Project BAOBAB

Theori

박세준

2019. 11. 07.

BAOBAB Team





THEORI

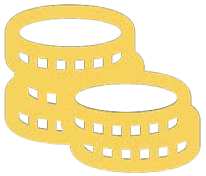
Research Project @ Theori

- Andrew Wesie
- Brian Pak
- Jinwoo Lee
- Juno Im
- Sihoon Lee
- Taeyang Lee

Background

@Theori

Global
Start-up / Corporate



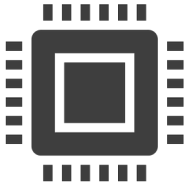
Cryptocurrency
Exchange

Financial
Institute

Game
Publisher

Blockchain as a
Service (BaaS)

Operating System
and Browsers



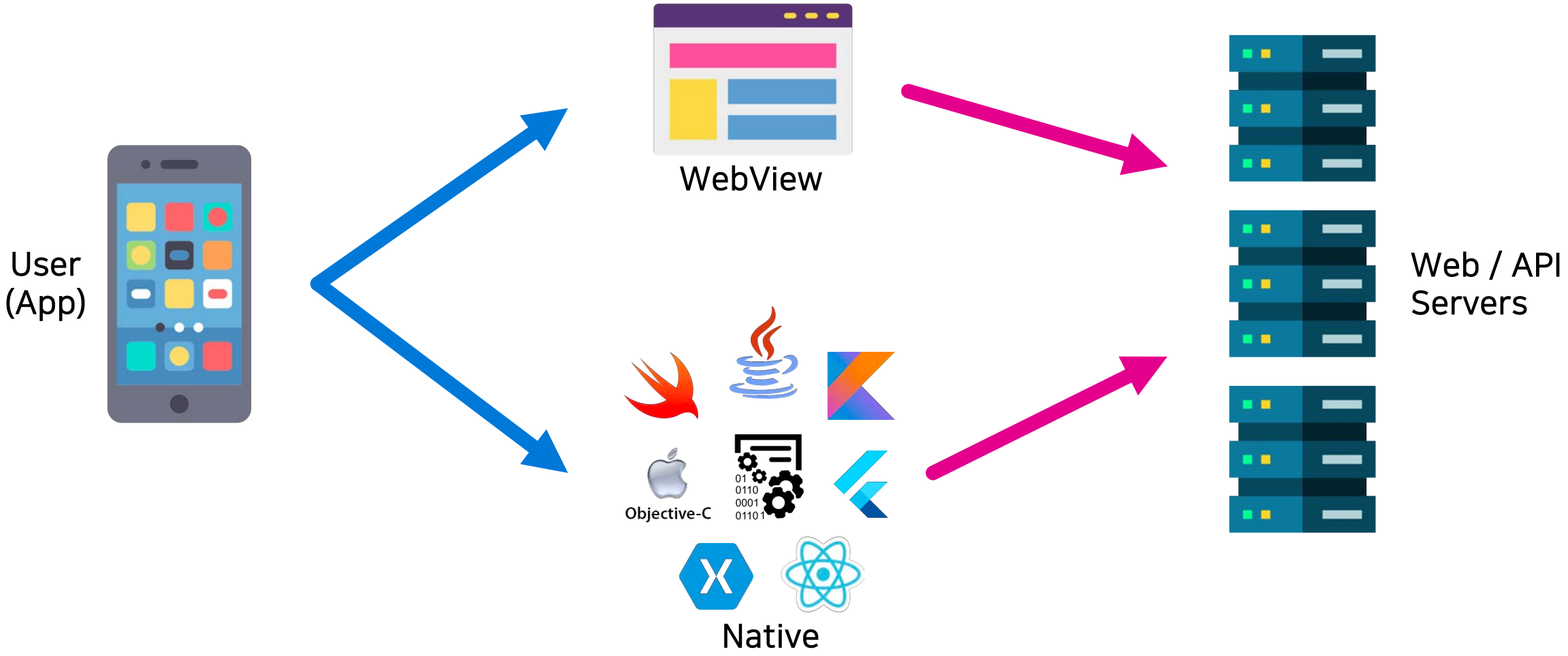
Electronics
Manufacturer

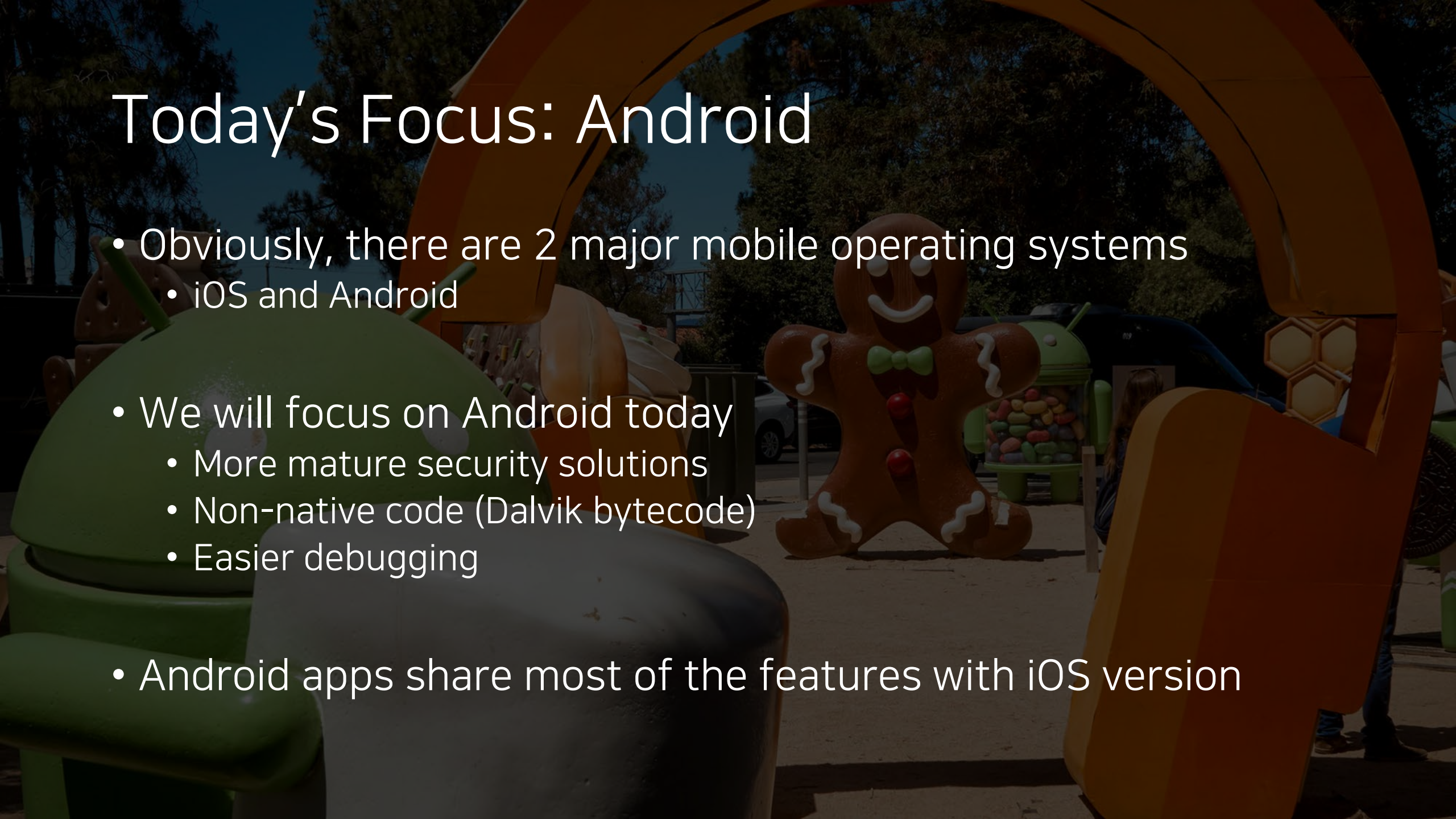
@Theori

- Many of our customers have their mobile apps
- They care about security
- They care about intellectual property

- They ask us:
  - What mobile security solution should they use?
  - Do they actually protect as they advertise?

Mobile Application





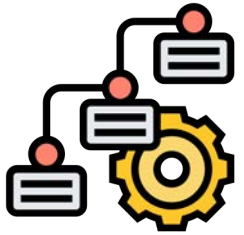
Today's Focus: Android

- Obviously, there are 2 major mobile operating systems
  - iOS and Android
- We will focus on Android today
  - More mature security solutions
  - Non-native code (Dalvik bytecode)
  - Easier debugging
- Android apps share most of the features with iOS version

Why are obfuscators used?

Protect Intellectual Property (IP)

Hide core business logic



Hide sensitive information

- Keys, tokens
- API URLs and parameters
- Intents

Stop and detect malicious users

- Repackaged games with cheats
- Backdoored banking apps

What are the features?

Hiding and removing information

- Obfuscated variable name, class name, method name
- Encrypted strings

Detection

- Rooted device / VM detection
- Debug detection

Code manipulation

- DEX, JNI encryption
- Java source, Dalvik bytecode, JNI code obfuscation

Integrity verification

- Memory checksum
- Resource checksum

Are obfuscators useful?

- Definitely increases the cost for the attackers or researchers
  - Useful for service providers (i.e. app developers)

- May help users to run the apps in safer environment
  - **Probably not** in reality
  - Running unnecessary code => More battery usage!

- Let's try to reverse engineer some apps, and we shall find out!

Let’s Try ‘em!

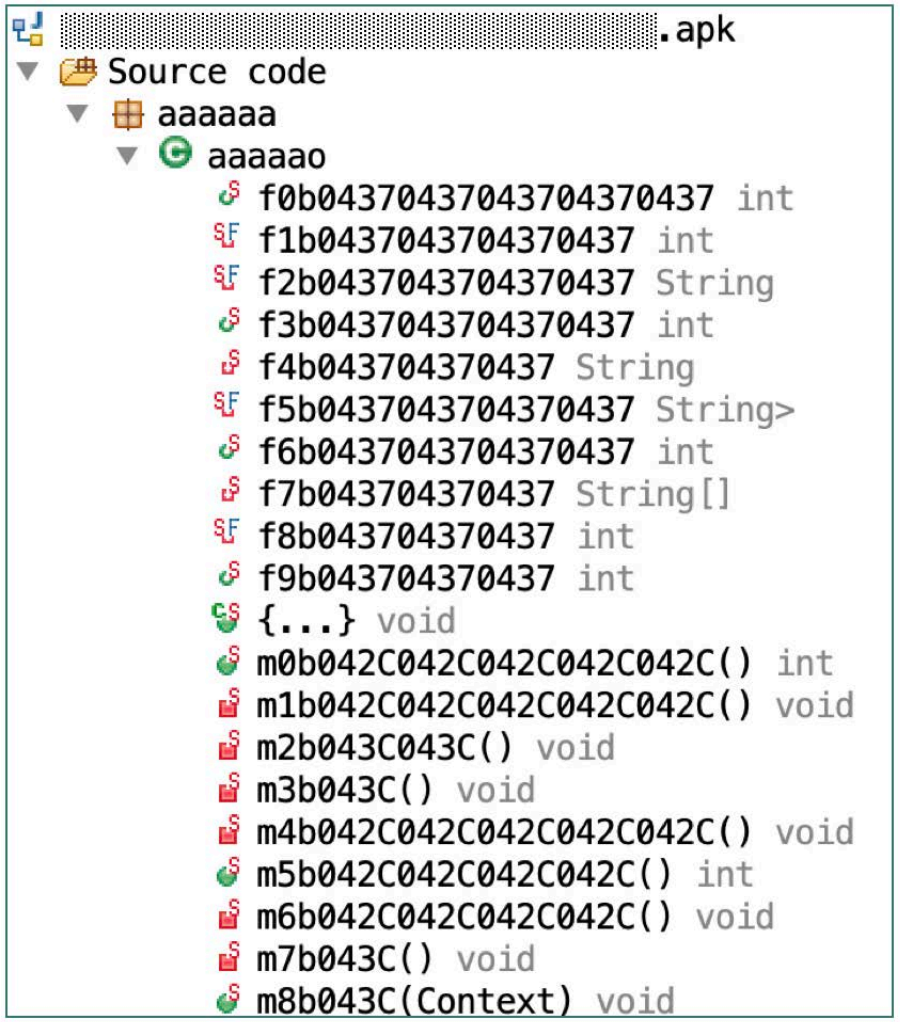
Example #1

```
/*  JADX ERROR: Method load error
    jadx.core.utils.exceptions.DecodeException: Load method exception: bogus element_width: 0000 in method
      at jadx.core.dex.nodes.MethodNode.load(MethodNode.java:139)
      at jadx.core.dex.nodes.ClassNode.load(ClassNode.java:249)
      at jadx.core.ProcessClass.process(ProcessClass.java:27)
      at jadx.api.JadxDecompiler.processClass(JadxDecompiler.java:311)
      at jadx.api.JavaClass.decompile(JavaClass.java:62)
    Caused by: jadx.core.utils.exceptions.DecodeException: bogus element_width: 0000 in method: asr.fpq(ja
      at jadx.core.dex.instructions.InsnDecoder.decodeInsns(InsnDecoder.java:55)
      at jadx.core.dex.nodes.MethodNode.load(MethodNode.java:124)
      ... 4 more
    Caused by: com.android.dex.DexException: bogus element_width: 0000
      at com.android.dx.io.instructions.InstructionCodec$36.decode(InstructionCodec.java:871)
      at jadx.core.dex.instructions.InsnDecoder.decodeRawInsn(InsnDecoder.java:70)
      at jadx.core.dex.instructions.InsnDecoder.decodeInsns(InsnDecoder.java:52)
      ... 5 more
    */
```

jadx output

- Method decompilation failure
  - baksmali, dx libraries aren’t resilient to edge cases

Example #2



- Proguard-like
  - Class / method name obfuscation
  - Unicode
  - Same names in different namespace

```
public void run() {
    if (da.iIIiiiiiii > 0) {
        da.IIIiiiIiii.removeCallbacks(da.iiiIiiiIII);
        if (da.IiiIiIiIiI == 0) {
            if (da.iIIiiiiiii >= 60) {
                da.IiiiiIiii.removeUpdates(this.IIIiiiIiii.IiiiiiiIII);
                this.IIIiiiIiii.m();
            } else {
                da.IIIiiiIiii.postDelayed(da.iiiIiiiIII, 1818);
            }
            int unused = da.iIIiiiiiii = da.iIIiiiiiii + 2;
        }
    } else {
        da.M();
        if (da.iIIiiiiiii == -6 || (da.iIIiiiiiii >= -8 && this.IIIiiiIiii.B() > 0)) {
            try {
                if (da.IiiiiIiii == null) {
                    LocationManager unused2 = da.IiiiiIiii = (LocationManager) this.IIIiiiIiii.getSystemService(t.b
                }
                da.IiiiiIiii.requestLocationUpdates(t.b("\nj\u001e"), 0, 0.0f, this.IIIiiiIiii.IiiiiiiIII);
                int unused3 = da.iIIiiiiiii = 1;
            } catch (Throwable th) {
            }
        }
        da.IIIiiiIiii.postDelayed(da.iiiIiiiIII, 2828);
    }
}
```

Example #3

```
/* renamed from: b043Cmmmmm reason: contains not printable characters */
private static void m3b043C() {
    f7b043704370437 = new String[]{aaoaaa.m38b041904190419("mqhp-0,`ik", 'z', '~', 1)};
}
```

jadx output

- String obfuscation / encryption
- Several depths of function
  - Mostly doing arithmetic computation on input
- Sometimes interleaves between native module (JNI) and Dalvik code

Example #4

```
v4 = a2;
v5 = a1;
v6 = a4;
v7 = a3;
v8 = calloc(1u, 0x20u);
v9 = v8;
if ( v8 )
{
  *v8 = sub_5F04;
  v8[1] = sub_5F44;
  v8[2] = sub_5F98;
  v8[3] = v5;
  sub_73D0(v7);
  sub_74E0(v7);
  v10 = ((int (__fastcall *)(JNIEnv *, int))(*v5)->NewStringUTF)(v5, v4);
  v14 = v6;
  v11 = ((int (__fastcall *)(JNIEnv *, int))(*v5)->NewStringUTF)(v5, v7);
  v12 = call_method(v5, (int)"java/lang/ClassLoader", (int)"getSystemClassLoader", (int)"()Ljava/lang/ClassLoader;");
  v9[4] = call_method(
            v5,
            (int)"dalvik/system/DexClassLoader",
            (int)"<init>",
            (int)"(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/ClassLoader;)V",
            v10,
            v11,
            0,
            v12);
  ((void (__fastcall *)(JNIEnv *, int))(*v5)->DeleteLocalRef)(v5, v10);
  ((void (__fastcall *)(JNIEnv *, int))(*v5)->DeleteLocalRef)(v5, v11);
  ((void (__fastcall *)(JNIEnv *, int))(*v5)->DeleteLocalRef)(v5, v12);
  v9[5] = sub_7394(v4);
  v9[6] = sub_7394(v7);
  v9[7] = sub_7394(v14);
}
return v9;
```

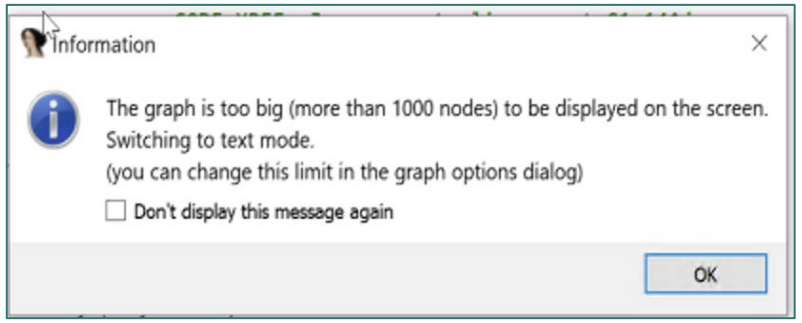
- DEX encryption
- DEX dynamic loading
  - Extract and load DEX files on the fly
- It can be done in both Dalvik and JNI

Example #5

```
public static Intent vWSe(Context context, D9bf d9bf, String str) {
    boolean z;
    Intent intent;
    char c;
    int i = 2 % 2;
    Intent intent2 = new Intent(context, WithdrawConfirmActivity.class);
    Intent intent3 = intent2;
    char[] cArr = {1, 2, 3, 4, 5, 6, ']', ']', 0, 8, 5, 3};
    int i2 = QQ6f;
    int i3 = i2 & 101;
    int i4 = -((i2 ^ 101) | (i2 & 101));
    int i5 = ((-i4) ^ i3) + ((i3 & (-i4)) << 1);
    i64H = i5 % 128;
    if (i5 % 2 != 0) {
    }
    intent2.putExtra(FJMD(cArr, 12, (byte) 11).intern(), d9bf.p233);
    int i6 = QQ6f;
    int i7 = (((i6 ^ 91) | (i6 & 91)) << 1) - (i6 ^ 91);
    i64H = i7 % 128;
    if (i7 % 2 != 0) {
    }
    String intern = FJMD(new char[]{1, 2, 3, 4, 9, 10, 11, 12, 13, 5, 7, 9}, 12, (byte) 51).intern();
    int i8 = QQ6f;
    int i9 = ((i8 & 68) << 1) + (i8 ^ 68);
    int i10 = ((i9 | -1) << 1) - (i9 ^ -1);
    i64H = i10 % 128;
    if (i10 % 2 != 0) {
        z = false;
    } else {
        z = true;
    }
    if (z) {
        intent3.putExtra(intern, str);
        intent = intent3;
    } else {
        intent3.putExtra(intern, str);
        intent = intent3;
        Object obj = null;
        super.hashCode();
    }
```

- Control flow obfuscation
- Often arithmetic based control flow divergence
- Very complex at first glance, but most of them are dead code!

Example #6



- JNI integration
  - Native ARM code
- Anything that can be done in Dalvik can be done in JNI using reflection APIs
- Some have obfuscated binary code as well

Example #7

```
v683 = sub_5100C(v616);
v684 = v683("/dev/qemu_pipe", 0LL);
if ( v684 )
{
  v685 = sub_5100C(v684);
  v686 = v685("/system/bin/nox-prop", 0LL);
  if ( v686 )
  {
    v704 = sub_5100C(v686);
    if ( v704("/system/bin/noxspeedup", 0LL) )
    {
      if ( MEMORY[0x34F0](&bad_string_arr, &unk_69418)
        && MEMORY[0x34F0]("ro.product.manufacturer", "motorola")
        && MEMORY[0x34F0]("ro.product.brand", "Android")
        && MEMORY[0x34F0]("ro.product.board", "shamu")
        && MEMORY[0x34F0]("ro.product.device", "shamu") )
      {
        v582 = 262152;
      }
      else if ( sub_368C("/proc/cpuinfo", "Intel(R) Core(TM) i") )
      {
        v582 = 262153;
      }
      else
      {
        v582 = sub_368C("/proc/misc", "vboxguest");
        if ( v582 )
```

- Rooting detection
- Many different, creative ways of detecting rooted devices, VM, debugging, etc.
  - File existence check (artifact)
  - Android property check
  - /proc/self/* check
  - /proc/net/unix check

Example #8

```
v1 = readApkDex((__int64)ptr, (int *)&v4);
if ( ptr )
{
  v3 = v1;
  free(ptr);
  v1 = v3;
  ptr = 0LL;
}
getSha1Hash((__int64)&v6, (__int64)v1, v4);
if ( !memcmp((const void *)IIIiIiiIii, &v6, 20u) )
  result = v0;
else
  result = 1LL;
```

- Re-packaging protection
- Integrity verification
  - Hash files
  - Cryptographic signatures
- Break re-packaging tools
  - Invalid attribute names in Manifest.xml

Alright, that was tough

How can we make it little bit more “clean”?

Fix bugs in open source tools (1/3)

- dexlib2 not handling array with element width of 0 with large element count correctly

fill-array-data-payload format

Name	Format	Description
ident	ushort = 0x0300	identifying pseudo-opcode
element_width	ushort	number of bytes in each element
size	uint	number of elements in the table
data	ubyte[]	data values

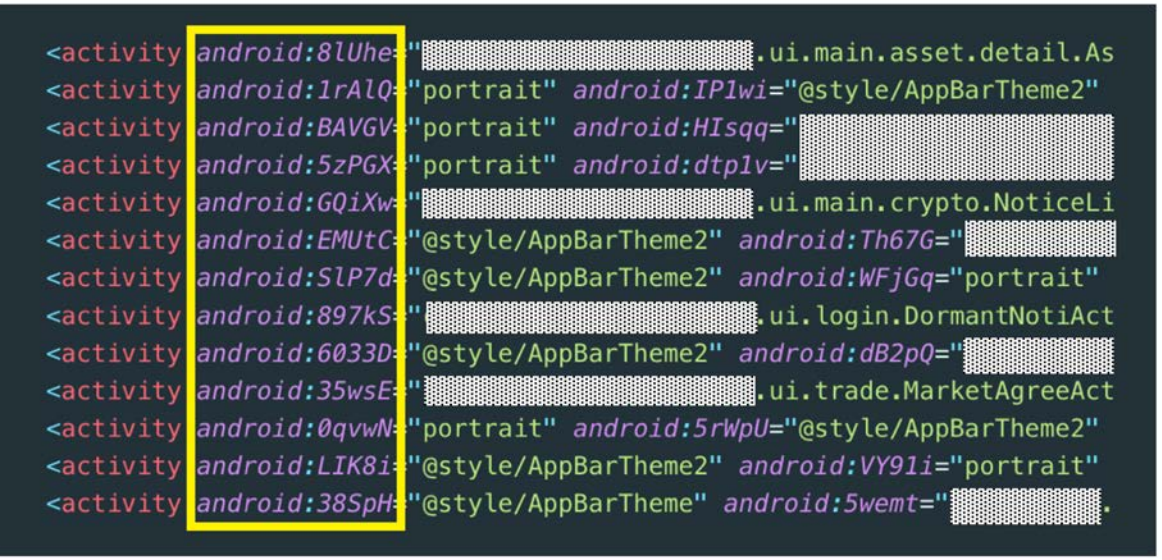
```
CODE:002A79A6                   .short 0x300
CODE:002A79A8                   .short 0
CODE:002A79AA                   .int 0xAFBDFA33
```

```
-       elementWidth = dexFile.getDataBuffer().readUshort(instructionStart + ELEMENT_WIDTH_OFFSET);
-       elementCount = dexFile.getDataBuffer().readSmallUint(instructionStart + ELEMENT_COUNT_OFFSET);
-       if (((long)elementWidth) * elementCount > Integer.MAX_VALUE) {
-           throw new ExceptionWithContext("Invalid array-payload instruction: element width*count overflows");
+       int localElementWidth = dexFile.getDataBuffer().readUshort(instructionStart + ELEMENT_WIDTH_OFFSET);
+
+       if (localElementWidth == 0) {
+           elementWidth = 1;
+           elementCount = 0;
+       } else {
+           elementWidth = localElementWidth;
+
+           elementCount = dexFile.getDataBuffer().readSmallUint(instructionStart + ELEMENT_COUNT_OFFSET);
+           if (((long) elementWidth) * elementCount > Integer.MAX_VALUE) {
+               throw new ExceptionWithContext("Invalid array-payload instruction: element width*count overflows");
+           }
```

- Fixed in smali version 2.3.1
- However, jadx uses v2.3 => Update build.gradle to use latest

Fix bugs in open source tools (2/3)

- Apktool not handling obfuscated attribute names in Android Manifest XML correctly




```
...
field public static final int useLevel = 16843167; // 0x101019f
field public static final int userVisible = 16843409; // 0x1010291
field public static final int usesCleartextTraffic = 16844012; // 0x10104ec
field public static final int value = 16842788; // 0x1010024
field public static final int valueFrom = 16843486; // 0x10102de
field public static final int valueTo = 16843487; // 0x10102df
field public static final int valueType = 16843488; // 0x10102e0
field public static final int variablePadding = 16843157; // 0x1010195
field public static final int vendor = 16843751; // 0x10103e7
field public static final int version = 16844057; // 0x1010519
field public static final int versionCode = 16843291; // 0x101021b
field public static final int versionCodeMajor = 16844150; // 0x1010576
field public static final int versionMajor = 16844151; // 0x1010577
...
```

- If Android namespace, use the binary value to resolve

  https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/api/current.txt

Fix bugs in open source tools (3/3)

- dexlib2 not gracefully handling a method with instruction that extends past the end of the method (truncated method)

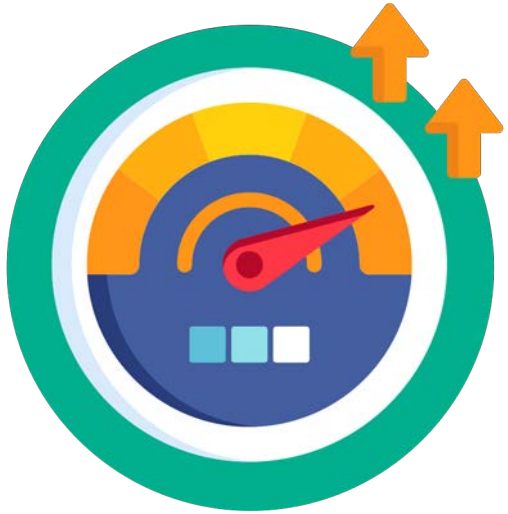
```
CODE:001A7CB4 # Method 2 (0x2)
CODE:001A7CB4 word_1A7CB4:    .short 0                # DATA XREF: CODE:003BAE69↓i
CODE:001A7CB4                                         # Number of registers : 0x0
CODE:001A7CB6                 .short 0                # Size of input args (in words) : 0x0
CODE:001A7CB8                 .short 0                # Size of output args (in words) : 0x0
CODE:001A7CBA                 .short 0                # Number of try_items : 0x0
CODE:001A7CBC                 .int 0                  # Debug info
CODE:001A7CC0                 .int 2                  # Size of bytecode (in 16-bit units): 0x2
CODE:001A7CC4 public static void a_lock.b()
CODE:001A7CC4                 invoke-super                    {v4, v0, v0, v0, v15}, <void a_lock.<clinit>() a_lock  clinit @V>
CODE:001A7CC4                                         # DATA XREF: CODE:003BAE91↓i
```

- Ignore truncated methods => Replace with NOP instruction
- Never gets called (otherwise Dalvik VM will do wrong things)

Code Deobfuscator + Optimizer

- Implement custom decompiler passes
  - Eliminate dead code
  - Optimize control flow
  - Constant evaluation and propagation
  - Heuristics

- Extended the existing tool: jadx
  - Fix decompiler bugs on the way ;)



Code Deobfuscator + Optimizer

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        try {
            XtAL = i2 % 128;
        } catch (ClassCastException e) {
            throw e;
        }
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        switch (i6 % 2 != 0 ? ')' : 'Y') {
            case 'Y':
                int i7 = 2 % 2;
                break;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1;
        try {
            XtAL = i9 % 128;
        } catch (IllegalStateException e2) {
            throw e2;
        }
        if (i9 % 2 == 0) {
        }
    }
}
```

Class constructor
(obfuscated)

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    try {
        eloq = i3 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    try {
        eloq = i5 % 128;
    } catch (ArrayStoreException e2) {
        throw e2;
    }
    switch (i5 % 2 != 0) {
        case false:
            int i6 = 96 / 0;
            return;
        default:
            return;
    }
}
```

XtAL method
(obfuscated)

koLf method
(obfuscated)

```
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = tqTj;
    int i3 = (i2 & 19) + ((i2 ^ 19) | (i2 & 19));
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i4 = eloq;
    int i5 = (((i4 ^ 5) | (i4 & 5)) << 1) - (((i4 & 5) ^ -1) & (i4 | 5));
    try {
        tqTj = i5 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i5 % 2 != 0) {
    }
    int i6 = tqTj;
    int i7 = (i6 & 39) + (i6 | 39);
    try {
        eloq = i7 % 128;
        if (i7 % 2 == 0) {
        }
        return intent2;
    } catch (ClassCastException e2) {
        throw e2;
    }
}
```

Code Deobfuscator + Optimizer

```java
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        try {
            XtAL = i2 % 128;
        } catch (ClassCastException e) {
            throw e;
        }
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        switch (i6 % 2 != 0 ? ')' : 'Y') {
            case 'Y':
                int i7 = 2 % 2;
                break;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1;
        try {
            XtAL = i9 % 128;
        } catch (IllegalStateException e2) {
            throw e2;
        }
        if (i9 % 2 == 0) {
        }
    }
}
```

Class constructor
(obfuscated)

- Switch-case statements are turned into if-else statements
- `default` case is used for `else` block
- `if` statement may not have `else` block
  - `switch` statement without `default` case

```java
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        try {
            XtAL = i2 % 128;
        } catch (ClassCastException e) {
            throw e;
        }
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        if ((i6 % 2 != 0 ? ')' : 'Y') == 'Y') {
            int i7 = 2 % 2;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1;
        try {
            XtAL = i9 % 128;
        } catch (IllegalStateException e2) {
            throw e2;
        }
        if (i9 % 2 == 0) {
        }
    }
}
```

Class constructor
(optimized-1)

Code Deobfuscator + Optimizer

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    try {
        eloq = i3 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    try {
        eloq = i5 % 128;
    } catch (ArrayStoreException e2) {
        throw e2;
    }
    switch (i5 % 2 != 0) {
        case false:
            int i6 = 96 / 0;
            return;
        default:
            return;
    }
}
```

XtAL method
(obfuscated)

- Switch-case statements are turned into if-else statements
- `default` case is used for `else` block
- `if` statement may not have `else` block
  - `switch` statement without `default` case

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    try {
        eloq = i3 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    try {
        eloq = i5 % 128;
    } catch (ArrayStoreException e2) {
        throw e2;
    }
    if (!(i5 % 2 != 0)) {
        int i6 = 96 / 0;
    }
}
```

XtAL method
(optimized-1)

```
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = tqTj;
    int i3 = (i2 & 19) + ((i2 ^ 19) | (i2 & 19));
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i4 = eloq;
    int i5 = (((i4 ^ 5) | (i4 & 5)) << 1) - (((i4 & 5) ^ -1) & (i4 | 5));
    try {
        tqTj = i5 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i5 % 2 != 0) {
    }
    int i6 = tqTj;
    int i7 = (i6 & 39) + (i6 | 39);
    try {
        eloq = i7 % 128;
        if (i7 % 2 == 0) {
        }
        return intent2;
    } catch (ClassCastException e2) {
        throw e2;
    }
}
```

koLf method
(unchanged)

Code Deobfuscator + Optimizer

```java
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        try {
            XtAL = i2 % 128;
        } catch (ClassCastException e) {
            throw e;
        }
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        if ((i6 % 2 != 0 ? ')' : 'Y') == 'Y') {
            int i7 = 2 % 2;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1;
        try {
            XtAL = i9 % 128;
        } catch (IllegalStateException e2) {
            throw e2;
        }
        if (i9 % 2 == 0) {
        }
    }
}
```

Class constructor
(optimized-1)

- Unnecessary try-catch statements are removed
- The exception wouldn't likely occur
- But even if it does, the code just re-throws the exception, thus it makes no semantic difference to remove the `catch` block

```java
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        XtAL = i2 % 128;
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        if ((i6 % 2 != 0 ? ')' : 'Y') == 'Y') {
            int i7 = 2 % 2;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1;
        XtAL = i9 % 128;
        if (i9 % 2 == 0) {
        }
    }
}
```

Class constructor
(optimized-2)

Code Deobfuscator + Optimizer

```java
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    try {
        eloq = i3 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    try {
        eloq = i5 % 128;
    } catch (ArrayStoreException e2) {
        throw e2;
    }
    if (!(i5 % 2 != 0)) {
        int i6 = 96 / 0;
    }
}
```

XtAL method
(optimized-1)

- Unnecessary try-catch statements are removed
- The exception wouldn't likely occur
- But even if it does, the code just re-throws the exception, thus it makes no semantic difference to remove the `catch` block

```java
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    eloq = i5 % 128;
    if (!(i5 % 2 != 0)) {
        int i6 = 96 / 0;
    }
}
```

XtAL method
(optimized-2)

```java
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = tqTj;
    int i3 = (i2 & 19) + ((i2 ^ 19) | (i2 & 19));
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i4 = eloq;
    int i5 = (((i4 ^ 5) | (i4 & 5)) << 1) - (((i4 & 5) ^ -1) & (i4 | 5));
    try {
        tqTj = i5 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i5 % 2 != 0) {
    }
    int i6 = tqTj;
    int i7 = (i6 & 39) + (i6 | 39);
    try {
        eloq = i7 % 128;
        if (i7 % 2 == 0) {
        }
        return intent2;
    } catch (ClassCastException e2) {
        throw e2;
    }
}
```

koLf method
(optimized-1)

koLf method
(optimized-2)

```java
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = tqTj;
    int i3 = (i2 & 19) + ((i2 ^ 19) | (i2 & 19));
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i4 = eloq;
    int i5 = (((i4 ^ 5) | (i4 & 5)) << 1) - (((i4 & 5) ^ -1) & (i4 | 5));
    tqTj = i5 % 128;
    if (i5 % 2 != 0) {
    }
    int i6 = tqTj;
    int i7 = (i6 & 39) + (i6 | 39);
    eloq = i7 % 128;
    if (i7 % 2 == 0)
    }
    return intent2;
}
```

Code Deobfuscator + Optimizer

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        XtAL = i2 % 128;
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        if ((i6 % 2 != 0 ? ')' : 'Y') == 'Y') {
            int i7 = 2 % 2;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1
        XtAL = i9 % 128;
        if (i9 % 2 == 0) {
        }
    }
}
```

Class constructor
(optimized-2)

- Replace private static fields with fixed constant value
- This is a specific heuristic for this obfuscator, but it helps the later stages – constant evaluation and propagation
  - The static field values are initialized to either 0 or 1, and stay the same throughout the code

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = ((0 | 29) << 1) - ((0 & -30) | ((0 ^ -1) & 29));
        XtAL = i % 128;
        if (i % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i2 = ((1 ^ 65) | (1 & 65)) << 1;
        int i3 = ((1 ^ -1) & 65) | (1 & -66);
        int i4 = ((-i3) & i2) + (i2 | (-i3));
        UKLD = i4 % 128;
        if ((i4 % 2 != 0 ? ')' : 'Y') == 'Y') {
            int i5 = 2 % 2;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i6 = ((0 & 49) - ((-(-((0 ^ 49) | (0 & 49)))) ^ -1)) - 1;
        XtAL = i6 % 128;
        if (i6 % 2 == 0) {
        }
    }
}
```

Class constructor
(optimized-3)

Code Deobfuscator + Optimizer

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    eloq = i5 % 128;
    if (!(i5 % 2 != 0)) {
        int i6 = 96 / 0;
    }
}
```

XtAL method
(optimized-2)

- Replace private static fields with fixed constant value
- This is a specific heuristic for this obfuscator, but it helps the later stages – constant evaluation and propagation
  - The static field values are initialized to either 0 or 1, and stay the same throughout the code

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = ((0 ^ 57) - ((-(-((0 & 57) << 1))) ^ -1)) - 1;
    eloq = i2 % 128;
    if (i2 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i3 = (0 ^ 85) + ((0 & 85) << 1);
    eloq = i3 % 128;
    if (!(i3 % 2 != 0)) {
        int i4 = 96 / 0;
    }
}
```

XtAL method
(optimized-3)

```
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = tqTj;
    int i3 = (i2 & 19) + ((i2 ^ 19) | (i2 & 19));
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i4 = eloq;
    int i5 = (((i4 ^ 5) | (i4 & 5)) << 1) - (((i4 & 5) ^ -1) & (i4 | 5));
    tqTj = i5 % 128;
    if (i5 % 2 != 0) {
    }
    int i6 = tqTj;
    int i7 = (i6 & 39) + (i6 | 39);
    eloq = i7 % 128;
    if (i7 % 2 == 0) {
    }
    return intent2;
}
```

koLf method
(optimized-2)

```
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = (0 & 19) + ((0 ^ 19) | (0 & 19));
    eloq = i2 % 128;
    if (i2 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i3 = (((1 ^ 5) | (1 & 5)) << 1) - (((1 & 5) ^ -1) & (1 | 5));
    tqTj = i3 % 128;
    if (i3 % 2 != 0) {
    }
    int i4 = (0 & 39) + (0 | 39);
    eloq = i4 % 128;
    if (i4 % 2 == 0) {
    }
    return intent2;
}
```

koLf method
(optimized-2)

Code Deobfuscator + Optimizer

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = ((0 | 29) << 1) - ((0 & -30) | ((0 ^ -1) & 29));
        XtAL = i % 128;
        if (i % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i2 = ((1 ^ 65) | (1 & 65)) << 1;
        int i3 = ((1 ^ -1) & 65) | (1 & -66);
        int i4 = ((-i3) & i2) + (i2 | (-i3));
        UKLD = i4 % 128;
        if ((i4 % 2 != 0 ? ')' : 'Y') == 'Y') {
            int i5 = 2 % 2;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i6 = ((0 & 49) - ((-(-((0 ^ 49) | (0 & 49)))) ^ -1)) - 1;
        XtAL = i6 % 128;
        if (i6 % 2 == 0) {
        }
    }
}
```

Class constructor
(optimized-3)

- Evaluate constant arithmetic operations
- Propagate and replace with the final constant literal value

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        XtAL = 29;
        if (1 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        UKLD = 66;
        if ((0 != 0 ? ')' : 'Y') == 'Y') {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        XtAL = 49;
        if (1 == 0) {
        }
    }
}
```

Class constructor
(optimized-4)

Code Deobfuscator + Optimizer

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = ((0 ^ 57) - ((-(-((0 & 57) << 1))) ^ -1)) - 1;
    eloq = i2 % 128;
    if (i2 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i3 = (0 ^ 85) + ((0 & 85) << 1);
    eloq = i3 % 128;
    if (!(i3 % 2 != 0)) {
        int i4 = 96 / 0;
    }
}
```

XtAL method
(optimized-3)

- Evaluate constant arithmetic operations
- Propagate and replace with the final constant literal value

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    eloq = 57;
    if (1 == 0) {
    }
    withdrawActivity.Eilq();
    eloq = 85;
    if (!(1 != 0)) {
        int i = 96 / 0;
    }
}
```

XtAL method
(optimized-4)

```
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = (0 & 19) + ((0 ^ 19) | (0 & 19));
    eloq = i2 % 128;
    if (i2 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i3 = (((1 ^ 5) | (1 & 5)) << 1) - (((1 & 5) ^ -1) & (1 | 5));
    tqTj = i3 % 128;
    if (i3 % 2 != 0) {
    }
    int i4 = (0 & 39) + (0 | 39);
    eloq = i4 % 128;
    if (i4 % 2 == 0) {
    }
    return intent2;
}
```

koLf method
(optimized-3)

```
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    eloq = 19;
    if (1 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    tqTj = 6;
    if (0 != 0) {
    }
    eloq = 39;
    if (1 == 0) {
    }
    return intent2;
}
```

koLf method
(optimized-4)

Code Deobfuscator + Optimizer

Class constructor (optimized-4)

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        XtAL = 29;
        if (1 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        UKLD = 66;
        if ((0 != 0 ? ')' : 'Y') == 'Y') {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        XtAL = 49;
        if (1 == 0) {
        }
    }
}
```

XtAL method (optimized-4)

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    eloq = 57;
    if (1 == 0) {
    }
    withdrawActivity.Eilq();
    eloq = 85;
    if (!(1 != 0)) {
        int i = 96 / 0;
    }
}
```

koLf method (optimized-4)

```
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    eloq = 19;
    if (1 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    tqTj = 6;
    if (0 != 0) {
    }
    eloq = 39;
    if (1 == 0) {
    }
    return intent2;
}
```

- Remove assignments to the private static fields that are not used any more
  - Dead code elimination
  - Constant values are already evaluated and propagated
- Specific heuristics for this obfuscator
  - It can be generically extended for other cases

Class constructor (optimized-5)

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        if (1 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        if ((0 != 0 ? ')' : 'Y') == 'Y') {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        if (1 == 0) {
        }
    }
}
```

XtAL method (optimized-5)

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    if (1 == 0) {
    }
    withdrawActivity.Eilq();
    if (!(1 != 0)) {
        int i = 96 / 0;
    }
}
```

koLf method (optimized-5)

```
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    if (1 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    if (0 != 0) {
    }
    if (1 == 0) {
    }
    return intent2;
}
```

Code Deobfuscator + Optimizer

Class constructor (optimized-5)

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        if (1 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        if ((0 != 0 ? ')' : 'Y') == 'Y') {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        if (1 == 0) {
        }
    }
}
```

XtAL method (optimized-5)

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    if (1 == 0) {
    }
    withdrawActivity.Eilq();
    if (!(1 != 0)) {
        int i = 96 / 0;
    }
}
```

koLf method (optimized-5)

```
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    if (1 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    if (0 != 0) {
    }
    if (1 == 0) {
    }
    return intent2;
}
```

- Perform constant evaluation on `if` statements
- Remove unreachable code path after evaluating constants
  - Example: `if(false){}`

Class constructor (optimized-6)

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        FJMD = new int[zfbe.XtAL.values().length];
        try {
        } catch (NoSuchFieldError unused) {
        }
        FJMD[zfbe.XtAL.koLf.ordinal()] = 1;
        char c = 'Y';
        if (c == 'Y') {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
        }
    }
}
```

XtAL method (optimized-6)

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    withdrawActivity.Eilq();
    boolean z = true;
    if (!z) {
        int i = 96 / 0;
    }
}
```

koLf method (optimized-6)

```
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    return intent2;
}
```

Code Deobfuscator + Optimizer

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        FJMD = new int[zfbe.XtAL.values().length];
        try {
        } catch (NoSuchFieldError unused) {
        }
        FJMD[zfbe.XtAL.koLf.ordinal()] = 1;
        char c = 'Y';
        if (c == 'Y') {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
        }
    }
}

static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    withdrawActivity.Eilq();
    boolean z = true;
    if (!z) {
        int i = 96 / 0;
    }
}

public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    return intent2;
}
```

Class constructor (optimized-6)

XtAL method (optimized-6)

koLf method (optimized-6)

- Remove unused variables
- Inline expressions
  - If PHI node only has one value, the value gets inlined

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        FJMD = new int[zfbe.XtAL.values().length];
        try {
        } catch (NoSuchFieldError unused) {
        }
        FJMD[zfbe.XtAL.koLf.ordinal()] = 1;
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
        }
    }
}

static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    withdrawActivity.Eilq();
}

public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    return intent2;
}
```

Class constructor (optimized-7)

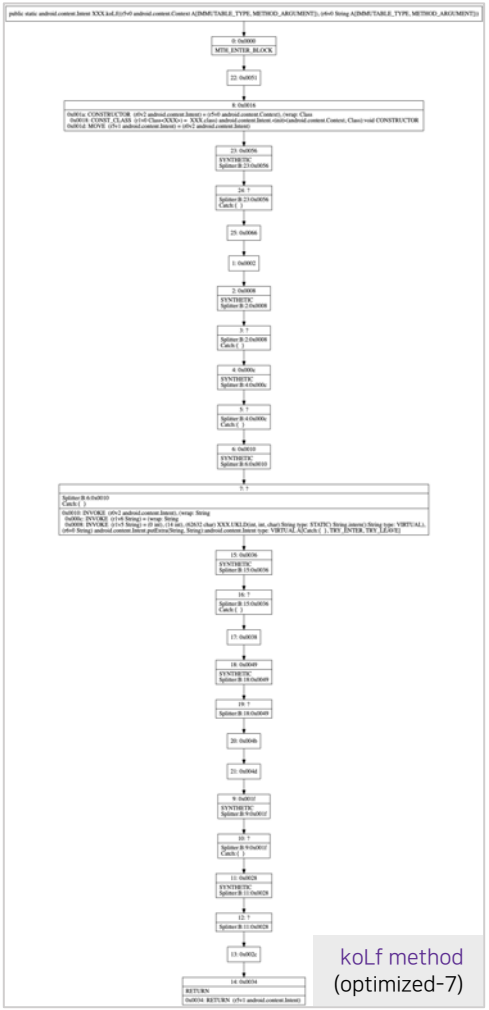
XtAL method (optimized-7)

koLf method (optimized-7)

Code Deobfuscator + Optimizer

```java
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    return intent2;
}
```

koLf method
(optimized-7)



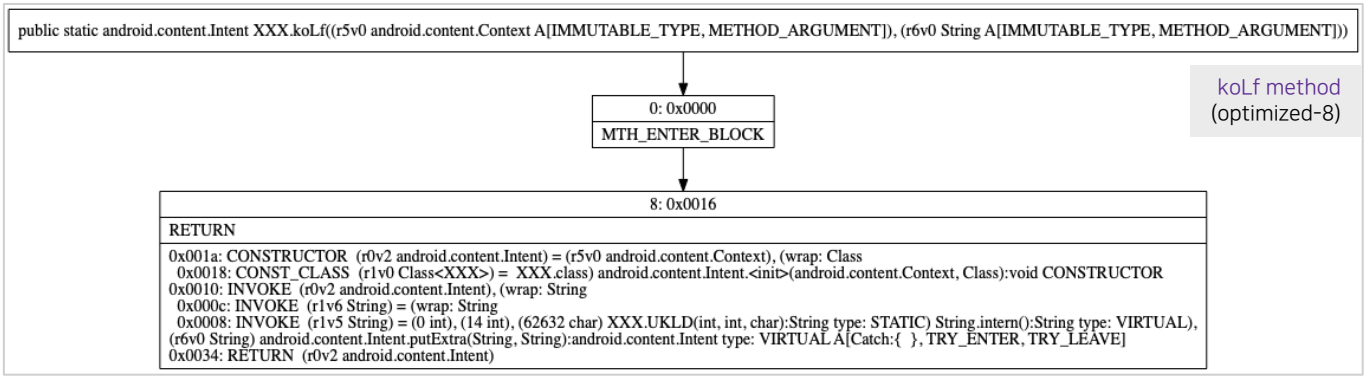
koLf method
(optimized-7)

- Clean up block nodes
- Remove empty blocks
  - aka, basic blocks that do not contain any instructions

- This helps jadx recognize more patterns in later decompilation passes

```java
public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    return intent;
}
```

koLf method
(optimized-8)



koLf method
(optimized-8)

Code Deobfuscator + Optimizer

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        FJMD = new int[zfbe.XtAL.values().length];
        try {
        } catch (NoSuchFieldError unused) {
        }
        FJMD[zfbe.XtAL.koLf.ordinal()] = 1;
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
        }
    }
}

static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    withdrawActivity.Eilq();
    return;
}

public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    return intent;
}
```

Class constructor (optimized-8)

XtAL method (optimized-8)

koLf method (optimized-8)

- Emulate the string decryption method invocations
  - Replace with String literal
- Uses smalivm by Caleb Fenton

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            FJMD[zfbe.XtAL.koLf.ordinal()] = 1;
        } catch (NoSuchFieldError unused) {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
        }
    }
}

static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    withdrawActivity.Eilq();
    return;
}

public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    intent.putExtra("PARAM_CURRENCY".intern(), str);
    return intent;
}
```

Class constructor (optimized-9)

XtAL method (optimized-9)

koLf method (optimized-9)

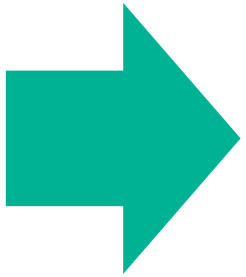
Code Deobfuscator + Optimizer

```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        int[] iArr;
        UKLD = 0;
        XtAL = 1;
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            iArr = FJMD;
        } catch (NoSuchFieldError unused) {
        }
        int i = UKLD;
        int i2 = ((i | 29) << 1) - ((i & -30) | ((i ^ -1) & 29));
        try {
            XtAL = i2 % 128;
        } catch (ClassCastException e) {
            throw e;
        }
        if (i2 % 2 == 0) {
        }
        iArr[zfbe.XtAL.koLf.ordinal()] = 1;
        int i3 = XtAL;
        int i4 = ((i3 ^ 65) | (i3 & 65)) << 1;
        int i5 = ((i3 ^ -1) & 65) | (i3 & -66);
        int i6 = ((-i5) & i4) + (i4 | (-i5));
        UKLD = i6 % 128;
        switch (i6 % 2 != 0 ? ')' : 'Y') {
            case 'Y':
                int i7 = 2 % 2;
                break;
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
            return;
        }
        int i8 = UKLD;
        int i9 = ((i8 & 49) - ((-(-((i8 ^ 49) | (i8 & 49)))) ^ -1)) - 1;
        try {
            XtAL = i9 % 128;
        } catch (IllegalStateException e2) {
            throw e2;
        }
        if (i9 % 2 == 0) {
        }
    }
}
```

```
static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    int i = 2 % 2;
    int i2 = tqTj;
    int i3 = ((i2 ^ 57) - ((-(-((i2 & 57) << 1))) ^ -1)) - 1;
    try {
        eloq = i3 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i3 % 2 == 0) {
    }
    withdrawActivity.Eilq();
    int i4 = tqTj;
    int i5 = (i4 ^ 85) + ((i4 & 85) << 1);
    try {
        eloq = i5 % 128;
    } catch (ArrayStoreException e2) {
        throw e2;
    }
    switch (i5 % 2 != 0) {
        case false:
            int i6 = 96 / 0;
            return;
        default:
            return;
    }
}
```

```
public static Intent koLf(Context context, String str) {
    int i = 2 % 2;
    Intent intent = new Intent(context, WithdrawActivity.class);
    Intent intent2 = intent;
    int i2 = tqTj;
    int i3 = (i2 & 19) + ((i2 ^ 19) | (i2 & 19));
    eloq = i3 % 128;
    if (i3 % 2 == 0) {
    }
    intent.putExtra(UKLD(0, 14, 62632).intern(), str);
    int i4 = eloq;
    int i5 = (((i4 ^ 5) | (i4 & 5)) << 1) - (((i4 & 5) ^ -1) & (i4 | 5));
    try {
        tqTj = i5 % 128;
    } catch (ClassCastException e) {
        throw e;
    }
    if (i5 % 2 != 0) {
    }
    int i6 = tqTj;
    int i7 = (i6 & 39) + (i6 | 39);
    try {
        eloq = i7 % 128;
        if (i7 % 2 == 0) {
        }
        return intent2;
    } catch (ClassCastException e2) {
        throw e2;
    }
}
```



```
static /* synthetic */ class AnonymousClass4 {
    static final /* synthetic */ int[] FJMD;
    private static int UKLD;
    private static int XtAL;

    static {
        FJMD = new int[zfbe.XtAL.values().length];
        try {
            FJMD[zfbe.XtAL.koLf.ordinal()] = 1;
        } catch (NoSuchFieldError unused) {
        }
        try {
            FJMD[zfbe.XtAL.vWSe.ordinal()] = 2;
        } catch (NoSuchFieldError unused2) {
        }
    }
}

static /* synthetic */ void XtAL(WithdrawActivity withdrawActivity) {
    withdrawActivity.Eilq();
    return;
}

public static Intent koLf(Context context, String str) {
    Intent intent = new Intent(context, WithdrawActivity.class);
    intent.putExtra("PARAM_CURRENCY".intern(), str);
    return intent;
}
```

Code Deobfuscator + Optimizer

```
public final void vWSe(yTB9 ytb9) {
    if (ytb9 instanceof haqS) {
        int codePointAt = (getResources().getString(R.string2.RES_2133459413).substring(4, 5).codePointAt(0) - 74) - 1;
        int i = (codePointAt & -1) + (codePointAt | -1);
        int codePointAt2 = getResources().getString(R.string2.RES_2133459899).substring(1, 3).codePointAt(0);
        int i2 = -((codePointAt2 & 0) | ((codePointAt2 ^ -1) & -1));
        int i3 = (i2 | -112) + (i2 & -112);
        int i4 = ((i3 | -1) << 1) - (i3 ^ -1);
        int i5 = -getPackageName().length();
        String intern = UKLD(i, i4, (char) (((-i5) & -26) + ((-i5) | -26))).intern();
        int i6 = -getResources().getString(R.string2.RES_2133459495).substring(0, 1).codePointAt(0);
        int i7 = (((-i6) | -36) << 1) - (-36 ^ (-i6));
        ApplicationInfo applicationInfo = getApplicationInfo();
        int length = getResources().getString(R.string2.RES_2133459577).substring(0, 3).length();
        if (!intern.equalsIgnoreCase(UKLD(i7, 6, (char) (((((length ^ -1) & 10229) | (-10230 & length)) - (((10229 & length) << 1) ^ -1)) - 1)).intern())) {
            String str = this.M5O8.FJMD;
            int i8 = -(-getResources().getString(R.string2.RES_2133460143).substring(0, 17).codePointAt(6));
            int i9 = (i8 & 0) | ((i8 ^ -1) & -1);
            int i10 = (((-i9) ^ -20) + (((-i9) & -20) << 1)) - 1;
            int length2 = (((getPackageName().length() + 18) - 1) + 0) - 1;
            int i11 = -getResources().getString(R.string2.RES_2133459749).substring(0, 6).length();
            String intern2 = UKLD(i10, length2, (char) (((-i11) & -6) + ((-i11) | -6))).intern();
            ApplicationInfo applicationInfo2 = getApplicationInfo();
            int codePointAt3 = getResources().getString(R.string2.RES_2133459931).substring(5, 7).codePointAt(0);
            int i12 = (codePointAt3 ^ -21) + ((codePointAt3 & -21) << 1);
            int codePointAt4 = getPackageName().codePointAt(6);
            int i13 = (codePointAt4 & 0) | ((codePointAt4 ^ -1) & -1);
            o3W8.FJMD(str, intern2, UKLD(121, i12, (char) (((((-i13) & -110) + ((-i13) | -110)) + 0) - 1)).intern(), null, this.yxFR);
        } else {
            String str2 = this.M5O8.FJMD;
            int length3 = "+100M".substring(0, 5).length();
            int i14 = ((length3 ^ 45) | (length3 & 45)) << 1;
            int i15 = -(((length3 & 45) ^ -1) & (length3 | 45));
            int i16 = (i14 ^ i15) + ((i14 & i15) << 1);
            int i17 = -getResources().getString(R.string2.RES_2133459749).substring(0, 6).codePointAt(1);
            int i18 = -i17;
            int i19 = ((i18 ^ -1) & -96) | (i18 & 95);
            int i20 = -(((-i17) & -96) << 1);
            int i21 = ((-i20) ^ i19) + ((i19 & (-i20)) << 1);
            int codePointAt5 = getResources().getString(R.string2.RES_2133458945).substring(0, 3).codePointAt(1);
            String intern3 = UKLD(i16, i21, (char) (((codePointAt5 | -115) << 1) - (codePointAt5 ^ -115))).intern();
            int length4 = getPackageName().length();
            int i22 = (length4 ^ 39) + ((length4 & 39) << 1);
            int codePointAt6 = getPackageName().codePointAt(23);
            int i23 = (codePointAt6 & 83) | ((codePointAt6 ^ -1) & -84);
            int i24 = -((codePointAt6 & -84) << 1);
            int i25 = ((-i24) ^ i23) + ((i23 & (-i24)) << 1);
            int codePointAt7 = getResources().getString(R.string2.RES_2133459480).substring(2, 4).codePointAt(1);
            o3W8.FJMD(str2, intern3, UKLD(i22, i25, (char) ((((62849 ^ codePointAt7) | (62849 & codePointAt7)) << 1) - (((62849 & codePointAt7) ^ -1) & (codePointAt7 | 62849)))).intern(), null, this.yxFR);
        }
    }
}
```

`getResources().getString()` gets a constant string, and `codePointAt()` returns a constant integer

`getPackageName().length()` gets a constant integer

Length of the String literal is a constant integer

Code Deobfuscator + Optimizer

```
public final void vWSe(yTB9 ytb9) {
    if (ytb9 instanceof haqS) {
        if (!UKLD(39, 5, (char) 0).intern().equalsIgnoreCase(UKLD(44, 6, (char) 10232).intern())) {
            o3W8.FJMD(this.M5O8.FJMD, UKLD(79, 42, (char) 0).intern(), UKLD(121, 11, (char) 0).intern(), null, this.yxFR);
        } else {
            o3W8.FJMD(this.M5O8.FJMD, UKLD(50, 15, (char) 0).intern(), UKLD(65, 14, (char) 62950).intern(), null, this.yxFR);
        }
    }
}
```

Code Emulation

```
intent.getStringExtra(FJMD(new char[]{1, 2, 3, 4, 5, 6, ']', ']', 0, 8, 5, 3}, 12, (byte) 11).intern());
 intent.getStringExtra(FJMD(new char[]{1, 2, 3, 4, 9, 10, 11, 12, 13, 5, 7, 9}, 12, (byte) 51).intern());
 intent.getStringExtra(FJMD(new char[]{1, 2, 3, 4, 7, 13, 13, 5, 0, 15, 206}, 11, (byte) 123).intern());
= intent.getBooleanExtra(FJMD(new char[]{1, 2, 3, 4, 19, 0, 17, 1, 7, 13, 13, 5, 0, 15, 195}, 15, (byte)
 intent.getStringExtra(FJMD(new char[]{1, 2, 3, 4, 23, 9, 2, 17, 2, 15, '\\'}, 11, (byte) 13).intern());
 intent.getStringExtra(FJMD(new char[]{1, 2, 3, 4, 22, 8, 8, 0, 16, 11, 2, 11, 20, 13, 3, 4, 15, 2, 17, 1
```

- Many depths of methods are called to “decrypt” the string
- Emulate the code execution in VM to compute the concrete value

```
public final class BFQR {
    public static int vWSe(int i, int i2) {
        return i / i2;
    }

    public static int UKLD(int i, int i2) {
        return i % i2;
    }

    public static int vWSe(int i, int i2, int i3) {
        return (i * i3) + i2;
    }

    public static int FJMD(int i, int i2) {
        return ((i + i2) - 1) % i2;
    }
}
```

```
private static String FJMD(char[] cArr, int i, byte r18) {
    int vWSe;
    int UKLD;
    int vWSe2;
    int UKLD2;
    char[] cArr2 = yxFR;
    int i2 = i;
    char c = CX1;
    char c2 = r18;
    char[] cArr3 = cArr;
    char[] cArr4 = new char[i2];
    if (i2 % 2 != 0) {
        i2--;
        cArr4[i2] = (char) (cArr3[i2] - c2);
    }
    if (i2 > 1) {
        int i3 = 0;
        for (i3 += 2; i3 < i2; i3 += 2) {
            char c3 = cArr3[i3];
            char c4 = cArr3[i3 + 1];
            if (c3 == c4) {
                cArr4[i3] = (char) (c3 - c2);
                cArr4[i3 + 1] = (char) (c4 - c2);
            } else if (UKLD == UKLD2) {
                int FJMD = BFQR.FJMD(vWSe, c);
                int FJMD2 = BFQR.FJMD(vWSe2, c);
                int vWSe3 = BFQR.vWSe(FJMD, UKLD, c);
                int vWSe4 = BFQR.vWSe(FJMD2, UKLD2, c);
                cArr4[i3] = cArr2[vWSe3];
                cArr4[i3 + 1] = cArr2[vWSe4];
            } else if (vWSe == vWSe2) {
                int FJMD3 = BFQR.FJMD(UKLD, c);
                int FJMD4 = BFQR.FJMD(UKLD2, c);
                int vWSe5 = BFQR.vWSe(vWSe, FJMD3, c);
                int vWSe6 = BFQR.vWSe(vWSe2, FJMD4, c);
                cArr4[i3] = cArr2[vWSe5];
                cArr4[i3 + 1] = cArr2[vWSe6];
            } else {
                int vWSe7 = BFQR.vWSe(vWSe, UKLD2, c);
                int vWSe8 = BFQR.vWSe(vWSe2, UKLD, c);
                cArr4[i3] = cArr2[vWSe7];
                cArr4[i3 + 1] = cArr2[vWSe8];
            }
        }
    }
    return new String(cArr4);
}
```

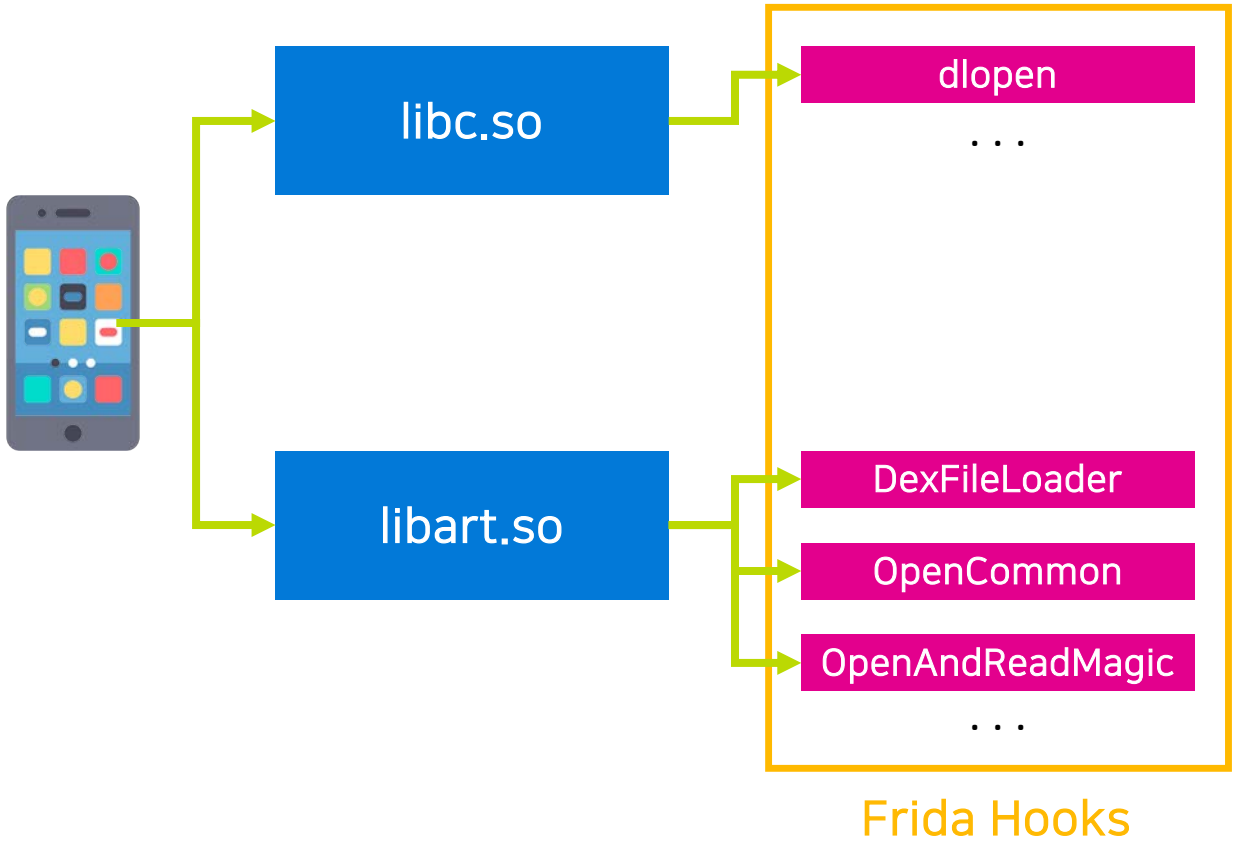
What about native (JNI) methods?

Frida – Dynamic code loading (1/2)

- Some DEX and/or .so executable files are stored encrypted
- When the app runs, they are loaded dynamically using the following APIs
  - DexClassLoader (Dalvik)
  - PathClassLoader (ART)
  - dlopen (Native shared object)

```
String absolutePath = next.getAbsolutePath();
sb.append(CoreConstants.COLON_CHAR);
sb.append(absolutePath);
int previousIndex = listIterator.previousIndex();
strArr[previousIndex] = absolutePath;
fileArr[previousIndex] = next;
zipFileArr[previousIndex] = new ZipFile(next);
StringBuilder insert = new StringBuilder().insert(0, absolutePath);
insert.append(t.b("4\t\u0015"));
dexFileArr[previousIndex] = DexFile.loadDex(absolutePath, insert.toString(), 0);
```

Frida – Dynamic code loading (2/2)



- Hook low-level APIs to intercept the DEX/.so files
  - DexFileLoader
  - OpenCommon
  - OpenAndReadMagic
  - dlopen
- Usually the obfuscator stub loads the “real” app code
  - Sometimes the app code itself is not obfuscated!

Frida – File deletion

- After certain files are decrypted and used, the obfuscator often unlink the files
  - May still be mapped in memory
- Hook `unlink` call and prevent the file deletion!
  - Some code checks for the return value => return 0 for success

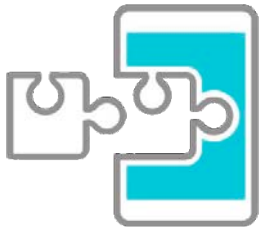
```
int __usercall unlink2@<eax>(int a1@<eax>, int ebp0@<ebp>, char *a3@<edi>, int a2)
{
  int v4; // esi
  int v5; // edx
  int result; // eax
  int (__cdecl *v7)(int); // eax
  char *v8; // eax
  int v9; // edx
  int v10; // ecx
  char v11; // [esp+1Ch] [ebp-70h]
  int v12; // [esp+48h] [ebp-44h]
  unsigned int v13; // [esp+7Ch] [ebp-10h]

  v4 = a1;
  v13 = __readgsdword(0x14u);
  v5 = sub_1810(a1, &v11);
  result = -1;
  if ( v5 >= 0 )
  {
    if ( v12 >= 0 && ((v7 = unlink) != 0 || (v7 = dword_304AC) != 0 || (v7 = sub_7DD0(143112), (dword_304AC = v7) != 0)) )
      result = v7(v4);
    else
      result = -1;
  }
  if ( __readgsdword(0x14u) != v13 )
  {
    v8 = sub_4550();
    result = sub_BA00(v8, v9, v10, ebp0, a3, a2);
  }
  return result;
}
```

```
while ( *v33 );
if ( v32 )
{
  if ( dword_35368 > 25 )
  {
    if ( "base.odex" != -5 && v32 != -1 )
      sub_5E80(v32 + 1, "odex");
  }
  else if ( "dex" && v32 != -1 )
  {
    sub_5E80(v32 + 1, "dex");
  }
  unlink2(v49, v3, v49, v37);
  *v32 = 0;
  unlink2(v49, v3, v49, v34);
}
```

Frida – Root detection (1/4)

- Detection of rooted devices is one of the key features that most solutions offer
- Since it's done in code by looking at the system artifacts, we can hook the right things to bypass the detection
- People have found clever, creative, and sometimes weird ways of detecting rooted devices



Frida – Root detection (2/4)

- File access
  - /system/app/Superuser.apk
  - /system/xbin/su
  - /system/bin/noxspeedup
  - magisk
  - …
- System info
  - /proc/cpuinfo
  - /dev/qemu_pip
  - pm list

```
Interceptor.attach(Module.findExportByName("libc.so", "stat"), {

  onEnter: function(args) {
    var name = Memory.readUtf8String(args[0]);
    if (name.includes("magisk")) {
      Memory.writeByteArray(args[0], utf8AbFromStr("/blah"));
      this.is_rootchk = true;
    }
  },

  onLeave: function(retval) {
    if (this.is_rootchk) {
      retval.replace(-1);
    }
  }

});
```

Frida – Root detection (3/4)

- Sometimes the solutions look for Frida hooks and processes
  - Ironically, this can be circumvented using Frida :p
- Unix socket check for Frida / remote debugging detection

```
Interceptor.attach(Module.findExportByName("libc.so", "connect"), {
  onEnter: function(args) {
    var memory = Memory.readByteArray(args[1], 64);
    var b = new Uint8Array(memory);
    if (b[2] == 0x69 && b[3] == 0xa2 &&                                  // Frida server port: 27042 (0x69a2)
        b[4] == 0x7f && b[5] == 0x00 && b[6] == 0x00 && b[7] == 0x01) {  // Frida server IP: 127.0.0.1 (0x7f000001)
      this.frida_detection = true;
    }
  },
  onLeave: function(retval) {
    if (this.frida_detection) {
      retval.replace(-1);
    }
  }
});
```

Frida – Root detection (4/4)

- Custom ROMs considered to be ”rooted” as well
- Basic utilities are executed to check any anomaly in system
  - Android properties
  - mount table
  - Binary existence
- Fake/virtual filesystem access emulation can be done as well

```
Interceptor.attach(Module.findExportByName("libc.so", "execve"),
 {
  onEnter: function(args) {
    var name = Memory.readUtf8String(args[0]);
    var arg = args[1];
    while (true) {
      var tmp = arg.readPointer();
      if (tmp.toInt32() == 0) break;
      log.push(tmp.readUtf8String());
      arg = arg.add(Process.pointerSize);
    }
    var black_list = ['getprop', 'which']
    for (var key in black_list) {
      if (name.includes(key)) {
        Memory.writeUtf8String(args[0], 'false');
      }
    }
  }
});
```

Frida – SSL bypass

- Most apps now protect their API calls with SSL

- Two ways to bypass SSL
  1. Man-in-the-middle (MITM) attack with a custom certificate and CA
     - Upside: It's easy, quick, and can log all streams at once
     - Downside: Many apps ship with SSL pinning built-in to the app
     - Bypassing SSL pinning can be done with Frida
  2. Log the master key for SSL communication
     - Upside: It's more generic since we can just hook openssl functions
     - Downside: Each handshake creates a new key; Multiple requests, multiple keys
     - Wireshark supports importing the SSL key

Conclusion

Conclusion

- Some questions remain to be answered
  - Is it effective?
  - How do customers quantify the value-add?
  - What about non-Korean apps?

Obscurity does not provide real security

Offensive Security Research is essential

Proper threat modeling and penetration testing are required

Secure code and design are way more effective than obfuscation :)

Future Work

Performance Comparison

How do obfuscators impact CPU and battery usage?

Build Better Obfuscator / Protector

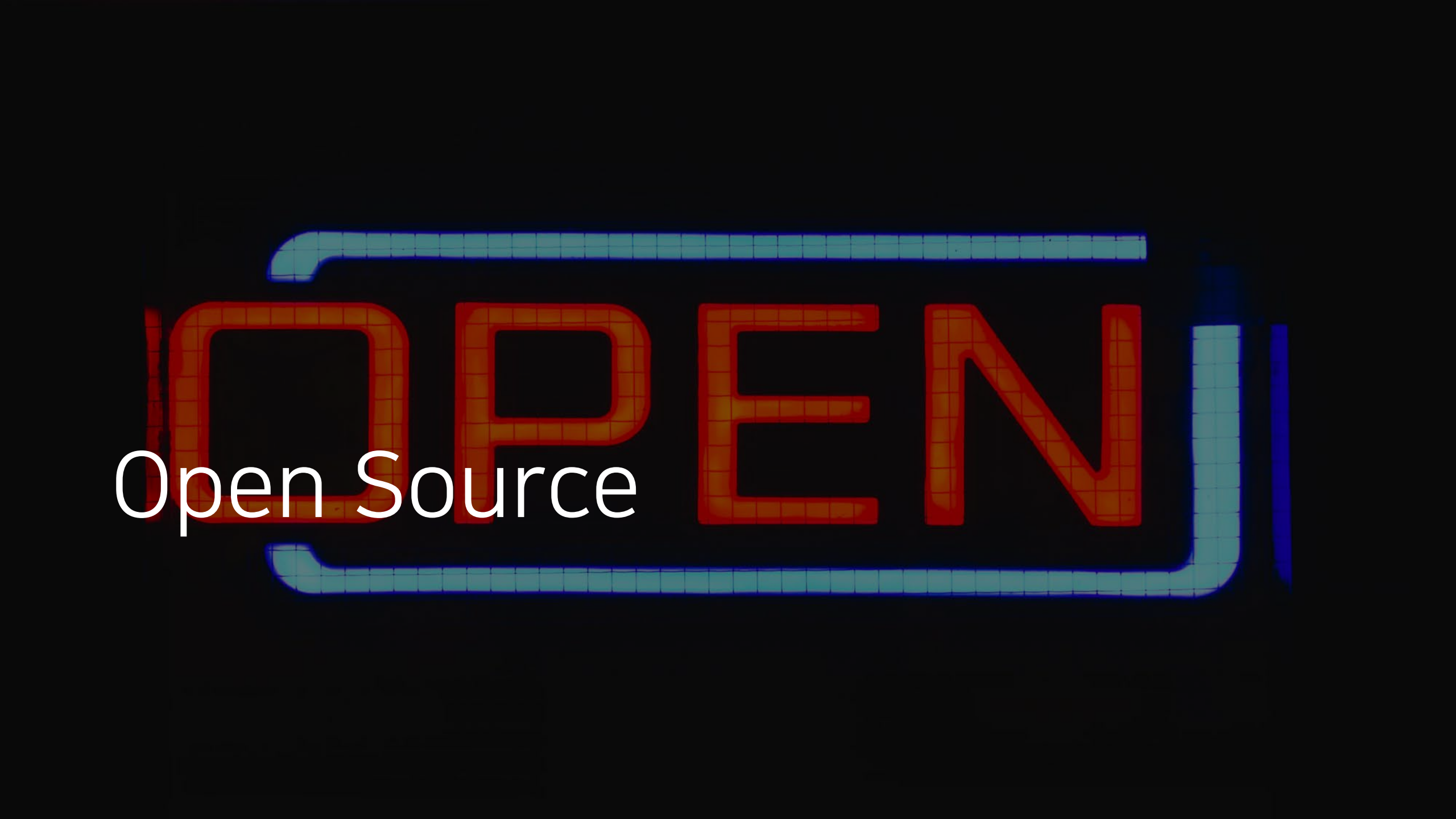
We know what will make people's lives a lot harder :p

Analysis of More Obfuscator Samples

We recently found more interesting, challenging samples!

Publish Whitepaper

More details, in-depth analysis and results will be shared


OPEN
Open Source

Open Source

- We plan to open source all code patches to improve existing tools (upstream)
  - More resilient to intentionally “broken” code and data
  - Optimizes obfuscated code for better readability of decompiled code

- We also plan to open source a few scripts
  - Bypass root / debug detection
  - Extract DEX and .so files
  - Bypass SSL pinning / Log SSL keys

https://github.com/theori-io

Thank You