

"I am not the D'r.0,1d You are Looking For": An Analysis of Android Malware Obfuscation

Samir Mody
Senior Manager
K7 Threat Control Lab



Making a Hash of it?



Default detection strategy
Checksums & Cloud-lookups



Resource limitations
Memory, Disk space, Power



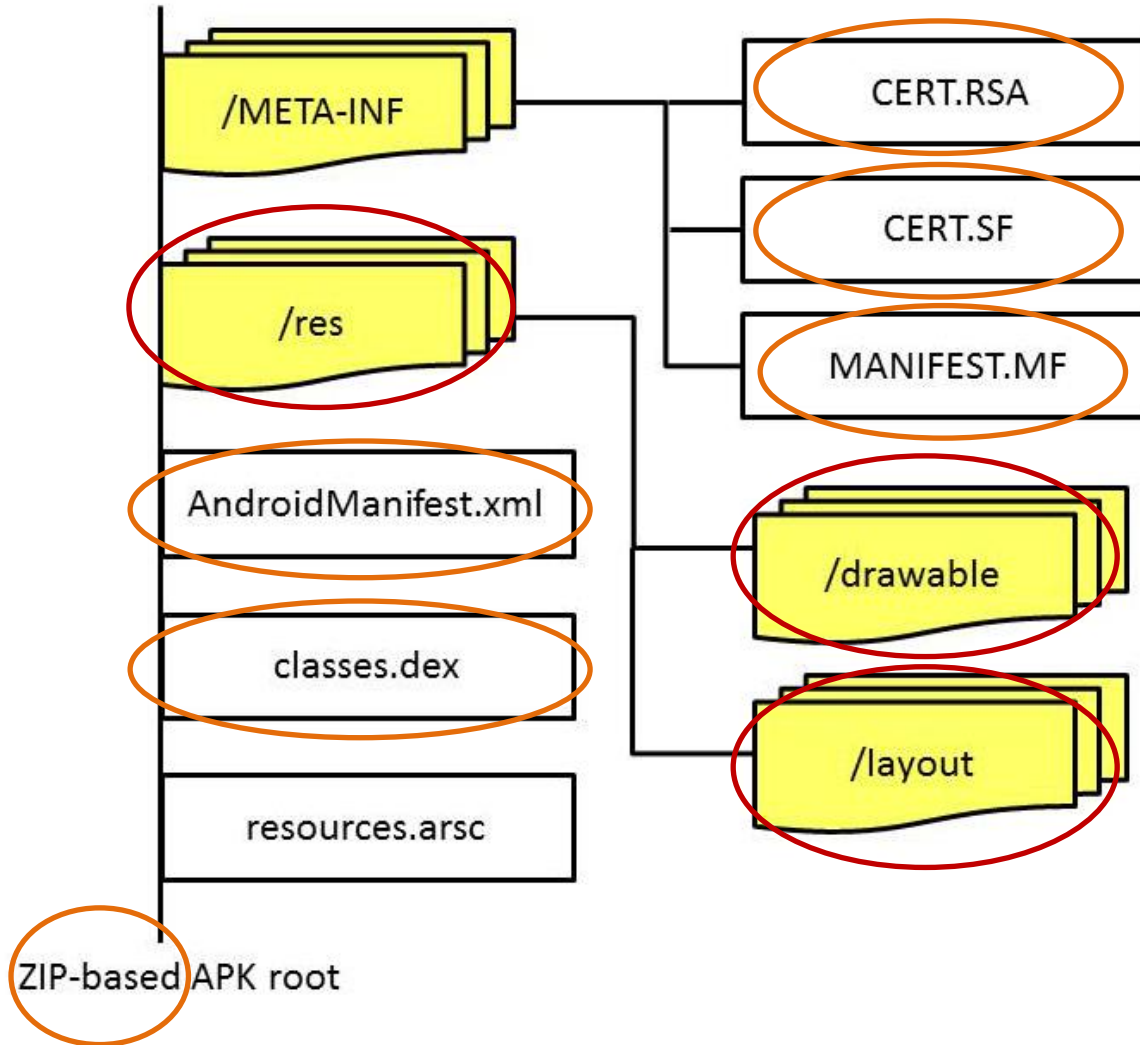
Relatively primitive obfuscation
Previously...

Metamorphosis

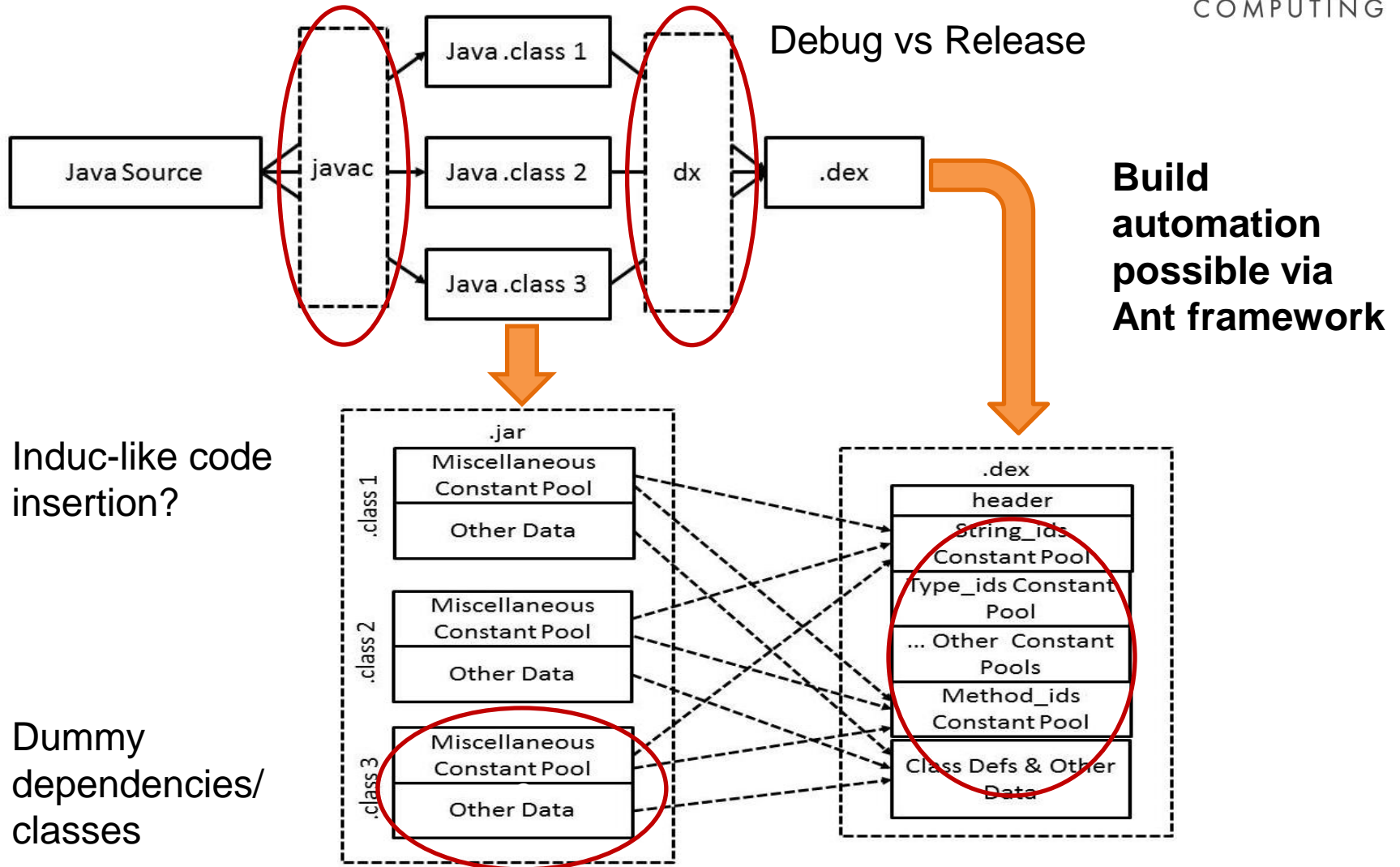


Windows style...

Typical APK Structure



Obfuscating Classes.dex via the Build



String Manipulation: Android.Obad.A

String Decryptor

```

[011b9c] com.android.system[011b9c] com.android.system.admin.oC11c11.oC11c11:(II)Ljava/lang/String;
011bac: d806 0662      |011bac: d806 0662      |0000
011bba: 2200 0700      |011bba: 2200 0700      |0002
private static String oC11c11(int iParam1, int iParam2)
{
    int j;
    byte[] arrayOfByte2 = o11c1c1c;
    int i = 0;
    byte[] arrayOfByte1 = new byte[11];
    int k;
    iParam1 = iParam1 + 98;
    if (arrayOfByte2 == null)
        k = 11;
    for (j = 0; ; j = arrayOfByte2[iParam2])
    {
        iParam2++;
        iParam1 = -1 + (k + j);
        arrayOfByte1[i] = (byte)iParam1;
        i++;
        if (i >= 11)
            break;
        k = iParam1;
    }
    return new String(arrayOfByte1, 0);
}

```

```

add-int/lit8 v6, v6, #int 98 // #62
new-instance v0, Ljava/lang/String; // type@0087
sget-object v5, Lcom/android/system/admin/oC11c11;<snip>
const/4 v4, #int 0 // #0
const/16 v1, #int 11 // #b
new-array v1, v1, [B // type@00a0
if-nez v5, 0015 // +000a
const/16 v2, #int 11 // #b
const/4 v3, #int 0 // #0
add-int/lit8 v7, v7, #int 1 // #01
add-int/2addr v2, v3
add-int/lit8 v6, v2, #int -1 // #ff
int-to-byte v2, v6
aput-byte v2, v1, v4
add-int/lit8 v4, v4, #int 1 // #01
const/16 v2, #int 11 // #b
if-lt v4, v2, 0023 // +0007
const/4 v2, #int 0 // #0
invoke-direct {v0, v1, v2}, Ljava/lang/String;<snip>
return-object v0
move v2, v6
aget-byte v3, v5, v7
goto 0010 // -0016

```

Randomised & tokenised variables

Proguard

“... **obfuscates** your code by removing unused code and **renaming classes, fields and methods** with **semantically obscure names**. The result is a smaller sized .apk file that is more **difficult to reverse engineer**”

- <http://developer.android.com/tools/help/proguard.html>

- **Classes:** `.myclass(es)` \rightarrow `.a(,b,c,d,...)`
- **Methods & variables:** explicit string references stripped

Massaging the DEX Header

Strong
content
validation

No
timestamp

No
unused
metadata
fields

Entry	Type	Description
Signature	BYTE[0x8]	DEX_FILE_MAGIC. Currently "dex\n035\0"
Checksum	DWORD	adler32 data corruption check
Hash	BYTE[0x14]	SHA1 data security check
File_size	DWORD	Size of entire file on disk
Header_size	DWORD	Currently 0x70
Endianness_flag	DWORD	0x12345678 (little-endian) or 0x78563412 (big-endian)
Link_size	DWORD	Link section size, or 0 if not statically-linked
Link_raw_offset	DWORD	Offset from start of file or 0 if not statically-linked
Map_raw_offset	DWORD	Offset from start of file to map item, 0 if absent
String_ids_sz_off_pair	{ DWORD, DWORD }	Size and location from start of file to string id list
Type_ids_sz_off_pair	{ DWORD, DWORD }	Size and location from start of file to type id list
Proto_ids_sz_off_pair	{ DWORD, DWORD }	Size and location from start of file to prototype id list
Field_ids_sz_off_pair	{ DWORD, DWORD }	Size and location from start of file to field id list
Method_ids_sz_off_pair	{ DWORD, DWORD }	Size and location from start of file to method id list
Class_def_sz_off_pair	{ DWORD, DWORD }	Size and location from start of file to class definition list
Data_sz_off_pair	{ DWORD, DWORD }	Size (aligned) and location from start of file to data section

DEX Byte-code Obfuscation: **Nop**

Original byte code

```
0009e0: 0000          |0000: nop // spacer *****
0009e2: 1a01 0000     |0001: const-string v1, ""
0009e6: 1200          |0003: const/4 v0, #int 0
0009e8: 0000          |0004: nop // spacer *****
0009ea: 6e10 2800 0700 |0005: invoke-virtual {v7},
Ljava/lang/String;.length: ()I
0009f0: 0a04          |0008: move-result v4
```

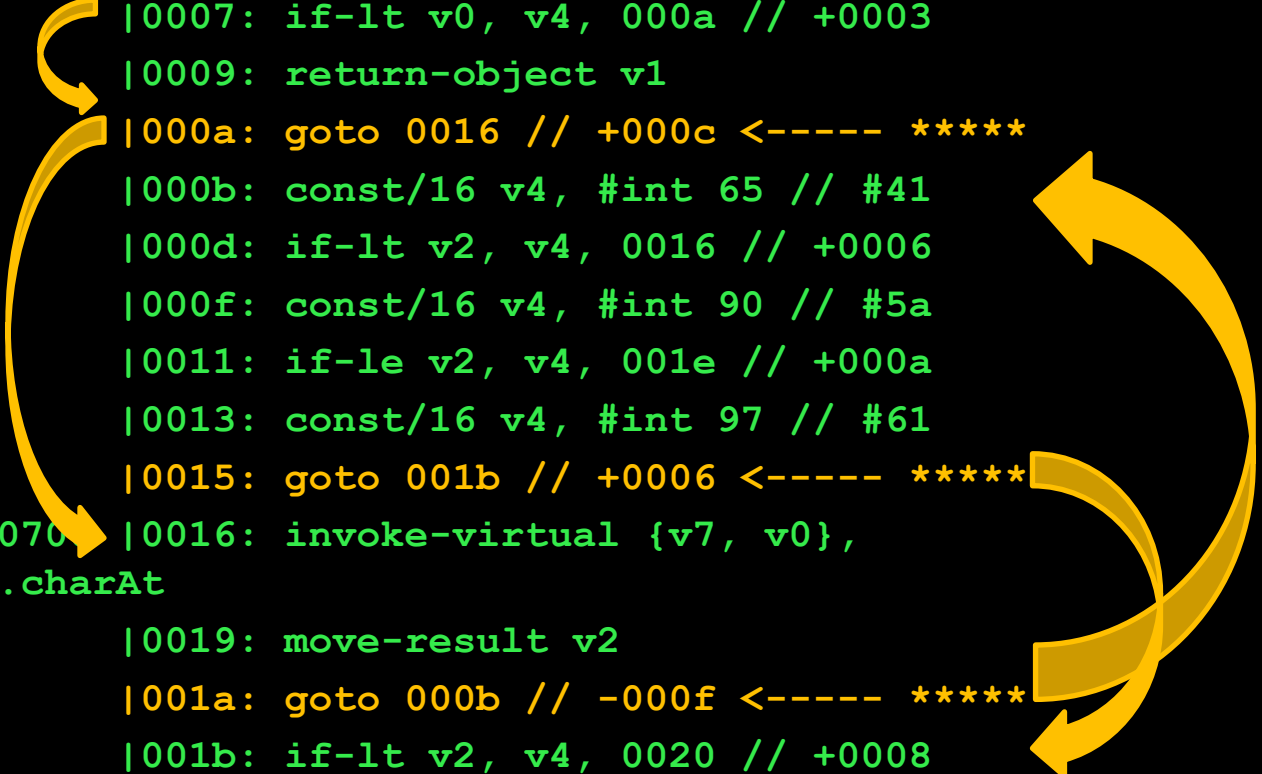
DEX Byte-code Obfuscation: **Goto**

Original byte code

```

0009ee: 3440 0300 |0007: if-lt v0, v4, 000a // +0003
0009f2: 1101      |0009: return-object v1
0009f4: 280c      |000a: goto 0016 // +000c <----- *****
0009f6: 1304 4100 |000b: const/16 v4, #int 65 // #41
0009fa: 3442 0600 |000d: if-lt v2, v4, 0016 // +0006
0009fe: 1304 5a00 |000f: const/16 v4, #int 90 // #5a
000a02: 3742 0a00 |0011: if-le v2, v4, 001e // +000a
000a06: 1304 6100 |0013: const/16 v4, #int 97 // #61
000a0a: 2806      |0015: goto 001b // +0006 <----- *****
000a0c: 6e20 2700 0700 |0016: invoke-virtual {v7, v0},
Ljava/lang/String;.charAt
000a12: 0a02      |0019: move-result v2
000a14: 28f1      |001a: goto 000b // -000f <----- *****
000a16: 3442 0800 |001b: if-lt v2, v4, 0020 // +0008

```



DEX Byte-code Obfuscation: **Move & Binop**



Original byte code

```
000a14: 1309 3a00 |001a: const/16 v9, #int 58 // #3a
000a18: d098 4000 |001c: add-int/lit16 v8, v9, #int 64 // #40
000a1c: 0184      |001e: move v4, v8
000a1e: 3642 0a00 |001f: if-gt v2, v4, 0029 // +000a
000a22: e006 0208 |0021: shl-int/lit8 v6, v2, #int 8 // #08
000a26: d766 0090 |0023: xor-int/lit16 v6, v6, #int -28672 //
#9000
000a2a: d762 00b0 |0025: xor-int/lit16 v2, v6, #int -20480 //
#b000
000a2e: e102 0208 |0027: shr-int/lit8 v2, v2, #int 8 // #08
000a32: 8e23      |0029: int-to-char v3, v2
```

DEX Byte-code Obfuscation: **Invoke**

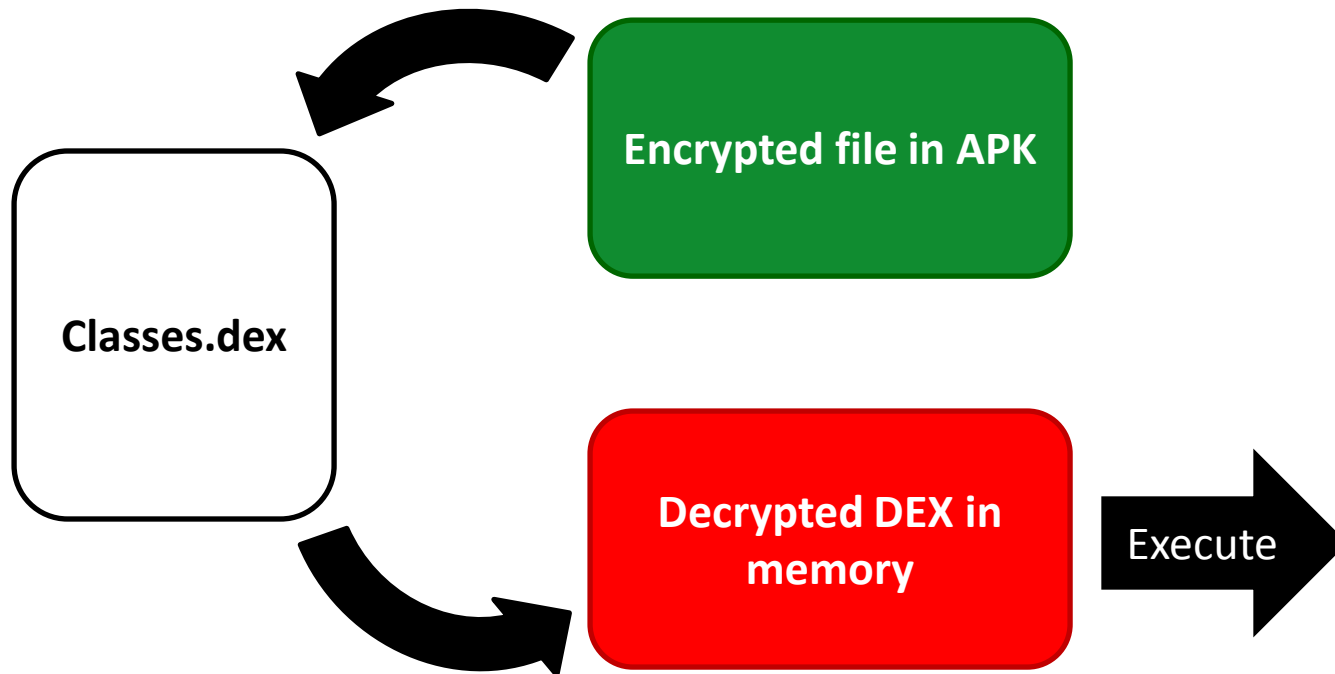
Dynamic method call via reflection

```
import java.lang.reflect.Method;
//Some other directives here
try {
Class cls = Class.forName("com.some.ClassName");
Object obj = cls.newInstance();
//Inspect class for method and invoke
Method mthd = cls.getDeclaredMethod(decr_mthd_str, null);
mthd.invoke(obj, null);
```

DEX Byte-code Obfuscation via JNI

Source: 'Code Protection in Android' – Schulz, uni-bonn.de

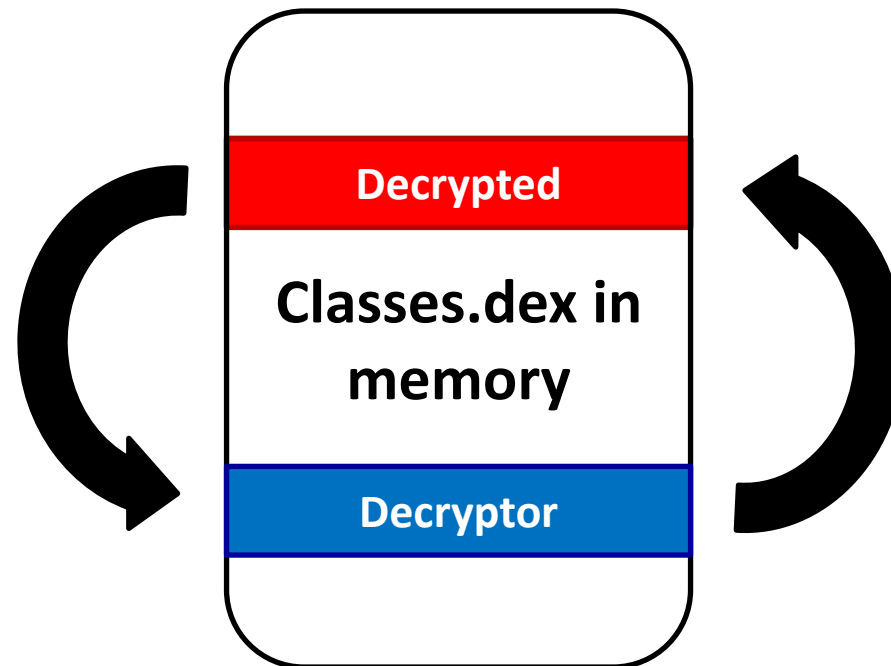
The Dropper



DEX Byte-code Obfuscation via JNI

Source: 'Code Protection in Android' – Schulz, uni-bonn.de

The Self-modifier



Detection Strategies



Checksums will not work very well
But judicious parsed object matching possible



Cloud scan: time lag and confidentiality issues



Dexopt: updated to flag non-standard code/data



IEEE Taggant System: Dexguard, Arxan, etc
But what about free Proguard?

Queries

