```

Adventures in Anti-Analysis: Fighting the Decompilers

- More exceptional fun! (example lifted from JF)



```
.method public ImNeverUsed()V
    .registers 4

    # Exception recursion of doom!
    :catch_0 # Catch block for exception 2
    :try_start_0
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 1
    :try_end_6
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_6} :catch_6

    :catch_6 # Catch block for exception 1
    :try_start_6
    new-instance v0, Ljava/lang/RuntimeException;
    invoke-direct {v0}, Ljava/lang/RuntimeException;-><init>()V
    throw v0 # Throw exception 2
    :try_end_c
    .catch Ljava/lang/Exception; {:try_start_6 .. :try_end_c} :catch_0

    const-string v0, "useless"
    new-instance v1, Ljava/lang/StringBuilder;
    invoke-static {v0}, Ljava/lang/String;->valueOf(Ljava/lang/Object;)Ljava/lang/String;
    move-result-object v2

    invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V
    const-string v2, "and stuff"
    invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
    move-result-object v1

    invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
    move-result-object v0

    return-void
.end method
```

Adventures in Anti-Analysis: Fighting the Decompilers

- This is clearly not valid code, impossible to write in java
- This *is* valid dalvik code - not something you want to run though
- However if this is gated is "dead code", it's legitimate to have inside a class
- Results of this are fun!

lookout™ MOBILE SECURITY

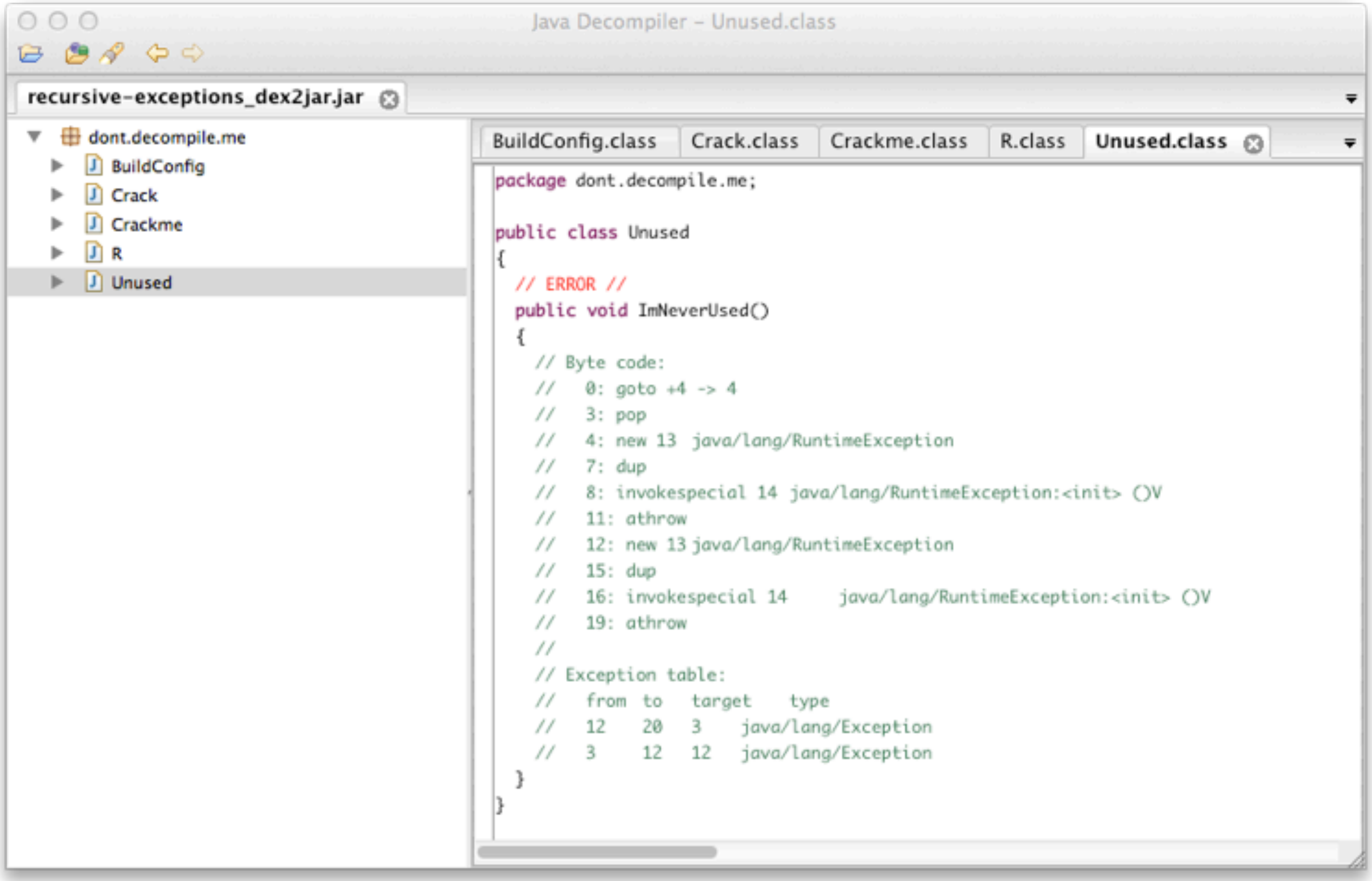
Adventures in Anti-Analysis: Fighting the Decompilers

```
champagne:recursive-exceptions tstrazzere$ jad Unused.class
Parsing Unused.class... Generating Unused.jad
Couldn't fully decompile method ImNeverUsed
Couldn't resolve all exception handlers in method ImNeverUsed
champagne: recursive-exceptions tstrazzere$ cat Unused.jad
// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov.
// Jad home page: http://www.kpdus.com/jad.html
// Decompiler options: packimports(3)

package dont.decompile.me;

public class Unused
{

    public Unused()
    {
    }

    public void ImNeverUsed()
    {
        JVM INSTR pop ;
        throw new RuntimeException();
        throw new RuntimeException();
    }
}
```

- JAD, check!

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Fighting the Decompilers

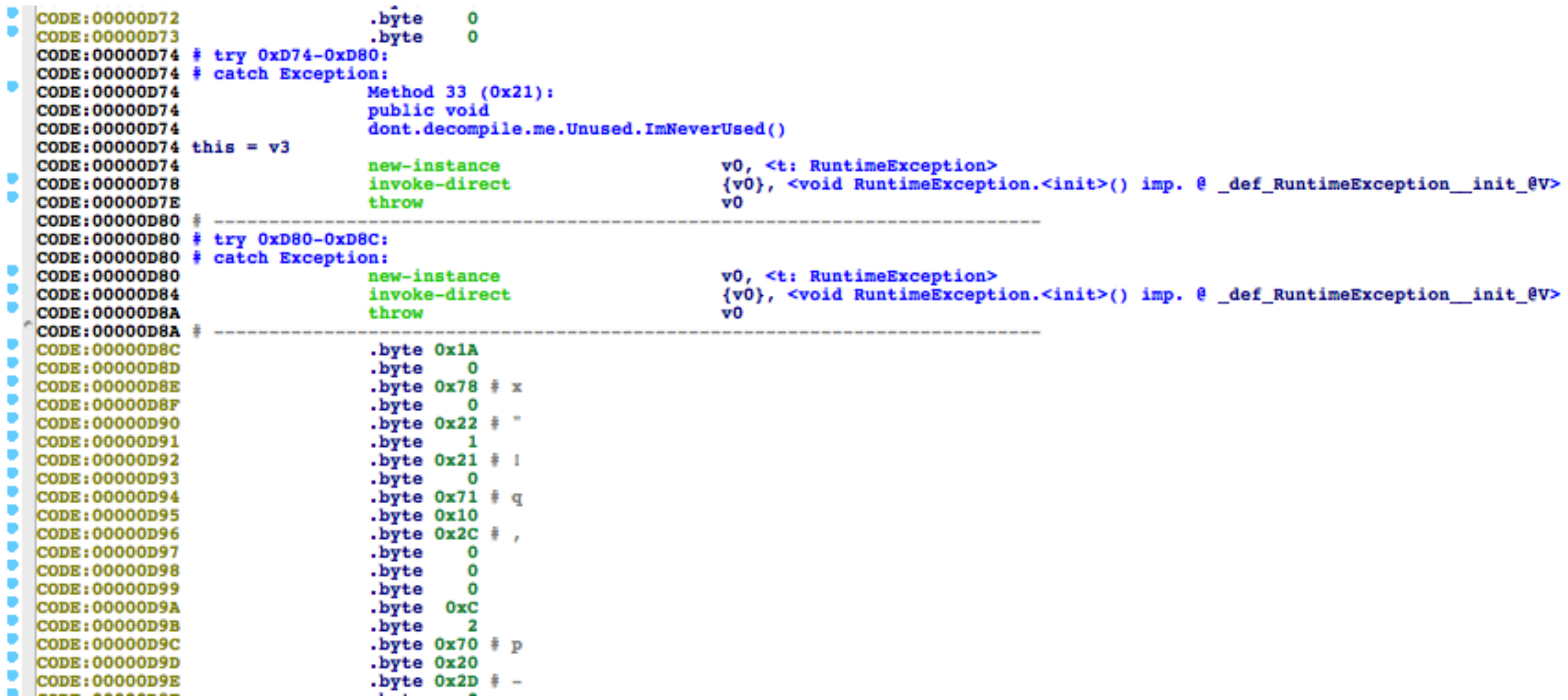


```
package dont.decompile.me;

public class Unused
{
  // ERROR //
  public void ImNeverUsed()
  {
    // Byte code:
    //   0: goto +4 -> 4
    //   3: pop
    //   4: new 13  java/lang/RuntimeException
    //   7: dup
    //   8: invokespecial 14  java/lang/RuntimeException:<init> ()V
    //   11: athrow
    //   12: new 13 java/lang/RuntimeException
    //   15: dup
    //   16: invokespecial 14     java/lang/RuntimeException:<init> ()V
    //   19: athrow
    //
    // Exception table:
    //   from  to   target    type
    //   12    20   3     java/lang/Exception
    //   3     12   12    java/lang/Exception
  }
}
```

- JD-Gui, check!

lookout™ MOBILE SECURITY

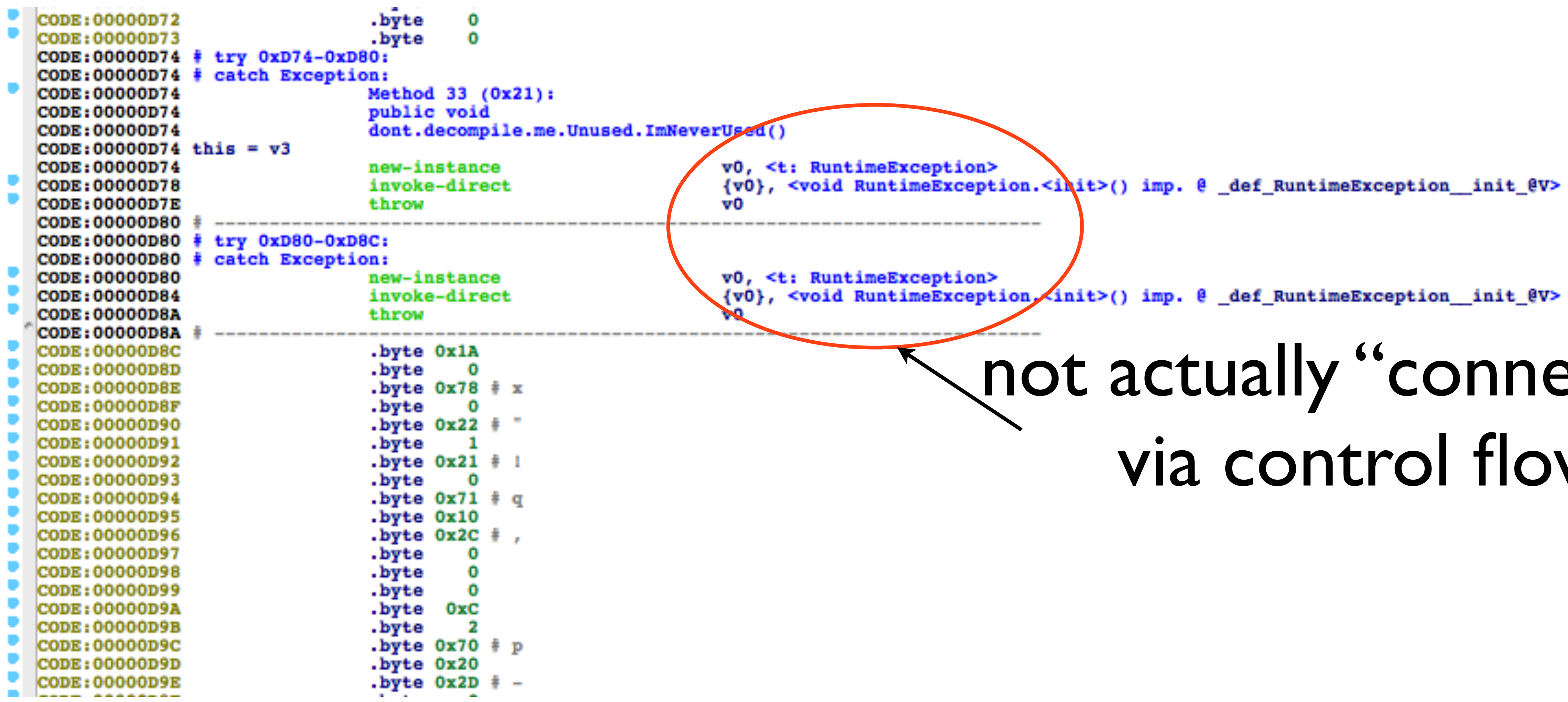
Adventures in Anti-Analysis: Fighting the Decompilers



```
CODE:00000D72                        .byte    0
CODE:00000D73                        .byte    0
CODE:00000D74 # try 0xD74-0xD80:
CODE:00000D74 # catch Exception:
CODE:00000D74                        Method 33 (0x21):
CODE:00000D74                        public void
CODE:00000D74                        dont.decompile.me.Unused.ImNeverUsed()
CODE:00000D74 this = v3
CODE:00000D74                        new-instance                   v0, <t: RuntimeException>
CODE:00000D78                        invoke-direct                  {v0}, <void RuntimeException.<init>() imp. @ _def_RuntimeException__init_@V>
CODE:00000D7E                        throw                          v0
CODE:00000D80 # ---------------------------------------------------------------------------
CODE:00000D80 # try 0xD80-0xD8C:
CODE:00000D80 # catch Exception:
CODE:00000D80                        new-instance                   v0, <t: RuntimeException>
CODE:00000D84                        invoke-direct                  {v0}, <void RuntimeException.<init>() imp. @ _def_RuntimeException__init_@V>
CODE:00000D8A                        throw                          v0
CODE:00000D8A # ---------------------------------------------------------------------------
CODE:00000D8C                        .byte 0x1A
CODE:00000D8D                        .byte    0
CODE:00000D8E                        .byte 0x78 # x
CODE:00000D8F                        .byte    0
CODE:00000D90                        .byte 0x22 # "
CODE:00000D91                        .byte    1
CODE:00000D92                        .byte 0x21 # !
CODE:00000D93                        .byte    0
CODE:00000D94                        .byte 0x71 # q
CODE:00000D95                        .byte 0x10
CODE:00000D96                        .byte 0x2C # ,
CODE:00000D97                        .byte    0
CODE:00000D98                        .byte    0
CODE:00000D99                        .byte    0
CODE:00000D9A                        .byte  0xC
CODE:00000D9B                        .byte    2
CODE:00000D9C                        .byte 0x70 # p
CODE:00000D9D                        .byte 0x20
CODE:00000D9E                        .byte 0x2D # -
```

- IDA Pro - check!

Adventures in Anti-Analysis: Fighting the Decompilers



```
CODE:00000D72                    .byte    0
CODE:00000D73                    .byte    0
CODE:00000D74 # try 0xD74-0xD80:
CODE:00000D74 # catch Exception:
CODE:00000D74                    Method 33 (0x21):
CODE:00000D74                    public void
CODE:00000D74                    dont.decompile.me.Unused.ImNeverUsed()
CODE:00000D74 this = v3
CODE:00000D74                    new-instance                v0, <t: RuntimeException>
CODE:00000D78                    invoke-direct               {v0}, <void RuntimeException.<init>() imp. @ _def_RuntimeException__init_@V>
CODE:00000D7E                    throw                       v0
CODE:00000D80 # ------------------------------------------------------------------------
CODE:00000D80 # try 0xD80-0xD8C:
CODE:00000D80 # catch Exception:
CODE:00000D80                    new-instance                v0, <t: RuntimeException>
CODE:00000D84                    invoke-direct               {v0}, <void RuntimeException.<init>() imp. @ _def_RuntimeException__init_@V>
CODE:00000D8A                    throw                       v0
CODE:00000D8A # ------------------------------------------------------------------------
CODE:00000D8C                    .byte 0x1A
CODE:00000D8D                    .byte    0
CODE:00000D8E                    .byte 0x78 # x
CODE:00000D8F                    .byte    0
CODE:00000D90                    .byte 0x22 # "
CODE:00000D91                    .byte    1
CODE:00000D92                    .byte 0x21 # !
CODE:00000D93                    .byte    0
CODE:00000D94                    .byte 0x71 # q
CODE:00000D95                    .byte 0x10
CODE:00000D96                    .byte 0x2C # ,
CODE:00000D97                    .byte    0
CODE:00000D98                    .byte    0
CODE:00000D99                    .byte    0
CODE:00000D9A                    .byte  0xC
CODE:00000D9B                    .byte    2
CODE:00000D9C                    .byte 0x70 # p
CODE:00000D9D                    .byte 0x20
CODE:00000D9E                    .byte 0x2D # -
```

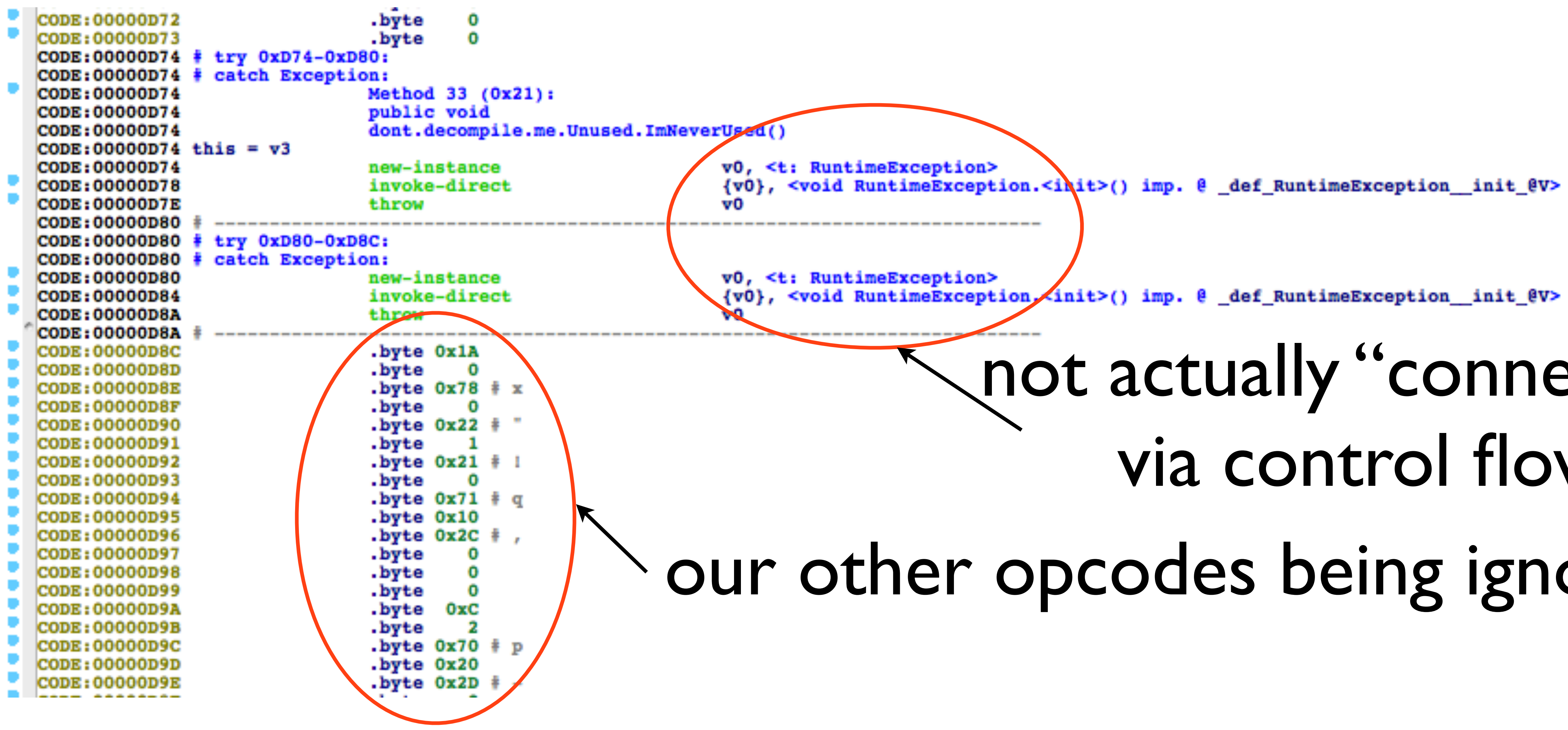
not actually "connected" via control flow

- Control flow garbage FTW!

lookout MOBILE SECURITY

Thursday, July 26, 12

Adventures in Anti-Analysis: Fighting the Decompilers



```
CODE:00000D72                          .byte    0
CODE:00000D73                          .byte    0
CODE:00000D74 # try 0xD74-0xD80:
CODE:00000D74 # catch Exception:
CODE:00000D74                          Method 33 (0x21):
CODE:00000D74                          public void
CODE:00000D74                          dont.decompile.me.Unused.ImNeverUsed()
CODE:00000D74 this = v3
CODE:00000D74                          new-instance                    v0, <t: RuntimeException>
CODE:00000D78                          invoke-direct                   {v0}, <void RuntimeException.<init>() imp. @ _def_RuntimeException__init_@V>
CODE:00000D7E                          throw                           v0
CODE:00000D80 # ---------------------------------------------------------------------------
CODE:00000D80 # try 0xD80-0xD8C:
CODE:00000D80 # catch Exception:
CODE:00000D80                          new-instance                    v0, <t: RuntimeException>
CODE:00000D84                          invoke-direct                   {v0}, <void RuntimeException.<init>() imp. @ _def_RuntimeException__init_@V>
CODE:00000D8A                          throw                           v0
CODE:00000D8A # ---------------------------------------------------------------------------
CODE:00000D8C                          .byte 0x1A
CODE:00000D8D                          .byte    0
CODE:00000D8E                          .byte 0x78 # x
CODE:00000D8F                          .byte    0
CODE:00000D90                          .byte 0x22 # "
CODE:00000D91                          .byte    1
CODE:00000D92                          .byte 0x21 # !
CODE:00000D93                          .byte    0
CODE:00000D94                          .byte 0x71 # q
CODE:00000D95                          .byte 0x10
CODE:00000D96                          .byte 0x2C # ,
CODE:00000D97                          .byte    0
CODE:00000D98                          .byte    0
CODE:00000D99                          .byte    0
CODE:00000D9A                          .byte  0xC
CODE:00000D9B                          .byte    2
CODE:00000D9C                          .byte 0x70 # p
CODE:00000D9D                          .byte 0x20
CODE:00000D9E                          .byte 0x2D # -
```

not actually "connected" via control flow

our other opcodes being ignored

- Confused as to what's going on after the exceptions

Adventures in Anti-Analysis: Fighting the Decompilers

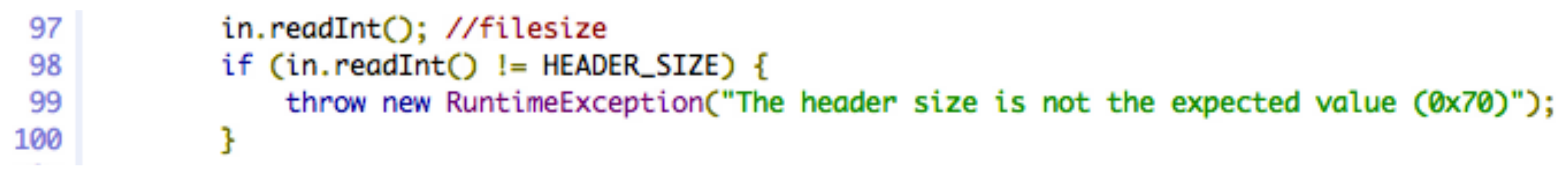
- Harder to detect automatically (could use pattern matching?)
- Might not matter for automated tools (???)
- Just makes it a pain for people using these tools manually to reverse (could be a big win depending on your goal)

Adventures in Anti-Analysis: Slightly Newer School

- We saw baksmali died on the LinkLocked section
- Anything else a quick win like that?

Adventures in Anti-Analysis: Slightly Newer School

- We saw baksmali died on the LinkLocked section
- Anything else a quick win like that?

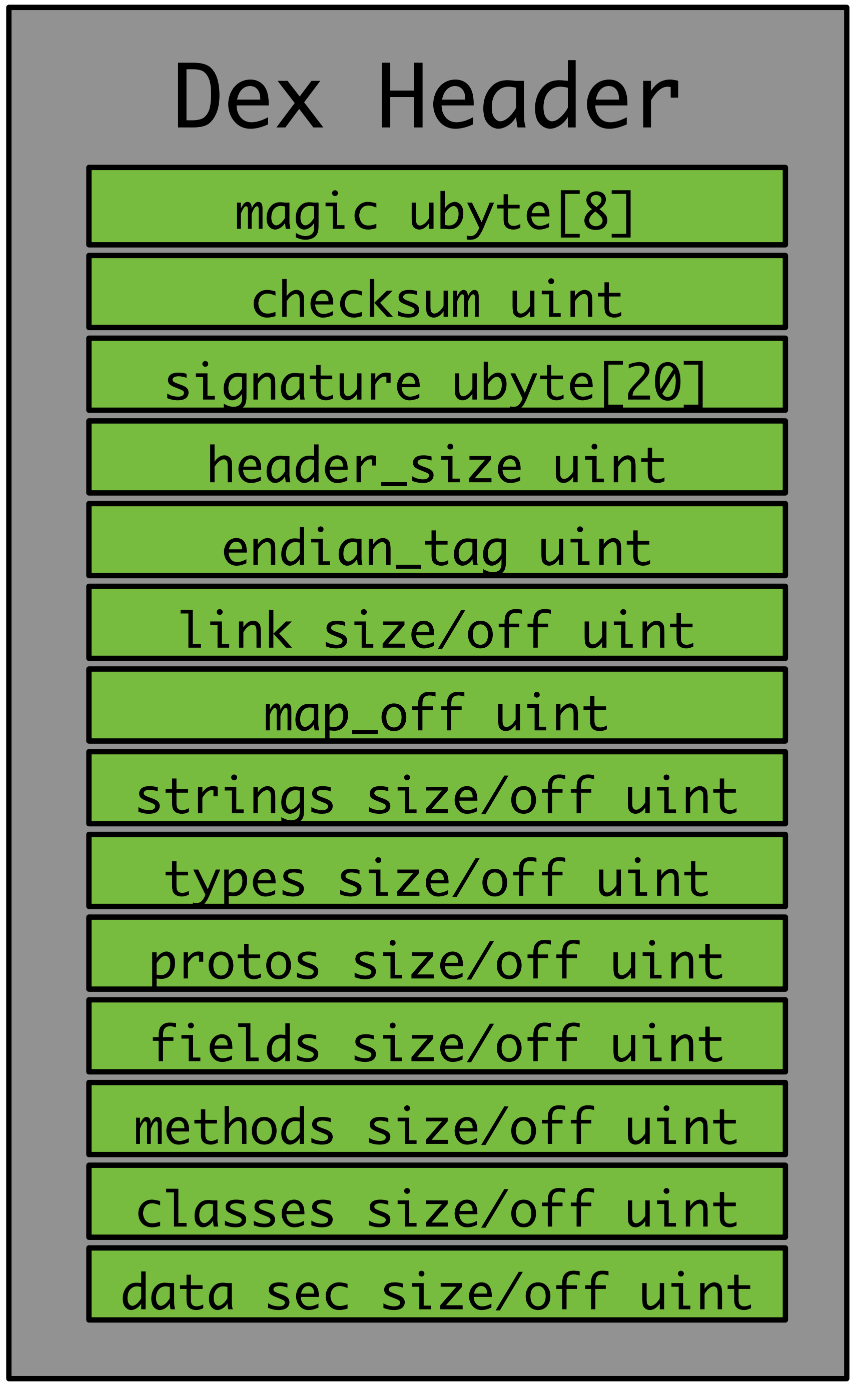


```
        in.readInt(); //filesize
        if (in.readInt() != HEADER_SIZE) {
            throw new RuntimeException("The header size is not the expected value (0x70)");
        }
```

- Header size is open for a quick attack!

lookout MOBILE SECURITY

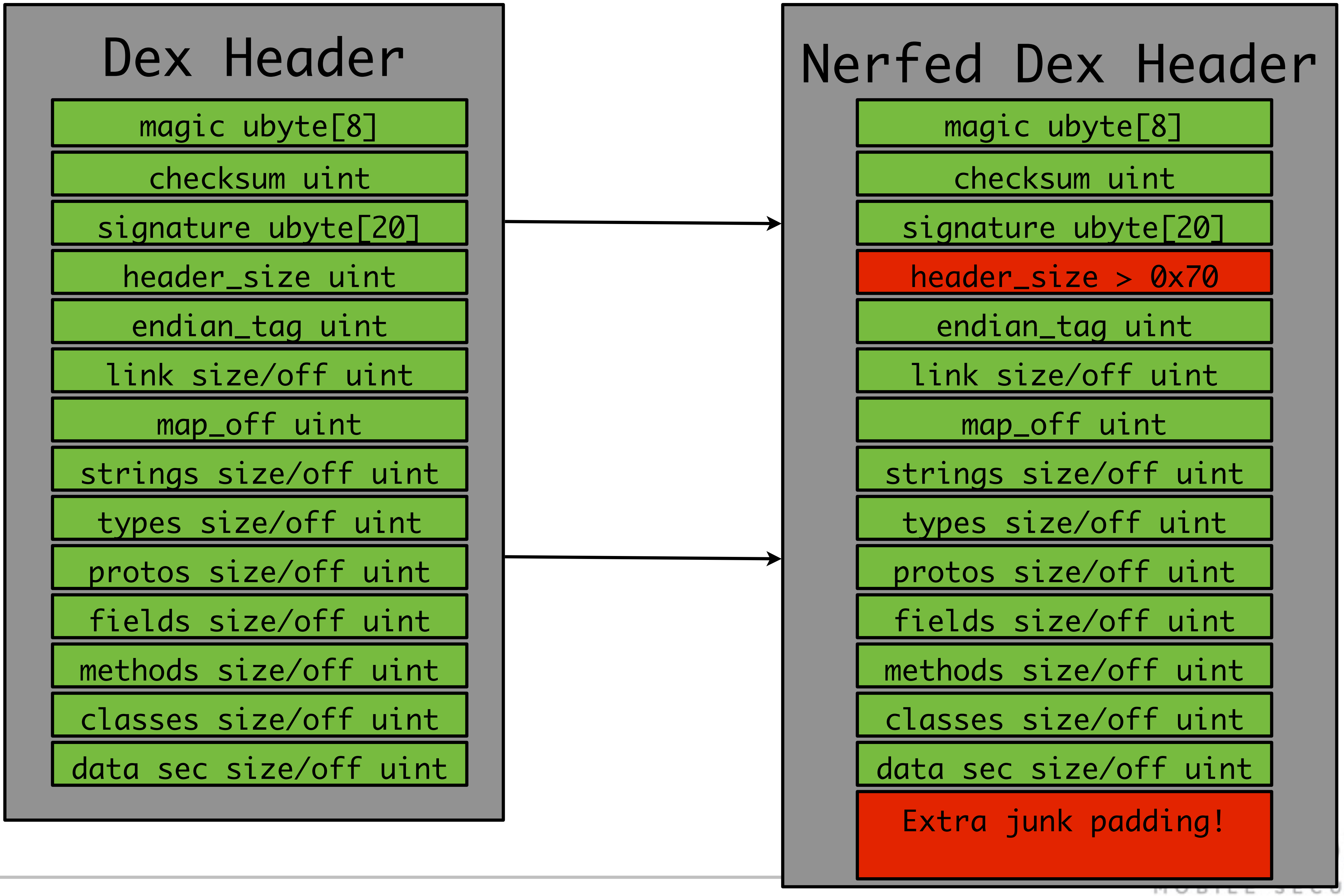
Adventures in Anti-Analysis: Slightly Newer School



- header_size is what we want
- Always currently 0x70 (112)
- Usage is for forward/backward compatibility
- Side effects: causes the verifier to skip over non-null bytes that are "inside" the header

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Slightly Newer School



Adventures in Anti-Analysis: Slightly Newer School

- Should be easy to implement
- Fix all the offsets in the header and it will be done

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Slightly Newer School

- Should be easy to implement
- Fix all the offsets in the header and it will be done

NOPE

Adventures in Anti-Analysis: Slightly Newer School

- ~~Should be easy to implement~~
- Fix all the offsets in the header ~~and it will be done~~
- Fix all the offsets inside every other structure, otherwise all items will be aligned

  i.e. - process every table and item linked in that table :\

  (PoC tool went from 100 lines to 2000+)

lookout MOBILE SECURITY

Adventures in Anti-Analysis: Slightly Newer School

- Fast forward to many hours later.... It works! But only against baksmali;

```
champagne:apkfuscator tstrazzere$ hexdump -C such_a_big_ego.dex | head
00000000  64 65 78 0a 30 33 35 00  a5 26 6f 22 c8 85 fc 5c  |dex.035..&o"...\|
00000010  83 be 45 21 d2 5c b9 9f  52 6a 0a 34 dc 55 19 f2  |..E!.\..Rj.4.U..|
00000020  71 19 00 00 78 00 00 00  78 56 34 12 00 00 00 00  |q...x...xV4.....|
00000030  00 00 00 00 18 08 00 00  8d 00 00 00 78 00 00 00  |............x...|
00000040  27 00 00 00 ac 02 00 00  1c 00 00 00 48 03 00 00  |'...........H...|
00000050  14 00 00 00 98 04 00 00  30 00 00 00 38 05 00 00  |........0...8...|
00000060  0b 00 00 00 b8 06 00 00  59 11 00 00 18 08 00 00  |........Y.......|
00000070  00 00 00 00 00 00 00 00  dc 08 00 00 df 08 00 00  |................|

champagne:apkfuscator tstrazzere$ baksmali such_a_big_ego.dex -o wontwork
UNEXPECTED TOP-LEVEL EXCEPTION:
org.jf.dexlib.Util.ExceptionWithContext: The header size is not the expected value (0x70)
        at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithContext.java:54)
        at org.jf.dexlib.Item.addExceptionContext(Item.java:176)
        at org.jf.dexlib.Item.readFrom(Item.java:78)
        at org.jf.dexlib.DexFile.<init>(DexFile.java:390)
        at org.jf.baksmali.main.main(main.java:265)
Caused by: java.lang.RuntimeException: The header size is not the expected value (0x70)
        at org.jf.dexlib.HeaderItem.readItem(HeaderItem.java:92)
        at org.jf.dexlib.Item.readFrom(Item.java:76)
        ... 2 more
header_item
```

lookout™ MOBILE SECURITY

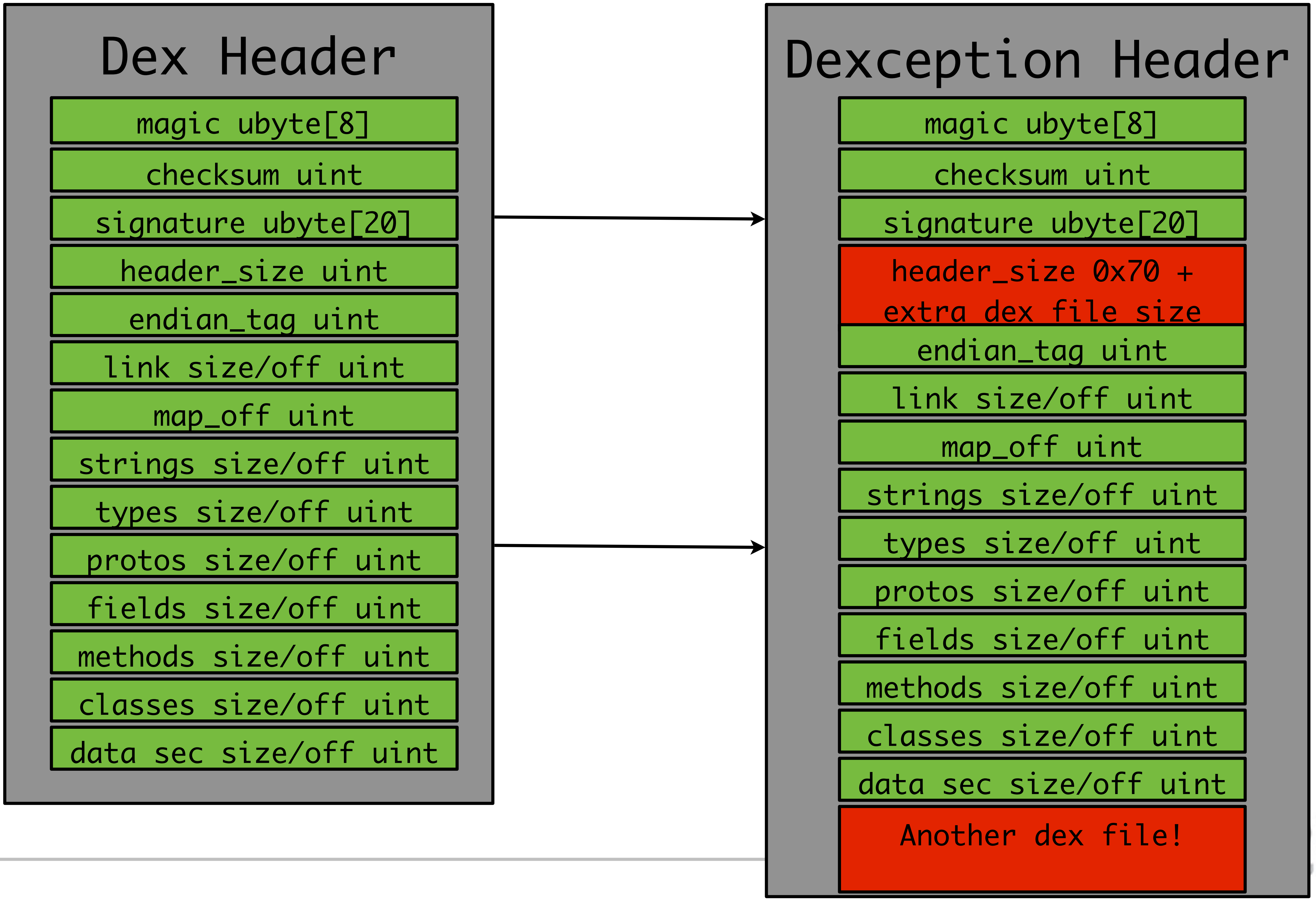
Adventures in Anti-Analysis: Slightly Newer School

- Great, but that is an easy fix, so what?
- Well, think of the possibilities this is data loaded into memory / dalvik-cache
- Hide data / resources inside the bloated header
- Hide a DEX file and load at runtime!

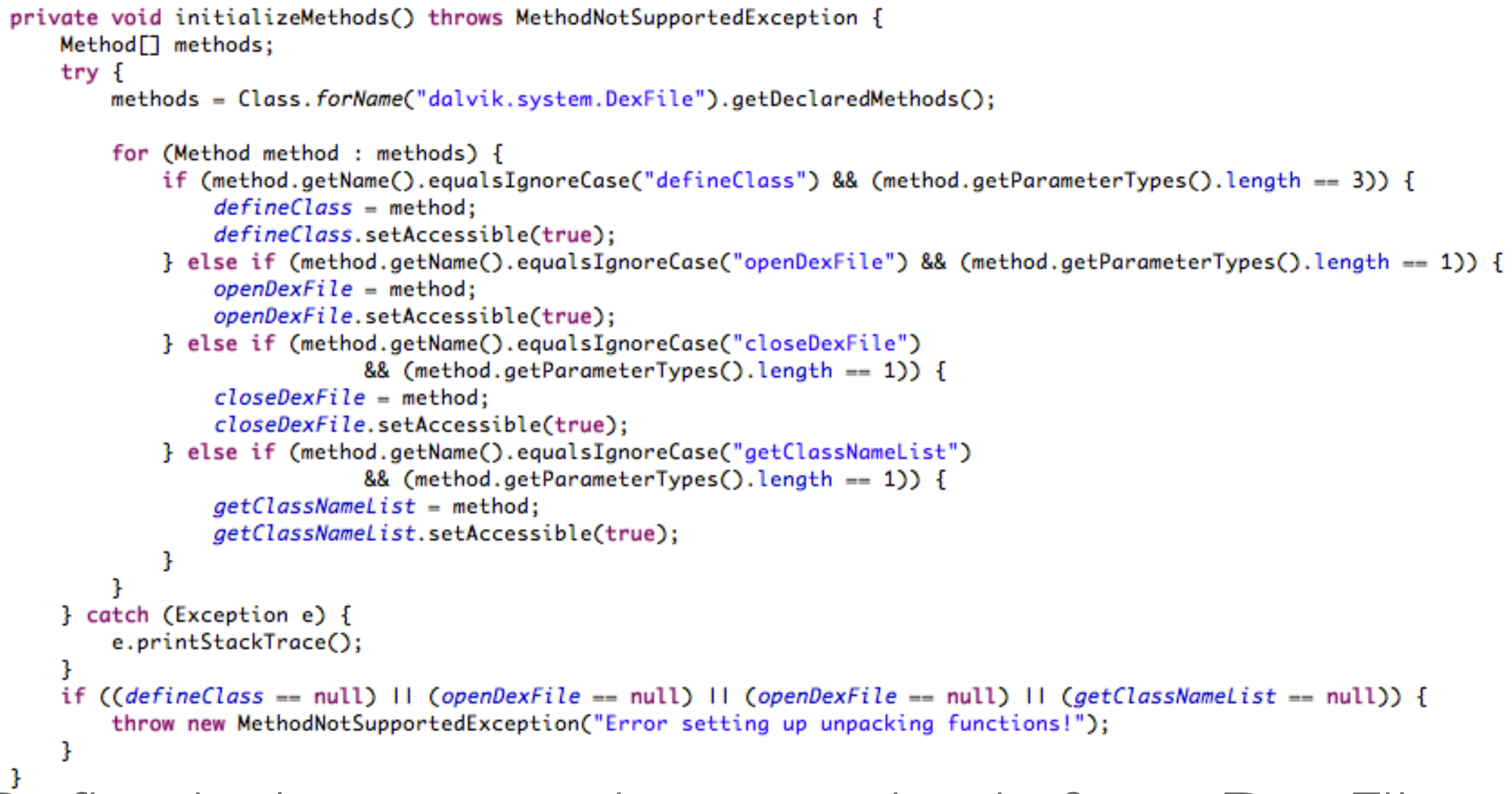
A dex inside a dex? Think of all the memes...

lookout™ MOBILE SECURITY

Adventures in Anti-Analysis: Dexception



Adventures in Anti-Analysis: Dexception

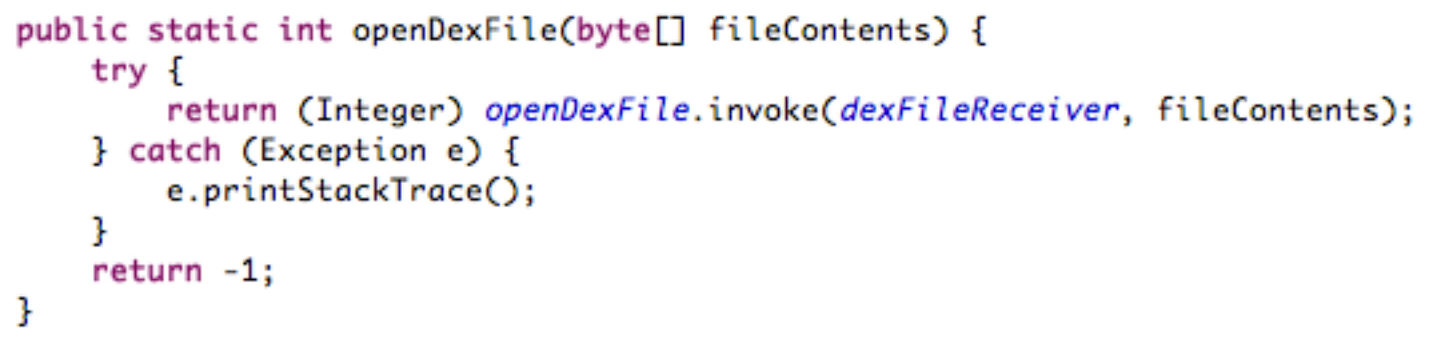


```
private void initializeMethods() throws MethodNotSupportedException {
    Method[] methods;
    try {
        methods = Class.forName("dalvik.system.DexFile").getDeclaredMethods();

        for (Method method : methods) {
            if (method.getName().equalsIgnoreCase("defineClass") && (method.getParameterTypes().length == 3)) {
                defineClass = method;
                defineClass.setAccessible(true);
            } else if (method.getName().equalsIgnoreCase("openDexFile") && (method.getParameterTypes().length == 1)) {
                openDexFile = method;
                openDexFile.setAccessible(true);
            } else if (method.getName().equalsIgnoreCase("closeDexFile")
                                && (method.getParameterTypes().length == 1)) {
                closeDexFile = method;
                closeDexFile.setAccessible(true);
            } else if (method.getName().equalsIgnoreCase("getClassNameList")
                                && (method.getParameterTypes().length == 1)) {
                getClassNameList = method;
                getClassNameList.setAccessible(true);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    if ((defineClass == null) || (openDexFile == null) || (openDexFile == null) || (getClassNameList == null)) {
        throw new MethodNotSupportedException("Error setting up unpacking functions!");
    }
}
```

- Reflectively access private methods from DexFile (ICS/JBean only for methods I wanted)

Adventures in Anti-Analysis: Dexception

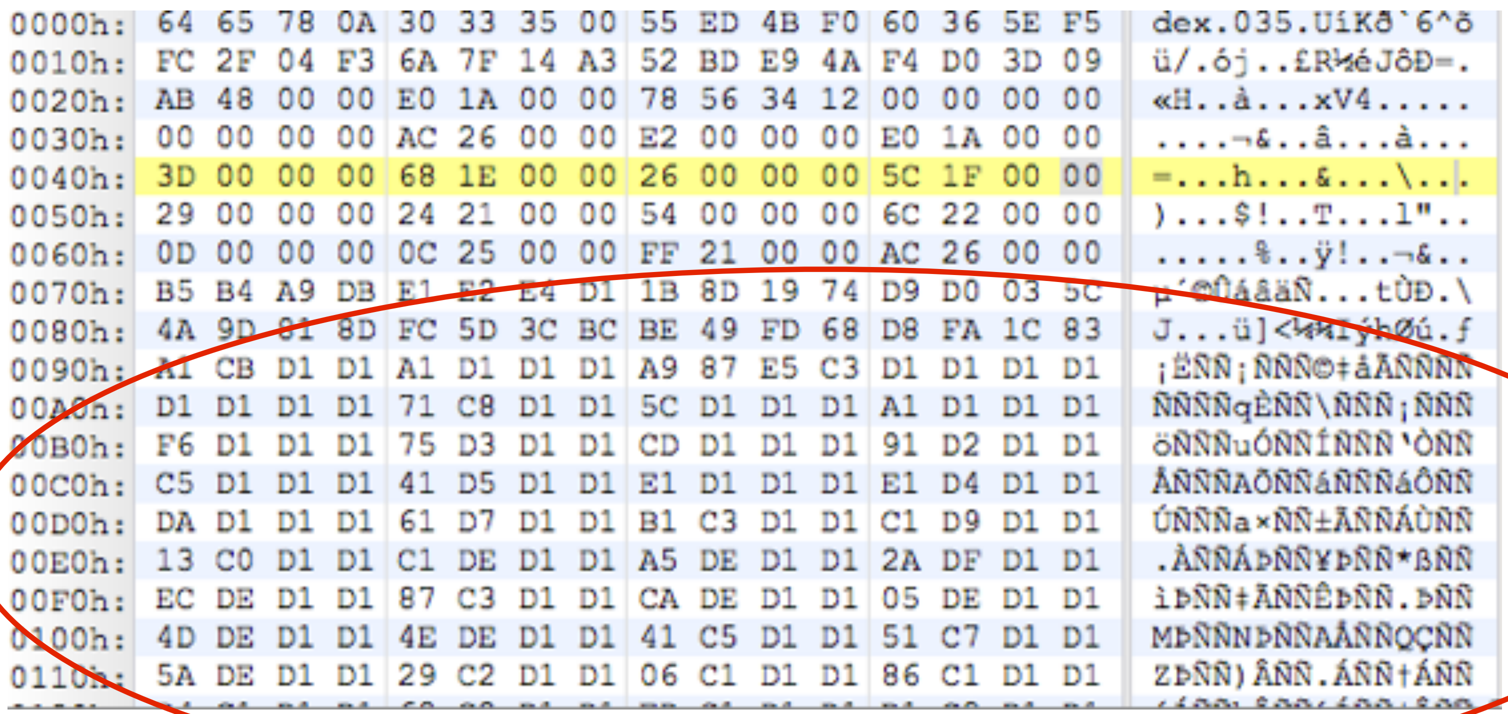
- The best method is actually openDexFile



```java
public static int openDexFile(byte[] fileContents) {
    try {
        return (Integer) openDexFile.invoke(dexFileReceiver, fileContents);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return -1;
}
```

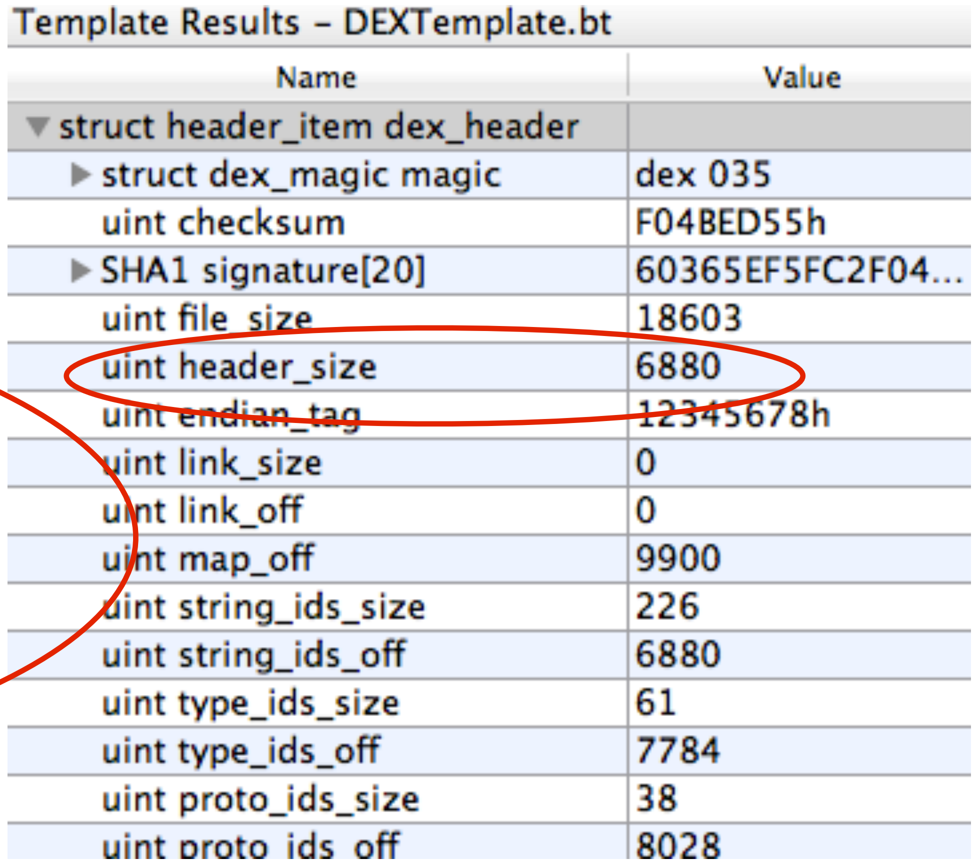
- This allows us to open a dex file from a byte[]
- Read the dex from publicSourceDir / Apps memory / dalvik-cache -- where ever
- Yay! We just created a packer/encryptor!

Adventures in Anti-Analysis: Dexception



```
0000h: 64 65 78 0A 30 33 35 00 55 ED 4B F0 60 36 5E F5  dex.035.UíKð`6^õ
0010h: FC 2F 04 F3 6A 7F 14 A3 52 BD E9 4A F4 D0 3D 09  ü/.ój..£R½éJôÐ=.
0020h: AB 48 00 00 E0 1A 00 00 78 56 34 12 00 00 00 00  «H..à...xV4.....
0030h: 00 00 00 00 AC 26 00 00 E2 00 00 00 E0 1A 00 00  ....¬&..â...à...
0040h: 3D 00 00 00 68 1E 00 00 26 00 00 00 5C 1F 00 00  =...h...&...\...
0050h: 29 00 00 00 24 21 00 00 54 00 00 00 6C 22 00 00  )...$!..T...l"..
0060h: 0D 00 00 00 0C 25 00 00 FF 21 00 00 AC 26 00 00  .....%..ÿ!..¬&..
0070h: B5 B4 A9 DB E1 E2 E4 D1 1B 8D 19 74 D9 D0 03 5C  µ´©ÛáâäÑ...tÙÐ.\
0080h: 4A 9D 81 8D FC 5D 3C BC BE 49 FD 68 D8 FA 1C 83  J...ü]<¼¾IýhØú.ƒ
0090h: A1 CB D1 D1 A1 D1 D1 D1 A9 87 E5 C3 D1 D1 D1 D1  ¡ËÑÑ¡ÑÑÑ©‡åÃÑÑÑÑ
00A0h: D1 D1 D1 D1 71 C8 D1 D1 5C D1 D1 D1 A1 D1 D1 D1  ÑÑÑÑqÈÑÑ\ÑÑÑ¡ÑÑÑ
00B0h: F6 D1 D1 D1 75 D3 D1 D1 CD D1 D1 D1 91 D2 D1 D1  öÑÑÑuÓÑÑÍÑÑÑ'ÒÑÑ
00C0h: C5 D1 D1 D1 41 D5 D1 D1 E1 D1 D1 D1 E1 D4 D1 D1  ÅÑÑÑAÕÑÑáÑÑÑáÔÑÑ
00D0h: DA D1 D1 D1 61 D7 D1 D1 B1 C3 D1 D1 C1 D9 D1 D1  ÚÑÑÑa×ÑÑ±ÃÑÑÁÙÑÑ
00E0h: 13 C0 D1 D1 C1 DE D1 D1 A5 DE D1 D1 2A DF D1 D1  .ÀÑÑÁÞÑÑ¥ÞÑÑ*ßÑÑ
00F0h: EC DE D1 D1 87 C3 D1 D1 CA DE D1 D1 05 DE D1 D1  ìÞÑÑ‡ÃÑÑÊÞÑÑ.ÞÑÑ
0100h: 4D DE D1 D1 4E DE D1 D1 41 C5 D1 D1 51 C7 D1 D1  MÞÑÑNÞÑÑAÅÑÑQÇÑÑ
0110h: 5A DE D1 D1 29 C2 D1 D1 06 C1 D1 D1 86 C1 D1 D1  ZÞÑÑ)ÂÑÑ.ÁÑÑ†ÁÑÑ
```

- header_size = 6880, win!



Template Results – DEXTemplate.bt

Name	Value
▼ struct header_item dex_header	
▶ struct dex_magic magic	dex 035
uint checksum	F04BED55h
▶ SHA1 signature[20]	60365EF5FC2F04...
uint file_size	18603
uint header_size	6880
uint endian_tag	12345678h
uint link_size	0
uint link_off	0
uint map_off	9900
uint string_ids_size	226
uint string_ids_off	6880
uint type_ids_size	61
uint type_ids_off	7784
uint proto_ids_size	38
uint proto_ids_off	8028

- Ok, maybe just a packer since we only XOR'ed everything with 0xd1...

Adventures in Anti-Analysis: Dexception

- This results in a pretty interesting issue
- Automated analysis tools see a valid dex and process it - though the avoid the hidden dex file
- Requires a special tool / hex editor / manual intervention to rip out the embedded dex file
- Many different options available to embedding files making it harder to automate

Adventures in Anti-Analysis: Dexception

- What if it's encrypted? Even a simple XOR could be hard to automate unpacking each time
- Throw an extra layer on top - zip the dex
- Throw in some native code? Load directly from memory?
- Possibilities are pretty much endless

Adventures in Anti-Analysis: Dexception

- The nice part about this "hiding" technique is ease of detection
- header_size > 0x70 ? alert!
- Maybe you just discovered the first Android KeyLime binary! (probably not)
- Someones definitely doing something weird, should definitely take a look at it

Adventures in Anti-Analysis: Endian Reversal Theory

- None of the current tools implement (maybe IDA does?) Reverse Endian support*
- Dalvik verifier/swapper can detect Reverse Endian dex files and swap it to fit the architecture support on the device
- Swap all bytes to Reverse Endian - break tools, still work on the device!
- Didn't actually implement this technique, but theory is sound (famous last words)

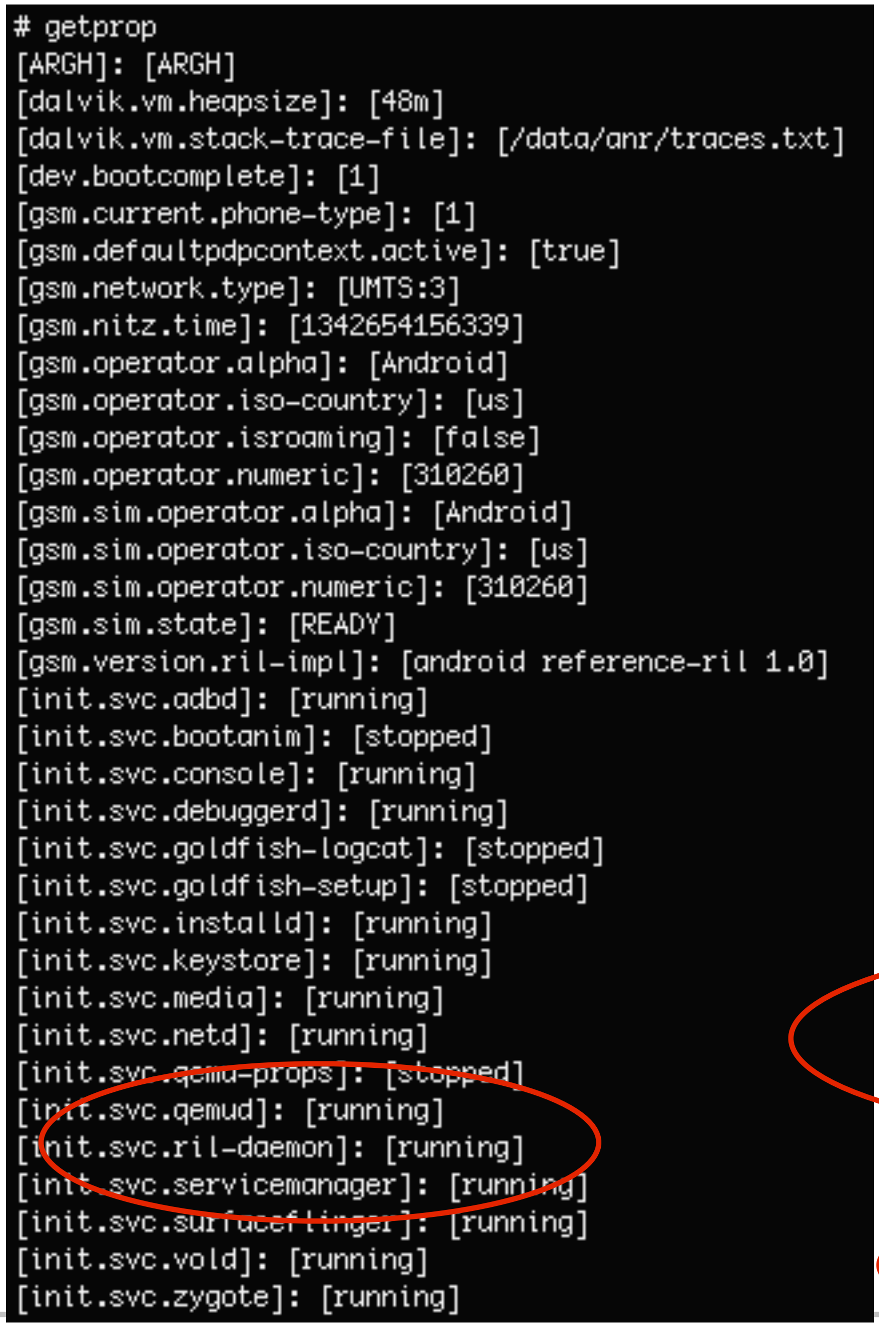
Adventures in Anti-Emulator

- Well covered subject (detecting qemu)
- Check out Jono + Charlie’s Summercon presentation on the Bouncer for good details!
- Only seen recommendations are for changing imei/ phone number/androidId
- Anything we can easily look for as an attacker?
- getprop FTW!

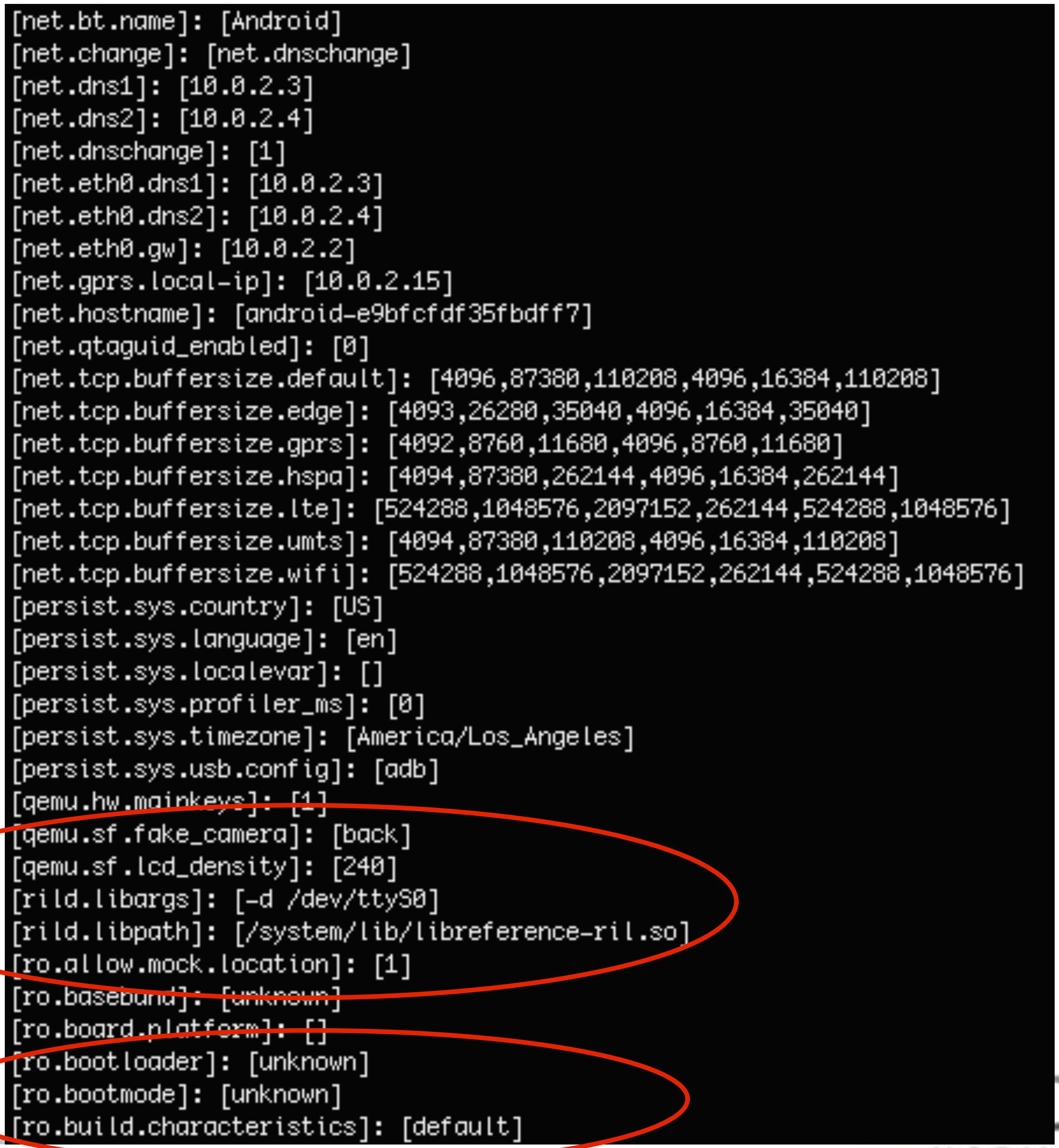
lookout™ MOBILE SECURITY

Adventures in Anti-Emulator

- Take your pick -- so many to choose from;



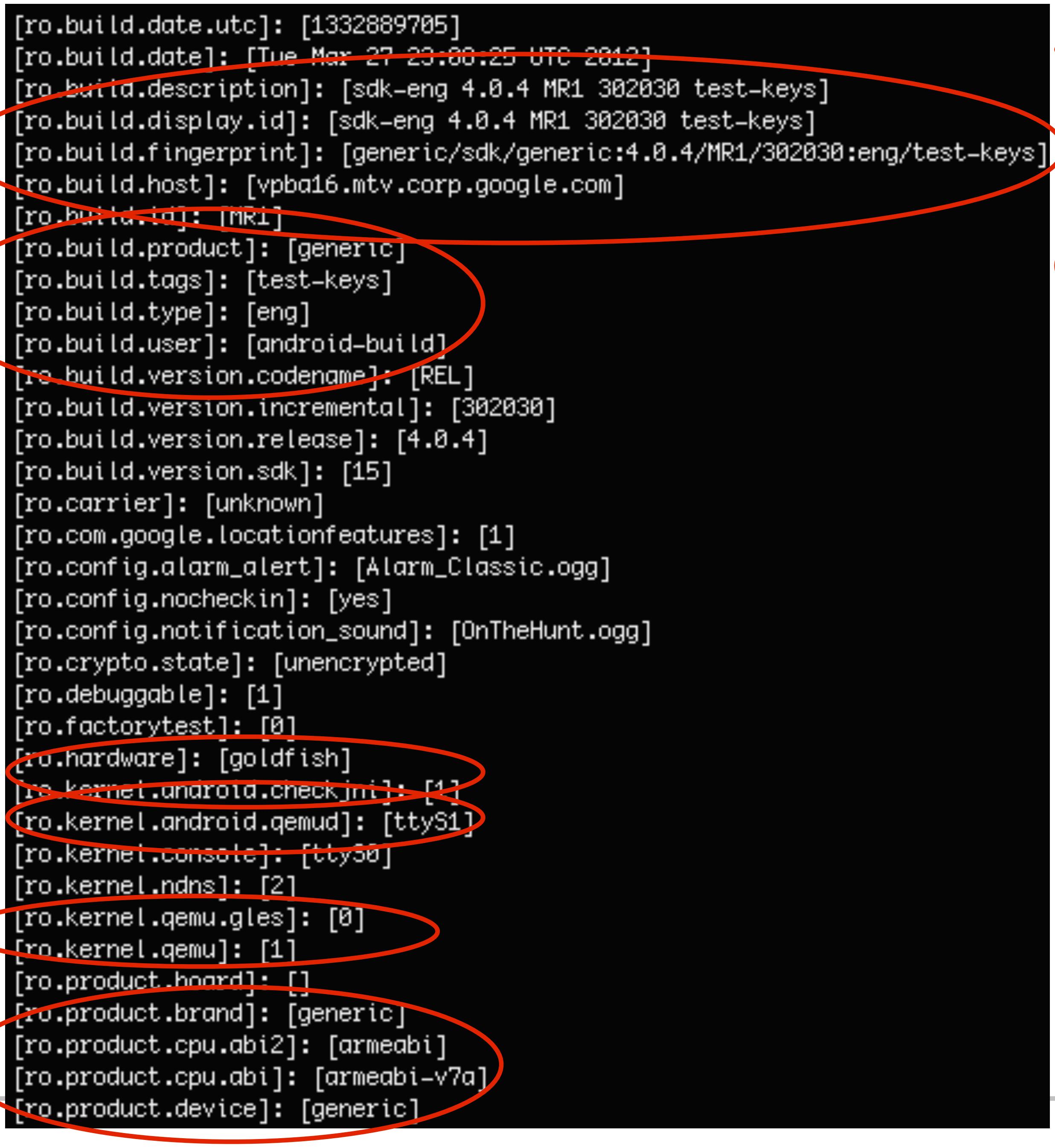
```
# getprop
[ARGH]: [ARGH]
[dalvik.vm.heapsize]: [48m]
[dalvik.vm.stack-trace-file]: [/data/anr/traces.txt]
[dev.bootcomplete]: [1]
[gsm.current.phone-type]: [1]
[gsm.defaultpdpcontext.active]: [true]
[gsm.network.type]: [UMTS:3]
[gsm.nitz.time]: [1342654156339]
[gsm.operator.alpha]: [Android]
[gsm.operator.iso-country]: [us]
[gsm.operator.isroaming]: [false]
[gsm.operator.numeric]: [310260]
[gsm.sim.operator.alpha]: [Android]
[gsm.sim.operator.iso-country]: [us]
[gsm.sim.operator.numeric]: [310260]
[gsm.sim.state]: [READY]
[gsm.version.ril-impl]: [android reference-ril 1.0]
[init.svc.adbd]: [running]
[init.svc.bootanim]: [stopped]
[init.svc.console]: [running]
[init.svc.debuggerd]: [running]
[init.svc.goldfish-logcat]: [stopped]
[init.svc.goldfish-setup]: [stopped]
[init.svc.installd]: [running]
[init.svc.keystore]: [running]
[init.svc.media]: [running]
[init.svc.netd]: [running]
[init.svc.qemu-props]: [stopped]
[init.svc.qemud]: [running]
[init.svc.ril-daemon]: [running]
[init.svc.servicemanager]: [running]
[init.svc.surfaceflinger]: [running]
[init.svc.vold]: [running]
[init.svc.zygote]: [running]
```



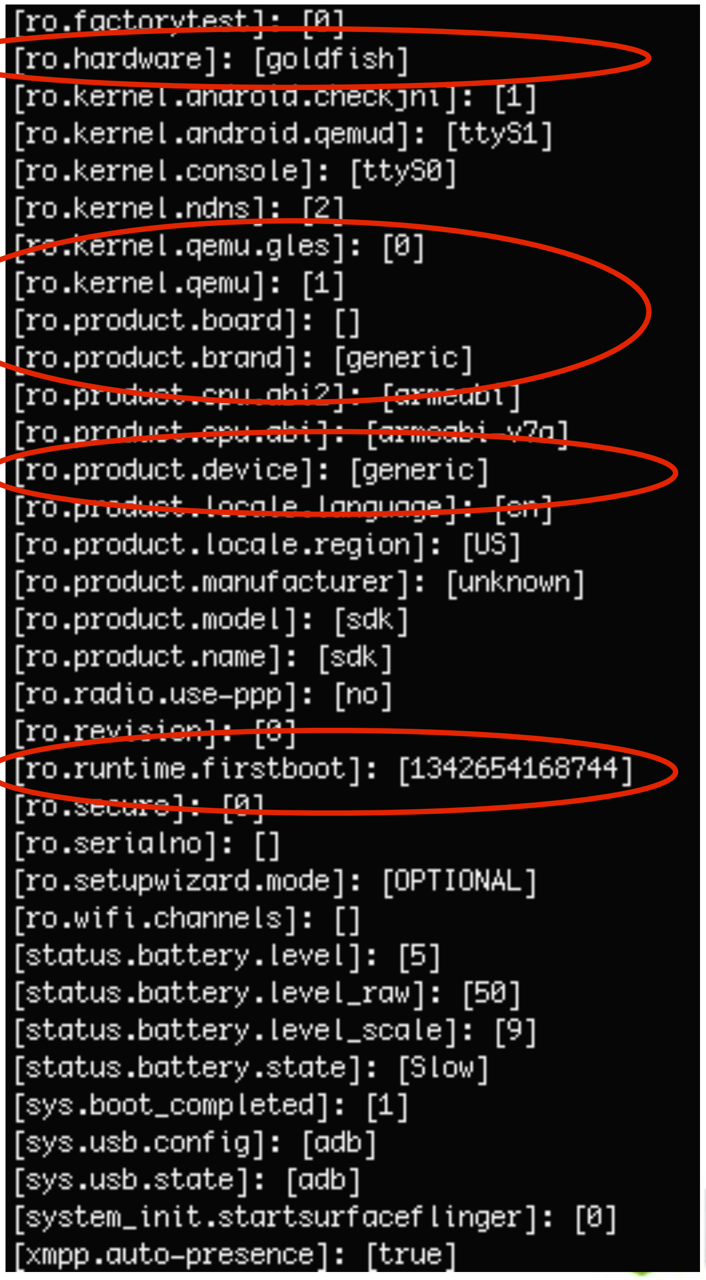
```
[net.bt.name]: [Android]
[net.change]: [net.dnschange]
[net.dns1]: [10.0.2.3]
[net.dns2]: [10.0.2.4]
[net.dnschange]: [1]
[net.eth0.dns1]: [10.0.2.3]
[net.eth0.dns2]: [10.0.2.4]
[net.eth0.gw]: [10.0.2.2]
[net.gprs.local-ip]: [10.0.2.15]
[net.hostname]: [android-e9bfcfdf35fbdff7]
[net.qtaguid_enabled]: [0]
[net.tcp.buffersize.default]: [4096,87380,110208,4096,16384,110208]
[net.tcp.buffersize.edge]: [4093,26280,35040,4096,16384,35040]
[net.tcp.buffersize.gprs]: [4092,8760,11680,4096,8760,11680]
[net.tcp.buffersize.hspa]: [4094,87380,262144,4096,16384,262144]
[net.tcp.buffersize.lte]: [524288,1048576,2097152,262144,524288,1048576]
[net.tcp.buffersize.umts]: [4094,87380,110208,4096,16384,110208]
[net.tcp.buffersize.wifi]: [524288,1048576,2097152,262144,524288,1048576]
[persist.sys.country]: [US]
[persist.sys.language]: [en]
[persist.sys.localevar]: []
[persist.sys.profiler_ms]: [0]
[persist.sys.timezone]: [America/Los_Angeles]
[persist.sys.usb.config]: [adb]
[qemu.hw.mainkeys]: [1]
[qemu.sf.fake_camera]: [back]
[qemu.sf.lcd_density]: [240]
[rild.libargs]: [-d /dev/ttyS0]
[rild.libpath]: [/system/lib/libreference-ril.so]
[ro.allow.mock.location]: [1]
[ro.baseband]: [unknown]
[ro.board.platform]: []
[ro.bootloader]: [unknown]
[ro.bootmode]: [unknown]
[ro.build.characteristics]: [default]
```

Adventures in Anti-Emulator

- And more...



```
[ro.build.date.utc]: [1332889705]
[ro.build.date]: [Tue Mar 27 23:08:25 UTC 2012]
[ro.build.description]: [sdk-eng 4.0.4 MR1 302030 test-keys]
[ro.build.display.id]: [sdk-eng 4.0.4 MR1 302030 test-keys]
[ro.build.fingerprint]: [generic/sdk/generic:4.0.4/MR1/302030:eng/test-keys]
[ro.build.host]: [vpba16.mtv.corp.google.com]
[ro.build.id]: [MR1]
[ro.build.product]: [generic]
[ro.build.tags]: [test-keys]
[ro.build.type]: [eng]
[ro.build.user]: [android-build]
[ro.build.version.codename]: [REL]
[ro.build.version.incremental]: [302030]
[ro.build.version.release]: [4.0.4]
[ro.build.version.sdk]: [15]
[ro.carrier]: [unknown]
[ro.com.google.locationfeatures]: [1]
[ro.config.alarm_alert]: [Alarm_Classic.ogg]
[ro.config.nocheckin]: [yes]
[ro.config.notification_sound]: [OnTheHunt.ogg]
[ro.crypto.state]: [unencrypted]
[ro.debuggable]: [1]
[ro.factorytest]: [0]
[ro.hardware]: [goldfish]
[ro.kernel.android.checkjni]: [1]
[ro.kernel.android.qemud]: [ttyS1]
[ro.kernel.console]: [ttyS0]
[ro.kernel.ndns]: [2]
[ro.kernel.qemu.gles]: [0]
[ro.kernel.qemu]: [1]
[ro.product.board]: []
[ro.product.brand]: [generic]
[ro.product.cpu.abi2]: [armeabi]
[ro.product.cpu.abi]: [armeabi-v7a]
[ro.product.device]: [generic]
```



```
[ro.factorytest]: [0]
[ro.hardware]: [goldfish]
[ro.kernel.android.checkjni]: [1]
[ro.kernel.android.qemud]: [ttyS1]
[ro.kernel.console]: [ttyS0]
[ro.kernel.ndns]: [2]
[ro.kernel.qemu.gles]: [0]
[ro.kernel.qemu]: [1]
[ro.product.board]: []
[ro.product.brand]: [generic]
[ro.product.cpu.abi2]: [armeabi]
[ro.product.cpu.abi]: [armeabi-v7a]
[ro.product.device]: [generic]
[ro.product.locale.language]: [en]
[ro.product.locale.region]: [US]
[ro.product.manufacturer]: [unknown]
[ro.product.model]: [sdk]
[ro.product.name]: [sdk]
[ro.radio.use-ppp]: [no]
[ro.revision]: [0]
[ro.runtime.firstboot]: [1342654168744]
[ro.secure]: [0]
[ro.serialno]: []
[ro.setupwizard.mode]: [OPTIONAL]
[ro.wifi.channels]: []
[status.battery.level]: [5]
[status.battery.level_raw]: [50]
[status.battery.level_scale]: [9]
[status.battery.state]: [Slow]
[sys.boot_completed]: [1]
[sys.usb.config]: [adb]
[sys.usb.state]: [adb]
[system_init.startsurfaceflinger]: [0]
[xmpp.auto-presence]: [true]
```

lookout™
MOBILE SECURITY

Adventures in Anti-Emulator

- Keep in mind, it is easy to do the inverse as well
- Instead of looking if you're in an emulator, see if you're just not a normal looking device
- Look for Google Experience app settings;

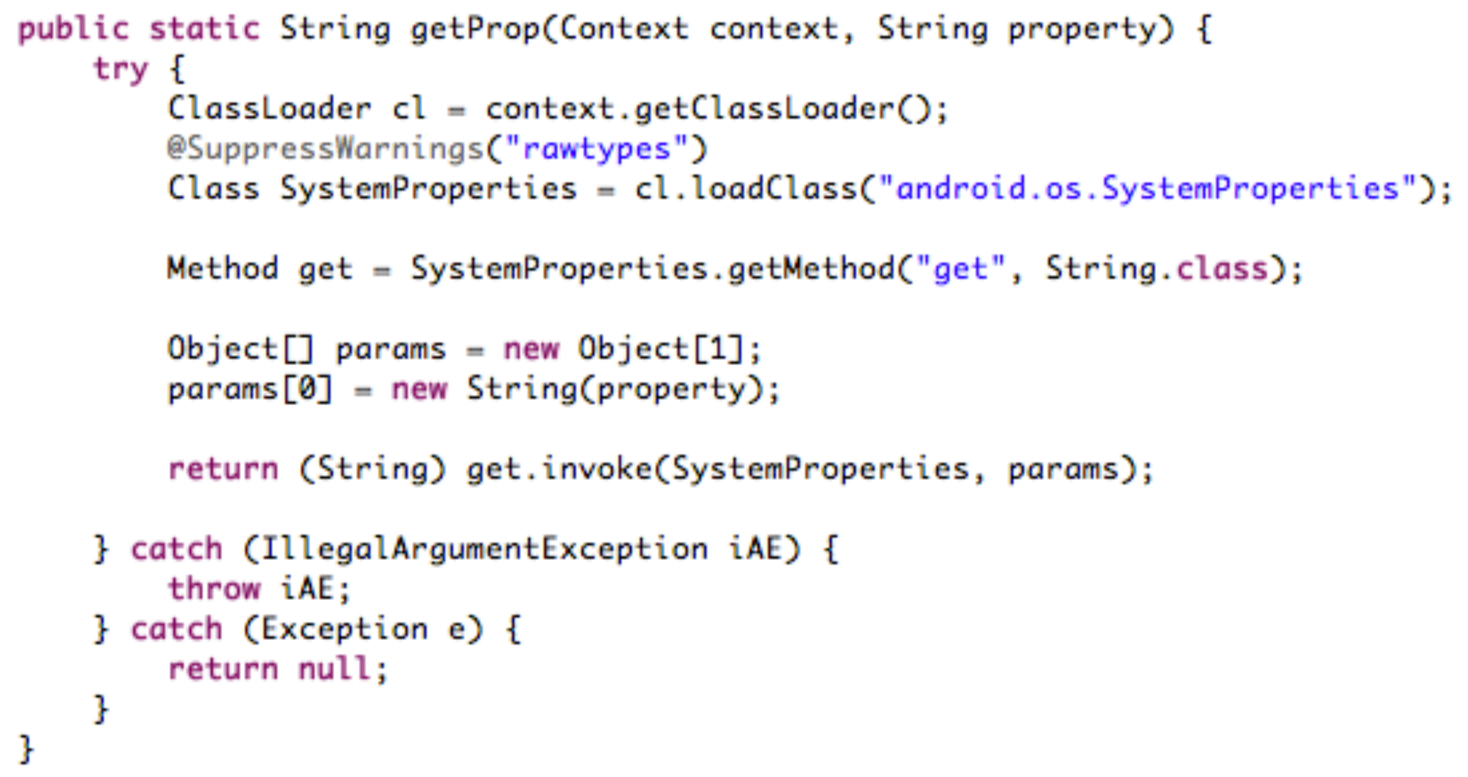
  ro.build.fingerprint
  ro.error.receiver.system.apps
  ro.url.legal.android_privacy
  ro.url.legal

Adventures in Anti-Emulator

- But how to use getprop? There is no public command?!

Adventures in Anti-Emulator

- But how to use getprop? There is no public command?!



```
public static String getProp(Context context, String property) {
    try {
        ClassLoader cl = context.getClassLoader();
        @SuppressWarnings("rawtypes")
        Class SystemProperties = cl.loadClass("android.os.SystemProperties");

        Method get = SystemProperties.getMethod("get", String.class);

        Object[] params = new Object[1];
        params[0] = new String(property);

        return (String) get.invoke(SystemProperties, params);

    } catch (IllegalArgumentException iAE) {
        throw iAE;
    } catch (Exception e) {
        return null;
    }
}
```

- Reflection to the rescue, yet again!

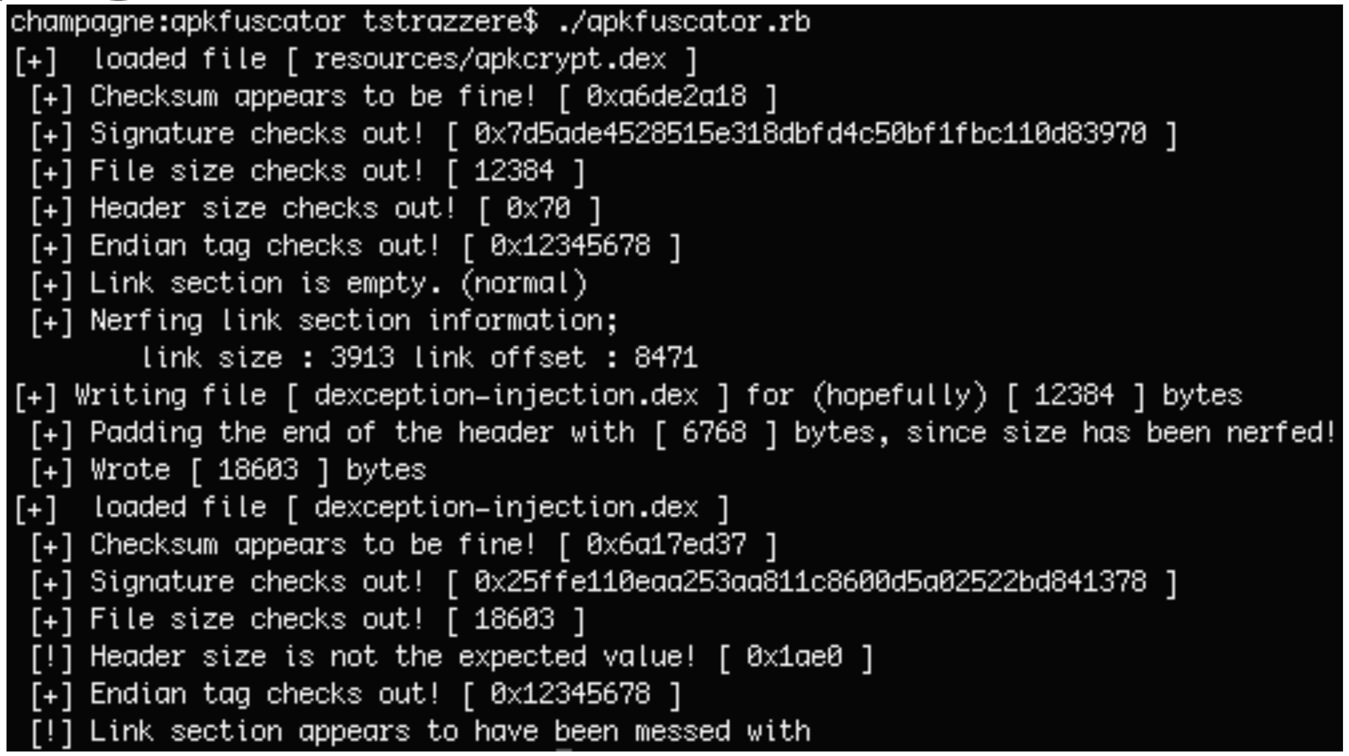
Adventures in Anti-Emulator

- Should be easy to flag statically, but this might not come up in dynamic analysis
- Could build more "believable" emulator images (build.prop)
- Can always flag *every* reflection, but that might cause massive false positives
- Hook the getprop command, maybe with pof's LD_PRELOAD example? (https://github.com/poliva/ldpreloadhook)

PoC and Lessons Learned

Stop Talking I Just Want the Code

- "apkfuscator" - PoC tool used to munge all the dex files / create examples open sourced (GPL) - check my github page soon!



```
champagne:apkfuscator tstrazzere$ ./apkfuscator.rb
[+]  loaded file [ resources/apkcrypt.dex ]
 [+] Checksum appears to be fine! [ 0xa6de2a18 ]
 [+] Signature checks out! [ 0x7d5ade4528515e318dbfd4c50bf1fbc110d83970 ]
 [+] File size checks out! [ 12384 ]
 [+] Header size checks out! [ 0x70 ]
 [+] Endian tag checks out! [ 0x12345678 ]
 [+] Link section is empty. (normal)
 [+] Nerfing link section information;
        link size : 3913 link offset : 8471
[+] Writing file [ dexception-injection.dex ] for (hopefully) [ 12384 ] bytes
 [+] Padding the end of the header with [ 6768 ] bytes, since size has been nerfed!
 [+] Wrote [ 18603 ] bytes
[+]  loaded file [ dexception-injection.dex ]
 [+] Checksum appears to be fine! [ 0x6a17ed37 ]
 [+] Signature checks out! [ 0x25ffe110eaa253aa811c8600d5a02522bd841378 ]
 [+] File size checks out! [ 18603 ]
 [!] Header size is not the expected value! [ 0x1ae0 ]
 [+] Endian tag checks out! [ 0x12345678 ]
 [!] Link section appears to have been messed with
```

- Beware of the code it bites -- quick hack and challenge to do heavy file manipulation in ruby (I know I'm a masochist)

lookout™ MOBILE SECURITY

PoC and Lessons Learned

But What Have We Truly Learned?!
(other than avoid ruby for file manipulation)

- When parsing dex files - expect the unexpected, don't throw an exception if you don't have to, but log that and possibly call attention to it
- Grok`ing the file format can be important, don't rely on all the pre-made tools to do it for you - you're a hacker, know what you're hacking!
- Part of defending is attacking your own weaknesses, stay sharp and keep your tools even sharper

lookout™
MOBILE SECURITY

PoC and Lessons Learned

But What Have We Truly Learned?!
(other than avoid ruby for file manipulation)

- If something explodes - use Jon Larimer's 010 Editor template (I've committed fixes for these methods) - makes analyzing / visualizing dex files so much easier
- Think outside the box - the world is full of people thinking "sms == bad", don't be a lazy analyst/reverser
- Have fun and break stuff - that's why we're reverse engineers

Thanks!

@timstrazz
strazzere.com/blog
github.com/strazzere

“diff-t” at
#droidsec / #smali on freenode

Greets:
fG+, Lohan+, jcase, tmw, jon larimer,
jono, zuk, jduck, syn, JF, pof, thomas cannon,
anthony desnos, snare, crypto girl and many others :)

lookout™ MOBILE SECURITY

References

http://code.google.com/p/smali/

http://code.google.com/p/androguard/

http://code.google.com/p/dex2jar/

http://siis.cse.psu.edu/ded/

http://hex-rays.com/products/ida/index.shtml

http://source.android.com/tech/dalvik/dex-format.html