APKID:
PEID FOR ANDROID APPS

EDUARDO NOVELLA

2018.12.06

BlackHat Europe London 2018

WHAT'S REDNAGA MATE?

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

WHO ARE WE

EDU

CALEB

DIFF

- @enovella_ @timstrazz @Caleb_Fenton

github.com/rednaga/APKiD

WHO AM I

- Sr. Security Consultant @ **Synopsys** (UK)
  - Mobile: Android Fingerprint (TEE RE), RASP
  - Embedded: RE/fuzzing/expl
- Previously @ Riscure (NL)
  - SW: Android Mobile Payment (HCE) & DRM
    - Smart-cards, -meters, IOT, …
  - HW: Side channel & Fault injection attacks
- **Likes**:
  - RE & Embedded & Exploiting & Crypto
  - Read/Write Source Code
  - Automate/Hook/Pwn/Outsmart stuff
- **Hobbies**:
  - Chess player, swimming, biking, running, nature, cooking, traveling

WHY ARE WE HERE

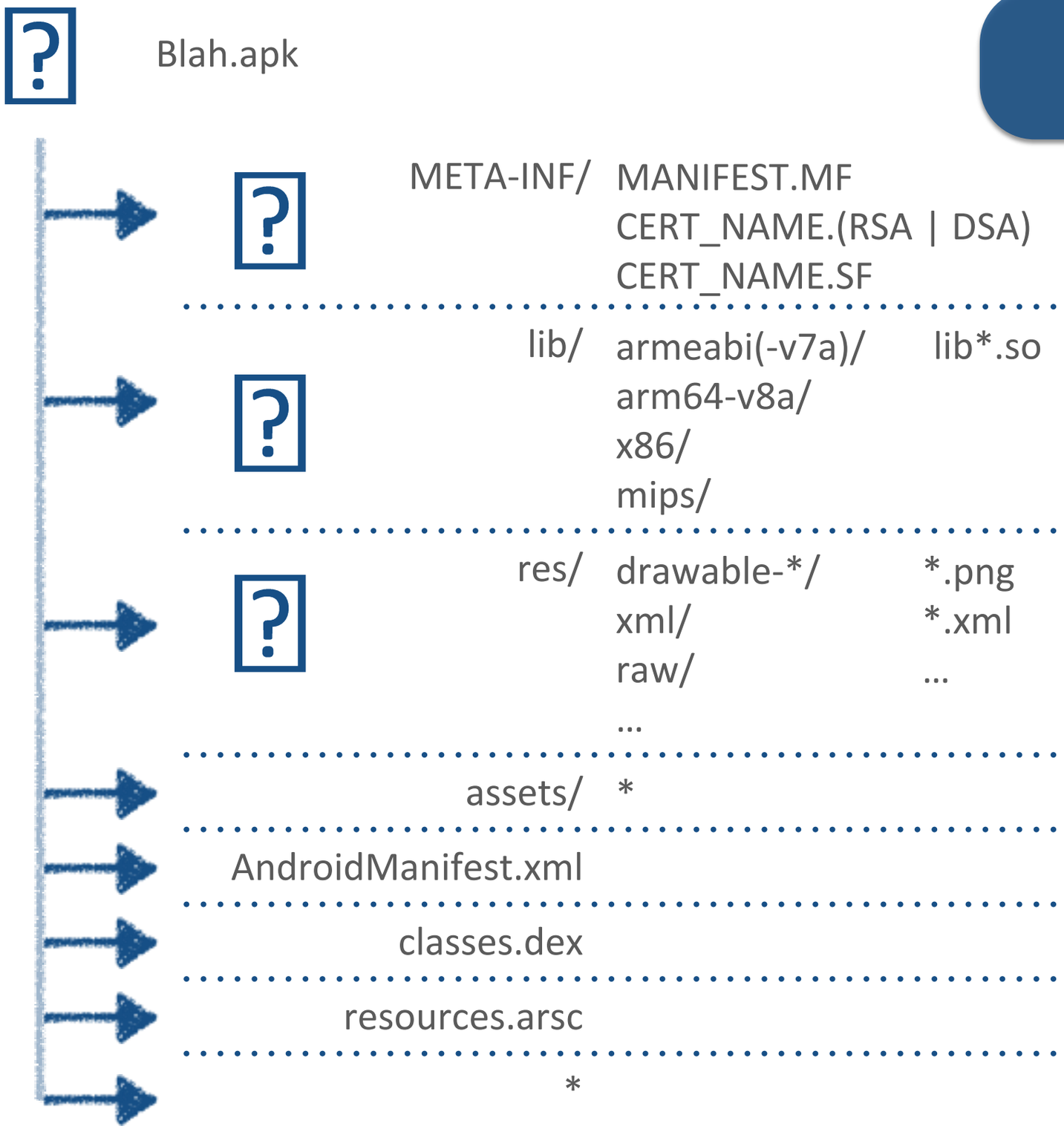
- Showcase **APKiD**; *The "PEID" of Android Apps*
- Identification of mobile security products, piracy, and how APK was built
- **APKiD** used for many purposes:
  - Which apps could potentially contain malware? Tampered?
  - **Which apps are obfuscated/packed? Which protector?**
  - Has the DEX been compiled in unusual ways? Anomalies?
  - Anti-debugging, anti-emulation, …

HOW APKID WORKS

The Basics

ANDROID APPLICATION PACKAGING (APK)

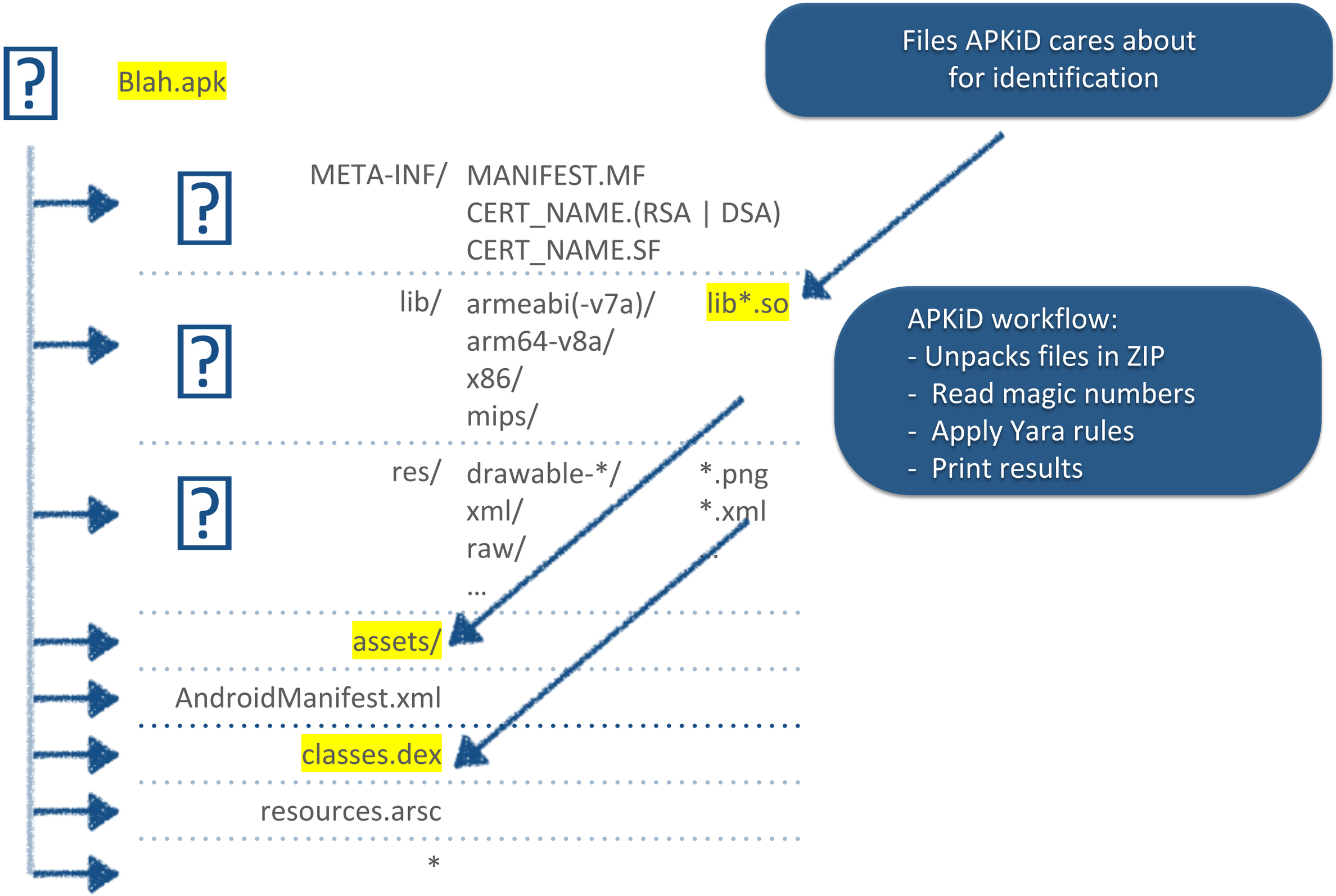
application/vnd.android.package-archive



APK = ZIP

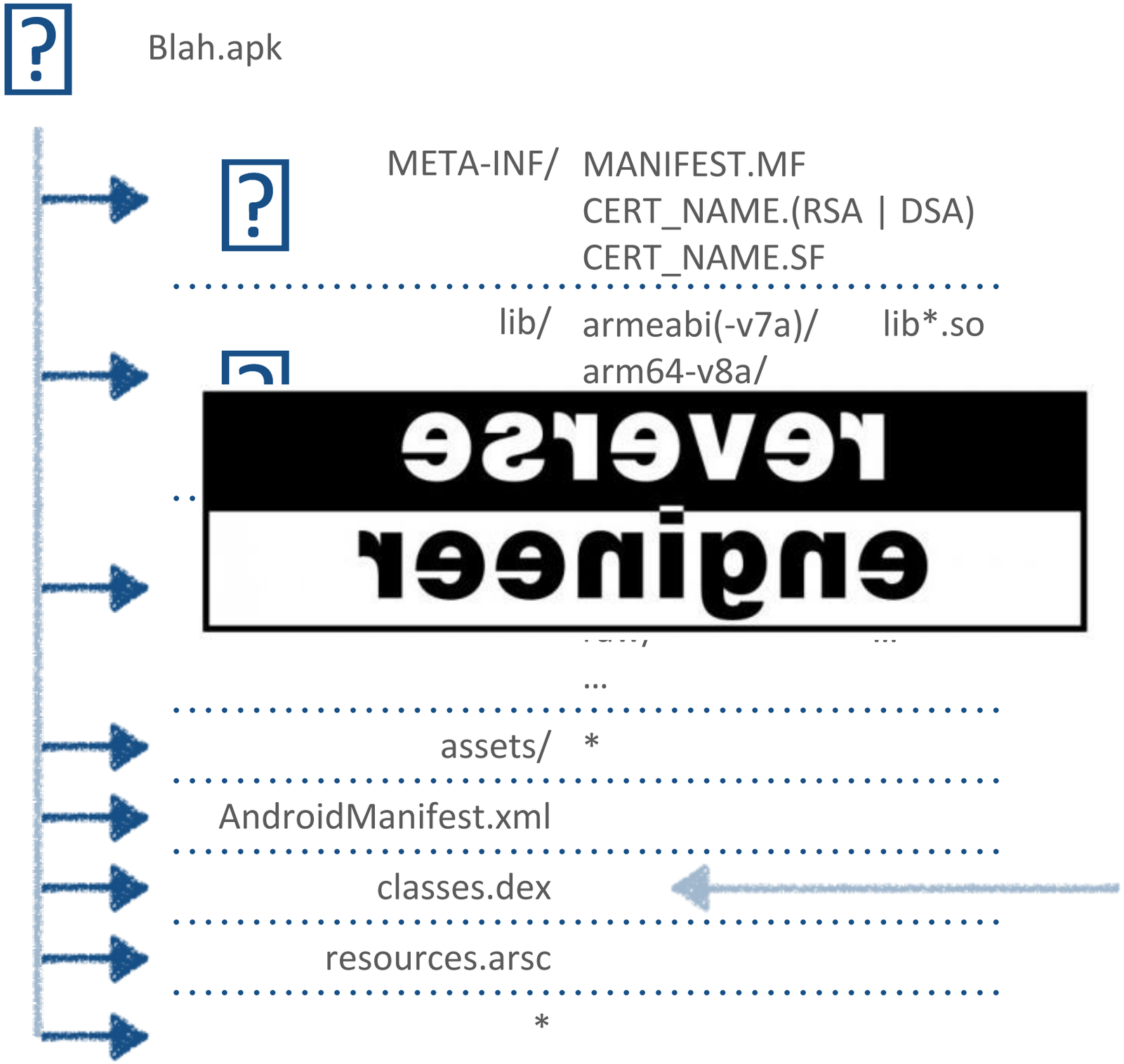
ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



ANDROID APPLICATION PACKAGING (APK)

application/vnd.android.package-archive



Dalvik EXecutable (DEX)
Compiled classes for DVM

Contains executable Dalvik code

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

Reverse with:
- smali / apktool
- IDA Pro
- Radare2
- Frida (dynamic)
- JEB
- Androguard
- Enjarify
- dex2jar + jad/jd
- JADX
- 010Editor Templates

MALWARE AND PIRACY DETECTION

Initial uses of APKiD
diff-caleb

THE QUESTION

Three main compilers:

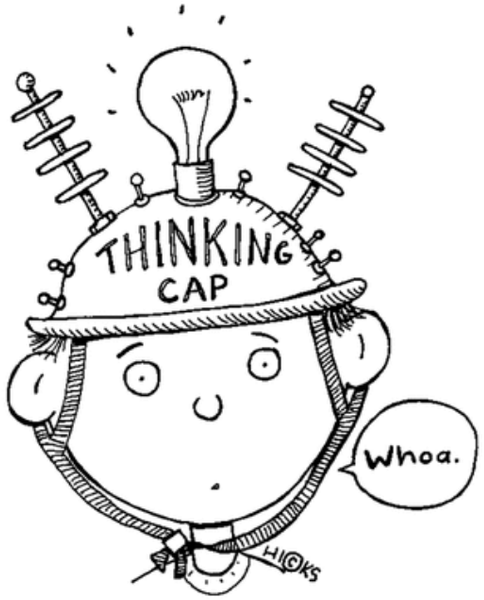
1. dx ← Java .class files (source code)
2. dexmerge ← Not used manually, only by IDEs (source code)
3. smali (**dexlib**) ← DEX files (not source code)

Why would a legitimate developer ever need to use ***smali***?

They have the source.

THE HYPOTHESIS

- If app compiled with dexlib, probably tampered
- If tampered, probably was not the developer
- Tampered apps are likely either:
  - pirated / cracked
  - malware



∴ app is tampered -> app is interesting

SAMPLE SET

- 20,000 APKs from each market
  - Top Play Apps, Aptoide, BlapkMarket, etc.
- 10,000 highest scoring “fraudulent” apps
  - Scored by experimental statical model
  - Fraud may just mean modified XML (not DEX)
- Up to 10 APKs per variant of all malware families

Further info: Android Compiler Fingerprint @ HITCON 2016

UNDERSTANDING OBFUSCATION SPICINESS

Spiciness levels = Obfuscation layers

ANDROID OBFUSCATORS / PACKERS

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 OBFUSCATORS / PACKERS

- **APKiD** has recently been enhanced to detect:
  - If a **DEX** file was altered or tampered (based on compiler fingerprint)
  - A bunch of **DEX/ELF obfuscators** (pattern, fingerprints and opcodes)
  - Tons of **packers** (hidden dex, dexclassloader, assets, …)

ANDROID OBFUSCATORS / PACKERS

- Arxan
- ADVObfuscator
- Allatori Demo
- AppSuit
- DexGuard
- DexProtector
- MetaFortress
- Firehash
- Obfuscator-LLVM
- AliPay obfuscator
- MD5 obfuscator
- Promon
- Gemalto Mobile Protector
- DxShield
- Amazon Kiwi
- Secenh, SecShell, SecNeo, Bangcle, …
- AppSealing
- AppSuit
- ApkGuard
- AppGuard
- APKProtect, AppSolid
- ApkToolPlus (Jiagu)
- Promon
- Gaoxor, ChornClickers
- UPX
- Kiro
- NQ Shield
- Medusah
- Qihoo 360, Baidu
- Yidun

ANDROID OBFUSCATION

```
        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'
    ((Map)v2_1).put(ffiiii.a("A=<@*62;;", 'e', '\u0001'), v0_1.c());
    ((List)v1).add(v2_1);
    return ((List)v1);
```

- **String encryption**
- **Class renaming**

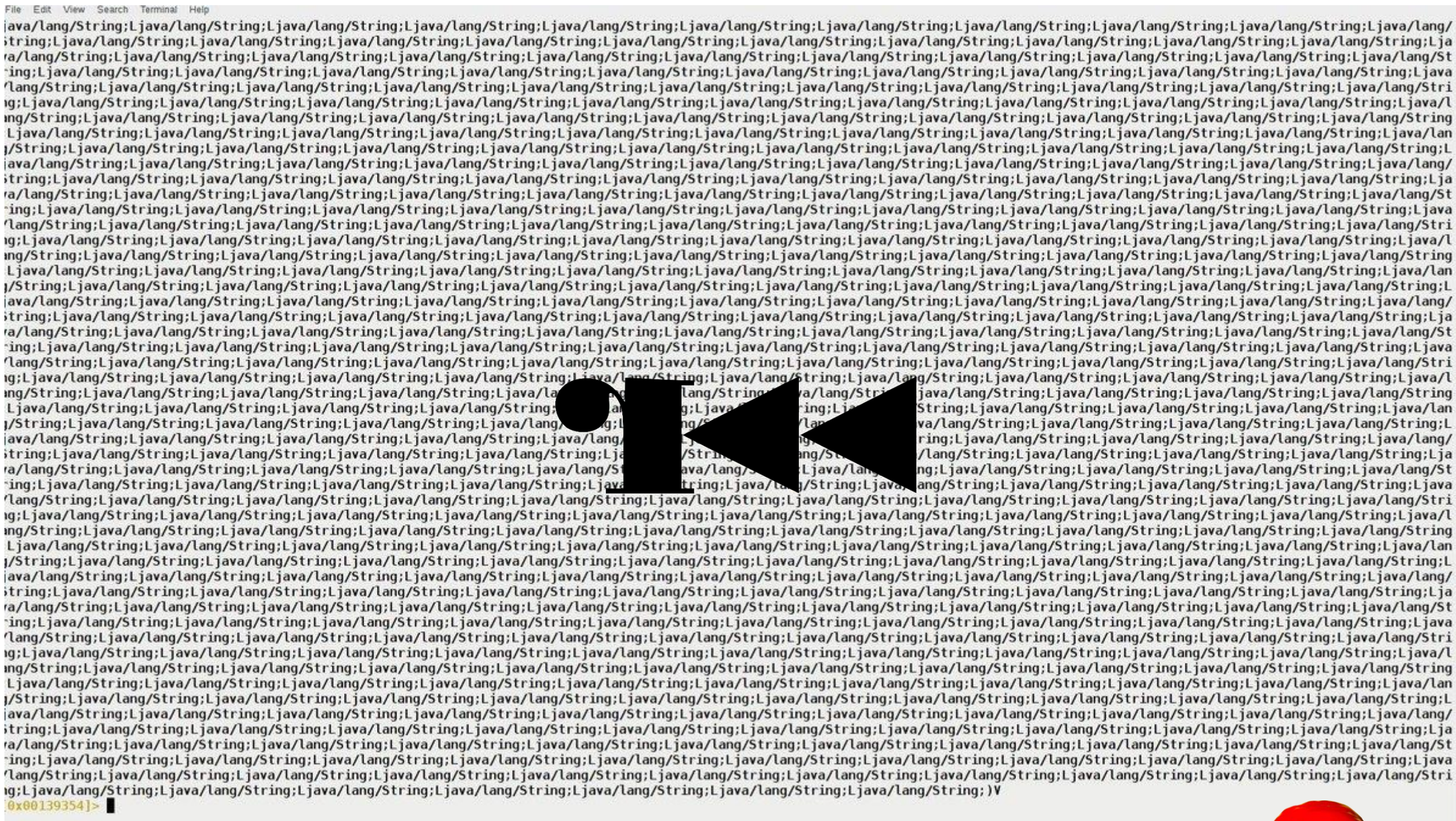
ANDROID OBFUSCATION

```
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$a.e[6])));
        Class[] array_class = new Class[3];
        array_class[0] = Integer.TYPE;
        array_class[1] = Class.forName(QC$a.j(0x63, ((byte)(-QC$a.e[6])), ((byte)QC$a.e[0x8F])));
        byte b = ((byte)(-QC$a.e[6]));
```

JEB

- **Class encryption**

ANDROID OBFUSCATION



Kudos to **Radare2** to resist spicy Chinese obfuscation ☺

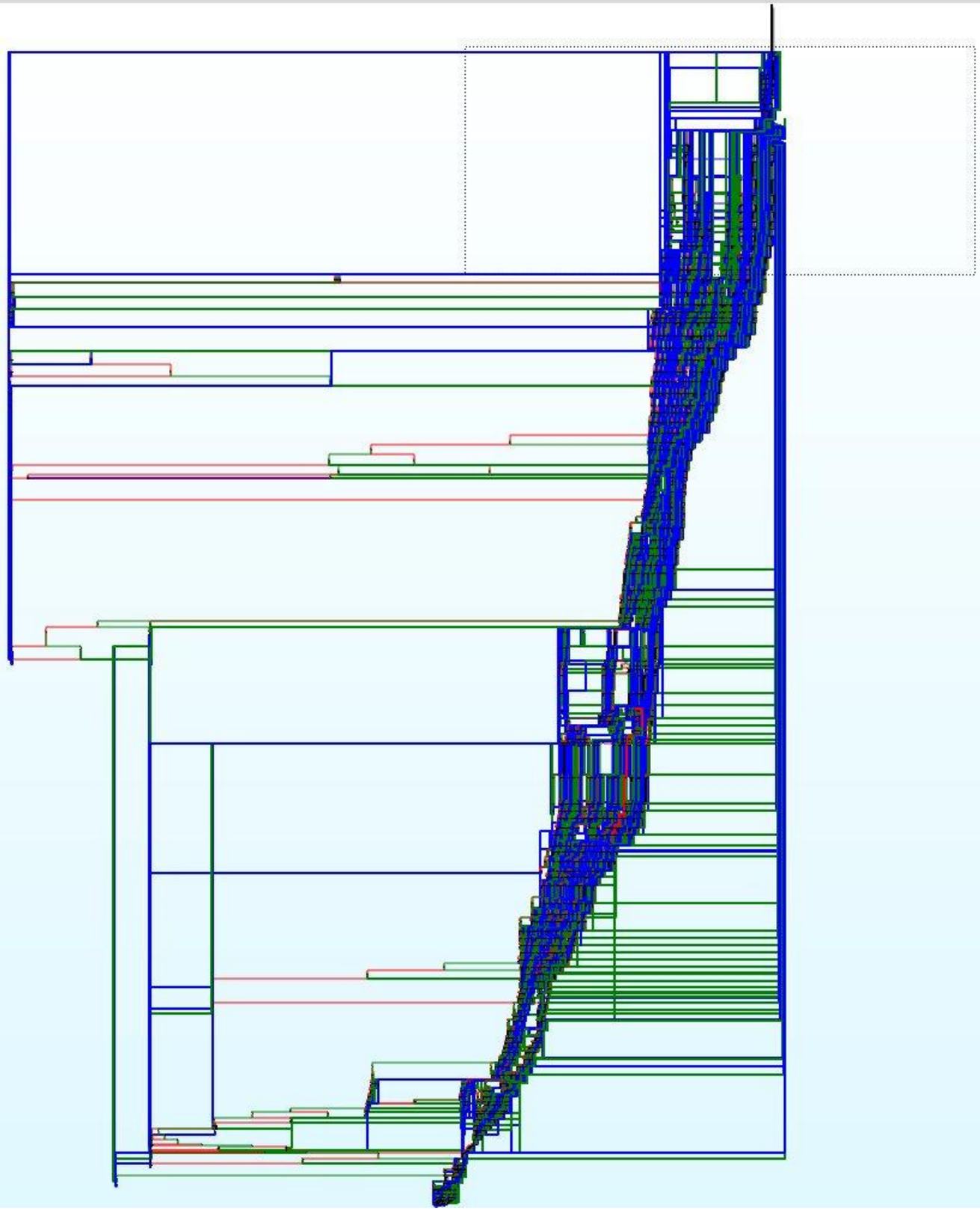
NATIVE OBFUSCATION

```
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);
```

Hex-Rays
state-of-the-art code analysis

- **String encryption**

NATIVE OBFUSCATION



NATIVE OBFUSCATION

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

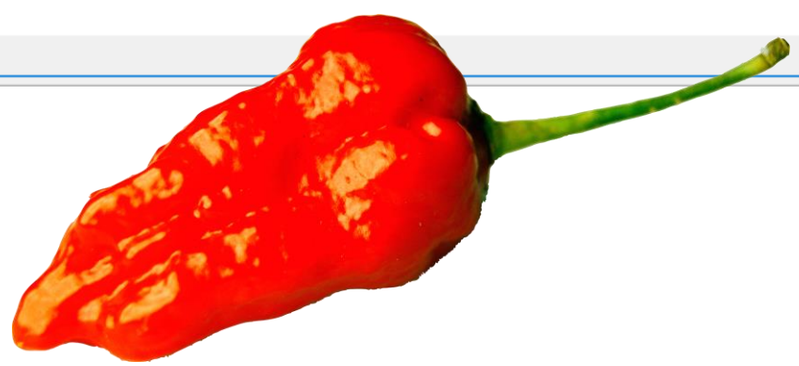
Line 196 of 1046

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)



NATIVE OBFUSCATION

```
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
```

ANDROID UNPACKING

```
[!] Received: [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.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)]
```

FRIDA

ANDROID UNPACKING

Project Explorer

- classes.dex.dat
- default.theme
- dp.arm-v7.so.dat
- dp.arm-v8.so.dat
- dp.arm.so.dat
- dp.mp3
- dp.x86.so.dat
- dp.x86_64.so.dat
- en.prop
- en_US.prop

Filter: type "Enter" to validate

Bytecode/Hierarchy

- com
  - konylabs
    - LibKonyApplication
    - ProtectedKonyApplication

Bytecode/Disassembly | LibKonyApplication/Source | ProtectedKonyApplication/Source | java<Unbound> | java<Unbound> | java<Unbound> | java<Unbound> | java<Unbound>

```
static {
    ProtectedKonyApplication.tcmI = new byte[]{78, -107, 84, 93, -31, -68, 88, 36, -24, -127, 52, 84, -65, -69, -82, 77, 33, -123, -43, -9, 44, -75, 46, 45, -85, -20,
}

public ProtectedKonyApplication() {
    super();
    ProtectedKonyApplication.H = this;
}

private void BgqzpebJ() {
    File v0 = new File(this.getDir(ProtectedKonyApplication.newArray("[illegible]"), 0), ProtectedKonyApplication.newArray("[illegible]"
```

Share

Project Explorer

- /tr[illegible]
  - c[illegible]
    - Manifest (bad)
    - Certificate
    - Bytecode
    - Resources
    - Assets

Filter: type "Enter" to validate

Bytecode/Hierarchy

- com
  - [illegible]
    - android
      - LibMyApplication
      - ProtectedMyApplicatior
        - <clinit>
        - <init>
        - Dk
        - Gfw
        - Imgpv
        - a
        - attachBaseContext
        - ayoyF
        - b

Bytecode/Disassembly | ProtectedMyApplication/Source | java<Unbound> | java<Unbound> | java<Unbound

```
        throw new IllegalArgumentException();
    label_34:
        String v4 = ProtectedMyApplication.a("[illegible]");
        String v1 = ProtectedMyApplication.a("[illegible]");
        goto label_9;
    label_19:
        v4 = ProtectedMyApplication.a("[illegible]");
        v1 = ProtectedMyApplication.a("[illegible]");
        goto label_9;
    label_5:
        v4 = ProtectedMyApplication.a("[illegible]");
        v1 = ProtectedMyApplication.a("[illegible]");
        goto label_9;
    label_24:
        v4 = ProtectedMyApplication.a("[illegible]");
        v1 = ProtectedMyApplication.a("[illegible]");
        goto label_9;
    label_29:
        v4 = ProtectedMyApplication.a("[illegible] \uDE93仮\uDCF6h[illegible]");
        v1 = ProtectedMyApplication.a("[illegible]");
    label_9:
        InputStream v3 = this.getAssets().open(v1);
        try {
            v2 = new InflaterOutputStream(new FileOut[illegible]
        }
        catch(Throwable v5) {
```

APKID

- Can detect all you have just seen in previous slides and many others too
- Important:
    - Detect obfuscators, packers, or other oddities on Android
    - APKiD does identification only! **Does NOT RE any Android protector**

```
[15:30 edu@snps 09-advobfuscator] >  apkid f9185b2b673e794ddb77f55e3f5088b0450a87ba745047c1fed5037f80dba23c.apk
[+] APKiD 1.2.1 :: from RedNaga :: rednaga.io
[*] f9185b2b673e794ddb77f55e3f5088b0450a87ba745047c1fed5037f80dba23c.apk!classes.dex
 |-> compiler : dexlib 2.x
[*] f9185b2b673e794ddb77f55e3f5088b0450a87ba745047c1fed5037f80dba23c.apk!lib/armeabi/libDexHelper.so
 |-> obfuscator : ADVobfuscator
[*] f9185b2b673e794ddb77f55e3f5088b0450a87ba745047c1fed5037f80dba23c.apk!lib/x86/libDexHelper.so
 |-> obfuscator : ADVobfuscator
[*] f9185b2b673e794ddb77f55e3f5088b0450a87ba745047c1fed5037f80dba23c.apk!lib/armeabi-v7a/libDexHelper.so
 |-> obfuscator : ADVobfuscator
[*] f9185b2b673e794ddb77f55e3f5088b0450a87ba745047c1fed5037f80dba23c.apk
 |-> packer : SecNeo
```

APKID

```
 |-> packer : UPX (unknown, modified)
[*] ./3cd4afa5ff274415565d641e003ae640527c471b48ffdca5a8f90c7c77dab90d.apk
 |-> packer : Ijiami
[*] ./57ffea49904872d93c0f20a233616d31d7c8b5e092494adfc83d0f061ea70c68.apk!classes.dex
 |-> compiler : dexlib 2.x
[*] ./57ffea49904872d93c0f20a233616d31d7c8b5e092494adfc83d0f061ea70c68.apk
 |-> packer : SecNeo
[*] ./42e033c5546573ae9b2c33dda2312d5116a049d424b0d1acd5cb7600a9a41008.apk!classes.dex
 |-> compiler : dexlib 2.x
[*] ./42e033c5546573ae9b2c33dda2312d5116a049d424b0d1acd5cb7600a9a41008.apk
 |-> packer : SecNeo
[*] ./f51697cbbe85ef802db85997f4c6bd57edbeee2829751e34c06821590a91748c.apk!classes.dex
 |-> compiler : dexlib 2.x
[*] ./f51697cbbe85ef802db85997f4c6bd57edbeee2829751e34c06821590a91748c.apk
 |-> packer : SecNeo
[*] ./b5b7b58a116a49481630acd9d87b9c888a227c98ea35c73d628826a273ed634e.apk!classes.dex
 |-> compiler : dx
[*] ./b5b7b58a116a49481630acd9d87b9c888a227c98ea35c73d628826a273ed634e.apk
 |-> packer : Qihoo 360
[*] ./f51e058d8c6ddc66790854f9e371ed770c61e75a91fe4a2efa3a1ad6f8a5ad14.apk!classes.dex
 |-> anti_vm : Build.BOARD check, Build.MANUFACTURER check, possible Build.SERIAL check, ro.build.type che
 |-> compiler : dx
[*] ./f51e058d8c6ddc66790854f9e371ed770c61e75a91fe4a2efa3a1ad6f8a5ad14.apk!lib/arm64-v8a/libkxqpplatform.s
 |-> obfuscator : Obfuscator-LLVM version 3.5
[*] ./f51e058d8c6ddc66790854f9e371ed770c61e75a91fe4a2efa3a1ad6f8a5ad14.apk!lib/armeabi/libkxqpplatform.so
 |-> obfuscator : Obfuscator-LLVM version 3.5
[*] ./f51e058d8c6ddc66790854f9e371ed770c61e75a91fe4a2efa3a1ad6f8a5ad14.apk!lib/x86/libkxqpplatform.so
 |-> obfuscator : Obfuscator-LLVM version 3.5
[*] ./f51e058d8c6ddc66790854f9e371ed770c61e75a91fe4a2efa3a1ad6f8a5ad14.apk
[*] ./9f85b3e134ee906d0f2e65e8560c8b4d440bb378cad8fc08328636c09a2b86fa.apk!classes.dex
 |-> compiler : dexlib 2.x
[*] ./9f85b3e134ee906d0f2e65e8560c8b4d440bb378cad8fc08328636c09a2b86fa.apk
 |-> packer : SecNeo
[*] ./c422f79810fef9006b775809079ebb48f25f3d4fbca3ba3f0065f743c65d7af3.apk!classes.dex
 |-> compiler : dx
[*] ./c422f79810fef9006b775809079ebb48f25f3d4fbca3ba3f0065f743c65d7af3.apk
 |-> packer : Qihoo 360
[*] ./7056c266614d2cb154b7ee851c17f6e0e32f3b270ce16df36072430d6e9a2a3d.apk!classes.dex
 |-> anti_vm : Build.FINGERPRINT check, Build.MANUFACTURER check, Build.MODEL check, Build.PRODUCT che
k, possible Build.SERIAL check, possible vm check, subscriber ID check
 |-> compiler : dx
[*] ./7056c266614d2cb154b7ee851c17f6e0e32f3b270ce16df36072430d6e9a2a3d.apk
 |-> packer : APKProtect
```

APKID DEMOS

THANKS!

EDUARDO NOVELLA
@ENOVELLA_
EDNOLO@INF.UPV.ES

Special Thanks to Caleb Fenton & Tim Strazzere!

Hats off to all the community opting for open source projects!

Join us on **Slack** at
slack.rednaga.io
on:
#rednaga
#apkid
#general
#random
#learn
#ios

06.12.2018

BLACKHAT EUROPE