NowSecure

CONNECT 2019

The Premier Mobile DevSecOps and OSS Event
June 3-4, 2019 | Washington, D.C.

APKiD

Fast Identification of Appshielding Products

Eduardo Novella

Outline

Main ideas

- **Introduction**

  Edu Novella and APKiD OSS (RedNaga community)

- **Android Malware Detection**

  What APKiD was initially designed for

- **Android Code Obfuscation**

  Identification of obfuscators, packers, appshielding and RASP products

- **Demos**

  Detection of the most common Android protectors in the market

- **Takeaways**

  Things learned after this presentation

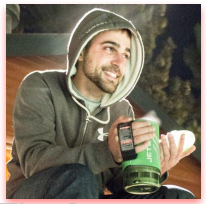
$ whoami

"I stay with problems longer"

- **Mobile Security Research Engineer @ NowSecure**
  - Focused on **Android** Reverse Engineering
- **Previously (Reverse Engineering)**
  - Android **mobile** security: cloud-based payments (HCE wallets), DRM and TEE solutions
  - **Embedded** security : smartcards, smartmeter, PayTV, HCE, modem, any hardened IoT dev
  - Crypto: side-channel & fault injection attacks (hw). Whitebox cryptography (sw)
- **Personal @ enovella.github.io**
  - Based in London (UK)
  - Chess player, swimmer and nature lover

RedNaga Security

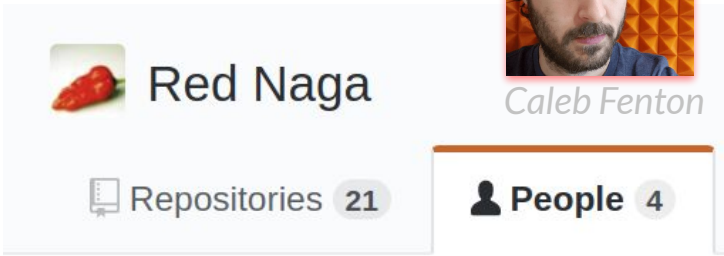
Mobile Security Team



Tim Strazzere

Caleb Fenton

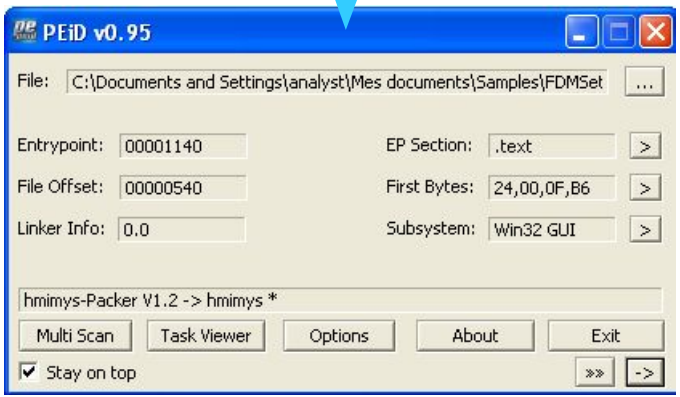
- Banded together by the love of **hot sauces** & 0-days
- Disclosures/lessons/training, **OSS** tools, malware analysis
- **Slack** channel talking about mobile sec
- Sources:
  - https://rednaga.io
  - https://github.com/rednaga
  - http://slack.rednaga.io



Why We Are Here

We love spicy assembly, and you?

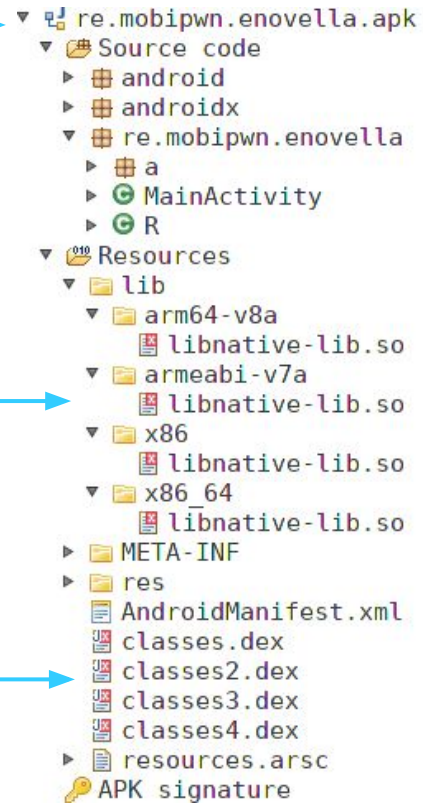
- Showcase **APKiD**: The "**PEiD**" of Android Apps
- APKiD can be used for many purposes:
  - Which apps could potentially contain malware? Tampered?
  - **Which apps are obfuscated/packed? Which protector?**
  - Has the DEX been compiled in unusual ways? Anomalies?
  - Anti-debugging, anti-emulation, …



How APKiD works

Android Application Packaging (APK)

- **APKiD workflow**
  - Unpack APK == ZIP
  - Read magic number
  - Apply Yara rules
  - Print results if matches
- **Yara rules**
  - APK
  - Native code
  - DEX bytecode



How APKiD works

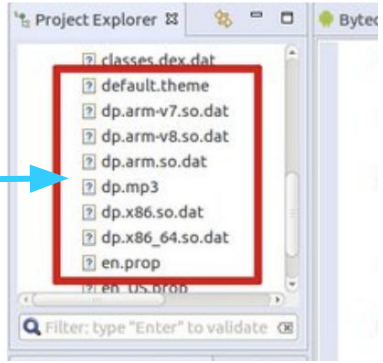
APK Yara rules

1

```
rule dexprotector_b : packer
{
  // Possible newer version

  meta:
    author      = "Eduardo Novella"
    description = "DexProtector"
    url         = "https://dexprotector.com/"
    sample      = "dca2a0bc0f2605072b9b48579e73711af816b0fa1108b825335d2d1f2418e2a7"

  strings:
    //              assets/com.package.name.arm.so.dat
    $encrptlib_1 = /assets\/[A-Za-z0-9.]{2,50}\.arm\-v7\.so\.dat/
    $encrptlib_2 = /assets\/[A-Za-z0-9.]{2,50}\.arm\-v8\.so\.dat/
    $encrptlib_3 = /assets\/[A-Za-z0-9.]{2,50}\.arm\.so\.dat/
    $encrptlib_4 = /assets\/[A-Za-z0-9.]{2,50}\.dex\.dat/
    $encrptlib_5 = /assets\/[A-Za-z0-9.]{2,50}\.x86\.so\.dat/
    $encrptlib_6 = /assets\/[A-Za-z0-9.]{2,50}\.x86\_64\.so\.dat/

    $encrptcustom = /assets\/[A-Za-z0-9.]{2,50}\.mp3/

  condition:
    is_apk and 2 of ($encrptlib_*) and $encrptcustom and
    not dexprotector_a and
    not dexprotector
}
```

2

3



How APKiD works

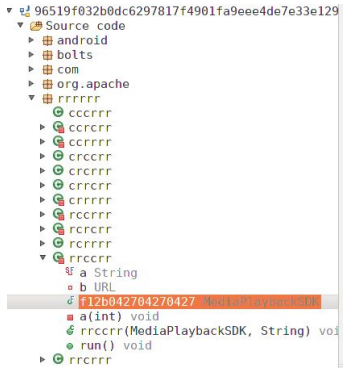
DEX Yara rules

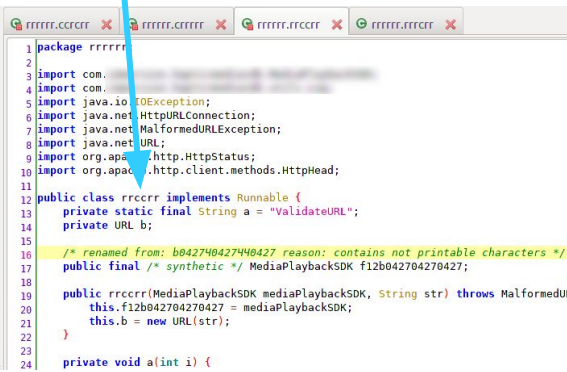
1

2

3

```
rule arxan : obfuscator
{
  meta:
    description = "Arxan"
    url         = "https://www.arxan.com/products/application-protection-mobile/"
    sample      = "7bd1139b5f860d48e0c35a3f117f980564f45c177a6ef480588b5b5c8165f47e"
    author      = "Eduardo Novella"

  strings:
    // Obfuscated Lpackage/class/: "L([a-z]\1{5}\/[a-z]{6}\/".
    // AFAIK, Yara does not support backreferences at the moment, thus this is the combo:
    $pkg = /L(a{6}|b{6}|c{6}|d{6}|e{6}|f{6}|g{6}|h{6}|i{6}|j{6}|k{6}|l{6}|m{6}|n{6}|o{6}|p{6}

    // Obfuscated methods are found to follow a pattern like:
    // 1 byte size + 1 byte ASCII + [7-26] non-ASCII bytes + 00 (null terminator)
    $m1 = { 10 62 (6? | 75) [14] 00 }
    $m2 = { (0b | 0d) 62 d0 [15] 00 }
    $m3 = { (0e | 10) 62 30 34 3? [15] 00 }
    $m4 = { (0b | 0d) 62 30 34 3? [13] 00 }
    $m5 = { (08 | 0b | 0d | 0e ) 62 [7-13] 00 }
    $m6 = { 0a 62 (30 34 3? | d? ?? ??) [11] 00 }
    $m7 = { (0d | 0b | 11) (62 d1 8? | 6? ?? ??) [14] 00 }

  condition:
    is_dex and
    $pkg and
    6 of ($m*)
}
```





NowSecure™

How APKiD works

ELF Yara rules

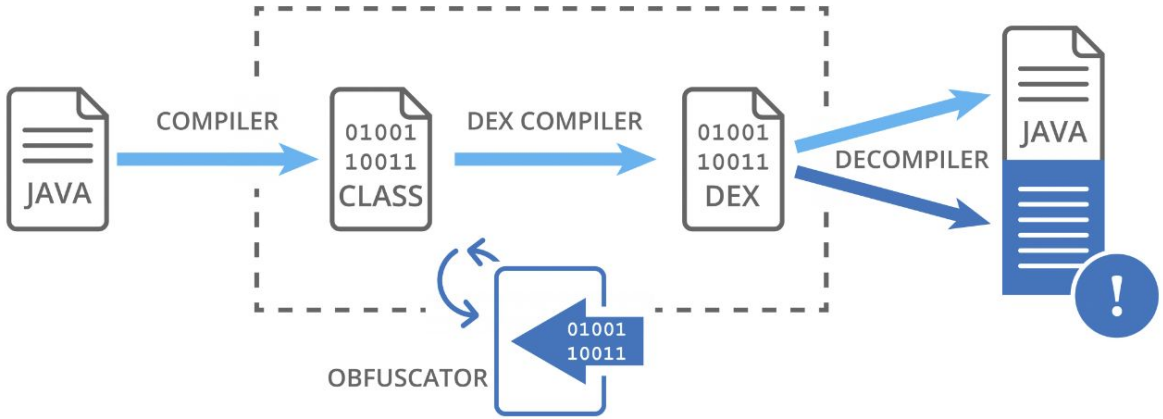
```
174
175     strings:
176       // Library below heuristic is found inside of is normally named "libaurorabridge.so"
177       $segment = ".firehash"
178       $opcodes_arm = {
179           04 00 2D E5  //  STR     R0, [SP,#var_4]!
180           00 00 0F E1  //  MRS     R0, CPSR
181           01 00 51 E1  //  CMP     R1, R1
182           02 00 00 ?A  //  BNE     loc_F0854
183           00 F0 29 E1  //  MSR     CPSR_cf, R0
184           04 00 9D E4  //  LDR     R0, [SP+4+var_4],#4
185           ?? ?? ?? EA  //  B       loc_F0828
186       }
187
188     condition:
189       elf.machine == elf.EM_ARM and all of them
190 }
```

Malware & Piracy detection

Initial uses of APKiD

Android Compiler Fingerprinting:

- dx → Java .class files (source code)
- dexmerge → Not used manually, only by IDEs (source code)
- **dexlib** (smali) → DEX files (**not** source code)

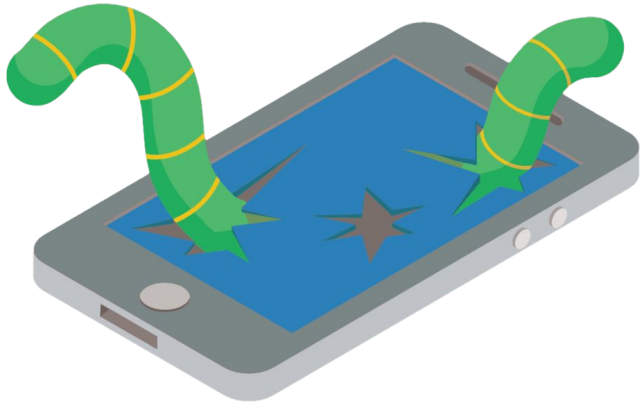


Why would a legitimate developer ever need to use smali?
They do have the source

Malware & Piracy detection

Initial uses of APKiD

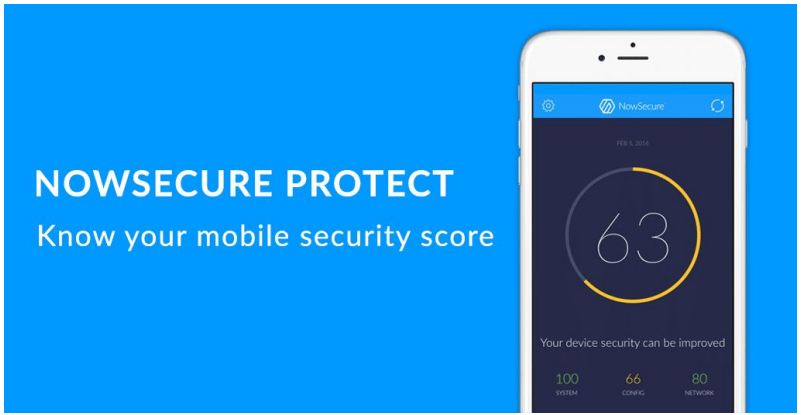
- **Hypothesis**:
- If app compiled with **dexlib**, probably tampered
- If tampered, probably was not the developer
- Tampered apps are likely either:
  - pirated / cracked
  - malware
  - Not necessarily tampered but obfuscated / packed



App is tampered ⟶ App is interesting

DEMO

NowSecure Protector APK
repackaged with Frida-gadget
via Objection



```
[00:23 edu@NowSecure 01-tampering] >  apkid NowSecureProtect_v1.0.apk
[+] APKiD 2.0.2 :: from RedNaga :: rednaga.io
[*] NowSecureProtect_v1.0.apk!classes.dex
 |-> anti_debug : Debug.isDebuggerConnected() check
 |-> anti_vm : Build.BOARD check, Build.FINGERPRINT check, Build.MANUFACTURER check, Build.MODEL check, Build.PRODUCT
.secure check, possible vm check
 |-> compiler : dx
[*] NowSecureProtect_v1.0.apk!classes2.dex
 |-> compiler : dx
[00:23 edu@NowSecure 01-tampering] >  objection patchapk --source NowSecureProtect_v1.0.apk
No architecture specified. Determining it using `adb`...
Detected target device architecture as: arm64-v8a
Using latest Github gadget version: 12.6.1
Patcher will be using Gadget version: 12.6.1
Unpacking NowSecureProtect_v1.0.apk
An error may have occurred while extracting the APK.
S: WARNING: Could not write to (/home/edu/.local/share/apktool/framework), using /tmp instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the de

App already has android.permission.INTERNET
Target class not specified, searching for launchable activity instead...
Reading smali from: /tmp/tmp_yc4aoyc.apktemp/smali/com/nowsecure/android/ui/welcome/WelcomeActivity.smali
Injecting loadLibrary call at line: 6
Attempting to fix the constructors .locals count
Current locals value is 0, updating to 1:
Writing patched smali back to: /tmp/tmp_yc4aoyc.apktemp/smali/com/nowsecure/android/ui/welcome/WelcomeActivity.smali
Creating library path: /tmp/tmp_yc4aoyc.apktemp/lib/arm64-v8a
Copying Frida gadget to libs path...
Rebuilding the APK with the frida-gadget loaded...
Rebuilding the APK may have failed. Read the following output to determine if apktool actually had an error:

S: WARNING: Could not write to (/home/edu/.local/share/apktool/framework), using /tmp instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the de

Built new APK with injected loadLibrary and frida-gadget
Signing new APK.
Signed the new APK
Performing zipalign
Zipalign completed
Copying final apk from /tmp/tmp_yc4aoyc.apktemp.aligned.objection.apk to NowSecureProtect_v1.0.objection.apk in curren
Cleaning up temp files...
[00:24 edu@NowSecure 01-tampering] >  apkid NowSecureProtect_v1.0.objection.apk
[+] APKiD 2.0.2 :: from RedNaga :: rednaga.io
[*] NowSecureProtect_v1.0.objection.apk!classes.dex
 |-> anti_debug : Debug.isDebuggerConnected() check
 |-> anti_vm : Build.BOARD check, Build.FINGERPRINT check, Build.MANUFACTURER check, Build.MODEL check, Build.PRODUCT
.secure check, possible vm check
 |-> compiler : dexlib 2.x
[*] NowSecureProtect_v1.0.objection.apk!classes2.dex
 |-> compiler : dexlib 2.x
```

NowSecure

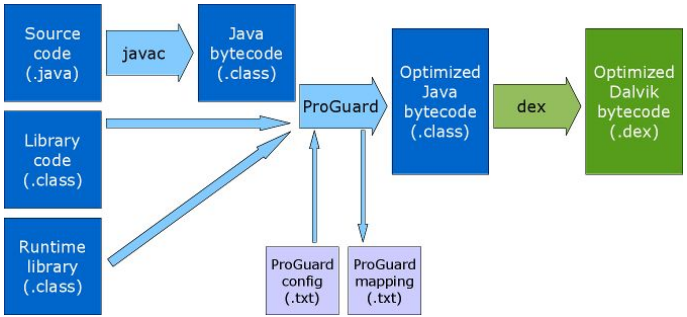
Android Code Obfuscation

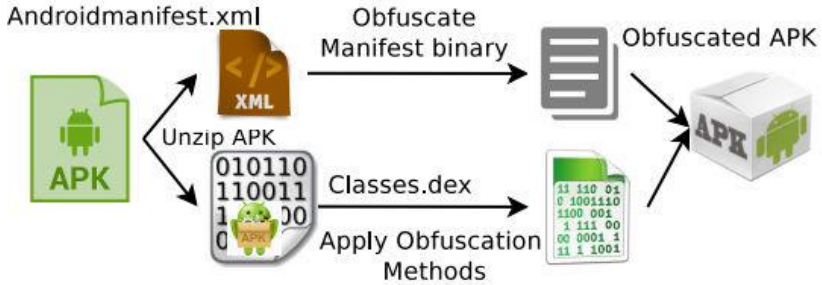
Obfuscators, packers, protectors, appshielding, RASP...

Android Code Obfuscation

Obfuscators, packers, protectors, appshielding, RASP...

- Protect intellectual property
- Make reverse engineering harder ⟶ slow down attackers
- Ensure mobile security: (**requirement**)
  - Mobile payment (HCE wallets [Analyzing the security of Cloud-Based Payment apps on Android])
  - Content protection (DRM)
  - Whitebox cryptography
  - Cryptocurrency wallets, time-based token generators, VPN client, ...
  - Any financial app in the market





NowSecure™

Android Code Obfuscation

Obfuscators, packers, protectors, appshielding, RASP...

Product	Anti-Tampering	Anti-Hooking	Anti-Debugging	Anti-Emulator	Code Obfuscation	White-Box Cryptography	Device Binding	Root Detection	Anti-Keylogger	Anti-Screen Reader	Data Encryption	Secure Communication
Arxan for Android	✓	✓	✓	.	✓	✓	.	✓	.	.	✓	.
DNP HyperTech CrackProof	✓	.	✓	✓	.	.	.	✓	.	.	.	.
Entersekt Transakt	✓	.	.	.	.	.	✓	✓	✓	.	.	✓
Gemalto Mobile Protector	.	✓	✓	.	✓	.	✓	✓	✓	.	✓	✓
GuardSquare DexGuard	✓	✓	✓	✓	✓	✓	.	✓	.	.	.	✓
Inside Secure Core for Android	✓	.	✓	.	✓	✓	.	✓	.	.	.	.
Intertrust WhiteCryption	✓	.	✓	.	✓	✓	✓	✓	.	.	.	.
PreEmptive DashO	✓	.	✓	✓	✓	.	✓	✓	.	.	.	.
Promon SHIELD	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SecNeo AppShield	✓	.	✓	.	✓	.	.	.	.	.	✓	.

Table 1: Overview of RASP products and their advertised features.

Source: “Honey, I shrunk your app security: The state of Android app Hardening”

Android Code Obfuscation

Obfuscators, packers, protectors, appshielding, RASP...

		Language-based			Runtime-based					TIRO	Sensitive APIs	
Sample	#	Reflection	Dynamic loading	Native code	DEX file hooking	Class data overwriting	ArtMethod hooking	Instruction hooking	Instruction overwriting	Iterations	Before	After
aliprotect	2	•	n	•	•	•				3	0	44
apkprotect	1	•	d	•						2	8	52
appguard	1	•		•	•					2	0	8
appsolid	1	•	n	•						2	0	82
baiduprotect	1	•	n	•	•	•				2	1	3
bangcle	1	•	n	•						2	1	4
dexguard	3	•								2	0	4
dexprotector	3	•	r	•						4	0	80
dxshield	2	•	n	•	•					2	3	26
ijiamipacker	2	•	n	•	•	•	•	•	•	2	1	98
liapp	1	•	n	•						2	4	90
naga	1	•	n	•	•					2	2	3
naga_pha	1	•	n	•	•	•	•	•	•	2	0	6
nqprotect	1	•	d	•						2	1	12
qihoopacker	3	•	n	•	•					2	3	217
secshell	2	•	r n	•	•	•				2	200	287
secneo	1	•	n	•						3	0	12
sqlpacker	2	•	d	•						2	1	31
tencentpacker	2	•	n	•	•					3	3	504
unicomsdk	2	•	d	•						2	226	227
wjshell	1	•	d	•	•					2	8	18

Android Code Obfuscation

Obfuscators, packers, protectors, appshielding, RASP...

Choose your sauce:

No Spicy

Mild

Medium Spicy

Very Spicy

Add a Kick with Dynomite Sauce!

Android Code Obfuscation

Renaming of classes, methods, fields and variables
String encryption

```
        v0_1 = new RootLog();
    }

label_8:
    HashMap v2_1 = new HashMap();
    ffiiii.a("\u000B\t\n\u0010{\u0011\u0013\u0001\u0015\u0017\u0016\u0003\b\u0015\u000B\r", '\u0087', '\u0000');
    ffiiii.a("G7=0F4@@5:8", '\u0016', v10);
    ffiiii.a("uqpt^qq]oolWi[Vgb`PS^RR", 'û', '\u0003');
    ffiiii.a("\u0011\u000F\u0010\u0016\u0002\u001A\n\u0018\u001A\u0011\u0018\u0018", '\u000F', v9);
    ffiiii.a("\u0017\f\t\u0007\t\u000B\u0003{\u0012\u007F\f\f\u0001\u0006\u0004", '\u0090', 'K', '\u0001');
    ffiiii.a("MKLR>LJUW", 'Z', v8);
    ((Map)v2_1).put(ffiiii.a("\u001C\u001A\u001B!\r\"$\u0012&(\'\u0014\u0019&\u001C\u001E", '8', '\u0000'), v0_1.a());
    ((Map)v2_1).put(ffiiii.a("%\u0015\u001B\u000E$\u0012\u001E\u001E\u0013\u0018\u0016", '¦', v10), wwwnww.b());
    ((Map)v2_1).put(ffiiii.a("JHIO;PR@TVUBVJGZWWIN[QS", 'ê', v9), v0_1.b());
    ((Map)v2_1).put(ffiiii.a("VTU[G_O]_V]]", '!', '\u0000'), wwwnww.b());
    ((Map)v2_1).put(ffiiii.a("\f\u0003\u0002\u0002\u0006\n\u0004~\u0017\u0007\u0015\u0017\u000E\u0015\u0015", '\u0018', v8), wwwnww.a());
    ((Map)v2_1).put(ffiiii.a("A=<@*62;;", 'e', '\u0001'), v0_1.c());
    ((List)v1).add(v2_1);
    return ((List)v1);
}
```

Android Code Obfuscation

Dynamic code loading / Class encryption
Reflection (anti-static analysis)

```
public static Object f() {
    Object object2;
    Object object1;
    Object object0;
    short s = ((short)QC$a.e[0x14]);
    String string0 = QC$a.j(s, ((byte)(s | 0x28)), ((byte)QC$a.e[0x8F]));
    short s1 = ((short)QC$a.e[0xD]);
    String string1 = QC$a.j(s1, ((byte)(s1 & 0x2A)), ((byte)QC$a.e[0x2F]));
    int i = 2;
    try {
        object0 = Class.forName(QC$a.j(0x4F, ((byte)(-QC$a.e[6])), ((byte)QC$a.e[9]))).getMethod(
                QC$a.j(0x9B, ((byte)QC$a.e[0x44]), ((byte)QC$a.e[0x47])), String.class, String.class)
                .invoke(null, string0, string1);
    }
    catch(Throwable throwable0) {
        throw throwable0.getCause();
    }

    byte[] array_b = new byte[]{-84, -71, 0x60, -63, 3, -24, 6, -120, -17, 2, -38, -32, -30, 0x29,
            -89, 0x29};
    String string2 = QC$a.j(0x61, ((byte)(QC$a.e[0x1E] - 1)), ((byte)QC$a.e[0x31]));
    int i1 = 2;
    try {
        object1 = Class.forName(QC$a.j(((short)QC$a.e[0x3A]), ((byte)(-QC$a.e[6])), ((byte)QC$a.
                e[0x3A]))).getDeclaredConstructor(byte[].class, String.class).newInstance(array_b,
                string2);
    }
    catch(Throwable throwable0) {
        throw throwable0.getCause();
    }

    byte[] array_b1 = new byte[]{0x5A, 6, -2, -75, 0x63, 0x4B, 0x20, 0x66, 0xC, -125, 0xB, 0x6E,
            -2, 9, 0x64, 0x4B};
    try {
        object2 = Class.forName(QC$a.j(0x73, ((byte)(-QC$a.e[6])), ((byte)(-QC$a.e[0x1D])))).getDeclaredConstructor(
                byte[].class).newInstance(array_b1);
    }
    catch(Throwable throwable0) {
        throw throwable0.getCause();
    }

    i1 = 3;
    try {
        array_object1 = new Object[i1];
        array_object1[2] = object2;
        array_object1[1] = object1;
        array_object1[0] = new Integer(2);
        Class class0 = Class.forName(QC$a.j(0x4F, ((byte)(-QC$a.e[6])), ((byte)QC$a.e[9])));
        string2 = QC$a.j(((short)QC$a.e[0x2F]), ((byte)QC$a.e[0x14]), ((byte)(-QC$a.e[6])));
        Class[] array_class = new Class[3];
        array_class[0] = Integer.TYPE;
```

NowSecure™

Android Code Obfuscation

Dynamic code loading
Class encryption

FRIDA

```
[Reflection => forName => javax.crypto.Cipher]
[!] Received: [Reflection => getMethod => getInstance => public static final javax.crypto.Cipher javax.crypto.Cipher.getInstance(java.lang.String,java.lang.String) thr
ows java.security.NoSuchAlgorithmException,java.security.NoSuchProviderException,javax.crypto.NoSuchPaddingException]
[!] Received: [Reflection => forName => javax.crypto.spec.SecretKeySpec]
[!] Received: [Reflection => forName => javax.crypto.spec.IvParameterSpec]
[!] Received: [Reflection => forName => javax.crypto.Cipher]
[!] Received: [Reflection => forName => java.security.Key]
[!] Received: [Reflection => forName => java.security.spec.AlgorithmParameterSpec]
[!] Received: [Reflection => getMethod => init => public final void javax.crypto.Cipher.init(int,java.security.Key,java.security.spec.AlgorithmParameterSpec) throws ja
va.security.InvalidKeyException,java.security.InvalidAlgorithmParameterException]
[!] Received: [Reflection => forName => javax.crypto.Cipher]
[!] Received: [Reflection => getMethod => doFinal => public final byte[] javax.crypto.Cipher.doFinal(byte[],int,int) throws javax.crypto.IllegalBlockSizeException,java
x.crypto.BadPaddingException]
[!] Received: [Reflection => forName => java.util.zip.Inflater]
[!] Received: [Reflection => forName => java.util.zip.Inflater]
[!] Received: [Reflection => getMethod => setInput => public void java.util.zip.Inflater.setInput(byte[],int,int)]
[!] Received: [Reflection => forName => java.util.zip.Inflater]
[!] Received: [Reflection => getMethod => inflate => public int java.util.zip.Inflater.inflate(byte[]) throws java.util.zip.DataFormatException]
[!] Received: [Reflection => forName => dalvik.system.DexFile]
[!] Received: [Reflection => getMethod => write => public void java.io.OutputStream.write(byte[],int,int) throws java.io.IOException]
[!] Received: [Reflection => getMethod => getFD => public final java.io.FileDescriptor java.io.FileOutputStream.getFD() throws java.io.IOException]
[!] Received: [Reflection => getMethod => sync => public native void java.io.FileDescriptor.sync() throws java.io.SyncFailedException]
[!] Received: [Reflection => getMethod => close => public void java.io.OutputStream.close() throws java.io.IOException]
[!] Received: [Reflection => getMethod => getAbsolutePath => public java.lang.String java.io.File.getAbsolutePath()]
[!] Received: [Reflection => getMethod => getAbsolutePath => public java.lang.String java.io.File.getAbsolutePath()]
[!] Received: [Reflection => forName => dalvik.system.DexFile]
[!] Received: [Reflection => getMethod => loadDex => public static dalvik.system.DexFile dalvik.system.DexFile.loadDex(java.lang.String,java.lang.String,int) throws ja
va.io.IOException]
[!] Received: [Reflection => forName => dalvik.system.DexFile]
[!] Received: [Reflection => getMethod => loadClass => public java.lang.Class dalvik.system.DexFile.loadClass(java.lang.String,java.lang.ClassLoader)]
[!] Received: [Reflection => forName => dalvik.system.DexFile]
[!] Received: [Reflection => getMethod => close => public void dalvik.system.DexFile.close() throws java.io.IOException]
[!] Received: [Reflection => getMethod => delete => public boolean java.io.File.delete()]
[!] Received: [Reflection => getMethod => delete => public boolean java.io.File.delete()]
[!] Received: [Reflection => forName => javax.crypto.Cipher]
[!] Received: [Reflection => getMethod => getInstance => public static final javax.crypto.Cipher javax.crypto.Cipher.getInstance(java
ows java.security.NoSuchAlgorithmException,java.security.NoSuchProviderException,javax.crypto.NoSuchPaddingException]
[!] Received: [Reflection => forName => javax.crypto.spec.SecretKeySpec]
[!] Received: [Reflection => forName => javax.crypto.spec.IvParameterSpec]
[!] Received: [Reflection => forName => javax.crypto.Cipher]
[!] Received: [Reflection => forName => java.security.Key]
[!] Received: [Reflection => forName => java.security.spec.AlgorithmParameterSpec]
[!] Received: [Reflection => getMethod => init => public final void javax.crypto.Cipher.init(int,java.security.Key,java.secu
va.security.InvalidKeyException,java.security.InvalidAlgorithmParameterException]
[!] Received: [Reflection => forName => javax.crypto.Cipher]
[!] Received: [Reflection => getMethod => doFinal => public final byte[] javax.crypto.Cipher.doFinal(byte[],int,int) throws javax.crypto.IllegalBlockSizeException,java
x.crypto.BadPaddingException]
[!] Received: [Reflection => forName => java.util.zip.Inflater]
[!] Received: [Reflection => forName => java.util.zip.Inflater]
```

Android Code Obfuscation

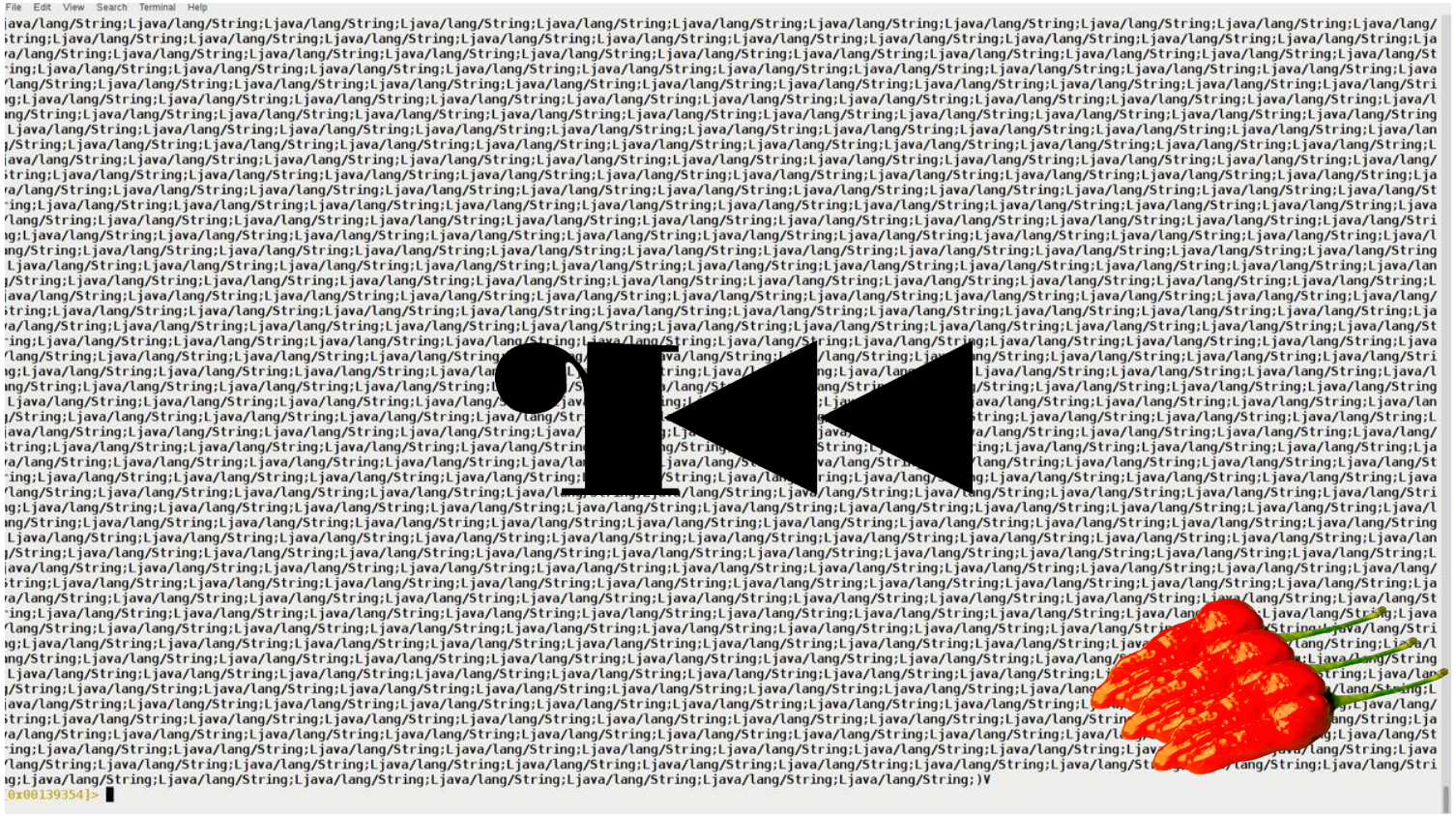
Anti-disassembly tricks

o.We

```
    }

    protected java.lang.String findLibrary(java.lang.String r9) {
        /* JADX: method processing error */
/*
Error: jadx.core.utils.exceptions.JadxRuntimeException: Unreachable block: B:5:0x0011
     at jadx.core.dex.visitors.blocksmaker.BlockProcessor.modifyBlocksTree(BlockProcessor.java:248)
     at jadx.core.dex.visitors.blocksmaker.BlockProcessor.processBlocksTree(BlockProcessor.java:52)
     at jadx.core.dex.visitors.blocksmaker.BlockProcessor.visit(BlockProcessor.java:38)
     at jadx.core.dex.visitors.DepthTraversal.visit(DepthTraversal.java:31)
     at jadx.core.dex.visitors.DepthTraversal.visit(DepthTraversal.java:17)
     at jadx.core.ProcessClass.process(ProcessClass.java:34)
     at jadx.api.JadxDecompiler.processClass(JadxDecompiler.java:296)
     at jadx.api.JavaClass.decompile(JavaClass.java:62)
*/
        /*
        r8 = this;
        goto L_0x0019;
    L_0x0001:
        r7 = move-exception;
        r0 = new java.lang.RuntimeException;
        r0.<init>(r7);
        throw r0;
    L_0x0008:
        r0 = 61;
        r2 = 61;
        $$11 = r2;
        goto L_0x0012;
        goto L_0x0013;
    L_0x0012:
        goto L_0x001c;
    L_0x0013:
        goto L_0x001c;
    L_0x0014:
        r0 = move-exception;
        r0 = r0.getCause();
    L_0x0019:
        r1 = o.We.class;
        goto L_0x0008;
    L_0x001c:
        r5 = 0;      Catch:{ all -> 0x0014 }
        r6 = r1.getClassLoader();   Catch:{ all -> 0x0014 }
        r0 = r6.getClass();    Catch:{ Exception -> 0x0001 }
        r1 = "findLibrary";    Catch:{ Exception -> 0x0001 }
        r2 = 1;      Catch:{ Exception -> 0x0001 }
24      r2 = new java.lang.Class[r2];    Catch:{ Exception -> 0x0001 }
24      r3 = java.lang.String.class;     Catch:{ Exception -> 0x0001 }
24      r4 = 0;      Catch:{ Exception -> 0x0001 }
24      r2[r4] = r3;      Catch:{ Exception -> 0x0001 }
24      r7 = r8.b(r0, r1, r2);      Catch:{ Exception -> 0x0001 }
24      r0 = 1;      Catch:{ Exception -> 0x0001 }
        r7.setAccessible(r0);  Catch:{ Exception -> 0x0001 }
```

Android Code Obfuscation

Anti-disassembly tricks



NowSecure

Android Code Obfuscation

Renaming

```
int v0_1 = 0;
while(v5 != null) {
    int v2 = v0_1 + 1;
    this.
    int[] v1_1 = this.
    int v4 = v2 + 1;
    int v0_2 = v5.
    v1_1[v2] = v0_2;
    int v1_2 = v4 + 1;
    this.
    int v2_1 = v1_2 + 1;
    this.
    int v1_3 = v2_1 + 1;
    this.
    int v2_2 = v1_3 + 1;
    this.
    if(v5.
        v0_1 = v2_2 + 1;
        this.
    }
    else {
        int v6 = v5.
        int v1_4 = v2_2 + 1;
        this.
        int v2_3 = 0;
        while(v2_3 < v6) {
            this.
            ++v2_3;
            ++v1_4;
        }

        v0_1 = v1_4;
    }

    v5 = v5.
}

this.
this.
this.
this.
this.
this.
```

NowSecure™

Native Code Obfuscation

String encryption

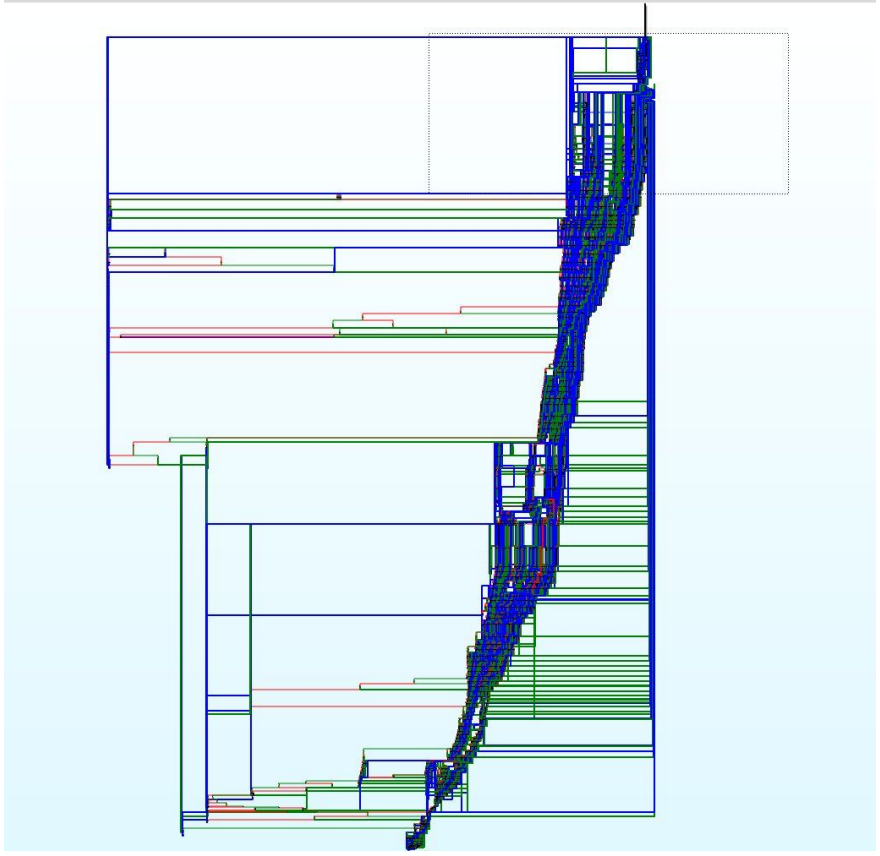
```
34  char v117; // r2
35  int v118; // r4
36  int (__fastcall *v119)(int, int); // r5
37  int v120; // r0
38  int v122; // [sp+17Ch] [bp+Ch]
39
40  v86 = a1;
41  a26 = -1836023616;
42  BYTE2(a29) = -91;
43  a27 = 1980321002;
44  a28 = 21516214;
45  a24 = 1535148118;
46  a23 = -588862745;
47  a25 = -1643676290;
48  LOWORD(a29) = 2347;
49  while ( (unsigned __int8)a23 != 238 )
50  {
51    *((_BYTE *)&a23 + (unsigned __int8)a23 % 0x1Au + 1) += 3;
52    LOBYTE(a23) = a23 + 1;
53  }
54  a11 = 622181664;
55  a12 = 1390984027;
56  a13 = 355746589;
57  a14 = -2112483426;
58  a15 = 9;
59  while ( (unsigned __int8)a11 != 40 )
60  {
61    *((_BYTE *)&a11 + (a11 & 0xF) + 1) += 3;
62    LOBYTE(a11) = a11 + 1;
63  }
64  a36 = 848482653;
65  a35 = 1986857018;
66  a31 = 777837714;
67  a32 = -1217557589;
68  a37 = 9969;
69  a30 = -1206877002;
70  a33 = -1791457705;
71  a34 = -1080671916;
72  while ( (unsigned __int8)a30 != 183 )
73  {
74    *((_BYTE *)&a86 + (unsigned __int8)a30 % 0x1Du - 227) += 3;
75    LOBYTE(a30) = a30 + 1;
76  }
77  v87 = (unsigned   int8)a32 ^ BYTE2(a35);
```

NowSecure™

Native Code Obfuscation

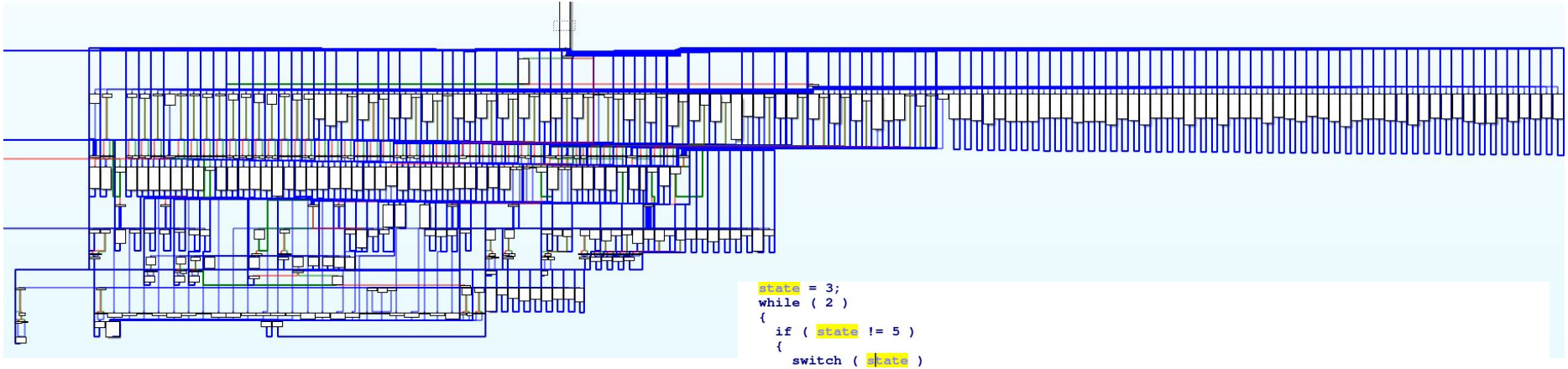
Junk code

Opaque predicates



Native Code Obfuscation

Control Flow Graph flattening



```
state = 3;
while ( 2 )
{
  if ( state != 5 )
  {
    switch ( state )
    {
      case 0:
        v80 = (0x38B09FBF * v54) >> 17;
        v72 += 0x65788BD2;
        state = -v80 & 7;
        continue;
      case 1:
        v72 *= 2;
        v72 &= 0xE27D31AD;
        goto LABEL_27;
      case 2:
        state = (unsigned __int64)sub_545B4(v130, *(_QWORD *)(v51 + 8), &v147, 2LL) != 0;
        v54 = 0x86E65A89;
        continue;
      case 3:
        v72 *= 0x69641CFA;
        state = 2;
        continue;
      case 4:
        v72 *= 2;
        state = 5;
        continue;
      default:
        continue;
    }
  }
  break;
```

Native Code Obfuscation

Anti-disassembly tricks

```
0xc774a [gp]
; CODE XREF from fcn.000c7684 (0xc772c)
; CODE XREF from syscall.0.1 (+0xd2)
0x000c774a 0020          movs r0, 0
0x000c774c 0d90          str r0, [sp + local_34h]
0x000c774e 0d9b          ldr r3, [sp + local_34h]
; 0xc7768
0x000c7750 05a0          adr r0, 0x14
0x000c7752 00ea0301      and.w r1, r0, r3
0x000c7756 4ff00202      mov.w r2, 2
0x000c775a 02fb01f1      mul r1, r2, r1
0x000c775e      .dword 0x0003ea80 ; aav.0x0003ea80
0x000c7762 0844          add r0, r1
0x000c7764 8746          mov pc, r0
```

```
0xc7bbe [gAv]
; CODE XREFS from fcn.000c7684 (0xc7b22, 0xc7bb4)
0x000c7bbe 4e98          ldr r0, [sp + local_138h]
0x000c7bc0 0121          movs r1, 1
0x000c7bc2 5090          str r0, [sp + local_140h]
0x000c7bc4 0128          cmp r0, 1
;-- aav.0x000c7bc6:
; UNKNOWN XREF from aav.0x00542704 (+0x18)
0x000c7bc6      .dword 0x0004f04f ; aav.0x0004f04e
0x000c7bca 08bf          it eq;[gBk]
```

f t

```
0xc7bcc [gBk]
; CODE XREF from fcn.000c7684 (0xc7bca)
0x000c7bcc 0221          movs r1, 2
```

NowSecure™

Native Code Obfuscation

Obfuscated function calls

Functions window

Function name	Segi
ObfuscatedCall<ObfuscatedAddress<void (*)(void)>>(...	.text
ObfuscatedAddress<void (*)(char *,int)>::original(void)	.text
ObfuscatedAddress<int (*)(void *,int)>::original(void)	.text
ObfuscatedAddress<void (*)(char *)>::original(void)	.text
strlen	.text
pAB7771827B1F51FA3E703EE417D1A4FE(char *,func_info...	.text
pDFB3268AF3FB1EBD7DCB3D5976048615(char *,func_inf...	.text
p88113D1ADA765D25AE4E7A343F98DAA8(char const*,fu...	.text
find_dexhunter_feature(void)	.text
sub_15E84	.text
ObfuscatedAddress<void (*)(_JNIEnv *)>::original(void)	.text
ObfuscatedAddress<void (*)(char *,char *,int,int)>::origi...	.text
ObfuscatedAddress<void (*)(char)>::original(void)	.text
ObfuscatedAddress<void (*)(void *,int,int)>::original(voi...	.text
JNI_OnLoad	.text
sub_19328	.text
sub_19A08	.text
sub_19AF4	.text
sub_19BDC	.text
sub_19C98	.text
pD1C5995441BDE309801AA2D6599D7D72	.text
p7F782C775179F8B6311393E00D93C5CF	.text
p75BAEF8CD621B5EF6A86CD883CFFE175	.text
pE4179E5B7B934B31C6DA2EFACBE38B40	.text

Line 196 of 1046

Graph overview

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1 | Structures | Enums

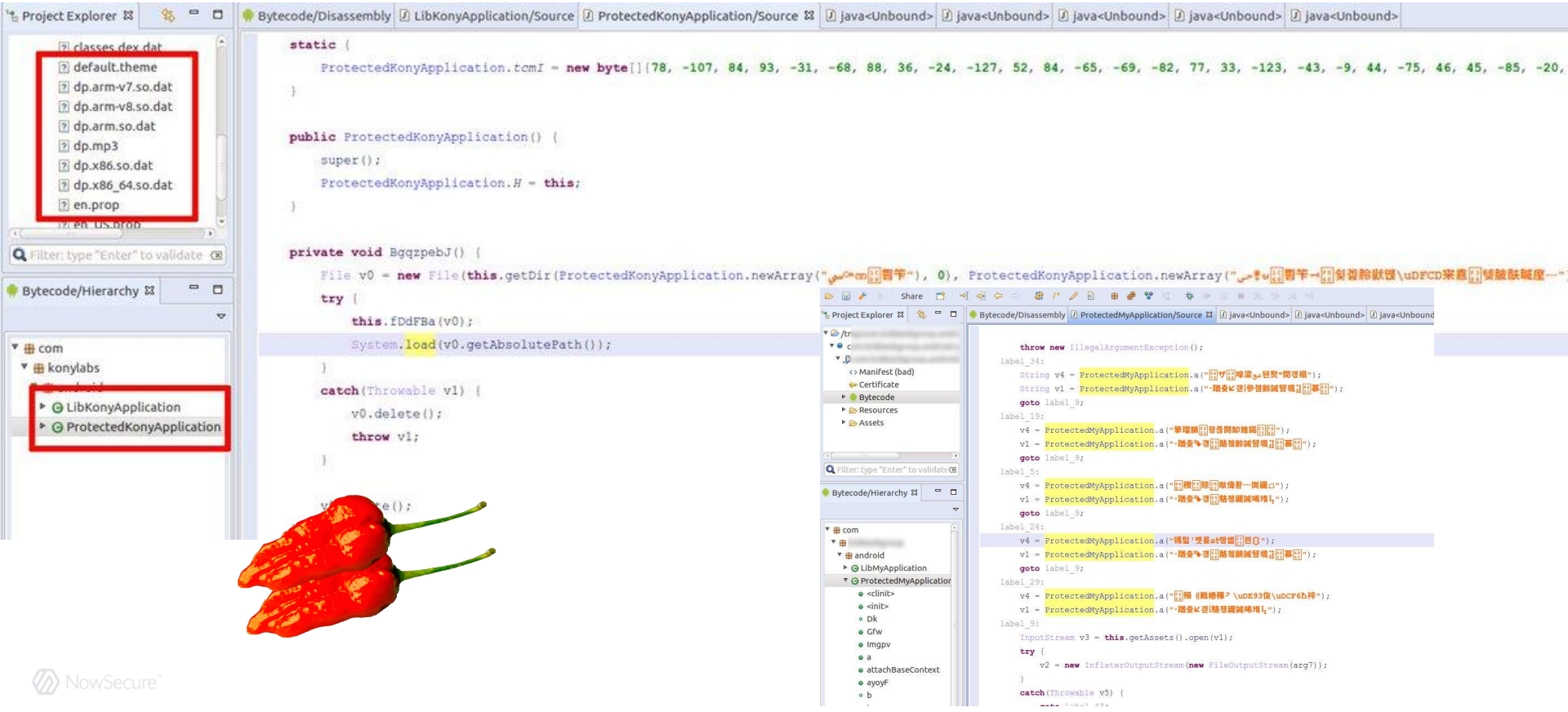
```
2666          }
2667        }
2668        if ( byte_4E088[4] )
2669          goto LABEL_150;
2670        if ( p3A986CD02384F03165D40910A7CD9F37 <= 0 )
2671        {
2672          if ( p87A0F4CD3F3E115EB8DF9B8F839AF245 == 1 )
2673          {
2674            v199 = &loc_1A6A4;
2675            v200 = 288;
2676            v43 = ObfuscatedCall<ObfuscatedAddress<void (*)(void)>>((char *)&loc_1A6A4 + 1, 288);
2677            goto LABEL_74;
2678          }
2679          v219 = &loc_DCF6;
2680          v220 = 449;
2681          v41 = (int (__fastcall *)(void *))ObfuscatedAddress<void (*)(_JNIEnv *)>::original(&v219);
2682          v42 = v194;
2683        }
2684        else
2685        {
2686          v217 = sub_41668;
2687          v218 = 527;
2688          v41 = (int (__fastcall *)(void *))ObfuscatedAddress<void (*)(char *)>::original();
2689          v42 = needle;
2690        }
2691        v43 = v41(v42);
2692LABEL_74:
2693        p75BAEF8CD621B5EF6A86CD883CFFE175(v43);
2694        memset(&src, 0, 0x15u);
2695        v667 = -36;
2696        v659 = -47;
2697        v662 = -43;
2698        v668 = -47;
```

000169F2 JNI_OnLoad:2681 (169F2)

NowSecure

Android Packers

Dynamic code loading



Android Code Protectors

Too many

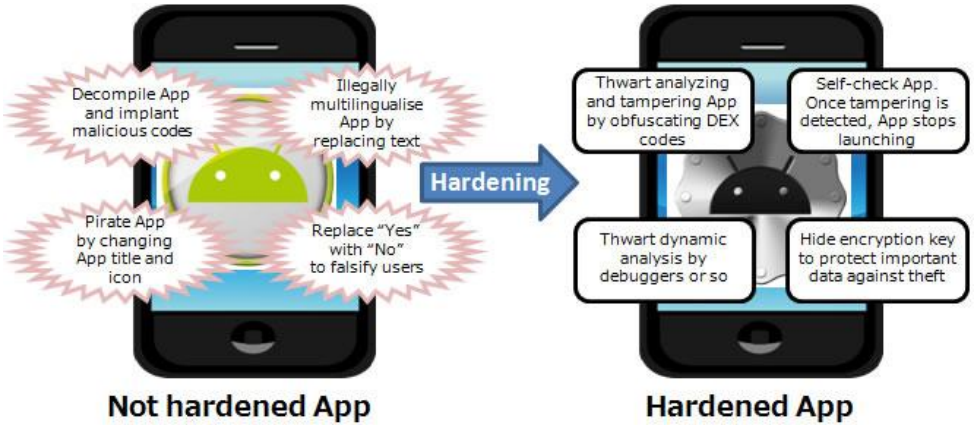
- Arxan
- Appdome
- ADVObfuscator
- Allatori
- AppSuit
- DexGuard
- DexProtector
- MetaFortress
- Firehash
- Obfuscator-LLVM
- AliPay obfuscator
- Promon
- Gemalto Mobile Protector
- DxShield
- Amazon Kiwi
- WhiteCryption
- Baidu
- ....
- Secenh
- SecShell
- SecNeo
- Bangcle
- AppSealing
- AppSuit
- ApkGuard
- AppGuard
- APKProtect
- AppSolid
- ApkToolPlus (Jiagu)
- Gaoxor
- ChornClickers
- UPX
- Kiro
- NQ Shield
- Medusah
- Qihoo 360
- Yidun ...

DEMOS

Takeaways

Before we go

- APKiD identifies **tampered** (repackaged), packed, obfuscated and **protected** Android apps
- **Automation** (multiple instances) → APKiD can store JSON data into DB
- Yara rules are **fast**→ **Combine** static + dynamic analysis
- APKiD can be easily **extended** for your own purposes
- Easy to reuse rules that might be applicable to **other** environments→ iOS, ELF binaries, …
- Once a new obfuscator rule released → Vendors often modify codebases to **avoid** detection



THANK YOU!
Q&A

Eduardo Novella
Mobile Security Research Engineer

enovella@nowsecure.com
@NowSecureMobile
@enovella_

Chat on:
rednaga.slack.com